IGUANA v0.3.0
Manual

December 1, 2015

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Goals . . . 3

2 Architecture **4**
- 2.1 Core . . . 4
- 2.2 LogClusterer . . . 5
  - 2.2.1 FEASIBLE . . . 5
- 2.3 DataGenerator . . . 6
- 2.4 Suite . . . 6
- 2.5 Connection . . . 7
- 2.6 Testcase . . . 7
  - 2.6.1 StressTestcase . . . 8
    - 2.6.1.1 Properties . . . 8
    - 2.6.1.2 Result Metrics . . . 10
  - 2.6.2 FederatedStressTestcase . . . 12
    - 2.6.2.1 Properties . . . 13
    - 2.6.2.2 Result Metrics . . . 13
  - 2.6.3 UploadShellTestcase . . . 13
    - 2.6.3.1 Properties . . . 13
    - 2.6.3.2 Result Metrics . . . 13

3 Configuration **14**
- 3.1 databases . . . 16
- 3.2 logclustering . . . 17
- 3.3 suite . . . 18
  - 3.3.1 randomfunction . . . 19
  - 3.3.2 test-db . . . 19
  - 3.3.3 testcases . . . 20
- 3.4 Example . . . 20
- 3.5 How to start IGUANA? . . . 24

4 Write your own ... **25**
- 4.1 Testcase . . . 25
- 4.2 LogClusterer . . . 27

5 Further information **29**
- 5.1 Links . . . 29
- 5.2 License . . . 29

A LGPL **31**

1 Introduction

BE AWARE THAT THIS MANUAL WILL BE STILL ENHANCED!

In this manual we describe IGUANA - A generic Benchmark Framework for SPARQL and SPARQL UPDATE Endpoints. First we describe why we think IGUANA is necessary in todays world. Then we'll describe the architecture and how you can configure IGUANA. Finally we'll show you how to write your own modules for IGUANA.

1.1 Motivation

Triplestores are the backbone of the semantic web, but to determine which of the stores is the perfect one for your datasets and the user given queries you should benchmark the triplestores which fits to your application. While there are several Benchmarks and some serves an execution platform none of them are generic. We provide with IGUANA a solution which handles SPARQL Queries and SPARQL UPDATES for any SPARQL endpoint supported by the jena remote driver. It doesn't care which dataset you have nor queries nor what you want to test. We provide a mighty execution framework which can handle SPARQL Benchmarks you desire.

1.2 Goals

IGUANA should be...

- ... easy to use
- ... easy to configure
- ... generic
- ... well documented
- ... nearly completely changeable

IGUANA should can ...

- ... use user defined testcases
- ... log clustering
- ... data generation
- ... work with different testcases
- ... test SPARQL and UPDATE workers
- ... upload testcases
- ... Shell scripting

2 Architecture

In this section we'll describe IGUANAs Architecture, it's core, the log clustering function, the data generator and the testcase interface as well as the implemented testcases.

