Experiment Control Using Gush

Jeannie Albrecht
Williams College
http://gush.cs.williams.edu

Gush
Tutorial Agenda

• Introduction, Slice/sliver initialization
• Gush configuration, Gush startup
• Gush overview (while PLab slivers are created)
• Basic Gush functionality
• Simple experiments using PLab and PG
• Experiments highlighting failure recovery
• Experiments involving barriers
• Very brief introduction to Nebula
• Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
VM login and Gush startup

• Start VM in VirtualBox using username and password provided
• Open terminal
• (Don’t forget `xterm -e ssh <username>@ops &`)
• Run:

```
$ sudo apt-get install keychain
$ cp /home/geni/.ssh/geni_key /home/geni/.ssh/id_rsa
$ cp /home/geni/.ssh/geni_key.pub /home/geni/.ssh/id_rsa.pub
$ keychain ~/.ssh/id_rsa
$ echo “Host *” >> ~/.ssh/config
$ echo “ StrictHostKeyChecking no” >> ~/.ssh/config
$ mkdir gush
$ cd gush
$ wget http://gush.cs.williams.edu/gush-bin-32bit.tgz
$ tar xzvf gush-bin-32bit.tgz
$ cp /home/geni/.gcf/omni_config .
```
<gush>
  <!--
  <resource_manager type="geni-pg" am="http://localhost:8348">
    <port_map slice="pgenigpolabbbncom_SLICENAME" port="61500"/>
  </resource_manager>
  -->
  <resource_manager type="geni-pg">
    <port_map slice="PG_SLICE" port="61600"/>
  </resource_manager>
</gush>
<?xml version="1.0" encoding="UTF-8"?>
<rspec type="request" ...
  <node
    component_id="urn:publicid:IDN+plc:williams+node+planetlab1.williams.edu"
    component_manager_id="urn:publicid:IDN+plc+authority+cm"
    component_name="planetlab1.williams.edu"
    exclusive="false" client_id="williamspl1">
      <hardware_type name="plab-pc"/>
      <hardware_type name="pc"/>
      <location country="unknown" latitude="42.71" longitude="-73.2"/>
      <sliver_type name="plab-vnode"/>
    </node>
  ...
</node>
</rspec>
PlanetLab Sliver Creation

- Copy RSpec to plab.rspec
- From gush directory:
  
  ```
  $ omni.py createslice SLICENAME
  $ omni.py -a plc createsliver --api-version 2
     SLICENAME plab.rspec
  ```

- Should get back info about 5 nodes
- (Note: We are using our ProtoGENI acct credentials to create PlanetLab slices and slivers.)
<?xml version="1.0" encoding="UTF-8"?>
<rspec type="request" ... >
    <node client_id="geni1" ... >
        <sliver_type name="raw-pc">
            <disk_image name="urn:publicid:IDN+emulab.net
                +image+emulab-ops//'FEDORA10-STD"/>
        </sliver_type>
        <services>
            <execute shell="sh" command="sudo hostname
                `cat /var/emulab/boot/realname` `cat
                /var/emulab/boot/mydomain`"/>
        </services>
    </node>
    <node client_id="geni2" ... > ... </node>
</rspec>
ProtoGENI Sliver Creation

- Copy RSpec to pg.rspec
- From gush directory:

  ```
  $ omni.py -a pg createslice PG_SLICE
  $ omni.py -n
  -a https://www.emulab.net/protogeni/xmlrpc/am
  createsliver PG_SLICE pg.rspec
  ```

- Should get back info about 2 nodes
Tutorial Agenda

- Introduction, Slice/sliver initialization
- Gush configuration, Gush startup
- **Gush overview (while PLab slivers are created)**
- Basic Gush functionality
- Simple experiments using PLab and PG
- Experiments highlighting failure recovery
- Experiments involving barriers
- Very brief introduction to Nebula
- Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Gush Overview

• How do experimenters actually use GENI?
• Goal: Develop abstractions and tools for addressing the challenges of managing distributed applications on GENI
  • Make it easy for a range of users to run experiments on GENI
• Strategy
  • Make minimal assumptions about resources and how they’re allocated
  • Leverage other services to locate resources and obtain credentials
  • Interface with other user tools to improve functionality
  • Hide complexity and use one common user interface to interact with different CFs (i.e., PlanetLab, ProtoGENI, ORCA)
Gush

