Unified treatment of output in Ganga

Current status of the prototype

Ivan Dzhunov

(CERN)

Motivation

- Unifying the concepts of outputdata and outputsandbox into one Job field called outputfiles
- Through user experience it turns out that more flexibility is required in the job output area – users want to put their ntuples (even stdout) into mass storage or they want a small DST file returned in the sandbox
- single field to configure where job output should go or what should be done with the output (compression)
- Proposed output methods are :
  - SandboxFile - return to the output workspace
  - CompressedFile - return gzip compressed to output workspace
  - MassStorageFile - write file into mass storage
  - LHCbDataFile - store file in Storage Element and register in LHCb file catalogue (LHCb only)
  - DQ2File - store file in SE and register in DQ2 (ATLAS only)
  - LCGStorageElementFile - upload to storage element.

Configuration / automatic file type detection

If we have this configuration in .gangarc

```
In .gangarc
#=============================================================================
#  configuration section for postprocessing the output
[Output]

#  list of output files that will be compressed after job is completed
CompressedFile = ['stdout', 'stderr']

# list of output files that will be written to mass storage after job is completed
MassStorageFile = ['*.dummy']

# list of output files that will be written to LCG SE after job is completed
LCGStorageElementFile = ['*.root']
```

then in Ganga we get automatic file type detection

```
In [1]:j = Job()

In [2]:j.outputfiles = ['fillrandom.root', 'fillrandom1.root', 'stdout', '*.dummy']

In [3]:j.outputfiles
Out[3]: [LCGStorageElementFile (name = 'fillrandom.root'), LCGStorageElementFile (name = 'fillrandom1.root'),
CompressedFile (name = 'stdout'), MassStorageFile (name = '*.dummy')]
```

both .root files get associated with LCGStorageElementFile, stdout with CompressedFile, *.dummy with MassStorageFile

```
In [4]:j.outputfiles = ['asd']

In [5]:j.outputfiles
Out[5]: [OutputFile (
 name = 'asd'
 )]
```

if can't find it in the config section, stays as OutputFile - the base class and will not be uploaded anywhere

Configuration / automatic file type detection 2

Automatic file type detection is done only for output files defined with strings

```
In .gangarc

#  configuration section for postprocessing the output
[Output]
CompressedFile = ['stdout', 'stderr']

In Ganga

In [6]:j.outputfiles = [MassStorageFile('stdout')]

In [7]:j.outputfiles
Out[7]: [MassStorageFile (name = 'stdout')]
```

Same file names defined for different output methods

```
#  configuration section for postprocessing the output

[Output]
MassStorageFile = ['stdout']
CompressedFile = ['stderr', 'stdout']

In [4]:j = Job()

In [5]:j.outputfiles = ['stdout']

ConfigError: filename stdout defined more than once in [Output] config section
In [6]:j.outputfiles
Out[6]: []
```

Remove duplicate entries -> same type and same name

```
In [2]:j = Job()
In [3]:j.outputfiles = ['stdout']
In [4]:j.outputfiles += ['stdout', '*.root']

In [5]:j.outputfiles
Out[5]: [CompressedFile (name = 'stdout'), LCGStorageElementFile (name = '*.root')]
```