2.1 Core

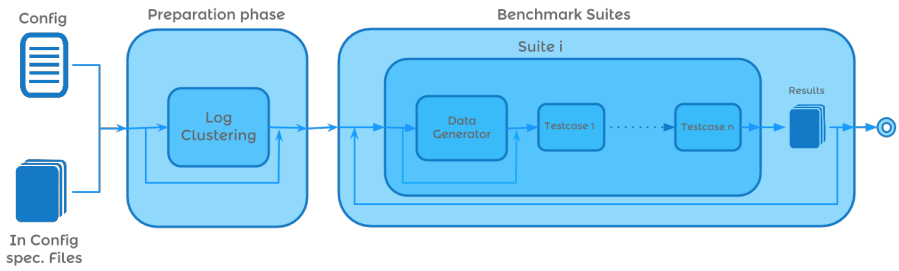

Figure 1: IGUANA Architecture v.0.3.0

As shown in Figure 1 IGUANA starts with a config file and the files/folder specified in this config file. We'll describe later on how the configuration file shoud look. IGUANA will parse the config file and start its preparation phase. In this phase IGUANA gives you the possiblity to cluster a log file to get queries which should be tested further on. This is optional and the user can define the clusterer IGUANA should usel. If the preparation phase finished the Benchmark suites starts to begin. A Benchmark suite is simply one Benchmark (or a part of it) which will test several testcases the user want to test indepedent of other benchmark suites. If the user wants to generate smaller or greated datasets out of a given dataset, IGUANA gives the possibility to generate them with the optional DataGenerator. After all the testcases which where specified in the suite will be tested. First if the user whishes there will be a script executed then the warmup will be started and then the testcase will be tested. If the testcase finished and the user defined a post script this will now be executed. This will be done for every testcase. After the suite finished the results will be saved in a folder called *results_i* while i is the i'th suite.
The testcases will be executed for all specified and in the current suite referenced connections and all specified datasets.
Be aware that you can change the suites in the config file which will follow the last one, while the previous suite is running. Also the testcases can be implemented ones as well as user written ones. If every Benchmark suite is finished IGUANA will be executed and if the user whishes send an email to the user.

2.2 LogClusterer

As previous told IGUANA can handle log clustering. While there is only one *LogClusterer* implemented in IGUANA, called FeasibleClusterer, the user can write an own LogClusterer. The LogClusterer can have several properties specified in the config. It should cluster a given log file to a query file with either query templates or queries itself. The LogCluster itself has 3 attributes: *class* which will tell the LogCluster class IGUANA should use. *path* will tell the LogClusterer where the log file(s) are located. *output-file* will tell the LogClusterer how the output file should be named. As of now you can implement an own LogClusterer if you create a project and implement a new Clusterer which must ...

- ... implement the *de.uni_leipzig.iguana.clusering.clusterer.Clusterer* Interface

Build your project and put the jars (and all the ones you needed which aren't in IGUANA yet) into the lib folder. We'll show you how to configure these later on.

2.2.1 FEASIBLE

As explaining Feasible would blow the workload we link to the hompage to Feasible [http://aksw.org/Projects/FEASIBLE.html] and just tell what parameters you can change in the FeasibleClusterer which will use Feasible.
Nevertheless the *FeasibleClusterer* will produce a query file containing a user specified number of queries which are query instances in one file. If you use this file in the StressTestcaseson it is important that you set the testcase property *is-pattern* to false.
the attribute *path* needs to be a file and not a path!
The class to put in the config is called: *de.uni_leipzig.iguana.clustering.clusterer.FeasibleClusterer*
The parameters are:

name	description	default value	optional
number-of-queries	The number of queries FEASIBLE should generate	-	no
output-dir	The folder in which the files should be saved in	.	yes
draw-voronoi-diagram		true	yes
feature-filter		""	yes
clause-filter		""	yes
ask		false	yes
describe		false	yes
select		false	yes
construct		false	yes
triple-patterns-count		true	yes
result-size		true	yes
join-vertices		true	yes
mean-join-vertices-degree		true	yes
mean-triple-pattern-selectivity		true	yes
bgps		true	yes
union		true	yes
filter		true	yes
optional		true	yes
distinct		true	yes
orderby		true	yes
groupby		true	yes
limit		true	yes
regex		true	yes
offset		true	yes
run-time		true	yes

2.3 DataGenerator

As the *DataGenerator* is currently implemented but not supported (it will be in future work) as it needs too much memory and too much time for even small datasets we decided to not describe how to use it and simply link to the paper how the datagenerator should be if it's finished. [1]

2.4 Suite

A Benchmark suite is simply a container for a *DataGenerator*, choosen *Connections* and several *Testcases*. It should be used to seperate phases of your benchmark. For example every dataset which should be tested should be in one own suite. So you can look at the results of the previous tested datasets while the next dataset is running.

2.5 Connection

A connection is a connection to an endpoint defined in *wookieConnection*. Currently only SPARQL/UPDATE Endpoints can be used. But if the user whishes to also test other databases the possibility to change the *wookieConnection* and implement a new connection there should be no problem to use those connections. But be aware that it can have sideeffects if you use a not SPARQL Connection with the implemented testcases as they are written specific for SPARQL queries. But still it can work. For example if you use SQL Queries (one per line) with the *SteressTestcase* and set *is-pattern* to false it should work without a problem. As of now you can implement an own connection if you create a project and implement a new Connection which must ...

- ... implement the *org.bio_gene.wookie.connection.Connection* Interface
- ... has a Constructor with one Parameter which must be a Map (these are the properties in the config file)

Build your project and put the jars (and all the ones you needed which aren't in IGUANA yet) into the lib folder. We'll show you how to configure these later on.

