HOW to run GNRS experiment in ORBIT

1. Loading a Node Image:

Use the GNRS testing image available on the ORBIT repository .

Command:

```
# On the Grid, use only Gen-3 nodes
omf load -i gnrs-1.0.ndz -t inventory:topo:gen3

# On a sandbox, image everything
omf load -i gnrs-1.0.ndz -t system:topo:all
```

2. Preparing Experiment Files:

a. Put the experiment in to a new folder, eg: “my-exp-test” in home.

Command:

```
# Change to our home directory
cd
# Create experiment directory
mkdir my-exp-test
cd my-exp-test
```

b. Then download the experiment files we need.

```
# The main GNRS compiled JAR file. Contains server and client code
wget https://bitbucket.org/romoore/gnrs/downloads/gnrs.jar
# Scripts for running on nodes
wget https://bitbucket.org/romoore/gnrs/raw/master/jserver/gnrsd
wget https://bitbucket.org/romoore/gnrs/raw/master/jserver/gbench
wget https://bitbucket.org/romoore/gnrs/raw/master/jserver/ggen
wget https://bitbucket.org/romoore/gnrs/raw/master/jserver/gnrsd.init
# Client trace generator
wget https://bitbucket.org/romoore/gnrs/raw/master/jserver/genTrace.sh
# Topology generation files
wget https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/topoGen.sh
wget
https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/degreeTopoGenerator.m
wget https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/foo.sh
wget
https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/jellyfishTopoGenerator.
m
wget
https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/locTopoGenerator.m
wget https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/locTopoProc.m
```

```
wget
https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/sizeTopoGenerator.m
wget https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/topoGen.sh
wget https://bitbucket.org/romoore/gnrs/raw/master/src/tools/topology_generator/topoGenerator.m
# The OMF experiment files
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/as-binding.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/gnrs_config.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/gnrs_group.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/gnrs_node.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/resources.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/simple.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/statscollect.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/utils.rb
# Preprocessing script
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/prepare.sh
# GNUPlot graphing script (Optional)
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/res-graph/process.rb
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/res-graph/plot-ins.gp
wget https://bitbucket.org/romoore/gnrs/raw/master/omf/res-graph/plot-lkp.gp
```

3. **Creating client trace.**

Client trace is a set of trace request such as Lookup and Insert

Command:

```
# Generate a trace file
chmod +x genTrace.sh
./genTrace.sh 1 10000 >client_1.trace
# Check that the file is correct
wc -l client_1.trace # should print 20,000
```

The output file is client_1.trace which would be used in prepare.sh. It contains 10000 Lookup and 10000 Insert.

4. Generating Topology

Command:

```
# The script below will generate a topology based on the largest prefixes. Requires GNU Octave
chmod +x topoGen.sh foo.sh
./topoGen.sh 1 4
```

1: method one. / 4: four ASes

Important output file are : prefix.data, topology.data

Code Running Details:

1) topoGen.sh:

   Download file: ASPrefixData.mat, topologyData.mat, shellMem.mat, hangMem.mat, clique.mat, linkPrefixofLoc.mat

   Run file “foo.sh 1 4”

2) topoGen.sh==> foo.sh

   source files that are waiting be call : degreeTopoGenerator.m, jellyfishTopologyGenerator.m, sizeTopoGenerator.m, topoGenerator.m(The .mat files downloaded above may be called by them)

3) topoGen.sh==> foo.sh==>topoGenerator(1, 4)

“topoGenerator(1, 4)”: function sizeTopoGenerator(1,4) is called. It would generate prefix.data, topology.data, link.data, datatraceInput.mat

5. Preparing Experiment Archive topology.tgz.

Command:

```
# Bundle up experiment files for nodes
chmod +x prepare.sh
./prepare.sh # Creates topology.tgz
```

Code Running Details:

(There are many lines commented. If they are executed, Nodes binding, AS binding and click delay module would be configured. However, here is not.

1) download required files:

map-ipv4.xml, spg.i386, as-uniq.pl

2) run “spg.i386 topology.data”
3) create topology/
4) copy prefix.data into topology/ named prefixes.ipv4
5) copy topology.data.route into topology/topology.data.route.
6) move map-ipv4.xml into topology/
7) copy client_1.trace into topology/
8) tar the files in topology/ to an topology.tgz file
9) remove as-uniq.pl, spg.i386, topology.data.route, topology/

(Note: now the folder topology/ has files:
prefix.data, topology.data.route, map-ipv4.xml, client_1.trace
)

6. Preparing the Web Server

Place some files in to *public_html* directory.

Command:

```
# Create directories
mkdir -p $HOME/public_html/gnrs/static
# Ensure the directories are accessible
chmod +rx $HOME/public_html $HOME/public_html/gnrs $HOME/public_html/gnrs/static
```

Move static files into $HOME/public_html/gnrs/static;
Unpack dynamic files into $HOME/public_html/gnrs;

Command：

```
cp gnrs.jar gnrsd gnrsd.init ggen gbench $HOME/public_html/gnrs/static/
cd $HOME/public_html/gnrs && tar -zxvf $HOME/my-exp-test/topology.tgz && cd $HOME/my-exp-test
```

Now in folder $HOME/public_html/gnrs/static/:
gnrs.jar, gnrsd, gnrsd.init, ggen gbench

Now in folder $HOME/public_html/gnrs/topology/:
prefix.data, topology.data.route, map-ipv4.xml, client_1.trace

7. Running the Experiment

First, turn on the nodes that are imaged with command:

```
omf tell -a on -t system:topo:imaged
```

Then, we adjust some common parameters to run the experiment.(Note: the complete list of parameter is in resource.rb)
Command:

```
omf exec simple.rb -- --tarUrl http://repository1:8080/~$USER/gnrs/tar/ --tmpDir
\$HOME/public_html/gnrs/tar --dataUrl http://repository1:8080/~$USER/gnrs/topology --scriptUrl
\http://repository1:8080/~$USER/gnrs/static --clientWait 60 --numServers 4 --numClients 1 --sNodes
1 \--cNodes 1 --messageDelay 250
```

From the above command, we set several parameters:

1) tarUrl:

   This is the URL from which nodes will download their .tgz files built by the experiment. This should be the web server where your user directory is located.

