RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Hands-On with SAGA Python (BlisS)

Presented by: Melissa Romanus
The RADICAL Group
http://radical.rutgers.edu
http://saga-project.org

Agenda

- Introduction
- Environment Set-Up and Installation
  - Logging into Lonestar
  - Setting up your environment for Python
  - Installing SAGA BlisS
- Job Submission with SAGA
  - Overview
  - Hands-On Example
- File Handling with SAGA
  - Hands-On Example
- SAGA Mandelbrot Example

Introduction

- What is SAGA BlisS?
  - Lightweight Python package that implements parts of OSG GFD.90 SAGA interface specification
    - File handling
    - Job submission
  - Provides plug-ins for different distributed middleware systems and services
- SAGA supports the following backends:
  - SSH - Allows job execution on remote hosts via SSH.
  - PBS(+SSH) - (includes TORQUE). Provides local and remote access to PBS/Torque clusters.
  - SGE(+SSH) - Provides local and remote access to Sun (Oracle) Grid Engine clusters.
  - SFTP - Provides remote filesystem access via the SFTP protocol.

Please Point Your Browser…

- Follow along with this tutorial by visiting….

XSEDE Tutorial Part 2: SAGA

https://github.com/saga-project/BigJob/wiki/XSEDE-Tutorial-Part-2:-SAGA

- Quick note: This tutorial utilizes the XSEDE resource Lonestar. The environment set-up details are specific to this resource.

Environment Set-Up

- Open a terminal (Linux/Mac: Terminal, Windows: PuTTy)

```
ssh <your username>@lonestar.tacc.utexas.edu
```

- Bootstrap your Local Python Environment

```
module load python
```

- Create your virtual environment

```
curl --insecure -s
https://raw.github.com/pypa/virtualenv/master/virtualenv.py
| python - $HOME/tutorial
```

Environment Set-Up cont'd

- Activate your newly installed Python environment to update your PYTHONPATH

```
source $HOME/tutorial/bin/activate
```

- Ensure batch jobs have the same Python environment by adding the following lines to $HOME/.profile
  - Open $HOME/.profile in your favorite text editor (e.g. vim, emacs, etc)

```
module load python
source $HOME/tutorial/bin/activate
```

Install BlisS

- You are now ready to install SAGA BlisS from PyPi!

```
pip install bliss
```

- You will notice some unpacking and installation of packages.

```
Successfully installed bliss paramiko-on-pypi pycrypto-on-pypi
Cleaning up...
```

- Check for successful installation by executing the following command:

```
python –c "import bliss; print bliss.version"
```

Job Submission with SAGA BlisS

Bliss provides the capability to submit jobs to local and remote queueing systems and resource managers

- The job submission and management capabilities of Bliss are packaged in the **bliss.saga.job** module.
  - **job.Service**: Provides a handle to the resource manager, e.g. a remote PBS cluster.
  - **job.Description**: Used to describe the executable, arguments, environment and requirements (e.g., number of cores, etc) of a new job.
  - **job.Job**: A handle to a job associated with a job.Service. It is used to control (start, stop) the job and query its status (e.g., Running, Finished, etc).

Creating Job Submission Scripts with BlisS

- First, we must import the BlisS python package.

```
import bliss.saga as saga
```

- Create a job service object that represents a local or cluster resource.
  - Takes URL as a parameter. URL tells Bliss what type of middleware or queuing system

```
js = saga.job.Service("sge://localhost")
```

Creating Job Submission Scripts with BlisS, cont'd

- Using job.Service, new jobs can be created and executed.
- To define a new job, you need a job.Description object which contains information about:
  - The executable
  - The arguments required by the executable
  - Environment that the job needs
  - What requirements we have for our job

```
jd = saga.job.Description()
# requirements
jd.queue = "development"
jd.wall_time_limit = 1 # minutes

# environment, executable & arguments
jd.environment = {'MYOUTPUT':'"Hello
from Bliss"'}
jd.executable = '/bin/echo'
jd.arguments = ['$MYOUTPUT ']

# output options
jd.output = "my1stjob.stdout"
jd.error = "my1stjob.stderr"
```

Hands-On: Putting it All Together

- You are now ready to run your first example script. Please refer to the wiki for this section.
- Verify that you are in your home directory

```
cd $HOME
```

- Open a new file name saga_example_1.py in your favorite text editor (vim, emacs, etc)

```
vim saga_example_1.py
```

- Copy and paste the contents from the website into your file and save it.

