

# LSGi Demo: Deployment, Configuration and Testing Document

Tere Gonzalez, Janneth Rivera, Hernan Laffite, Fei Chen, Krishna Viswanathan

Dec 17, 2015, Updated July 21, 2016

## Contents

<b>LSGi Demo: Deployment, Configuration and Testing Document</b>	<b>1</b>
<b>1. Overview</b>	<b>1</b>
<b>2. LSGi Package Content</b>	<b>2</b>
<b>3. LSGi Deployment and Configuration</b>	<b>3</b>
<b>3.1. Initial Requirements</b>	<b>4</b>
<b>3.2. Download the package</b>	<b>4</b>
<b>3.3. Update Hosts File</b>	<b>5</b>
<b>3.4. SSH Passwordless</b>	<b>5</b>
<b>3.5. Deploy package</b>	<b>6</b>
<b>3.6. Configure the package</b>	<b>8</b>
<b>3.7. Run test cases</b>	<b>8</b>
<b>4. Manual Steps to Run single node demo and verify the results</b>	<b>10</b>
<b>4.1. How to verify that single node run was successfully</b>	<b>11</b>
<b>5. Run multi-node demo</b>	<b>12</b>
<b>6. Query inference states</b>	<b>13</b>
<b>7. History</b>	<b>16</b>

## 1. Overview

This document summarizes how to configure LSGi engine and run the demo in order to test graph inference models running in I4tm. The document describes methods to deploy, configure and test the engine.

### About LSGi

The LSGi engine uses a graphical inference algorithm to compute a joint probability of the graph given a small evidence in order to predict unknown events. Each vertex is associated to a random variable which denotes a discrete event or state. Our approximate inference algorithm infer a new state using iterative computation over all the vertices of the graph until convergence is reached.

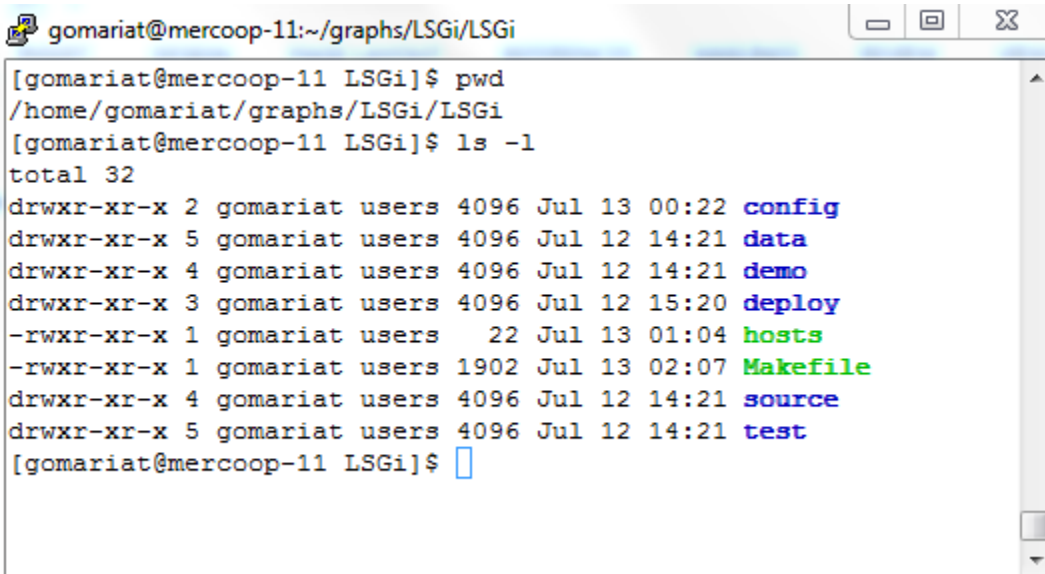
## About the Architecture of LSGi

The system is designed to run in a multicore and multinode environment using a graph-data parallel approach. The graph is partitioned over N multi-nodes and the engine launch N processes to compute the inference for each partitioned graph –local graph. In order to compute the global inference, the processes push their local predicted states to other processes; at the same time the processes pull from the other remote states to their local computation. The LSGi engine is designed to use a shared memory approach as communication medium to minimize communication overheads. The states can be stored in: a) **/dev/shm** while running multiple processes in a single server; or b) in the librarian file system **/lfs/** – a high bandwidth and low latency network file system while running the processes in multiple computing nodes connected.

For further background on conceptual design and model considerations, please see the documents posted in our repository.