2) tmpDir:

   local directory that serves the .tgz files for nodes to download.

3) dataUrl:

This is where the topology.tgz was unpacked and where the experiment can find the dynamic files.

4) scriptUrl:
   the URL where the experiment will get static files (gnrs.jar, gnrsd.init, etc.)
5) clientWait:
   how long to wait for the trace to complete, in seconds.
6) numServers:
   How many GNRS servers to run total.
7) numClient:
   How many GNRS clients to run total.
8) sNodes:
   How many ORBIT nodes used to run servers.
9) cNodes:
   How many ORBIT nodes used to run clients.
10) messageDelay:
    How long the client should wait between messages, in microseconds

Code Running Detail:

1) Step One:

It has a function "doMainExperiment" to control the whole process of experiment.

a. simple.rb==>utils.rb(import GNRSnode classes, read resource.rb)

b. simple.rb==>utils.rb==>doInitSetip()==>defineGroups()

doInitSetup calls *defineGroups()* which calculate serversPerNode, clientsPerNode, assign hostname/group/nodelist/ipaddress to every node, assign asNumber/port/group to every server in this node.

serverMap[ ] contains gnrsGroups of servers, asMap[ ] contain gnrsNodes, clientMap[ ] contains gnrsGroup of clients

serverMap=[S1, S2,....]; S1 is the number of servers in the first node as server
clientMap=[C1, C2, ...]; C1 is the number of clients in the first node as client

2) Step Two:

a. simple.rb==>doMainExperimnet()==>utils.rb==>prepareNode()

prepareNodes()---> mkdir /gnrs/tar/group.hostname(eg: node1-2.orbit-lab.org)

b. simple.rb==>doMainExperimnet()==>utils.rb==>prepareDelayModule()==>makeDelayConfig()

generate delay value between severs, between clinets, and between one sever and one client.

c. simple.rb==>doMainExperimnet()==>utils.rb==>prepareDelayModule()==>makeDelayScript()

prepareDelayModule()----> generate a click script into "gnrs/tar/group.hostname/delayModule.click"
use these click script to apply delayconfig to every server and clients

3) ***Step Three:***
simple.rb==>doMainExperimnet()==>utils.rb==>installConfigs()

a. Make dir
/gnrs/tar/group.hostname/stats#server.asNumber#hash.new(server.asNumber);
/gnrs/tar/group.hostname/etc/gnrs; /gnrs/tar/group.hostname/usr/local/bin/gnrs;
/gnrs/tar/group.hostname/init.d

b.Download server configuration files "prefixes.ipv4,map-ipv4.xml,gnrs.jar,ggen, gbench,gnrsd"

c. makeBindingFile():
format: asNumber, group IP, server port

d. makeServerConfig() (from gnrs_config.rb): create
/gnrs/tar/group.hostname/etc/gnrs/server_#node.asNumber.xml

makeServerNetConfig():create /gnrs/tar/group.hostname/etc/gnrs/net-ipv4_#node.asNumber.xml

makeBerkelyDBConfig(): create
/gnrs/tar/group.hostname/etc/gnrs/berkelydb_#node.asNumber.xml

makeServerInit():start the server service. Create
/gnrs/tar/group.hostname/etc/init.d/gnrsd_#node.asNumber.xml

copy prefixed.ipv4 to /gnrs/tar/group.hostname/etc/gnrs/
copy map-ipv4.xml to /gnrs/tar/group.hostname/etc/gnrs/
copy gnrs.jar to /gnrs/tar/group.hostname/usr/local/bin/gnrs/
copy gnrsd to /gnrs/tar/group.hostname/usr/local/bin/gnrs/
echo asBinding to /gnrs/tar/group.hostname/etc/gnrs/topology.bind

makeClientConfig: create client#(node.asNumber)R#{asCount(node.asNumber)}
copy gnrs.jar, ggen, gbench

4) ***Step Four:***
a. simple.rb==>doMainExperimnet()==>utils.rb==>buildTarballs()
generate the compressed .tgz files that includes configuration.
b. simple.rb==>doMainExperimnet()==>utils.rb==>getHostTarballs()
unpack the tarballs and set authority.

5) ***Step Five:***

a.simple.rb==>doMainExperimnet()==>utils.rb==>installDelayModule()
copy the existed delayModule files to new dir and execute them
b.simple.rb==>doMainExperimnet()==>utils.rb==>installInit()
Install server init scripts
c.simple.rb==>doMainExperimnet()==>utils.rb==>launchServers()
run gnrsd_#(node.asNumber)
d.simple.rb==>doMainExperimnet()==>utils.rb==>launchGUID()
launching trace clients
e.simple.rb==>doMainExperimnet()==>utils.rb==>stopServers()
stop servers service

6) ***Step Six:***
a. simple.rb==>doMainExperimnet()==>utils.rb==>collectClienttStats()
b. simple.rb==>doMainExperimnet()==>utils.rb==>collectServerStats()
c. simple.rb==>doMainExperimnet()==>utils.rb==>removeExperimentFIles()
delete experiment-related files form nodes which is temporary

Finally, dont forget turn the power off.

```
omf tell -a offh
```