This is the unofficial page of the Ganga CMSSW Plugin (GangaCMS)

At Uniandes, we have developed a plugin in Ganga for CMS software framework. We called it **GangaCMS**. Ganga is a great application for job submission to any batch system including the Grid. Its motto is "Configure once - run anywhere".

Disclaimer: this is not an official initiative from CMS nor from the Ganga team. The information contained in this website is for general information purposes and valid only within our research group.

Get started with Ganga

- First of all you need to make sure you can run Ganga on your machine (or from where you will submit jobs). Details on installation are found on the official pages. Ganga is installed on the lxplus machines and one way to get it setup from your account is to add an *alias* in your shell configuration file (.bashrc or *.cshrc):

```
# this is to get Ganga added to your environment
alias gangaenv 'eval `/afs/cern.ch/sw/ganga/install/etc/ganga_setup.py --version=latest --interac
```

- Ganga in our **Tier-3** (installed under /opt/exp_soft/ganga/):

```
# this is to get Ganga added to your environment
alias gangaenv 'eval `/opt/exp_soft/ganga/ganga_setup.py --version=latest --interactive --experim
```

Get the Plugin

- Get a copy of the plugin. At the moment it is in our group SVN repository (not included in the official Ganga release):

```
svn co svn+ssh://svn.cern.ch/reps/cmsuniandes/Users/aosorio/Code/GangaCMS
```

First time Configuration

- After downloading the plugin, you are now ready to run Ganga for the first time. Setup the environment and start Ganga with option "-g":

```
<lxplus249> gangaenv
.....
Setting up Ganga 5.3.1 (csh,generic)
<lxplus249> ganga -g
```

The option -g creates a hidden configuration file for Ganga (**.gangarc**). We need to add a few lines in there to drive the plugin. Open **.gangarc** in your favourite editor and add/edit the following lines (you need to adapt them to **your** case):

```
## ....... Edit the following lines

RUNTIME_PATH = /afs/cern.ch/user/a/aosorio/Work/GangaCMS

gangadir = /afs/cern.ch/user/a/aosorio/scratch0/gangadir

## ....... Add the following lines anywhere in the configuration file
## .. dataTwiki is optional - this is the URL to a twiki where dataset files are located (in asci

[CMSSW]
```

```
dataOutput = /castor/cern.ch/user/a/aosorio/gridfiles/ganga
dataTwiki = https://twiki.cern.ch/twiki/pub/Main/CMSUniandesGroupSUSY

[CMSCAF]
copyCmd = rfcp
mkdirCmd = rfmkdir
```

- These options correspond to:
  - **RUNTIME_PATH**: to tell Ganga where the plugin is located
  - **gangadir**: Ganga creates a repository for your jobs. Tell Ganga where you want this repository to be created (needs space on disk)
  - **[CMSSW]** and **[CMSCAF]**: these sections contain the corresponding options to be given to the GangaCMS plugin (for example, **dataOutput** tells Ganga where is the massive storage element to save the output from your jobs)

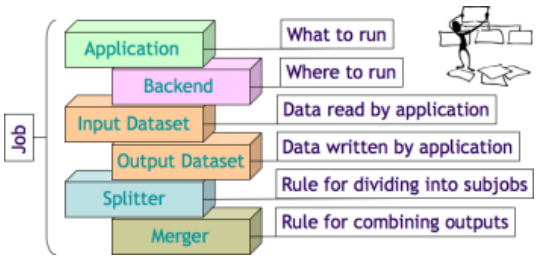
- Regarding Grid, the following lines could be edited in the **.gangarc** file:

```
#  Enables/disables the support of the GLITE middleware
EDG_ENABLE = False

#....

#  Enables/disables the support of the GLITE middleware
GLITE_ENABLE = True

#  sets the LCG-UI environment setup script for the GLITE middleware
# for lxplus
GLITE_SETUP = /afs/cern.ch/cms/LCG/LCG-2/UI/cms_ui_env.sh

# for yali
GLITE_SETUP = /opt/glite/etc/profile.d/grid-env.sh
#.....

#  sets the name of the grid virtual organisation
VirtualOrganisation = cms
```