## 2. LSGi Package Content

The LSGi Package has the following modules:



```

gomariat@mercoop-11:~/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ pwd
/home/gomariat/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ ls -l
total 32
drwxr-xr-x 2 gomariat users 4096 Jul 13 00:22 config
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 data
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 demo
drwxr-xr-x 3 gomariat users 4096 Jul 12 15:20 deploy
-rwxr-xr-x 1 gomariat users  22 Jul 13 01:04 hosts
-rwxr-xr-x 1 gomariat users 1902 Jul 13 02:07 Makefile
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 source
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 test
[gomariat@mercoop-11 LSGi]$

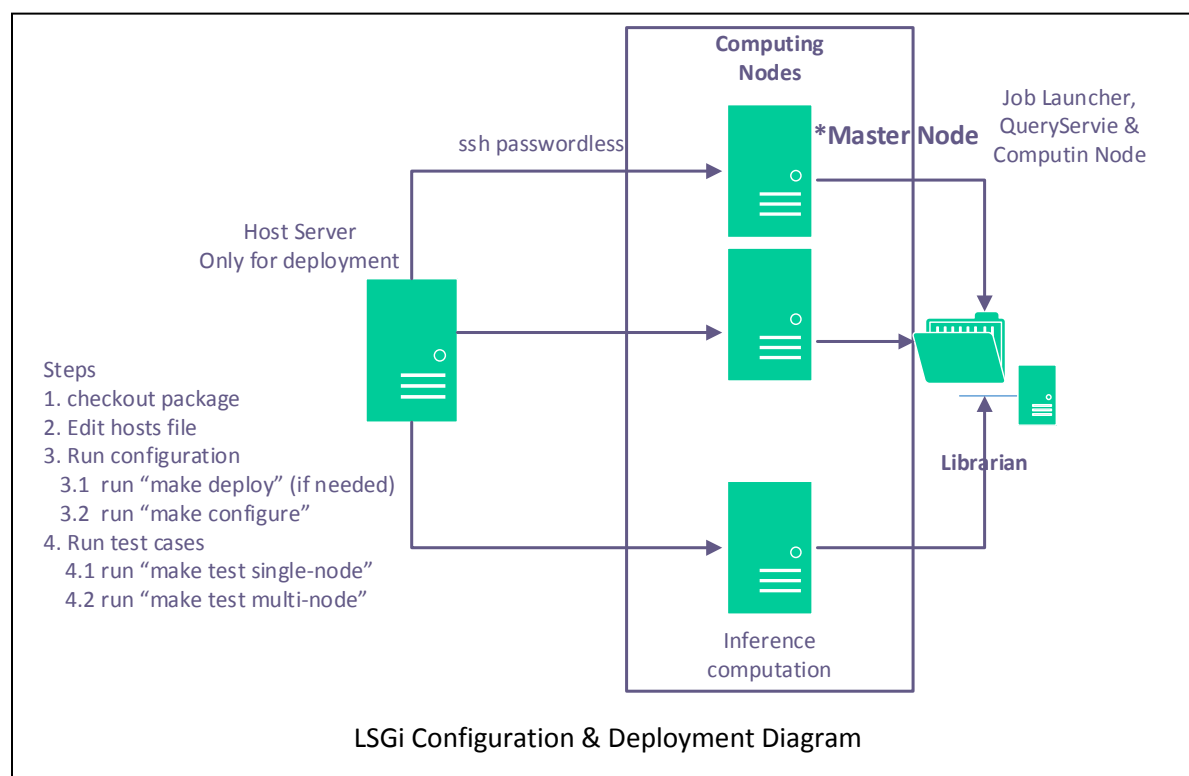
```

- **/config.** Scripts to configure remote nodes: compile and ssh passwordless.
- **/data:** It contains input graphs as binary files, configuration files and will host local outputs of the inferences like local states and statistics.
- **/demo.** It contains scripts to demo the existing functionality for the project that comprises:
  - run a single node inference
  - run multimode inference
  - run query client to request inference states

- **/deploy.** Scripts to deploy the local checkout to remote nodes.
- **/docs:** It contains multiple documents that describe the inference engine.
- **/source:** It contains the main source code for the modules: inference, query service and query client. To compile and generate the executables it should be executed “make clean” and then “make” commands.
- **/test.** It contains several basic test cases to test librarian access, pmem persistence and sync that are fundamental to verify that the shared states are working in the environment.
- **hosts:** main file to configure the computing nodes.
- **Makefile:** main configuration and deployment goals

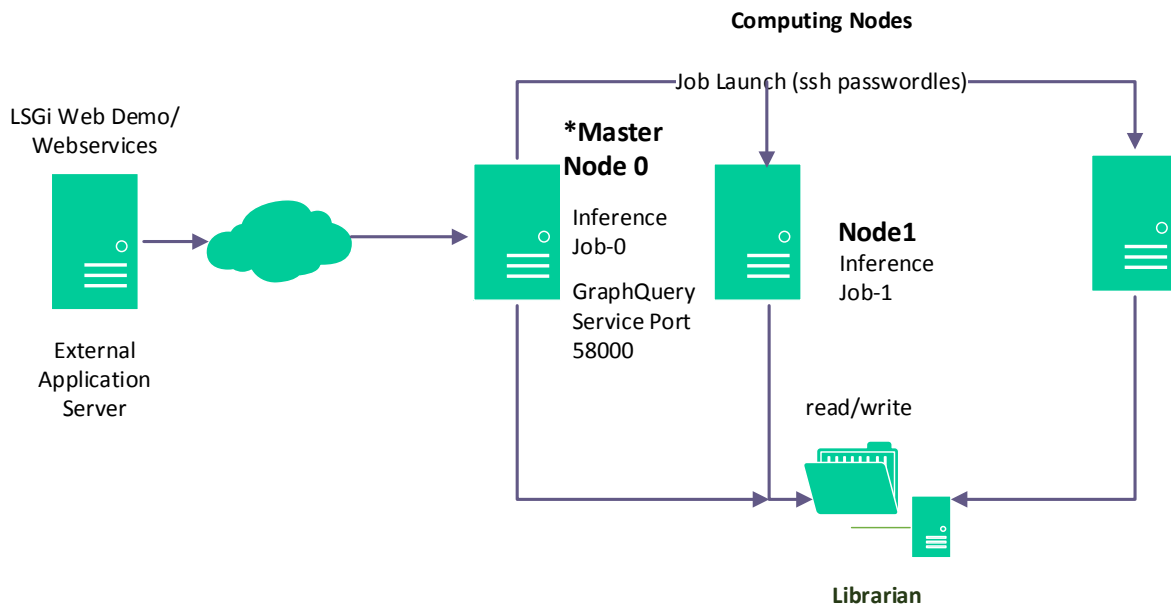
### 3. LSGi Deployment and Configuration