Hands-On: Putting it All Together, cont'd

- Execute the script

```
python saga_example_1.py
```

- The output from Bliss to command line will look something like this:

```
Job ID : [sge://localhost]-[None]
Job State : saga.job.Job.New

...starting job...
Job ID : [sge://localhost]-[644240]
Job State : saga.job.Job.Pending

...waiting for job...
Job State : saga.job.Job.Done
Exitcode : None
```

So… what did I just do?

- Now that the job has completed, open up my1stjob.stdout

```
vim my1stjob.stdout
```

- Notice that the contents of this file are “Hello from Bliss” as specified in the job description

```
TACC: Setting memory limits for job 658157 to unlimited KB
TACC: Dumping job script:
------------------------------------------------------
[[ Bash script ommitted ]]
------------------------------------------------------
TACC: Done.
Hello from Bliss
TACC: Cleaning up after job: 658157
TACC: Done.
```

File Handling with SAGA BlisS

- SAGA has remote file and directory handling capabilities. These capabilities are packaged in:

```
bliss.saga.filesystem
```

- This package contains two main classes:

```
filesystem.File #provides a handle to a remote file
```

```
filesystem.Directory #provides a handle to a remote directory
```

- Can traverse and modify local and remote file systems
- SFTP support, more plug-ins in development

Hands-On: Listing a Remote Directory

- SAGA has remote file and directory handling capabilities. These capabilities are packaged in:

```
bliss.saga.filesystem
```

- This package contains two main classes:

```
filesystem.File #provides a handle to a remote file
```

```
filesystem.Directory #provides a handle to a remote directory
```

- Can traverse and modify local and remote file systems
- SFTP support, more plug-ins in development

Hands-On: Listing a Remote Directory

- Note: SAGA **does not** support SSH auth via username/password. To use SFTP on remote, keys must be set up. For this tutorial, we will use localhost for simplicity.

```
cd $HOME
```

- Open a new file name saga_example_1.py in your favorite text editor (vim, emacs, etc)

```
vim saga_example_2.py
```

- Copy and paste the contents from the website into your file and save it.

Hands-On: Listing a Remote Directory, cont'd

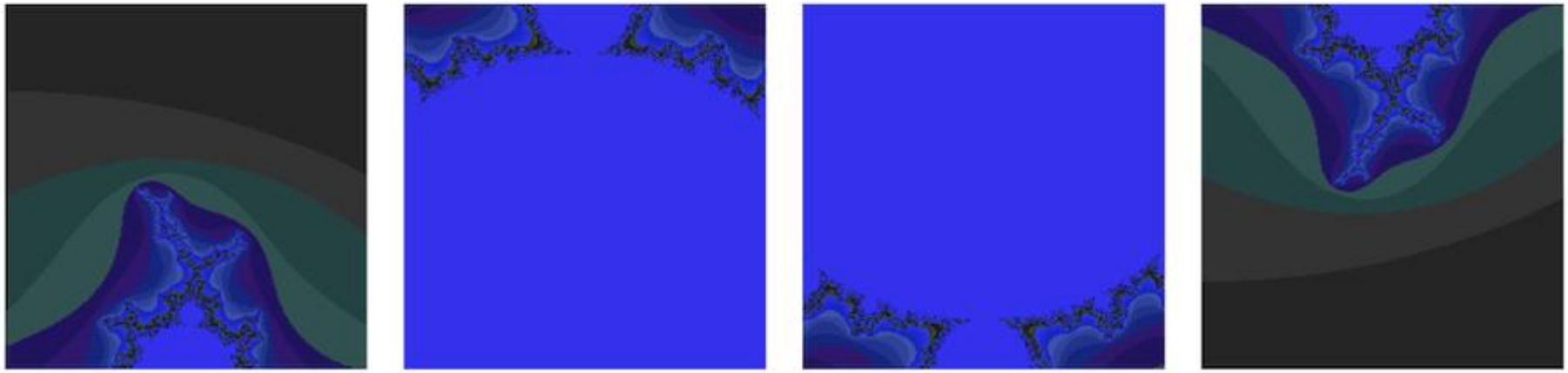
- Execute the script

```
python saga_example_2.py
```