A full list of the current options and their defaults is described here:

[CMSSW]	Description	Default
Option		
arch	platform/architecture	slc4_ia32_gcc345
cmsswdir	Path to CMSSW	$VO_CMS_SW_DIR
version	Default version of CMSSW	CMSSW_3_2_5
dataOutput	The place where Outputdata should go	$HOME/scratch0
dataTwiki	Points to a Twiki where datafile list can be uploaded	empty
workArea	Top directory where user creates its CMSSW working area	$HOME
dbsPath	Path to dbsCommandLine.py script	/afs/.../cms/dbs-client/DBS_2_0_6/lib/DBSAPI
dbsCommand	The command line DBS client	dbsCommandLine.py

Definition of a Job in Ganga



From Ganga homepage

The Application

A job in Ganga is make upon different building blocks. The two main blocks are the application and the backend. In our case, the application is the **cmsRun** executable.

At the moment, the **cmsRun** application has the following attributes:

Attribute	Description	Default
platform	OS platform where the application is supposed to run	slc4_ia32_gcc345
version	CMSSW version	CMSSW_3_2_5
args	simple list of arguments associated to a list of parameters	empty
uselibs	if the application depends upon user build libraries (0=false, 1=true)	0
cfgfile	configuration file for cmsRun	empty

Splitters and Mergers

Job splitting

SplitByFiles

A very simple job splitter was implemented: **SplitByFiles**. As its name indicates, a job is splitted in n subjobs given a partition of the input dataset. You will need to construct the splitter object by passing a Ganga File, containing a plain list of the dataset file names, and tell what type of data is going to be used ("local" prepends the prefix "file:", "castor" prepends "rfio:" etc). Here a show a snippet of the splitter definition:

```
ff      = File(name='/opt/CMS/CMSSW_2_2_3/src/ForGangaTest/SimpleAnalyzer/files.txt')
fdata = CMSDataset( ff , 'local' )
myjob.inputdata = fdata
```

ArgSplitter

Ganga comes with an **Argument Splitter**: your provide a list of n-arguments and Ganga builds n-subjobs having each one the specific set of arguments. We adapted this functionality to modify the configuration file that drives the **cmsRun** application. You will need first to understand what parameter, its type and value/argument:

```
# You want the following configuration parameter to change in each job:

process.source = cms.Source("EmptySource",
                                                    firstEvent = cms.untracked.uint32(1001) )
#-------------------------------------------------------------------------------/
```

You need to use the **cmsRun()** application method **add_cfg_Parameter** which takes two arguments: parameter and its type. Ganga will append at the end of the **cfg.py** file the appropriate line and pass the argument. You can add as many lines you want but make sure they match also the number of arguments in the

list i.e.

```
subjob_1:
parameter_1 = parameter_type_1 ------> [ [ arg_1_1,
parameter_2 = parameter_type_2 ------>     arg_1_2,
...
parameter_n = parameter_type_n ------>     arg_1_n ],

subjob_2:
parameter_1 = parameter_type_1 ------>    [ arg_2_1,
parameter_2 = parameter_type_2 ------>      arg_2_2,
...
parameter_n = parameter_type_n ------>      arg_2_n ],
...
```

```
# app is your application object of type cmsRun()
# in this example we will change the first event of each job.
app.add_cfg_Parameter('process.source.firstEvent','cms.untracked.uint32')

# List of Arguments for this job
arguments = [ [1] , [11] , [21] , [31] ]

# Set the Argument Splitter for this job
myjob.splitter = ArgSplitter( args = arguments )

# ArgSplitter will produce 4 subjobs in this case
```

The effect on the **cfg.py** file will be: **process.source.firstEvent = cms.untracked.uint32(1)** for subjob 1 for example.

More: ArgSplitter in the Ganga documention.

