# Configuration to launch LSGi Jobs

Tere Gonzalez, Aug 25 2016

Contents

[Configuration to launch LSGi Jobs 1](#_Toc461199631)

[Overview. 1](#_Toc461199632)

[1. Input files: 1](#_Toc461199633)

[2. Configuration Parameters 2](#_Toc461199635)

[1. Execution configuration file: 3](#_Toc461199636)

[2. Command line parameters 3](#_Toc461199637)

[3. Running a LSGi Job 4](#_Toc461199638)

# Overview.

This document describes the requirements of how to launch LSGi jobs from command line. LSGi engine has configuration rules in order to successfully execute the job in single or multiple computing nodes. The following sections describe the inputs and configuration parameters require to run a job with for a given graph.

# Input files:

# The engines uses 3 input files that have to be placed in the same folder location. The files are:

1. **Graph file (edge list+ factors): in binary format.**

The graph file is the edge list (pair of connected vertices) and their factors. For example, the graph filename is grap8.bin. We have separated program to convert text file of the edge list to a binary code. See source/binaryConvertor. Example of edge list with factors is: vertex #1 connected to #10000 and its relation is (0,0)=0.75; (0,1) =0.25; (1,0)= 0.74;(1,1)=0.6. The vertex identifiers are consecutive integer numbers and they are used as indices of the data structures, so they are not tags. Consider element (0,0) is P(X1=0 | X2=0)=0.75 and (0,1) is P(X1=0 | X2=1)=0.6. The factor values should be log numbers.

1 10000 // 0.75 0.25 0.74 0.6  
2 10000 // 0.75 0.25 0.74 0.6  
…  
1 2941 // 0.75 0.25 0.74 .04  
…

In the current LSGi package input graphs are stored at: **LSGi/data/inputGraphs/graph8.bin**

1. **Labels file: known evidence of the graph as vertices list and the known states(0,1).**

The filename for the labels should contain the filename of the graph as prefix follow by .labels word, for example, grap8.bin.labels. In the next example, we can see that vertex 1 is labeled in state 0 and 1204 and 2941 as state1. The format of the file is: [vertice#] [state];

1 0  
1204 1  
2941 1

The labels file should be stored at the same location than the graph file. In the current LSGi package, input graphs and labels are stored at: **LSGi/data/inputGraphs/graph8.bin.labels**

1. **Partitions file**: When the engine launches N jobs, the engines requires as an input the size of vertices that are going to be processed in each node. The filename should contain the filename of the graph as prefix follow of .par. word and follow for the number of port or nodes to launch. For example, if the engine launches 1 or 8 jobs, the engine will look for the following files: graph8.par.1 or graph8.par.8. Each file should contain the same lines as the number of jobs. The summation of the partition size (an integer in each line in the file) should add up to the total number of vertices in the graph.

For example, let’s assume that the file graph8.bin contains 10 vertices. The first file is graph8.bin.par.1 and the second is the example of 8 nodes partitions graph8.bin.par.8.

2  
2  
1  
1  
1  
1  
1  
1

10

The partition size file should be stored at the same location than the graph file. In the current LSGi package, input graphs, labels and partition size fille are stored at: **LSGi/data/inputGraphs/graph8.bin.par.1 or LSGi/data/inputGraphs/graph8.bin.par.8.**

# 2. Configuration Parameters

As part of the configuration, we have 2 main elements: 1) execution related file and 2) command line parameters.

# Execution configuration file:

The content of execution configuration file is:

#Threads,#samples,outputFilename,numasocketsTouse(1;2),#SamplesSnapshop,threshold  
**1,100,../../data/outputStats/outgdns,1,10,0.03**

The list of parameters are:

1. #Threads: Number of threads to use for inference processing. The work is evenly partitioned by the number of threads for each process.
2. #samples: Number of iterations to run. For benchmark we use 100.
3. outputFilename: Filename and path for the local ouput files, the filename is used as
4. numasocketsTouse(1): This parameters should set ALWAYS to 1.
5. #SamplesSnapshop: Number of iterations to test convergence of the graph.
6. threshold: Threshold to verify if the vertex has convergence. We use (0.03 <=) to decide if the vertex id has converge.
7. Execution configuration file:

# Command line parameters

To launch the job from command line requires the following parameters:

./gibbsApp <executionConfigurationFile> <input graph> <JobIndex> <totalJobs> [-FS=]

The list of parameters are:

1. ./gibbsApp: Program that runs the inference using gibbs sampling algorithm.
2. <executionConfigurationFile>: the file described in the previous section(4,1) which contains threads and sampling info.
3. <input graph>: input graph filename as described (1,1).
4. <jobIndex>: job identifier for synchronization among computing nodes. The value is between 0 to N; where N is the maximum number of launched jobs.
5. <totalJobs>: maximum number of launched jobs.
6. <-FS=>: file system to use, for example: /dev/shm/ -in shared memory; /lfs/ in FAM; or local folder. The parameter is optional and the default value is /lfs/

For example:

1) Launch a single job for graph8.bin; the command line is:

./gibbsApp ../data/config/configure\_dns ../data/inputGraphs/graph8.bin 0 1 –FS=/dev/shm

2) Launch a 2 jobs for graph8.bin; the command line is:

Process 1:  
./gibbsApp ../data/config/configure\_dns ../data/inputGraphs/graph8.bin 0 2 –FS=/dev/shm

Process 2:  
./gibbsApp ../data/config/configure\_dns ../data/inputGraphs/graph8.bin 0 2 –FS=/dev/shm

# Running a LSGi Job

In order to run a job, the LSGi package includes a folder “demo/inference”. This folder includes a set of scripts that allow run a inference job, track or kill them.

Despite there are several related scripts customized to specific dataset, the main script to run the inference over multiple dataset and nodes is:

**launchMultiNodeDemo.sh:**

@Description:  
Launch processes in each node listed on hostfile. The script connects to the nodes via ssh and run the inference program in each of them. The script also launch the query service that will retrieve states information for the query client.

@Params:   
<N> Number of process to launch (MAX)  
<dataset> Dataset path to run.  
<port> QueryService TCPI/IP port for accepting query request. (Optional, default 58000)

@Requirements:  
1. Make sure 'hosts' file is updated with the list (name or IP) of connected nodes.  
2. Make sure the dataset file has the related files require to run: binary graph, labels and partitions. See section 1 of this document.

@How to run it:

Command line: ./launchMultiNodeDemo.sh <N> <dataset> <port>

For example:

1. Using default QS port:

./launchMultiNodeDemo.sh 2 ../../data/inputGraphs/dns\_graph.alchemy.factors.bin

This line runs 2 nodes for the dns\_graph.alchemy.factors.bin. The query service for this dataset will start at default port.(58000)

1. Using custom QS port:

./launchMultiNodeDemo.sh 2 ../../data/inputGraphs/dns\_graph.alchemy.factors.bin 580003

This line runs 2 nodes for the dns\_graph.alchemy.factors.bin. The query service will start at the port 58000.

Note. The script has another variable to define if the shared states will be stored on /dev/shm or /lfs/ or other file system. The default file system is /lfs/. To change the location of shared states, update the script in the variable $SharedFileSystem.