2.6 Testcase

The testcase interface is designed so the users can define their own testcases as well as use implemented ones. Testcases are a part of a benchmark, while all defined testcases represent the whole benchmark or at least a big part of a benchmark. Testcases needs to implement several functions. These are a simple *start* method, a Method to get the results called getResults(), a Function to add Results which were derived previously for other connections, called addCurrentResults(Collection¡ResultSet¿), further on to set the properties specified in the config setProperties(Properties), the current Connection setConnection(Connection), setting the XML-Node where the connections are specified setConnectionNode(Node, String), set the current connection name setCurrentDBName(String) and the current dataset setCurrentPercent(String) and at last if it's a test which doesn't seperate the results in datasets isOneTest() (this is needed for example in the UploadShellTestcase). As of now you can implement an own Testcases if you create a project and implement a new Testcase which must ...

- ... implement the *de.uni_leipzig.iguana.testcases.Testcase* Interface

Build your project and put the jars (and all the ones you needed which aren't in IGUANA yet) into the lib folder. We'll show you how to configure these later on.

2.6.1 StressTestcase

The StressTestcase is designed to test one Connetion at a time with several SPARQL workers and several UPDATE workers, those will simulate a real testcase scenario where several Users will request the connection with SPARQL queries while in the background some users will update the connections dataset. This testcase should simulate a real world scenario. While the updates will be executed ordered in a row (it is a little bit more complex due to several strategies the user can define, this will be told later on) the SPARQL Queries will be executed a little bit different. If you use query instances every SPARQL Worker will start random (but always at the same query) at one query and will test this query. If it's finished the next query in the line will be tested and so on. If there are no more query instances the worker will begin at the first one and will loop until the testcase is finished. If you use query templates the templates will be converted into several query instances which will be saved in one file per template. The workers will start at a random query template get a random instance of it (of course the workers will have the same order for every connection!) and test it. Then the next query template will be took and the next instance of it will be tested and so on.

2.6.1.1 Properties

name	description	default value	optional
sparql-users	number of sparql workers(can be 0)	-	no
queries-path	path or filename of the sparql queries	-	no
is-pattern	are the sparql queries templates/patterns or real queries	-	no
timelimit	time in ms the stresstest should run	-	no
latency-strategy[0-9]+	Latency strategy between queries/updates	-	yes
latency-amount[0-9]+	only with latency-strategy[0-9]+	-	yes

sparql-users is an integer which tells you how many sparql worker should be used. If this is greater than 0 than you need the*queries-path* which is the filename or path to the sparql queries. *is-pattern* is a boolean which tells iguana if the queries specified in *queries-path* are templates/pattern or query instances. The file needs to have one query per line. Every query will be validated. If a query can't be validated it will not be tested.
Query Templates can have variables which must be structure like %%v([0-9]?)+%%. For example:

```
PREFIX dbo: <http://dbpedia.org/ontology/> ASK FROM <urn:graph> {
     ?x dbo:name %%v1%% ; dbo:country %%v2%%}
```

The variables in the template will be exchanged through variables. For example %%v9%% will be exchanged to ?v9. After this the Prefixes will be cut out and the head will be removed, the template will look as following:

```
{ ?x dbo:name %%v1%% ; dbo:country %%v2%%}
```

Now the prefixes will be added again and a head with SELECT DISTINCT ¡all variables which should be exchaged¿ FROM ¡graph-uri¿ Will be added. The user defined limit (default=5000) will be added/replaced and the final query will look like following:

```
PREFIX dbo: <http://dbpedia.org/ontology/> SELECT DISTINCT ?v1 ?
    v2 FROM <urn:graph> { ?x dbo:name %%v1%% ; dbo:country %%v2
    %%} LIMIT 5000
```

