Tutorial Session: Experiment Control Using Gush

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

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions

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

- Start VM in VirtualBox using username and password provided
- Open terminal
- Run:

```bash
$ bash
$ keychain ~/.ssh/id_rsa
$ echo 'StrictHostKeyChecking no' >> ~/.ssh/config
$ mkdir gush
$ cd gush
$ cp /usr/local/geni/gush/gush-bin-32bit.tgz .
$ tar xzvf gush-bin-32bit.tgz
```
[omni]
users = guest31

[plc]
type=sfa
authority=plc.gec11
user=plc.gec11.guest31
cert=~/.sfi/plc.gec11.guest31.gid
key=~/.sfi/guest31.pkey
registry=http://www.planet-lab.org:12345
slicemgr=http://www.planet-lab.org:12347

[guest31]
urn=urn:publicid:IDN+plc:gec11+user+guest31
keys=~/.ssh/id_rsa.pub
<?xml version="1.0" encoding="UTF-8"?>
<gush>
  <resource_manager type="geni-plc">
    <port_map slice="gec11_slice31" port="61500"/>
  </resource_manager>
  
  <resource_manager type="geni-pg" cf="plc">
    <port_map slice="slice31" port="61600"/>
  </resource_manager>
</gush>
ProtoGENI RSpec (2-Fedora10 PCs)

```xml
<?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 myslice.rspec
- DON’T RUN THIS YET! We need to rate limit.
- From gush directory:
  
  ```
  $ helper-scripts/handle-geni.py -n -f plc
  -a https://www.emulab.net/protogeni/xmlrpc/am
  createsliver slice31 myslice.rspec
  ```

- Should get back info about 2 nodes
- If you get a “server busy” error, try again in a few minutes…
Starting Gush

- Now we’re ready to start Gush!
  - But we won’t actually connect to anything just yet… we need to wait for our VMs to boot
- In gush directory (pick your favorite port):

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

gush> Gush has learned about the slice gec11_slice31. Gush has learned about the slice slice31.

gush> Updated information on the slice gec11_slice31 is available.
Updated information on the slice slice31 is available.
```
There are 17 known nodes:

<table>
<thead>
<tr>
<th>Node</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planet2.scs.cs.nyu.edu">gec11_slice31@planet2.scs.cs.nyu.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@ricepl-1.cs.rice.edu">gec11_slice31@ricepl-1.cs.rice.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planet1.scs.stanford.edu">gec11_slice31@planet1.scs.stanford.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab1.cs.uchicago.edu">gec11_slice31@planetlab1.cs.uchicago.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab1.cs.umass.edu">gec11_slice31@planetlab1.cs.umass.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab1.cs.uoregon.edu">gec11_slice31@planetlab1.cs.uoregon.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab2.cs.uoregon.edu">gec11_slice31@planetlab2.cs.uoregon.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab3.cs.uoregon.edu">gec11_slice31@planetlab3.cs.uoregon.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab02.cs.washington.edu">gec11_slice31@planetlab02.cs.washington.edu</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab1.koganei.wide.ad.jp">gec11_slice31@planetlab1.koganei.wide.ad.jp</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab1.arizona-gigapop.net">gec11_slice31@planetlab1.arizona-gigapop.net</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab2.arizona-gigapop.net">gec11_slice31@planetlab2.arizona-gigapop.net</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:gec11_slice31@planetlab3.arizona-gigapop.net">gec11_slice31@planetlab3.arizona-gigapop.net</a>:61500</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:guest31@pc419.emulab.net">guest31@pc419.emulab.net</a>:61600</td>
</tr>
<tr>
<td>P</td>
<td><a href="mailto:guest31@pc430.emulab.net">guest31@pc430.emulab.net</a>:61600</td>
</tr>
</tbody>
</table>
Add Resources to PlanetLab Slice