LSGi package includes the steps to deploy and configure the system. It assumes a set of computing nodes which will have a replica of the full package. All computing nodes should have access to a shared memory like /lfs or /dev/shm/. In case of the package is not deployed in the computing nodes, the Makefile includes a goal for deploy which copies to the list of nodes included on the hosts file. One of the nodes, the first of the host file configuration, will be named as Master node and the configuration step will create automatically an ssh passwordless between the master and the rest of the nodes. Finally, once it is configured, LSGi package includes 2 main test cases to verify if the configuration has been done successfully.



Once the computing nodes are configured, the master node is responsible for launching the inference jobs over the computing nodes. Each node will execute a local inference computation and access data

from librarian to synchronize global inference states. The master node is also responsible for launching a query service which is a TCP/IP service that also access librarian data to retrieve results to the LSGi web service in an external tomcat server.



Operational LSGi Deployment Diagram

### 3.1. Initial Requirements

- 1) Git: The LSGi package is on GitHub HPE.

```
$ sudo apt-get install git
```

- 2) g++ compiler >= 4.8
- 3) Lpmmem
- 4) Pthreads

- 5) Valid user to login to from host to computing nodes, and from master to other nodes.

### 3.2. Download the package

First, you need to checkout the LSGi package to have access to the deployment and configuration scripts. The package is located at the following URL:

<https://github.com/HewlettPackard/LSGi/>

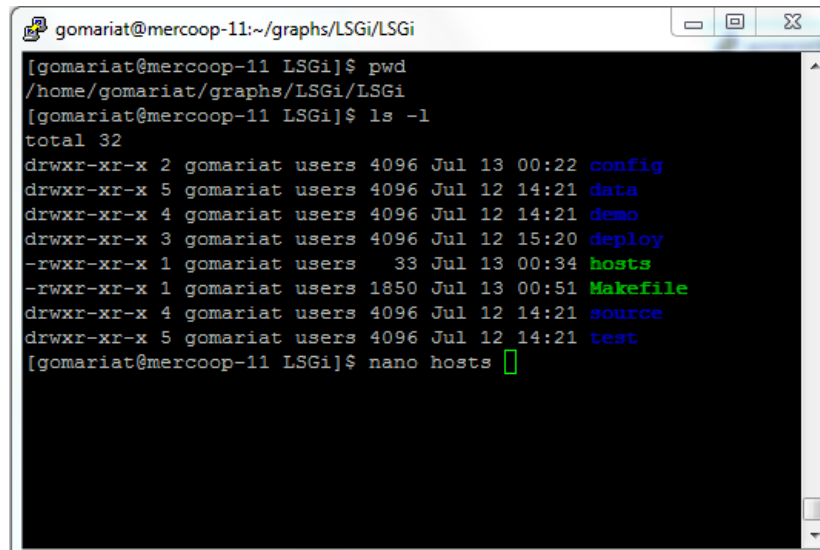
```
#Checkout the package from GitHub
```

```
$ git clone https://github.com/HewlettPackard/LSGi/
$ cd LSGi/
```

### 3.3. Update Hosts File

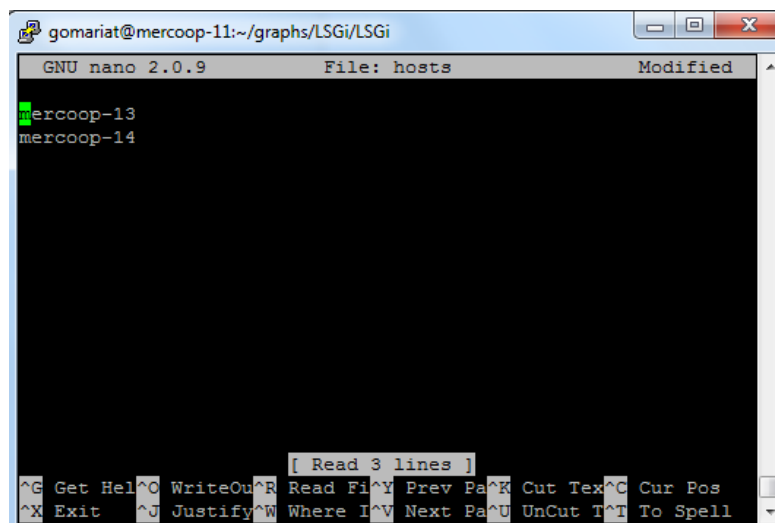
The hosts file should contain the computing nodes that are going to be configured in the system. In the previous checkout folder, go to LSGi/ directory which is the **LSGi root**.

```
$ cd LSGi/
$ nano hosts
```



```
gomariat@mercoop-11:~/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ pwd
/home/gomariat/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ ls -l
total 32
drwxr-xr-x 2 gomariat users 4096 Jul 13 00:22 config
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 data
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 demo
drwxr-xr-x 3 gomariat users 4096 Jul 12 15:20 deploy
-rwxr-xr-x 1 gomariat users 33 Jul 13 00:34 hosts
-rwxr-xr-x 1 gomariat users 1850 Jul 13 00:51 Makefile
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 source
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 test
[gomariat@mercoop-11 LSGi]$ nano hosts
```