This query will now be requested against the reference connection. And all instances will be saved in one File.
timelimit tells IGUANA how long the testcase should run (in ms). Be aware that the testcase will run longer if the last queries/updates take longer. In other words: the last queries/update will be executed no matter what but if they will be finished after the time limit exceeds they will not effect the results.
If you want to simulate network latency or any kind of latency you can add several latencies to the testcase. These will be defined as *latency-strategy*. Their are 3 *latency-strategy*s. NONE, VARIABLE and FIXED. NONE is as simple as not being there. It is like a placeholder. FIXED will wait a given time for every query. VARIABLE will get a random number out of a given or calculated intervall every time before a query/update will be executed. This number will be wait. Everything is in ms. If there are more than one latencies, the summation of all will be waited. You can add latencies by adding the property latency-strategy0. if you want more than one latencies simply add the property latency-strategy1, latency-strategy2, ... To set an amount or intervall, or let be calculate an intervall simply put for every latency-strategy you defined a *latency-amount* property with either an integer or an intervall as follows: "[n;m]" where n is the lower bound and m the upper bound. If you defined the strategy VARIABLE but simply put an integer in the related latency-amount an intervall will be calculated as follows assumed x is the related latency-amount:

$$[x - \lfloor\sqrt{x}\rceil; x + \lfloor\sqrt{x}\rceil] \tag{1}$$

For every *latency-strategy* you need a *latency-amount*. They are related by the following number. For example: latency-strategy8 is related to latency-amount8.

name	description	default value	optional
update-users	number of update workers(can be 0)	-	no
update-strategy		NONE	yes
update-path		-	no
linking-strategy		DI	yes
worker-strategy[0-9]+		NEXT	yes

while you do not only have sparql-users and general properties there is also *update-users* which basically tells IGUANA how many update workers should be used. While they do need a path with update files you need to assign a *update-path*. This path can be relative.
The files in the update-path should have the following formats. First a 6 digit number which tells the order of the files. Than if it's an insertion file: added or if it's a deletion file: removed. You can have related deletion and insertion files. For example a path can look as following:
00001.added.nt, 000002.added.nt, 000002.removed.nt. The latter two files are related. This will be important for the *linking-strategy*.
Further on you can define an *update-strategy* which is either NONE, FIXED or VARIABLE. Exactly the same as latency-strategy with one difference. The update-strategy uses the following equation as x:

$$x = \frac{timelimit}{|updatepath|} \tag{2}$$

You can't define an own amount.
Further on you can define a linking-strategy which can be DI, ID, D or I. D means it will first update every deletion files and then every insertion files. I means it will first update every insertion files than every deletion files, DI will update first deletion than its related insertion file. Sorted by their numbers. ID will do the same other wise around.
The *worker-strategy[0-9]+* tells every worker (worker0 has the worker strategy worker-strategy0 and so on) which files it should upload. Either it'll only update ADDED or REMOVED or it'll simply take the NEXT file it can get which is still not updated.

2.6.1.2 Result Metrics

It will produce several results, with the metrics failed queries, succeeded queries, Queries per second, Total time of queries, Query Mixes per TimeLimit for every worker and also their SPARQL, as well as UPDATE summation, as their means. Queries Per Second will measure for every query how many queries per second the connection can handle. Total time of Queries will sum up for every query how much time all request of this query took. Query Mixes per TimeLimit will measure how many queries the connection can handle in the given timelimit. Failed and succeeded queries measures for every query the number of succeeded and failed queries. Every result will be saved as csv files. **How should i read the csv file?** Usually the header (first line) tells you the query number and the following lines are the results of each connection (one connection per line). The cell tells you the result for the query/mix for the connection. The cell for the query 10 with the metric queries total time tells you how many ms every request of query 10 the connection needed.

