Microservices in Python with PyRods and EmbedPython

Presented at iRODS User Group Meeting 2012

By Hao Xu xuh@cs.unc.edu
PyRods and EmbedPython

- Developed & Maintained by Jerome Fuselier
- Both projects are hosted at http://code.google.com/p/irodspython/
- Functionality:
  - PyRods: Python binding for iRODS client api and data structures
  - EmbedPython: an iRODS module that allows calling microservices written in Python from iRODS Rule Engine
Write Microservices in C

IRODS Communication Protocol

ICOMMANDS → IRODS Client API → IRODS Server API → Core, iCAT, ...

C Microservice

Rule Engine
Write Microservices using PyRODS and EmbedPython

IRODS Communication Protocol

icommands ➔ IRODS Client API ➔ IRODS Server API ➔ Core, iCAT, ...

PyRODS

Python Microservices ➔ EmbedPython ➔ Rule Engine
General Idea 1

- Compare with C code:
  ```c
  int <msi name>(msParam_t *param1, ..., msParam_t *paramX,
                  ruleExecInfo_t *rei) {
    /* parse input params */
    /* do stuff */
    /* fill in out params */
  }
  ```

- Python code:
  ```python
  def <function name>(param1, ..., paramX, rei):
    # parse input params
    # do stuff
    # fill in out params
  ```
General Idea 2

- Rule Engine code:
  
  msiPyInitialize;

  msiLocalPythonX(<script path>, <function name>,
                  <recursion test flag>, arg1, ..., argX);

  msiPyFinalize;

- Alternatively, you can call
  - msiRodsPythonX
  - MsiPythonExec

- Wrap the code in an adapter rule
  
  <msi name>(param1, ..., paramX) { <RE code> }
Example: Goal

`/tmp/test.r`

test {
    ...
    msiReadFromDataObj(*path, *contents);
    ...
}
...
...
from irods import *
def pyReadFromDataObj(path, contents, rei):
    path_str = path.parseForStr()
    f = iRodsOpen(rei.getRsComm(), path_str)
    contents_str = f.read()
    f.close()
    fillStrInMsParam(contents, contents_str)
Example: Adapter Rule

```c
msiReadFromDataObj(*path, *contents) {
    msiPyInitialize;
    msiLocalPython2("/tmp/test.py", "pyReadFromDataObj",
                    "noRecursionTest", *path, *contents);
    msiPyFinalize;
}
```
Example: Rules

```
/test/test.r

/test/test.r

test {
    msiReadFromDataObj(*path, *contents);
    writeString("stdout", *contents);
}
msiReadFromDataObj(*path, *contents) {
    msiPyInitialize;
    msiLocalPython2("/tmp/test.py", "msiReadFromDataObj",
                    "noRecursionTest", *path, *contents);
    msiPyFinalize;
}

INPUT *path="/tempZone/home/rods/test.txt"
OUTPUT ruleExecOut
```
Another Example

test {

    msiReadFromDataObj(*path1, *contents1);
    ...
    msiReadFromDataObj(*pathN, *contentsN);

}
msiReadFromDataObj(*path, *contents) {
    msiPyInitialize;
    msiLocalPython2("/tmp/test.py", "msiReadFromDataObj",
        "noRecursionTest", *path, *contents);
    msiPyFinalize;
}
Another Example

test {
    msiPyInitialize;
    msiReadFromDataObj(*path1, *contents1);
    ...
    msiReadFromDataObj(*pathN, *contentsN);
    msiPyFinalize;
}
msiReadFromDataObj(*path, *contents) {

    msiLocalPython2("/tmp/test.py", "msiReadFromDataObj",
                    "noRecursionTest", *path, *contents);

}
A Few Tips

- If there is any error, look at the rodsLog* files for more information.
- Available PyRods objects/functions can be found in the PyRods document
Building PyRods and EmbedPython

Diagram:
- PyRods
- EmbedPython
- IRODS Server
- Dependency
- Module
- iRODS object files

Diagram Description:
- PyRods depends on EmbedPython.
- EmbedPython has a module relationship with IRODS Server.
- PyRods receives iRODS object files from IRODS Server.
Build Order

• Build PyRods
  • Build iRODS server
  • Build PyRods, link with iRODS object files

• Build EmbedPython
  • Build iRODS server with EmbedPython module, link with PyRods object files
  • Build PyRods again
A Few Tips On Building PyRods and EmbedPyRods

- Install python and python-dev
- Search for PyRods and iRODS 3.0 on iRODS Chat
- Use -fPIC to build iRODS server and modules
- Set PYTHONPATH
- Set LD_LIBRARY_PATH to include libodbc.so.1
- To run the test rules in EmbedPython, modify the rules or copy microservices.py so that the paths match
- If you build PyRods with code from iRODS svn (post 3.0), you may need to make some changes to PyRods code
Write Microservices using Python: A Possibly More Efficient Model