Writing your own DMLite plugin

Alejandro Álvarez Ayllón
on behalf of the LCGM development team
Overview

- Introduction
- Setting the development environment
- Writing first dummy functionality
- Loading our plugin into the stack
- Adding a new Catalog
- Listing directories on a Hadoop namespace
Introduction

• DMLite aims to make plugin development easy
  – And as safe as possible
• We are going to write a simple plugin that provides a couple of methods only to show the bases
  – No need to implement every single method!
Setting the development environment

- Since it is a small plugin, we will just use a simple Makefile
- We will keep two different files: one for the plugin registration, and another one for the plugin itself
  - Makefile
  - MyPlugin.cpp
  - MyCatalog.h
  - MyCatalog.cpp
Setting the development environment

- We need to install dmlite-devel to get the headers
  - `yum install dmlite-devel`
- Our plugin will need to link against dmlite
  - `LDFLAGS=-ldmlite`
- If we are development on SL5, we need to point to the Boost 1.41 headers (coming from EPEL)
  - `/usr/include/boost141`
We are going to start with the bare minimum: registering our plugin and printing every configuration entry that is read from the configuration files.
Loading our plugin into the stack

• We need to create a new file: 
  /etc/dmlite.conf.d/00-myplugin.conf
  – Starts with 00 to be the first one
• And add the LoadPlugin directive
  – LoadPlugin plugin_my <so location>
• If now we restart HTTPD, we will see every 
  configuration directive dumped to the stdout
  – Which is our plugin does!
Adding a new Catalog

• We need to create a new class that inherits from Catalog
• And our factory must inherit from CatalogFactory
• We implement the missing methods in the Factory
  – MyFactory::createCatalog
• And, at least, getImplId on out Catalog
• And register the factory as a CatalogFactoryFactory
  – PluginManager::registerCatalogFactory
Adding a new Catalog

• Now, we rename /etc/dmlite.conf.d/00-myplugin.conf to /etc/dmlite.conf.d/zmyplugin.conf
  – We want it to be the last one loaded
  – Remember, plugins are called as LIFO!

• We restart httpd
  – And we refresh the browser
  – It obviously fails: we haven’t implemented anything yet
We will need at least these methods to list a directory:
- Catalog::changeDir
- Catalog::extendedStat
- Catalog::openDir
- Catalog::readDirx
- Catalog::closeDir
And our new plugin...

- Is accessible through HTTP/DAV
- And GridFTP
- And xroot
- And NFS
- And SRM
But I don't like C++ :'(  

- You will be able to do it in Python!

```python
import pydmlite as pd
from dmlite.exceptions import *

class pyINode(pd.INode):

    ...  
    def getComment(self, inode):
        comment = None

        if inode in self.inode_dict:
            comment = self.inode_dict[inode].comment

        if comment == None:
            raise PyDmException(pd.DMLITE_NO_COMMENT, "There is no comment for inode %ld" % inode)

        return comment

    def setComment(self, inode, newcomment):
        if inode in self.inode_dict:
            self.inode_dict[inode].comment = newcomment

    def deleteComment(self, inode):
        try:
            self.inode_dict[inode].comment = None
        except KeyError:
            raise
```