ArgSplitterX

The ArgSplitterX is an extended version of the normal ArgSplitter already described. In its functionality, it works the same as the ArgSplitter however it has new added feature(s):

- For MC simulation, each subjob is assigned a unique random seed to the generator. How it works:
- You can maintain in your script the same structure used for the ArgSplitter but replace it with the **ArgSplitterX** type:

```
myjob.splitter = ArgSplitterX()
myjob.splitter.args = arguments #where arguments are the ones you have previously define
```

- To have the random seed added to each subjob, you just need to add the following lines to your script:

```
myjob.splitter.AddRndSeed()
```

- That's all ... in principle this should work fine. How it works: for the moment the random seed is taken based on time in milliseconds (+min + seconds). This is not ideal yet but it is first good approximation given the limitation I have on the size of the integer. (More work and tests to be done)

Job merging

Ganga comes with some great merging plugins, among them **RootMerger** which collects the root output from all jobs and merges it (using hadd). The RootMerger has attributes files, overwrite and ignorefailed, all of

them self explanatory. I put here an example of the RootMerger definition:

```
rm = RootMerger()
rm.files = ['histo.root']         #  files to merge
rm.overwrite = True             # Overwrite output files
rm.ignorefailed = True        # ignore root files that failed to open
```

Examples

The following script illustrates all main characteristics of a job configuration:

```
from GangaCMS.Lib.CMSexe import *

#construct the cmsRun application object
app = cmsRun()
app.uselibs = 1
app.cfgfile = File(name='/opt/CMS/CMSSW_2_2_3/src/ForGangaTest/SimpleAnalyzer/simpleanalyzer_cfg.
app.version = 'CMSSW_2_2_3'

#construct the job with backend Local
myjob = Job( application = app, backend = 'Local' )

#set the data you want to get back from the job
myjob.outputsandbox.append('histo.root')

#define a data set
ff      = File(name='/opt/CMS/CMSSW_2_2_3/src/ForGangaTest/SimpleAnalyzer/files.txt')
fdata = CMSDataset( ff , 'local' )
myjob.inputdata = fdata

#create a splitter
sp = SplitByFiles()
sp.filesPerJob = 1
sp.maxFiles = -1

#create a root merger
rm = RootMerger()
rm.files = ['histo.root']
rm.overwrite = True
rm.ignorefailed = True

myjob.splitter = sp
myjob.merger = rm

#submit job
myjob.submit()
```

LXPLUS and CAF queues

This table summarizes the CAF batch queues available in addition to the usual LSF queue on lxplus (1nh, 1nd, 1nw):

Name	cmscaf1nh	cmscaf1nd	cmscaf1nw
max jobs/user	100	100	10
max length	1 norm. hour	1 norm. day	1 norm. week

(cmscaf1nd replaces cmscaf8nh from July 2009).

Backends

Available Backends Runtime Handlers

Here is a list of the backends runtime handlers we have implemented so far:

Backend	Tested	Dataset
Local	✅	CMSDataset
LSF	✅	CMSDataset
CRAB	✅	CMSDatasetPath
LCG	✅	CMSDataset
PBS	✅	CMSDataset

LCG

With the LCG backend one sends jobs directly to the Grid using the GLITE middleware. According to your needs, some configuration is required:

- In your **.gangarc**:

search for [CMSSW]

```
# this is the path to your output files on your favorite Storage Element: for example
dataOutput = /dpm/uniandes.edu.co/home/cms/user/a/aosorio/gridfiles/ganga
```

search for [LCG]:

```
#here you put the name of the Server that runs as Storage Element
DefaultSE = moboro.uniandes.edu.co
```

- In your job script make sure you select the LCG backend:

```
#... construct the job with backend Local

myjob = Job( application = app, backend = 'LCG' )
myjob.backend.middleware = 'GLITE'

#... specify here the Computing Element where you want your job to run
myjob.backend.CE = 'kuragua.uniandes.edu.co:2119/jobmanager-lcgpbs-cms'
```

CRAB

A very simple CRAB wrapper has been implemented. It basically helps creating in a consistent way a job configuration file and submits to CRAB.

Tools

DBS Search

In CMS, the Data Bookkeeping Service or DBS is design to help finding datasets. There are currently two ways of accessing the information: through a Web form or through the command line using a small CLI application. At the core this database, there is the Query Language (QL). In Ganga, we have implemented a small wrapper around the CLI application called **dbsCommandLine.py**.