- A distributed application management infrastructure
  - Designed to simplify deployment of distributed applications
  - Provides abstractions for configuration and management
  - Allows users to “remotely control” computers running distributed applications
Step 1: Describe Application

- Describe experiment using application “building blocks”
- Create customized control flow for distributed applications
- Application specification blocks are described using XML

Application Block

Component Block 1
Senders
- Process Block 1
  Prepare Files
- Process Block 2
  Join Network
- Barrier Block 1
  Phase 1 Barrier
- Process Block 3
  Send Files

Component Block 2
Receivers
- Process Block 1
  Join Network
- Barrier Block 1
  Phase 1 Barrier
- Process Block 2
  Receive Files
Step 1: App Spec

<project name="simple">
  <software name="SimpleSoftwareName" type="tar">
    <package name="Package" type="web">
      <path>http://sysnet.cs.williams.edu/~jeannie/software.tar</path>
      <dest_path>software.tar</dest_path>
    </package>
  </software>
  <component name="Cluster1">
    <rspec>
      <num_hosts>20</num_hosts>
    </rspec>
    <software name="SimpleSoftwareName" />
    <resources>
      <resource type="geni-pg" group="pgenigpolabbbncom_SLICENAME" />
    </resources>
  </component>
  <experiment name="simple">
    <execution>
      <component_block name="cb1">
        <component name="Cluster1" />
        <process_block name="p2">
          <process name="cat">
            <path>cat</path>
            <cmdline>
              <arg>software.txt</arg>
            </cmdline>
          </process>
        </process_block>
      </component_block>
    </execution>
  </experiment>
</project>
Gush supports experimentation with PlanetLab, ProtoGENI, and ORCA resources.

How can we find “good” resources?
- We may want machines with specific characteristics.
- Reserve machines a priori with specific characteristics on ProtoGENI and ORCA.
- Use SWORD (http://sword.cs.williams.edu) on PlanetLab to find “good” machines.
Gush Resource Directory

<?xml version="1.0" encoding="UTF-8"?>
<gush>
  <resource_manager type="geni-plc">
    <port_map slice="pgenigpolabbbncom_SLICENAM" port="61500"/>
  </resource_manager>
  <resource_manager type="geni-pg">
    <port_map slice="slice31" port="61600"/>
  </resource_manager>
  <resource_manager type="geni-orca">
    <port_map slice="orca-slice31" port="61700"/>
  </resource_manager>
</gush>

Note: ProtoGENI and ORCA slivers must be created BEFORE starting Gush!
Step 3: Configure Resources

- Connect to and configure selected resources
  - **Controller** connects to and “remotely controls” the Gush clients on the experimenter’s behalf
  - Controller SSHs to resource, installs Gush client and other software as needed, starts client, client makes TCP connection to controller
Controller issues commands to clients telling them to start running applications/experiments

- **Senders** begin running sender processes
- **Receivers** begin running receiver processes
Step 5: Monitor Application

- We want to make sure the processes keep running
- Clients monitor experiment processes for failures
  - If a failure is detected, client notifies controller
  - Controller may tell client to restart failed process, or find new resource
Step 6: Cleanup

- Gush clients make sure all programs exited cleanly
- Remove logs and software from remote machines (optional)
- Disconnect clients from controller
Recall our simple example…
Gush in Action

gush> load tests/simple.xml
Project "simple" is selected.
Experiment "simple" is selected.
gush> run
Starting experiment run.
Running experiment simple...
gush> The configuration matcher has finished matching.
The resource allocator has finished successfully.
pgenigpolabbbncom_SLICENAME@planetlab1.cs.umass.edu:61500 has joined the mesh.
The file transfer of Package to planetlab1.cs.umass.edu has been completed.
The software installation of Package on planetlab1.cs.umass.edu was successful.
pgenigpolabbbncom_SLICENAME@planetlab1.cs.uoregon.edu:61500 has joined the mesh.
The file transfer of Package to planetlab1.cs.uoregon.edu has been completed.
The software installation of Package on planetlab1.cs.uoregon.edu was successful.
pgenigpolabbbncom_SLICENAME@planetlab1.cs.umass.edu:61500,24301: Hello World
pgenigpolabbbncom_SLICENAME@planetlab1.cs.uoregon.edu:61500,27717: Hello World
The experiment has ended.
gush> disconnect
pgenigpolabbbncom_SLICENAME@planetlab1.cs.umass.edu:61500 has decided to leave the mesh.
pgenigpolabbbncom_SLICENAME@planetlab1.cs.uoregon.edu:61500 has decided to leave the mesh.
gush> quit
Tutorial Agenda