How it works

```
#  configuration section for postprocessing the output
[Output]
CompressedFile = ['stdout']
MassStorageFile = ['fillrandom.root']
LCGStorageElementFile = ['fillrandom1.root']

In [1]:j = Job(application=Root(),backend=Local())

In [2]:j.application.script = File('/afs/cern.ch/user/i/idzhunov/public/fillrandom2.py')

In [3]:j.outputfiles = ['stdout', 'fillrandom.root', 'fillrandom1.root']

In [4]:j.outputfiles
Out[4]: [CompressedFile (name = 'stdout' ), MassStorageFile ( name = 'fillrandom.root' ),
 LCGStorageElementFile ( se_type = '' , name = 'fillrandom1.root' , srm_token = '' , lfc_host = 'lfc-dteam.cern.ch' ,
  se_rpath = '' , se = 'srm-public.cern.ch'  )]

In [5]:j.submit()

In [6]:
Ganga.GPIDev.Lib.Job               : INFO     job 163 status changed to "completed"

In [6]:jobs[-1].outputfiles[1].location()
Out[6]: ['/castor/cern.ch/user/i/idzhunov/ganga/fillrandom.root']

In [7]:jobs[-1].outputfiles[2].location()
Out[7]: ['guid:61d1b532-4ef0-4e52-b682-3f79038ed94a']

In [8]:jobs[-1].outputfiles[1].get('/afs/cern.ch/user/i/idzhunov/public')
19 bytes in 0 seconds through eth0 (in) and local (out)
19 bytes in remote file

In [9]:jobs[-1].outputfiles[2].get('/afs/cern.ch/user/i/idzhunov/public')
job output downloaded here /afs/cern.ch/user/i/idzhunov/public/79038ed94a

in job.outputdir stdout is compressed
~/gangadir/workspace/idzhunov/LocalXML/163/output $ ll
total 2
-rw-r--r-- 1 idzhunov cg 86 Feb  1 14:05 __jobstatus__
-rw-r--r-- 1 idzhunov cg  0 Feb  1 14:05 stderr
-rw-r--r-- 1 idzhunov cg 27 Feb  1 14:05 stdout.gz
-rw-r--r-- 1 idzhunov cg  0 Feb  1 14:05 __syslog__
```

What happens behind the scenes

- on submit and resubmit, depending on what j.outputfiles contains,a __postprocessoutputfile__ file is created and placed in the input sandbox

```
for the example

In [4]:j.outputfiles
Out[4]: [CompressedFile (name = 'stdout' ), MassStorageFile ( name = 'fillrandom.root' ),
 LCGStorageElementFile ( se_type = '' , name = 'fillrandom1.root' , srm_token = '' , lfc_host = 'lfc-dteam.cern.ch' ,
  se_rpath = '' , se = 'srm-public.cern.ch'  )]

contents of __postprocessoutputfile__ are

zipped stdout
massstorage fillrandom.root nsmkdir rfcp nsls /castor/cern.ch/user/i/idzhunov/ganga
lcgse fillrandom1.root lfc-dteam.cern.ch lcg-cr --vo dteam  -d srm-public.cern.ch
```

- Preferably post-processing should happen on the WN
  - in the job script we parse this file and take the required actions – compressing or sending the output to mass storage or LCG SE
  - another text file __postprocesslocations__ is created with the locations of the uploaded output files, this file is shipped back to the client within the outputsandbox so that we are able to set the locations of the MassStorageFile and LCGStorageElementFile objects
  - setting the locations of MassStorageFile and LCGStorageElementFile happens in a new backend.postprocess method which is called at the end of the postprocess_hook in the Job class

More about IBackend's postprocess method

```
class IBackend(GangaObject):

...

    def postprocess(self, outputfiles, outputdir):
        """ This is a hook called after the job is completed
        to postprocess the output files
        """
        pass

...


class Job(GangaObject):

...
    def postprocess_hook(self):
        self.application.postprocess()
        self.getMonitoringService().complete()
        self.backend.postprocess(self.outputfiles, self.getOutputWorkspace().getPath())

...


class LCG(IBackend):


...
    def postprocess(self, outputfiles, outputdir):
        '''
        here parse __postprocesslocations__ file,
        set locations of the OutputFile objects and delete
        __postprocesslocations__ file
        '''
        ...

...
```

Post-processing on the client

- Sometimes post processing cannot happen on the WN -> in these cases we do it on the Ganga client, in the Backend.postprocess method
  - Batch backend
    - compressing stdout and stderr, because at the time when jobscript is executed these files are still not ready
    - upload to LCG SE, because on the batch system we might not have a valid proxy (pass the file in the outputsandbox, upload it to LCG SE and delete it from outputsandbox)
  - LCG and CREAM backends
    - upload to mass storage, again first pass the files to the outputsandbox, make the upload and delete them from the sandbox