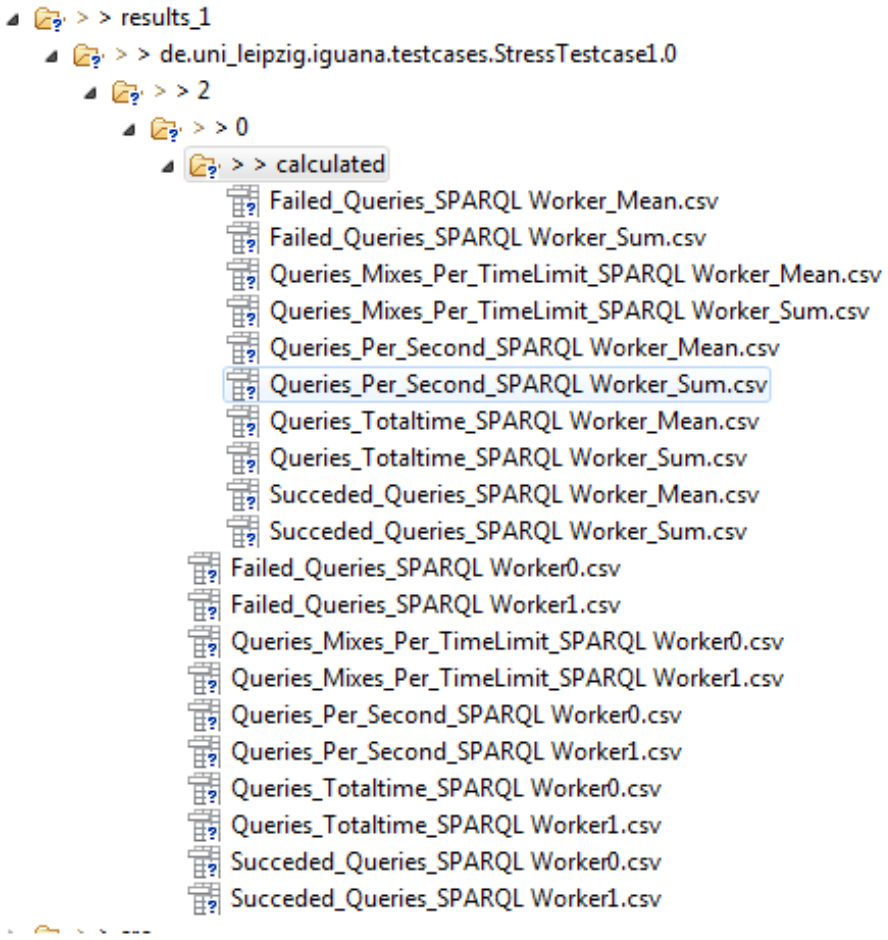

Figure 2: StressTest result folder overview

```
1 Connection;0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19
2 fuseki;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
3 virtuoso_4;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
4 blazegraph;0;0;0;0;0;0;0;1;0;0;0;0;1;0;0;0;0;1;0;0;0;0;0;0
5
```

Figure 3: Failed Queries

```
Connection;0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19
fuseki;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1
virtuoso_4;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
blazegraph;1;1;0;1;0;1;0;0;0;0;1;0;0;0;0;0;0;0;1;1;1;0;1;1;1
```

Figure 4: Succeeded Queries

```
Connection;0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19
fuseki;43;157;51;61;58;94;88;136;43;82;89;32;189;75;155;45;3
virtuoso_4;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
blazegraph;271;77;0;161;0;394;0;0;0;0;66;0;0;0;0;0;0;0;375;4
```

Figure 5: Queries Total Time

```
Connection;Mix
fuseki;180
virtuoso_4;6
blazegraph;62
```

Figure 6: Query Mixes per TimeLimit

```
Connection;0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19
fuseki;2;7;3;2;4;2;6;4;2;4;4;1;9;3;7;2;1;3;12;4;4;12;3;9;3;2
virtuoso_4;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
blazegraph;13;3;8;0;19;0;0;0;0;3;0;0;0;0;0;0;0;0;18;21;9;0;5
```

Figure 7: Queries Per Second

2.6.2 FederatedStressTestcase

Same as StressTestcase but the update workers can work on different connections, so a Federated Connection can be tested without any problem. The actual endpoint will be the QueryEngine in front of the federated system and the update worker will get the connections behind the federated system. For example if a federated system has localhost:8080/sparql as the sparql endpoint for all conections and a virtuoso and a blazegraph connection is behind this federated system the update workers need to know into which of them they should upload. This you can specify

2.6.2.1 Properties

Nearly as same as StressTestcase but with one more option and one difference

name	description	default value	optional
worker[0-9]+	worker0 tells the connection id into which the update worker0 needs to upload to	-	no
update-path[0-9]+	Tells the worker0 which update-path is assigned to the worker	-	no

Be aware that the update-path[0-9]+ must be the same path for every worker with the same connection id. For example a worker with the connection id "fuseki" and a different worker also with the connection id "fuseki" needs to have the same update-path!

2.6.2.2 Result Metrics

See the result metrics at StressTestcase with one difference. The summation and mean of the UPDATE workers will be seperated. It will get the summation and the mean for every different connection. Every result will be saved as csv files.