• Introduction, Slice/sliver initialization
• Gush configuration, Gush startup
• Gush overview (while PLab slivers are created)
• **Basic Gush functionality**
• Simple experiments using PLab and PG
• Experiments highlighting failure recovery
• Experiments involving barriers
• Very brief introduction to Nebula
• Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Now we’re ready to start Gush (with PG)!

But we won’t actually connect to anything just yet…we need to wait for our VMs to boot and our keys to be installed.

In gush directory (pick your favorite port):

```
guest31@geni-vm:~/.gush$ ./gush -P 15000
```

```
gush> Gush has learned about the slice PG_SLICE.
gush> Updated information on the slice PG_SLICE is available.
```
Verify Resource Pool

gush> info nodes
There are 2 known nodes:

[ P ] guest31@pc419.emulab.net:61600(pref=0) (Disconnected.)
[ P ] guest31@pc430.emulab.net:61600(pref=0) (Disconnected.)

gush> quit
PLEASE DON’T DO THIS NOW!

Gush can add resources to PlanetLab slices

```plaintext
gush> slice add SLICENAME ucsd

The hosts assigned to slice pgenigpolabbbncom_SLICENAME have been updated. Updated information on the slice pgenigpolabbbncom_SLICENAME is available.

gush> info nodes

There are 10 known nodes:

- `pgenigpolabbbncom_SLICENAME@planet2.scs.cs.nyu.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@ricepl-1.cs.rice.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planet1.scs.stanford.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planetlab1.cs.uchicago.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planetlab1.ucsd.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planetlab2.ucsd.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planetlab3.ucsd.edu:61500(pref=0)` (Disconnected.)
- `pgenigpolabbbncom_SLICENAME@planetlab1.cs.umass.edu:61500(pref=0)` (Disconnected.)

...
Remove Resources from PLab Slice

- PLEASE DON’T DO THIS NOW!
- Gush can remove resources from PlanetLab slices

```
gush> slice remove pgenigpolabbbncom_SLICENAME ucsd
```
```
gush> The hosts assigned to slice pgenigpolabbbncom_SLICENAME have been updated.
Updated information on the slice pgenigpolabbbncom_SLICENAME is available.
```
```
gush> info nodes
There are 7 known nodes:
[ P ] pgenigpolabbbncom_SLICENAME@planet2.scs.cs.nyu.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@ricepl-1.cs.rice.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planet1.scs.stanford.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planetlab1.cs.uchicago.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planetlab1.cs.umass.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planetlab1.cs.uoregon.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planetlab2.cs.uoregon.edu:61500(pref=0) (Disconnected.)
[ P ] pgenigpolabbbncom_SLICENAME@planetlab3.cs.uoregon.edu:61500(pref=0) (Disconnected.)
...```
Renew PlanetLab Slice

- PLEASE DON’T DO THIS NOW!
- Gush can renew PlanetLab slices

```
gush> slice renew pgenigpolabbbncom_SLICENAME
gush> The slice pgenigpolabbbncom_SLICENAME has been renewed.
```

- Gush can re-query PlanetLab server for updates

```
gush> slice update pgenigpolabbbncom_SLICENAME
```
```
gush> Updated information on the slice pgenigpolabbbncom_SLICENAME is available.
```
Basic Gush Functionality

• Go back to VM and start Gush

```
./gush -P 15000
```

• Run “info nodes” and verify that resource pools have been created

```
gush> info nodes

There are 17 known nodes:
[   ] pgenigpolabbbncom_SLICENAME@planet2...:61500(pref=0) (Disconnected.)
[   ] pgenigpolabbbncom_SLICENAME@ricepl-1...:61500(pref=0) (Disconnected.)
[   ] pgenigpolabbbncom_SLICENAME@planet1...:61500(pref=0) (Disconnected.)
```
Useful Commands