Preliminary

On lxplus all defaults are already set. However, you will need to set the environment for CMSSW (cmsenv) first.

Example 1

Suppose you want to find a given set of files by specifying Run number and dataset:

```
#... first get an instance of DBSSearch()
In [2]:dbs = DBSSearch()
Defaults:
http://cmsdbsprod.cern.ch/cms_dbs_prod_global/servlet/DBSServlet
```

This is the URL of the DBS servlet by default. For searching data the **cms_dbs_prod_global** instance of the DBS is the correct one. It is possible to change it at anytime, we will do it in the second example. Now you can perform the search for Run id = 111125 and dataset = /Cosmics/CRAFT09-PromptReco-v1/RECO using the **searchFiles** method:

```
In [2]:result = dbs.searchFiles(111125,'/Cosmics/CRAFT09-PromptReco-v1/RECO')

In [3]:result
Out[3]: True
```

You are done. Now dbs contains the list of files. You can get that list in a CMSDataset() object, ready to be inserted in your inpudata of your job:

```
In [4]:inputdata = CMSDataset()

In [5]:inputdata = dbs.getCMSDataset()

In [6]:inputdata
Out[6]: <GangaCMS.Lib.CMSDataset.CMSDataset.CMSDataset object at 0xf75be6cc>

In [7]:inputdata.names
Out[7]: ['/store/data/CRAFT09/Cosmics/RECO/v1/000/111/125/FEEFDE56-288D-DE11-9A6E-0030486780B8.ro
```

You can also search at which Sites the data is located:

```
In [8]:result = dbs.searchSites(111125,'/Cosmics/CRAFT09-PromptReco-v1/RECO')

In [10]:result
Out[10]: True

In [11]:dbs.getSites()
Out[11]: ['srm-v2-cms.cr.cnaf.infn.it', 'srm-cms.cern.ch', 'cmssrm.fnal.gov', 'cit-se.ultralight.
```

Example 2

MC samples and PAT productions don't have a Run id. For this case the tool uses a different method **searchMCFiles()**: However, we will need to change de DBS instance to 'cms_dbs_ph_analysis_02' (this needs a bit of more work).

```
In [2]:dbs = DBSSearch()
Defaults:
http://cmsdbsprod.cern.ch/cms_dbs_prod_global/servlet/DBSServlet

Out[12]: DBSSearch (
 dbsServer = 'cmsdbsprod.cern.ch' ,
```

```
 dbsInstance = 'cms_dbs_prod_global'
 )

In [13]:dbs.dbsInstance = 'cms_dbs_ph_analysis_02'


In [14]:dbs.updateUrl()
New Url:
http://cmsdbsprod.cern.ch/cms_dbs_ph_analysis_02/servlet/DBSServlet
```

Now we can perform a search on some PAT-SUSY samples:

```
In [15]:result = dbs.searchMCFiles('/SUSY_LM0-sftsht/susy-SUSY_LM0_229_SUSYPAT_V5_v1*')

In [16]:result
Out[16]: True

In [18]: dbs.getCMSDataset().names
Out[18]: ['/store/group/user/susy/fronga/SUSY_LM0-sftsht/SUSY_LM0_229_SUSYPAT_V5_v1/00a24c1fa2b61
```

and then you can also check where the data is located.

```
In [19]:result = dbs.searchSites(1, '/SUSY_LM0-sftsht/susy-SUSY_LM0_229_SUSYPAT_V5_v1*')

In [20]:result
Out[20]: True

In [21]:dbs.getSites()
Out[21]: ['srm.ihepa.ufl.edu', 'pccms2.cmsfarm1.ba.infn.it', 'hephyse.oeaw.ac.at', 'grid-srm.phys
```

Please note the "1" in **searchSites**.

To do

There are lots of exciting things to be developed:

- Write some utility that matches the site output from DBS to a CE.
- Integrate with other APIs (DBS-API for instance)
- Talk to CRAB Server directly
- Cleaning up the code, anything else?

-- AndresOsorio - Mar 2009

This topic: Main > CMSUniandesGroupGangaCMS
Topic revision: r28 - 2009-11-14 - AndresOsorio