GENI Experiment Control Using Gush

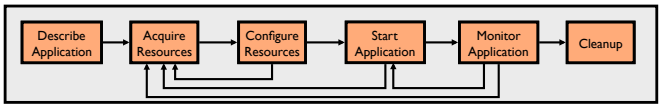
Jeannie Albrecht
Williams College
http://gush.cs.williams.edu
GEC 5

Overview

- How do we actually use GENI?
- Goal: Develop abstractions and tools for addressing the challenges of managing distributed applications
  - Make it easy for a range of users to run a variety of experiments on GENI
- Strategy
  - Interact with geniwrapper to locate resources and obtain credentials (similar to sfi)
  - Interface with other user tools (i.e., Raven)
  - Hide complexity and use one user interface to interact with different underlying systems (i.e., PlanetLab, MAX, GpENI, etc.)

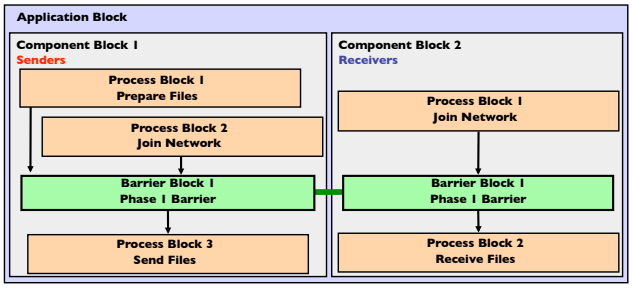
Gush

- 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 worldwide



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 Specification (Demo)

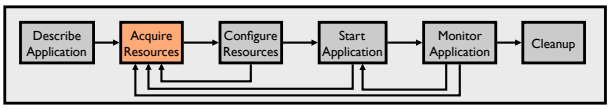
```
<gush>
 <project name="simple">
  <software name="SimpleSoftwareName" type="none">
   <package name="Package" type="web">
    <path>http://sysnet.cs.williams.edu/~jeannie/software.tar</path>
    <dest_path>software.tar</dest_path>
   </package>
  </software>
  <component name="Cluster1">
   <rspec>
    <num_hosts>3</num_hosts>
   </rspec>
   <software name="SimpleSoftwareName" />
   <resources>
    <resource type="planetlab" group="williams_gush" />
    <resource type="gpeni" group="gpeni_gush" />
    <resource type="max" group="maxpi_gush" />
   </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>
```

SOFTWARE

DEFINE RESOURCE POOL

DEFINE PROCESSES (EXECUTION)

Step 2: Acquire Resources

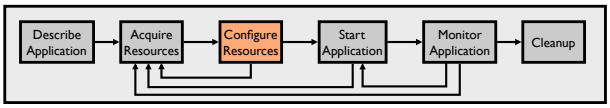


- How can we find "good" machines?
  - We may want machines with specific characteristics
  - Rspec? (TBD)
- Gush interfaces directly with geniwrapper
  - Define basic information in Gush config file
  - Send this basic info to geniwrapper to obtain resources

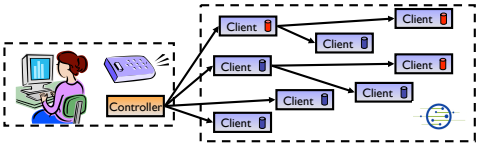
Gush Resource Directory

```
<gush>                                                    GENIWRAPPER
  <resource_manager type="planetlab">
    <user>jeannie@cs.williams.edu</user>  ──────►  plc.williams.jeannie
    <allsites>allsites.xml</allsites>
    <port_map slice="williams_gush" port="15413"/>  ──►  plc.williams.gush
  </resource_manager>
                                  ──────►  <configFile>planetlab_sfi_config</configFile>
  <resource_manager type="gpeni">
    <user>jeannie@cs.williams.edu</user>
    <port_map slice="gpeni_gush" port="15414"/>
  </resource_manager>

  <resource_manager type="max">  ──────►  plc.max.maxpl.jeannie
    <user>jeannie@cs.williams.edu</user>
    <port_map slice="maxpl_gush" port="15415"/>
  </resource_manager>

</gush>
```

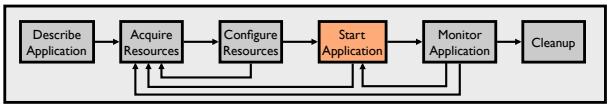
Step 3: Configure Resources



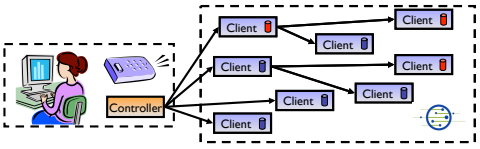
- Connect to and configure selected resources
  - Optionally create a tree for achieving scalability in communication
  - **Controller** "remotely controls" the **clients** on our behalf
  - Install software on clients