- First connection takes the longest (have to install client)
- Should always disconnect from nodes when finished
- Should periodically remove *logfile* in your directory

gush> connect <host>: Start/connect to a Gush client @ <host>.
gush> connect pat <regex> <num>: Connect to <num> hosts that match the regular expression pattern <regex>.
gush> disconnect: Close all open connections.
gush> disconnect <host>: Disconnect from <host>.
gush> info nodes: Print summary information on all nodes.
gush> info node <host>: Print information about <host>.
gush> info mesh: Print the mesh status (membership).
gush> shell “<quoted string>”: Run the string as a shell command on all connected nodes.
gush> quit: Quit Gush.
Useful Commands: Running Experiments

- Always load xml project file (app spec) first, then run
- Manipulate host selection with “prefer <regex>”
- Save preferred hosts with “save prefer <prefer.xml>”
- Check preference value with “info nodes”

```
gush> prefer <regex>: Tell Gush to increase preference value for nodes matching regex. Nodes with higher preference values are used first.
gush> load <file.xml>: Read a project/preferences file.
gush> run: Start executing the experiment in file.xml.
gush> info control: Print controller state information.
gush> debug fail node <host>: Mark <host> as failed and find replacement (only during experiment).
gush> save prefer <prefer.xml>: Save current host preferences (can load them back in later).
```
Tutorial Agenda

- Introduction, Slice/sliver initialization
- Gush configuration, Gush startup
- Gush overview (while PLab slivers are created)
- Basic Gush functionality
- Simple experiments using PLab and PG
- Experiments highlighting failure recovery
- Experiments involving barriers
- Very brief introduction to Nebula
- Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Simple Experiments using PL & PG

- **SimpleSummerCampExample**
- Check out simple.xml
- Modify it to use:
  - PlanetLab resources
  - ProtoGENI resources
- Be careful when reusing project/experiment names in app spec (or always restart Gush)
Tutorial Agenda

- Introduction, Slice/sliver initialization
- Gush configuration, Gush startup
- Gush overview (while PLab slivers are created)
- Basic Gush functionality
- Simple experiments using PLab and PG
- Experiments highlighting failure recovery
- Experiments involving barriers
- Very brief introduction to Nebula
- Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Failure Experiments using PL & PG

• Check out sleep.xml on Gush wiki page (FailureSummerCampExample)

• We will try to simulate three failure scenarios:
  • Process failure
  • Host failure/unresponsiveness
  • Gush client failure

• Faking failures is tricky…
  • Restart Gush after each attempt to avoid confusion
Tutorial Agenda

• Introduction, Slice/sliver initialization
• Gush configuration, Gush startup
• Gush overview (while PLab slivers are created)
• Basic Gush functionality
• Simple experiments using PLab and PG
• Experiments highlighting failure recovery
• Experiments involving barriers
• Very brief introduction to Nebula
• Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Barriers and Predecessors

• By default, all process blocks run in parallel
• Order can be defined in app spec using “predecessors”
• Predecessor relationships only provide ordering on each resource
• If global ordering is desired, barriers can be defined in app spec
• See BarrierSummerCampExample
Barriers Across Components

Application Block

Component Block 1
Senders

- Process Block 1
  - Prepare Files

- Process Block 2
  - Join Network

- Barrier Block 1
  - Phase 1 Barrier

- Process Block 3
  - Send Files

Component Block 2
Receivers

- Process Block 1
  - Join Network

- Barrier Block 1
  - Phase 1 Barrier

- Process Block 2
  - Receive Files
Tutorial Agenda

• Introduction, Slice/sliver initialization
• Gush configuration, Gush startup
• Gush overview (while PLab slivers are created)
• Basic Gush functionality
• Simple experiments using PLab and PG
• Experiments highlighting failure recovery
• Experiments involving barriers
• Very brief introduction to Nebula
• Questions

http://gush.cs.williams.edu/trac/gush/wiki/GushSummerCamp
Nebula

- Nebula (GUI) allows users to describe, run, monitor, & visualize applications
- XML-RPC interface for managing applications programmatically
Questions?

For more info:
http://gush.cs.williams.edu