Add the host name or IP of the computing nodes as list, one in each line as follows:



```
gomariat@mercoop-11:~/graphs/LSGi/LSGi
GNU nano 2.0.9      File: hosts      Modified
mercoop-13
mercoop-14

[ Read 3 lines ]
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

### 3.4. SSH Passwordless

In order to proceed with the deployment and configuration, it is required to setup passwordless SSH from host server to configuration nodes. To generate ssh key and copy to the computing nodes, go to LSGi root folder and run the script as follows:

```
$ make configureSSH
```

```

gomariat@mercoop-11:~/graphs/LSGi/LSGi
/home/gomariat/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ ls -l
total 32
drwxr-xr-x 2 gomariat users 4096 Jul 13 00:22 config
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 data
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 demo
drwxr-xr-x 3 gomariat users 4096 Jul 12 15:20 deploy
-rwxr-xr-x 1 gomariat users 22 Jul 13 01:04 hosts
-rwxr-xr-x 1 gomariat users 1842 Jul 13 01:05 Makefile
drwxr-xr-x 4 gomariat users 4096 Jul 12 14:21 source
drwxr-xr-x 5 gomariat users 4096 Jul 12 14:21 test
[gomariat@mercoop-11 LSGi]$ make configureSSH
#Configure SSH passwordless among the master node (first host),
the master itself and the rest of the nodes
cd config; ./sshCopy.sh
>> Setting up passwordless key >>
-----
Creating ssh key in from node:mercoop-11
Generating public/private rsa key pair.
Enter file in which to save the key (/home/gomariat/.ssh/id_rsa
):
/home/gomariat/.ssh/id_rsa already exists.
Overwrite (y/n)?
SSh-key copy to mercoop-13
The authenticity of host 'mercoop-13 (16.111.39.157)' can't be
established.
RSA key fingerprint is ba:27:51:a9:68:99:cb:85:e7:98:19:d5:f7:2
6:21:b3.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mercoop-13,16.111.39.157' (RSA) to
the list of known hosts.
gomariat@mercoop-13's password:
Now try logging into the machine, with "ssh 'mercoop-13'", and
check in:

    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expec
ting.

SSh-key copy to mercoop-14
The authenticity of host 'mercoop-14 (16.111.39.158)' can't be
established.
RSA key fingerprint is 31:73:7d:be:25:ce:3a:0b:d3:77:58:e4:51:4
d:43:5d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mercoop-14,16.111.39.158' (RSA) to
the list of known hosts.
gomariat@mercoop-14's password:
Now try logging into the machine, with "ssh 'mercoop-14'", and
check in:

    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expec
ting.

-----
Done passwordless between master to hosts
[gomariat@mercoop-11 LSGi]$

```

### 3.5. Deploy package

If the computing nodes do not have the LSGi package, you can deploy it using the following command:

```
$ make deployment
```

This deployment goal use the current path as source folder to copy. The destination folder is the home directory of the user. If you would like to change the paths, you can edit the Makefile to change the

LSGi\_PATH\_SRC and LSGi\_PATH\_DEST.

```

gomariat@mercoop-11:~/graphs/LSGi/LSGi
[gomariat@mercoop-11 LSGi]$ clear
[gomariat@mercoop-11 LSGi]$ make deployment
#Deploy LSGi package to remote hosts
#[Step #1]
cd deploy/; ./setPermissions.sh /home/gomariat/graphs/LSGi
>> Setting permissions to local package
/home/gomariat/graphs/LSGi/LSGi/config
/home/gomariat/graphs/LSGi/LSGi/demo/inference
/home/gomariat/graphs/LSGi/LSGi/demo/graphQuery
/home/gomariat/graphs/LSGi/LSGi/deploy
/home/gomariat/graphs/LSGi/LSGi/test/deploy
/home/gomariat/graphs/LSGi/LSGi/demo/inference
/home/gomariat/graphs/LSGi/LSGi/demo/graphQuery
-----
Done with permissions
#[Step #2]
cd deploy/; ./copyPkg.sh /home/gomariat/graphs/LSGi ~/
>> Copying files to target nodes.
copy to mercoop-13...
test                100%    3      0.0KB/s   00:00
Makefile            100% 1842    1.8KB/s   00:00
Makefile            100%  248    0.2KB/s   00:00
processSync.h       100% 7579    7.4KB/s   00:00
BarrierMonitor.o     100% 201KB 201.3KB/s 00:00
BarrierMonitor.cpp   100% 1423    1.4KB/s   00:00
BarrierMonitor       100% 102KB 101.9KB/s 00:00
REAME               100%  187    0.2KB/s   00:00
.project            100%  846    0.8KB/s   00:00
.cproject           100% 3713    3.6KB/s   00:00
Makefile            100%  451    0.4KB/s   00:00
test_createUpdatePersistLoop 100% 3019    3.0KB/s   00:00
REAME.txt           100%  655    0.6KB/s   00:00