- PLEASE DON’T DO THIS NOW!
- Gush can add resources to PlanetLab slices

```
gush> slice add gec11_slice31 ucsd
gush> The hosts assigned to slice gec11_slice31 have been updated.
Updated information on the slice gec11_slice31 is available.

gush> info nodes
There are 20 known nodes:
[ P ] gec11_slice31@planet2.scs.cs.nyu.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@ricepl-1.cs.rice.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@planet1.scs.stanford.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@planetlab1.cs.uchicago.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@planetlab1.ucsd.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@planetlab2.ucsd.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@planetlab3.ucsd.edu:61500(pref=0) (Disconnected.)
[ P ] gec11_slice31@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 gec11_slice31 ucsd
  gush> The hosts assigned to slice gec11_slice31 have been updated.
  Updated information on the slice gec11_slice31 is available.

  gush> info nodes
  There are 17 known nodes:
  [ P   ] gec11_slice31@planet2.scs.cs.nyu.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@ricepl-1.cs.rice.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@planetlab1.cs.stanford.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@planetlab1.cs.uchicago.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@planetlab1.cs.umass.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@planetlab2.cs.uoregon.edu:61500(pref=0)  (Disconnected.)
  [ P   ] gec11_slice31@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 gec11_slice31
    gush> The slice gec11_slice31 has been renewed.
```

- Gush can re-query PlanetLab server for updates

```
gush> slice update gec11_slice31
    gush> Updated information on the slice gec11_slice31 is available.
```
Tutorial Agenda

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions
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
Step 1: App Spec

```xml
<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-plc" group="gec11_slice31"/>
      <resource type="geni-pg" group="slice31"/>
      <resource type="geni-orca" group="orca-slice31"/>
    </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.
<?xml version="1.0" encoding="UTF-8"?>
<gush>
  <resource_manager type="geni-plc">
    <port_map slice="gec11_slice31" 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
Step 4: Start Application

- 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.

gec11_slice31@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.

gec11_slice31@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.

gec11_slice31@planetlab1.cs.umass.edu:61500,24301: Hello World
gec11_slice31@planetlab1.cs.uoregon.edu:61500,27717: Hello World
The experiment has ended.

gush> disconnect
gec11_slice31@planetlab1.cs.umass.edu:61500 has decided to leave the mesh.
gec11_slice31@planetlab1.cs.uoregon.edu:61500 has decided to leave the mesh.

gush> quit
Tutorial Agenda

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions
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:

  [   ] gec11_slice31@planet2... :61500(pref=0) (Disconnected.)
  [   ] gec11_slice31@ricepl-1...:61500(pref=0) (Disconnected.)
  [   ] gec11_slice31@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”

```plaintext
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

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions
Simple Experiments using PL & PG

- Check out tests/simple.xml
- Modify it to use:
  - Only PlanetLab resources
  - Only ProtoGENI resources
  - A mix of both
- Be careful when reusing project/experiment names in app spec (or always restart Gush)
- Notice the different “synchronization” behaviors when using 2 components. We’ll come back to this later.
Tutorial Agenda

• 3:30 – Introduction, VM startup, account distribution
• 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
• 4:00 – Gush overview (while slivers are created)
• 4:20 – Basic Gush functionality
• 4:30 – Simple experiments using PL and PG
• 4:45 – Experiments highlighting failure recovery
• 5:00 – Experiments involving SWORD and barriers
• 5:15 – Very brief introduction to Nebula
• 5:20 – Questions
Failure Experiments using PL & PG

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

• 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

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions
Finding “good” resources on PlanetLab can be challenging at times!

Gush can use SWORD to find machines that meet a specified criteria.

See [http://sword.cs.williams.edu](http://sword.cs.williams.edu) for more info on SWORD.

See SwordExample on Gush wiki for more info on using SWORD with Gush.
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 BarrierTutorialExample
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

- 3:30 – Introduction, VM startup, account distribution
- 3:45 – Slice/sliver initialization, Gush configuration, Gush startup
- 4:00 – Gush overview (while slivers are created)
- 4:20 – Basic Gush functionality
- 4:30 – Simple experiments using PL and PG
- 4:45 – Experiments highlighting failure recovery
- 5:00 – Experiments involving SWORD and barriers
- 5:15 – Very brief introduction to Nebula
- 5:20 – Questions
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