Post-processing on WN / client

	Compressing	LCG SE	Mass storage
Local	WN	WN	WN
Batch	Client	Client	WN
LCG	-	WN	Client
CREAM	-	WN	Client

Configuration for uploads to LCG SE and Mass storage

```
#=========================================================================
#  configuration section for storing of the output to mass storage

[MassStorageOutput]
# Command used to create a directory in the mass storage location
mkdir_cmd = nsmkdir

# Command used to copy data to the mass storage location
cp_cmd = rfcp

#Command used to list files in the mass storage location
ls_cmd = nsls

#Path to the mass storage location where the files will be stored
path = /castor/cern.ch/user/i/idzhunov/ganga



#=========================================================================
#  configuration section for storing of the output to LCG storage element

[LCGStorageElementOutput]
# LFC host for Logical File Name association with the uploaded output file
LFC_HOST = lfc-dteam.cern.ch

# SRM where the output file should be uploaded
dest_SRM = srm-public.cern.ch
```

Output files API

```
class OutputFile(GangaObject):
    """OutputFile represents base class for output files, such as CompressedFile,
       MassStorageFile, LCGStorageElementFile, etc
    """
    _schema = Schema(Version(1,1), {'name': SimpleItem(defvalue="",doc='name of the file')})

    ...

    def location(self):
        """
        Return list with the locations of the post processed files (if they was configured to upload the output somewhere)
        """
        raise NotImplementedError

    def get(self, dir):
        """
        Retrieves locally all files matching this OutputFile object pattern
        """
        raise NotImplementedError


class CompressedFile(OutputFile):
    """CompressedFile represents a class marking a file for compressing
    """
    _schema = Schema(Version(1,1), {'name': SimpleItem(defvalue="",doc='name of the file')})

    ...


class MassStorageFile(OutputFile):
    """MassStorageFile represents a class marking a file to be written into mass storage (like Castor at CERN)
    """
    _schema = Schema(Version(1,1), {'name': SimpleItem(defvalue="",doc='name of the file')})

    #overridden location and get methods

    def location(self):
        ...
    def get(self, dir):
        ...
```

Output files API 2

```
class LCGStorageElementFile(OutputFile):
    """LCGStorageElementFile represents a class marking an output file to be written into LCG SE
    """
    _schema = Schema(Version(1,1), {
        'name'        : SimpleItem(defvalue="",doc='name of the file'),
        'se'          : SimpleItem(defvalue='', copyable=1, doc='the LCG SE hostname'),
        'se_type'     : SimpleItem(defvalue='', copyable=1, doc='the LCG SE type'),
        'se_rpath'    : SimpleItem(defvalue='', copyable=1, doc='the relative path to the VO directory on the SE'),
        'lfc_host'    : SimpleItem(defvalue='', copyable=1, doc='the LCG LFC hostname'),
        'srm_token'   : SimpleItem(defvalue='', copyable=1, doc='the SRM space token, meaningful only when se_type is set to
srmv2')
})

    ...
    def __init__(self,name='', **kwds):
        """ name is the name of the output file that has to be written into LCG SE
        """
        super(LCGStorageElementFile, self).__init__(name, **kwds)

        lcgSEConfig = getConfig('LCGStorageElementOutput')

        self.lfc_host = lcgSEConfig['LFC_HOST']
        self.se = lcgSEConfig['dest_SRM']
    ...

    def getUploadCmd(self):
        #constructs the lcg-cr command, i.e. lcg-cr --vo dteam -d srm-public.cern.ch ...
        ...

    def setLocation(self, location):
        #sets the location of the uplaoded to LSG SE file, i.e. guid:234523sdrfwer254er
        ...

    #overridden location and get methods

    def location(self):
        ...
    def get(self, dir):
        ...
```