master              100%  198    0.2KB/s   00:00
HEAD                100%  198    0.2KB/s   00:00
pre-rebase.sample   100% 4951    4.8KB/s   00:00
post-receive.sample 100%  548    0.5KB/s   00:00
pre-commit.sample   100% 1578    1.5KB/s   00:00
pre-apppatch.sample 100%  398    0.4KB/s   00:00
post-commit.sample  100%  160    0.2KB/s   00:00
commit-msg.sample   100%  896    0.9KB/s   00:00
applypatch-msg.sample 100%  452    0.4KB/s   00:00
update.sample       100% 3611    3.5KB/s   00:00
post-update.sample   100%  189    0.2KB/s   00:00
prepare-commit-msg.sample 100% 1239    1.2KB/s   00:00
pack-301da65eadac24475f5ae4 100% 50MB 25.2MB/s 00:02
pack-301da65eadac24475f5ae4 100% 18KB 18.0KB/s 00:00
index               100% 45KB 44.6KB/s 00:00
HEAD                100%  23      0.0KB/s   00:00
packed-refs         100%  94     0.1KB/s   00:00
config              100% 257     0.3KB/s   00:00
description          100%  73     0.1KB/s   00:00
README.md            100% 1066    1.0KB/s   00:00
-----
Done copy to nodes
[gomariat@mercoop-11 LSGi]$

```

### 3.6. Configure the package

Once the package exists on the remote nodes, you can run the configure script as follow goal:

```
$ make configure
```

This configure goal will copy host file changes into the computing nodes and compile remote LSGi packages to have it ready for test cases and demo.

```
[gomariat@mercoop-11 LSGi]$ make configure
#Copy hostfile to remote nodes
cd config/; ./copyHostFile.sh ~/LSGi
despath=/home/gomariat/LSGi
>> Copy hostfile...
-----
Hostfile to destination path:/home/gomariat/LSGi
mercoop-13...
hosts          100%   22    0.0KB/s   00:00
mercoop-14...
hosts          100%   22    0.0KB/s   00:00
-----
>>Done compiling
#Compile LSGi package in remote hosts
cd config/; ./compile.sh ~/LSGi
despaht=/home/gomariat/LSGi
>> Compiling...
-----
Compilation using current path:/home/gomariat/LSGi
mercoop-13...
/home/gomariat/LSGi/LSGi/source/inference
rm -f gibbsDNS-ArrayPartitioned.o gibbsDNSApp
g++ -Wall -Wno-sign-compare -lm -lpthread -lpmem -o gibbsDNSApp gibbsDNS-ArrayP
artitioned.cpp

/home/gomariat/LSGi/LSGi/source/queryService/clientService
rm -f QueryClient.o QueryClient
g++ -O2 -g -Wall -fmessage-length=0 -c -o QueryClient.o QueryClient.cpp
g++ -o QueryClient QueryClient.o
cp QueryClient ../../demo/graphQuery
-----
>>Done compiling
[gomariat@mercoop-11 LSGi]$
```

After the goal, the compilation should be done with 0 Errors. Otherwise, check the configuration path.

### 3.7. Run test cases

#### 3.7.1. Fists, add passwordless between master node and the other compute nodes

```
$ ssh <masterNode> #the first node of the hosts file
$ cd ~/LSGi/LSGi
$ make configureSSH
```

#### 3.7.2.Run Single Node Test

```
$ make testSingleNode
```

The single node launcher will execute the inference in verbose mode. The program outputs the trace on the console. If everything is correct, the user should see the legend ***"TEST RESULT: Success!!!"*** on the summary of the execution as follows:



```

gomariat@vm10-l4tm: ~
gomariat@simpl-14tm-580-35:~/LSGi/LSGi$ make testSingleNode
#Run test for SingleNode in remote hosts
cd test/deploy/; ./testSingleNode.sh ~/LSGi

LSGi by Hewlett Packard Labs
-----
Test case:          LSGi Single Node
Script:             /home/gomariat/LSGi/LSGi/demo/inference/launchSingleNode8g.sh
SharedFileSystem:
-----
Removing output files in vm1-l4tm at
/home/gomariat/LSGi/LSGi/data/outputStats
rm: cannot remove 'out*.*': No such file or directory
Launching in vm1-l4tm at:
/home/gomariat/LSGi/LSGi/demo/inference
Large Scale Graph Inference (LSGi)
Version 1.0-l4tm, @Hewlett Packard labs
-----
Launching single-local process
BEGIN!

===== Inference Parameters =====

input filename:      ../../data/inputGraphs/graph8.alchemy.bin
configuration filename: ../../data/config/configure_g8
Update frequency:    1 per iteration
processID:           0
Total Process:       1
Shared File System: /dev/shm/
===== Execution configuration:

User Configurations: 2,2000,../../data/outputStats/outg8,1,10,0.03
END COMPUTATION

===== Generating output =====

===== Summary of Execution =====
Graph Load Time(sec) = 3.0007
Gibbs Run Time(sec)  = 0.024378
VMEM usage (GB)      = 0.0942192
RSS MEM usage(GB)    = 0.00282288
Input Configuration  = 2,2000,../../data/outputStats/outg8,1,10,0.03
Output file          = ../../data/outputStats/outg8_p1_p0
Graph input file     = ../../data/inputGraphs/graph8.alchemy.bin
sync time (sec)      = 0.000229

Going to the next EXP..
===== Execution configuration:

Releasing topology
Releasing distribution
DONE!
done
Verifying output in vm10-l4tm at:
/home/gomariat/LSGi/LSGi/data/outputStats
-rw-r--r-- 1 gomariat gomariat 400 Jul 20 15:34 outg8_p1_p0.conv
-rw-r--r-- 1 gomariat gomariat 349 Jul 20 15:34 outg8_p1_p0.timer
-rw-r--r-- 1 gomariat gomariat 32 Jul 20 15:34 outg8_p1_p0.tsv
-----
TEST RESULT: Success!!! :)
LSGi single node has run and output files were generated.

```

### 3.7.3.Run Multi Node test

```
$ make testMultiNode
```

The multi node launcher will execute the inference in verbose mode. The program outputs the trace on the console. If everything is correct, the user should see the legend ***“TEST RESULT: Success!!!”*** on summary of the execution as follows:

```
gomariat@vm10-l4tm: ~
gomariat@simpl-l4tm-580-25:~/LSGi/LSGi$ make testMultiNode
#Run test for MultiNode in master node
cd test/deploy/; ./testMultiNode.sh ~/LSGi

LSGi by Hewlett Packard labs
-----
Test case:      Multi Node LSGi
Script:         /home/gomariat/LSGi/LSGi/demo/inference/launchMultiNode40G.sh
SharedFileSystem:
Found nodes:    10 (maxNodes=40, available:1,2,4,8,16,32,40)
-----
Removing output files in at:vm1-l4tm
/home/gomariat/LSGi/LSGi/data/outputStats
rm: cannot remove 'out*.*': No such file or directory
Launching in at:
/home/gomariat/LSGi/LSGi/demo/inference
Waiting for execution ....
Waiting for execution ....
Waiting for execution ....
Verifying output in at:
/home/gomariat/LSGi/LSGi/data/outputStats
-rw-r--r-- 1 gomariat gomariat 400 Jul 20 16:42 outg40_p10_p0.conv
-rw-r--r-- 1 gomariat gomariat 358 Jul 20 16:42 outg40_p10_p0.timer
-rw-r--r-- 1 gomariat gomariat  87 Jul 20 16:42 outg40_p10_p0.tsv
-----
TEST RESULT: Success!!! :)
LSGi multi-node has run and output files were generated.
```

## 4. Manual Steps to Run single node demo and verify the results

First, clean up previous run results and then you can run manually the launcher of 8-vertex graph as follows:

```
# Clean and remove previous run results
$ rm LSGi/data/outputStats/ *

#run small graph – 8 vertex graph
$ cd LSGi/demo/inference
$ ./launchSingleNode8G.sh
```

You can browse the content of the demo folder, which include all the scripts to launch, monitor, and kill the inference jobs.

```

gomariat@mercado-3:~/sshConn
gomariat@build-14tm-2:~/graphs/code/LSGi_DH/LSGi/LSGi/demo/inference$ ls -l
total 188
-rwxrwxr-x 1 gomariat gomariat 84742 Jul  9 00:24 gibbsDNSApp
-rw-rw-r-- 1 gomariat gomariat  10 Jul  8 23:47 hosts
-rwxrwxr-x 1 gomariat gomariat  323 Jul  8 23:47 jobStatus.sh
-rwxrwxr-x 1 gomariat gomariat  108 Jul  8 23:47 killGraphQService.sh
-rwxrwxr-x 1 gomariat gomariat  371 Jul  8 23:47 killInferenceAll.sh
-rwxrwxr-x 1 gomariat gomariat  219 Jul  8 23:47 killInferenceLocal.sh
-rwxrwxr-x 1 gomariat gomariat  367 Jul  8 23:47 launchGraphQService.sh
-rwxrwxr-x 1 gomariat gomariat 1952 Jul  8 23:47 launchMultiNode1B.sh
-rwxrwxr-x 1 gomariat gomariat 2067 Jul  8 23:47 launchMultiNode40G.sh
-rwxrwxr-x 1 gomariat gomariat 1812 Jul  8 23:47 launchMultiNode8G.sh
-rwxrwxr-x 1 gomariat gomariat 1925 Jul  8 23:47 launchMultiNodeDemo.sh
-rwxrwxr-x 1 gomariat gomariat 1925 Jul  8 23:47 launchMultiNode.sh
-rwxrwxr-x 1 gomariat gomariat  827 Jul  8 23:47 launchSingleNode8g.sh
-rwxrwxr-x 1 gomariat gomariat  759 Jul  8 23:47 launchSingleNode.sh
drwxrwxr-x 2 gomariat gomariat  154 Jul  8 23:47 LFS_local
-rw-rw-r-- 1 gomariat gomariat 3745 Jul  9 00:27 log.txt
-rw-rw-r-- 1 gomariat gomariat   0 Jul  9 00:27 QSlog.txt
-rwxrwxr-x 1 gomariat gomariat 36154 Jul  9 00:24 QueryService
-rwxrwxr-x 1 gomariat gomariat 1369 Jul  8 23:47 runQueryService1b.sh
-rwxrwxr-x 1 gomariat gomariat 1266 Jul  8 23:47 runQueryService_local.sh
-rwxrwxr-x 1 gomariat gomariat 1266 Jul  8 23:47 runQueryService.sh
-rw-rw-r-- 1 gomariat gomariat   0 Jul  8 23:47 Total
gomariat@build-14tm-2:~/graphs/code/LSGi_DH/LSGi/LSGi/demo/inference$

```

#### 4.1. How to verify that single node run was successfully

The single node launcher will execute the inference on the local machine in verbose mode. The program outputs the trace on the console. If everything is correct, the user should see the summary of the execution as follows:

```

gomariat@mercado-3:~
===== Generating output =====

===== Summary of Execution =====
Graph Load Time(sec) = 0.048896
Gibbs Run Time(sec)  = 0.006063
VMEM usage (GB)      = 0.0883598
RSS MEM usage(GB)    = 0.00291824
Input Configuration  = 1,100,.../data/outputStats/outg8,1,10,0.03
Output file          = ../../data/outputStats/outg8_p1_p0
Graph input file     = ../../data/inputGraphs/graph8.alchemy.bin
sync time (sec)      = 0.000175

Going to the next EXP..
===== Execution configuration:

Releasing topology
Releasing distribution
DONE!
done
root@node04:~/LSGi/demo/inference#

```

### 1. Verify the output results:

```
#go to the output folder
$ ls ../../data/outputStats/outg8_*
$ cat ../../data/outputStats/outg8_p1_p0.tsv
```

0

843

1336

517

346

2000

712

862

#the columns is yellow should be 0 and 2000, the other values should vary but they should be very close.

## 5. Run multi-node demo

```
$ cd LSGi/demo/inference

#specify the number of process to run,
#for example 4 or 2 , one in each node of the host file
$ ./launchMultiNode8G.sh 2
```

```

gomariat@mercado-3:~/sshConn
sh
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/inference$ ./launchMultiNode8G.sh 2
Large Scale Graph Inference (LSGi)
Version 1.0-14tm, @Hewlett Packard labs

-----
Running experiment for [ 2 ] processes
-----
/home/gomariat/graphs/code/LSGi/demo/inference
lauching on host: localhost
lauching on host: localhost
Launch done for [2] nodes
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]

-----
Graph Query Service for LSGi
-----
Graph: [ ../../data/inputGraphs/graph8.alchemy.bin ]
Shared on [2 nodes ]

Partition Node Size
-----
node [ 0 ]      4
node [ 1 ]      4

Shared Vertex States
-----
node [ 0 ]      graph8.alchemy.bin.2_0
                  /dev/shm/graph8.alchemy.bin.2_0.pdist

@@@ stats[0]:0
    stats[1]:2000
    stats[2]:0
node [ 1 ]      graph8.alchemy.bin.2_1
                  /dev/shm/graph8.alchemy.bin.2_1.pdist

@@@ stats[0]:3
    stats[1]:41
    stats[2]:4
Query Server starting...

-----
Connected on port [ 58000 ]
Query Server Ready!...
Waiting New query request: [ ? ]

```

The user should see the trace of the process launched as is showed above. This script will also startup a Query Service that waits for user queries and retrieve the graph states requested. See next section.

## 6. Query inference states

Once the inference is running, the user can request states of the vertices as follows:

### a. Query By Vertex identifier

#### Test 1:

```

$ cd LSGi/demo/graphquery/

#query state for specific vertex id
$ ./queryByVertexId.sh 0

```

```

gomariat@mercado-3:~/sshConn
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/graphQuery$ ./queryByVertexId.sh 0

-- Getting inference state: connecting to port=58000--
Query Type: [ Search by Vertex Id ]
vertex#:    [ 0 ]

Inference Results
-----
vertex# state  prob_0 iteration#
-----
0      1      0      2000
-----

Elapsed runtime:      240
Non-converged vertices: 0
Convergence:      100
state0=200010

Done query for [ 0 ]
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/graphQuery$

```

The iteration number should go up to 2000 when the inference finished and the state is =1 and prob\_0=0 for vertex 1. If you query before the inference has finished, if everything is running, the iteration should be changing up to 2000.

## Test 2.

```
#query state for specific vertex id
$ ./queryByVertexId.sh 5
```

```

gomariat@mercado-3:~/sshConn
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/graphQuery$ ./queryByVertexId.sh 5

-- Getting inference state: connecting to port=58000--
Query Type: [ Search by Vertex Id ]
vertex#:    [ 5 ]

Inference Results
-----
vertex# state  prob_0 iteration#
-----
5      0      1      2000
-----

Elapsed runtime:      240
Non-converged vertices: 0
Convergence:      100
state0=200010

Done query for [ 5 ]
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/graphQuery$

```

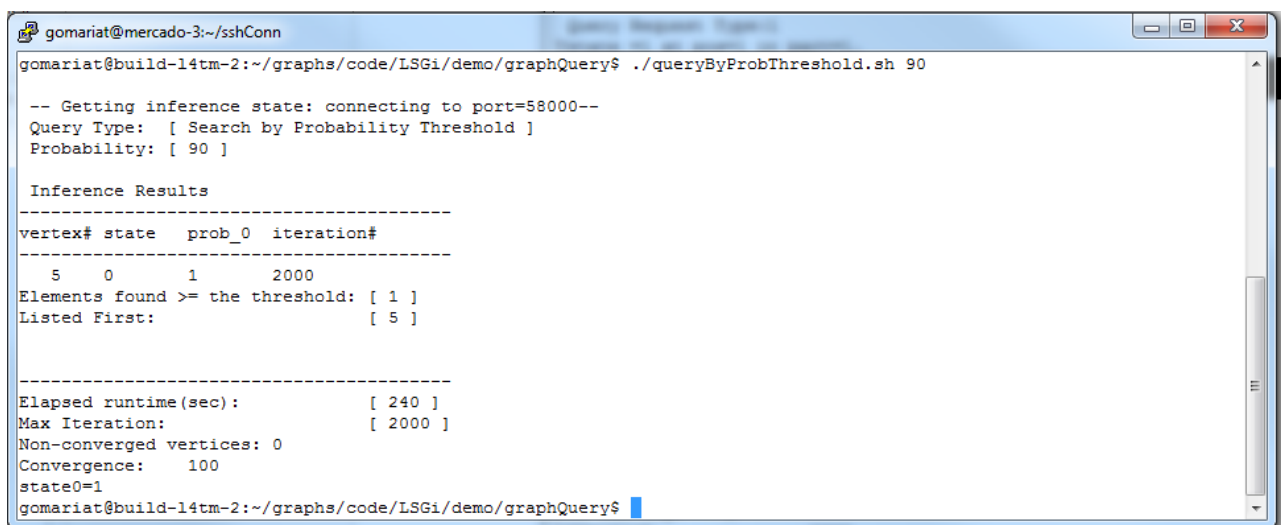
The iteration number should go up to 2000 when the inference finished and the state is =0 and prob\_0=1 for vertex 5. If you query before the inference has finished, if everything is running, the iteration should be changing up to 2000.

## b. Query By Probability Threshold.

### Test 1:

```
$ cd LSGi/demo/graphquery/
#query state for a subset of vertices which probability on infected above the threshold
#query for 90 probability of state 0
$ ./queryByProbThreshold.sh 90
```

The output should display only 1 element over 90% probability of being state=0;



```
gomariat@mercado-3:~/sshConn
gomariat@build-l4tm-2:~/graphs/code/LSGi/demo/graphQuery$ ./queryByProbThreshold.sh 90

-- Getting inference state: connecting to port=58000--
Query Type: [ Search by Probability Threshold ]
Probability: [ 90 ]

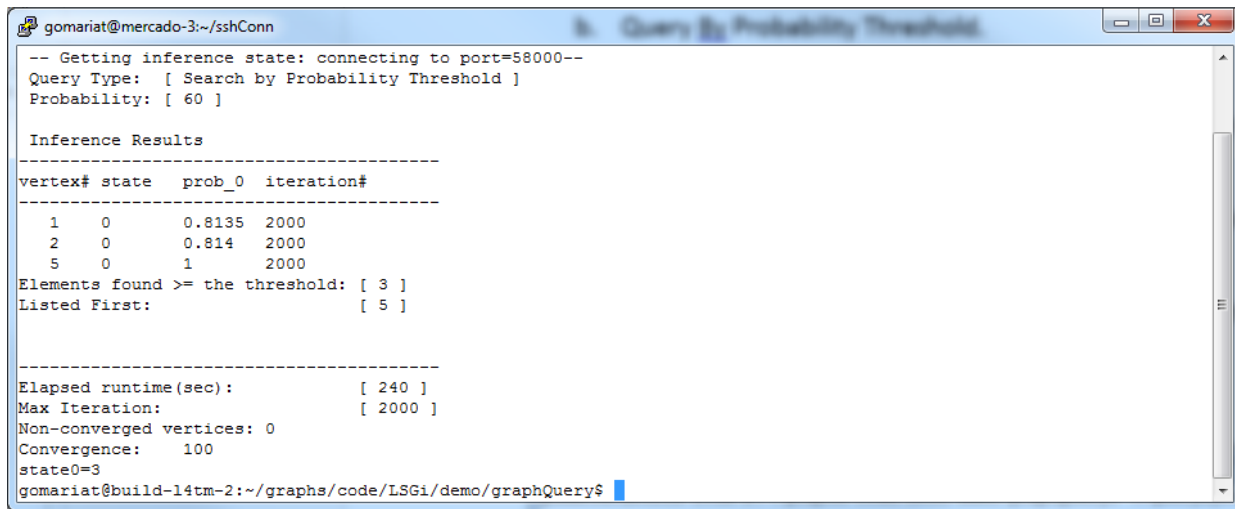
Inference Results
-----
vertex# state  prob_0  iteration#
-----
5      0      1      2000
Elements found >= the threshold: [ 1 ]
Listed First:                    [ 5 ]

-----
Elapsed runtime(sec):           [ 240 ]
Max Iteration:                  [ 2000 ]
Non-converged vertices: 0
Convergence: 100
state0=1
gomariat@build-l4tm-2:~/graphs/code/LSGi/demo/graphQuery$
```

### Test 2:

```
$ ./queryByProbThreshold.sh 60
```

The output should display 3 elements over 60% probability of being state=0.



```

gomariat@mercado-3:~/sshConn
-- Getting inference state: connecting to port=58000--
Query Type: [ Search by Probability Threshold ]
Probability: [ 60 ]

Inference Results
-----
vertex# state prob_0 iteration#
-----
1 0 0.8135 2000
2 0 0.814 2000
5 0 1 2000
Elements found >= the threshold: [ 3 ]
Listed First: [ 5 ]

-----
Elapsed runtime(sec): [ 240 ]
Max Iteration: [ 2000 ]
Non-converged vertices: 0
Convergence: 100
state0=3
gomariat@build-14tm-2:~/graphs/code/LSGi/demo/graphQuery$

```

## 7. History

Created Dec 14, 2015 Version 1.0

Updated June 2, 2016 version 1.1

Updated July 8, 2016 version 1.2

Updated July 12, 2016 version 2.0

+update deployment method.

+ Change gitrepo URL

+ Add results verification

Updated July 15, 2016 version 2.1

+Correct typos

+better screenshots