- The output from Bliss to command line will look something like this:

```
sftp://localhost/tmp/train115//saga_example_2.py (1029 bytes)
 sftp://localhost/tmp/train115//motd (1758 bytes)
```

Mandelbrot Fractals



- I conceived and developed a new geometry of nature and implemented its use in a number of diverse fields. It describes many of the irregular and fragmented patterns around us, and leads to full-fledged theories, by identifying a family of shapes I call *fractals*. -- *Mandelbrot*

Hands-On: Distributed Mandelbrot

- Install the Python Image Library.

```
pip install PIL
```

- Download the Mandelbrot fractal generator

```
curl --insecure -Os https://raw.github.com/saga-project/bliss/master/examples/advanced/mandelbrot/mandelbrot.py
```

- Test the mandelbrot script via command line (1024 x 1024 fractal).

```
python mandelbrot.py 1024 1024 0 1024 0 1024 frac.png
```

Hands-On: Distributed Mandelbrot, cont'd

- Ensure that you are in your home directory

```
cd $HOME
```

- Open a new file name saga_mandelbrot.py in your favorite text editor (vim, emacs, etc)

```
vim saga_mandelbrot.py
```

- Copy and paste the contents from the website into your file and save it.

Hands-On: Distributed Mandelbrot, cont'd

- Execute the script

```
python saga_mandelbrot.py
```

- The output from Bliss to command line:

```
* Submitted [sge://localhost]-[652593]. Output will be written to: tile_x0_y0.png
* Submitted [sge://localhost]-[652594]. Output will be written to: tile_x0_y1.png
* Submitted [sge://localhost]-[652595]. Output will be written to: tile_x1_y0.png
* Submitted [sge://localhost]-[652596]. Output will be written to: tile_x1_y1.png
* Job [sge://localhost]-[652593] status: saga.job.Job.Pending
* Job [sge://localhost]-[652594] status: saga.job.Job.Pending
* Job [sge://localhost]-[652595] status: saga.job.Job.Pending
* Job [sge://localhost]-[652596] status: saga.job.Job.Pending
* Job [sge://localhost]-[652593] status: saga.job.Job.Running
* Job [sge://localhost]-[652594] status: saga.job.Job.Running
* Job [sge://localhost]-[652595] status: saga.job.Job.Running
* Job [sge://localhost]-[652596] status: saga.job.Job.Running
...
* Job [sge://localhost]-[652593] status: saga.job.Job.Done
* Job [sge://localhost]-[652594] status: saga.job.Job.Done
* Job [sge://localhost]-[652595] status: saga.job.Job.Done
* Job [sge://localhost]-[652596] status: saga.job.Job.Done
* Copying sftp://localhost//scratch/0000/train115/mbrot//tile_x0_y0.png back to /home1/0000/train115
* Copying sftp://localhost//scratch/0000/train115/mbrot//tile_x0_y1.png back to /home1/0000/train115
* Copying sftp://localhost//scratch/0000/train115/mbrot//tile_x1_y1.png back to /home1/0000/train115
* Copying sftp://localhost//scratch/0000/train115/mbrot//tile_x1_y0.png back to /home1/0000/train115
* Stitching together the whole fractal: mandelbrot_full.png
```

Hands-On: Distributed Mandelbrot, cont'd

- To view the png file you just created, you can scp the file to your local machine (Linux/Mac). Refer to web for Windows.
- On your LOCAL machine, execute the following command:

```
scp <your_lonestar_username>@lonestar.tacc.utexas.edu:mandelbrot_full.png .
```

- This will place the file mandelbrot_full.png in whatever directory you execute this command from.
- Image can be viewed using any image viewer installed on your machine.

Conclusion

- SAGA BlisS is a Python implementation of OGF GFD.90 SAGA
- SAGA BlisS can be used for:
  - Job submission
  - Remote file handling
  - Complex distributed workflows
- SAGA provides support for the following backends:
  - SSH
  - PBS(+SSH)
  - SGE(+SSH)
  - SFTP

BlisS/SAGA Support

- General Public Support and Related Issues mailing list:
  - saga-bliss@googlegroups.com
- Development Related mailing list
  - bliss-dev@googlegroups.com

Questions?