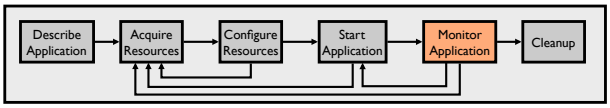
Step 4: Start Application



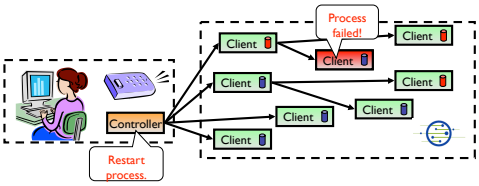
- Controller issues commands to clients telling them to start running our application
  - Senders begin running sender processes
  - Receivers begin running receiver processes



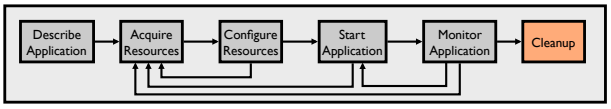
Step 5: Monitor Application



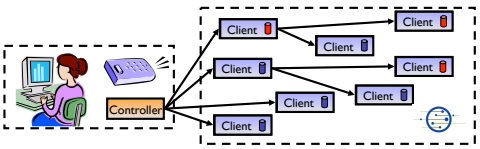
- We want to make sure the processes keep running
- Gush clients monitor experiment processes for failures
  - If a failure is detected, client notifies controller
  - Controller decides to tell client to restart failed program or process



Step 6: Cleanup



- Gush clients make sure all programs exited cleanly
- Remove logs and software from remote machines
- Disconnect clients from controller



Demo

```
albrecht:trunk jeannie$ ./gush  -P 15000
gush> Gush has learned about the slice gpeni_gush.
Gush has learned about the slice maxpl_gush.
Gush has learned about the slice williams_gush.
info nodes
There are 15 known nodes:
[ P      ] williams_gush@planetlab1.ucsd.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab2.ucsd.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab3.ucsd.edu:15413(pref=0) (Disconnected.)
[  U     ] jeannie@sysnet.cs.williams.edu:15400(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab1.williams.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab2.williams.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab3.williams.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab4.williams.edu:15413(pref=0) (Disconnected.)
[ P      ] williams_gush@planetlab5.williams.edu:15413(pref=0) (Disconnected.)
[ P      ] gpeni_gush@geni-planetlab-1.ksu.gpeni.net:15414(pref=0) (Disconnected.)
[ P      ] gpeni_gush@geni-planetlab-1.ku.gpeni.net:15414(pref=0) (Disconnected.)
[ P      ] maxpl_gush@planetlab2.dragon.maxgigapop.net:15415(pref=0) (Disconnected.)
[ P      ] maxpl_gush@planetlab3.dragon.maxgigapop.net:15415(pref=0) (Disconnected.)
[ P      ] maxpl_gush@planetlab4.dragon.maxgigapop.net:15415(pref=0) (Disconnected.)
[ P      ] maxpl_gush@planetlab5.dragon.maxgigapop.net:15415(pref=0) (Disconnected.)
```

Demo

```
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.
gpeni_gush@geni-planetlab-1.ksu.gpeni.net:15414 has joined the mesh.
The file transfer of Package to geni-planetlab-1.ksu.gpeni.net has been completed.
The software installation of Package on geni-planetlab-1.ksu.gpeni.net was successful.
williams_gush@planetlab1.williams.edu:15413 has joined the mesh.
maxpl_gush@planetlab2.dragon.maxgigapop.net:15415 has joined the mesh.
The file transfer of Package to planetlab1.williams.edu has been completed.
The software installation of Package on planetlab1.williams.edu was successful.
The file transfer of Package to planetlab2.dragon.maxgigapop.net has been completed.
The software installation of Package on planetlab2.dragon.maxgigapop.net was successful.
gpeni_gush@geni-planetlab-1.ksu.gpeni.net:15414,31821: Hello World
williams_gush@planetlab1.williams.edu:15413,19548: Hello World
maxpl_gush@planetlab2.dragon.maxgigapop.net:15415,26459: Hello World
The experiment has ended.
```

Status and Next Steps

- Gush works with current PlanetLab XML-RPC API (PLCAPI)
  - Have written code for geniwrapper integration
  - Needs to be debugged/tested
- Cluster Integration so far
  - PlanetLab
  - GpENI
  - MAX
  - Raven
- Next steps
  - Continue obtaining user feedback to enhance usability and provide additional functionality
  - Continue to integrate with other Cluster B projects