2.6.3 UploadShellTestcase

This testcase is there to give you the option to upload several datasets to the current connection via shell script. The script can have several arguments called "%%PERCENT%%", "%%DBNAME%%", "%%FILE%%" which will be replaced by the current percentage, the current connection name and the current file/path to upload to.
These arguments needs to stand in the script-name argument. For example: script-name="./upload.sh %%DBNAME%% %%FILE%%"

2.6.3.1 Properties

name	description	default value	optional
file	Tells the file which should be uploaded	-	no
script-name	tells the name of the script which should be executed	-	no

2.6.3.2 Result Metrics

This will produce only one resultset for the whole suite. It'll measure the time the shell script took for each percent and each connection. Every result will be saved as csv files. The one csv file which will be resulting has as the header the dataset percentage and for each line the connection and the time in ms it took to execute the shell script.

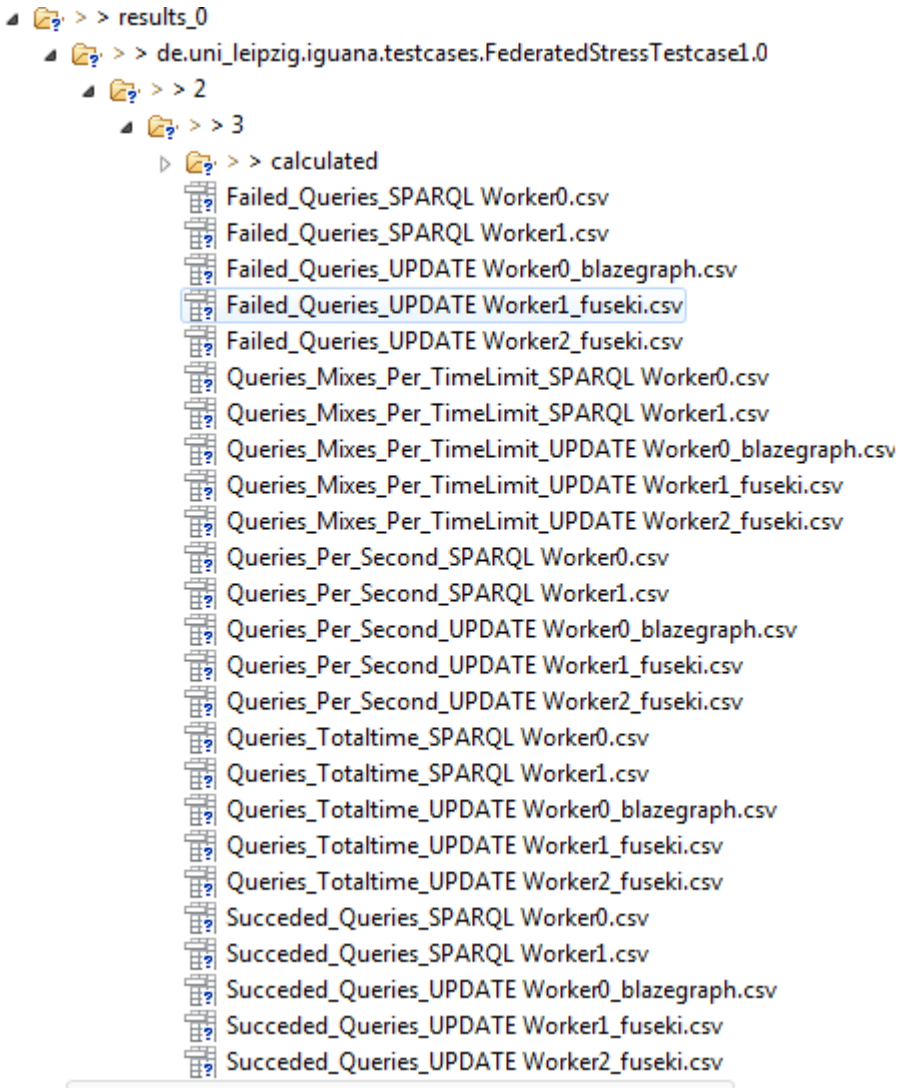

Figure 8: FederatedStressTest result folder overview

3 Configuration

In this section we'll describe what you can configure in IGUANA and after this an examples The root element looks like follow:

```
<?xml version="1.0" encoding="UTF-8"?>
<iguana xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
```

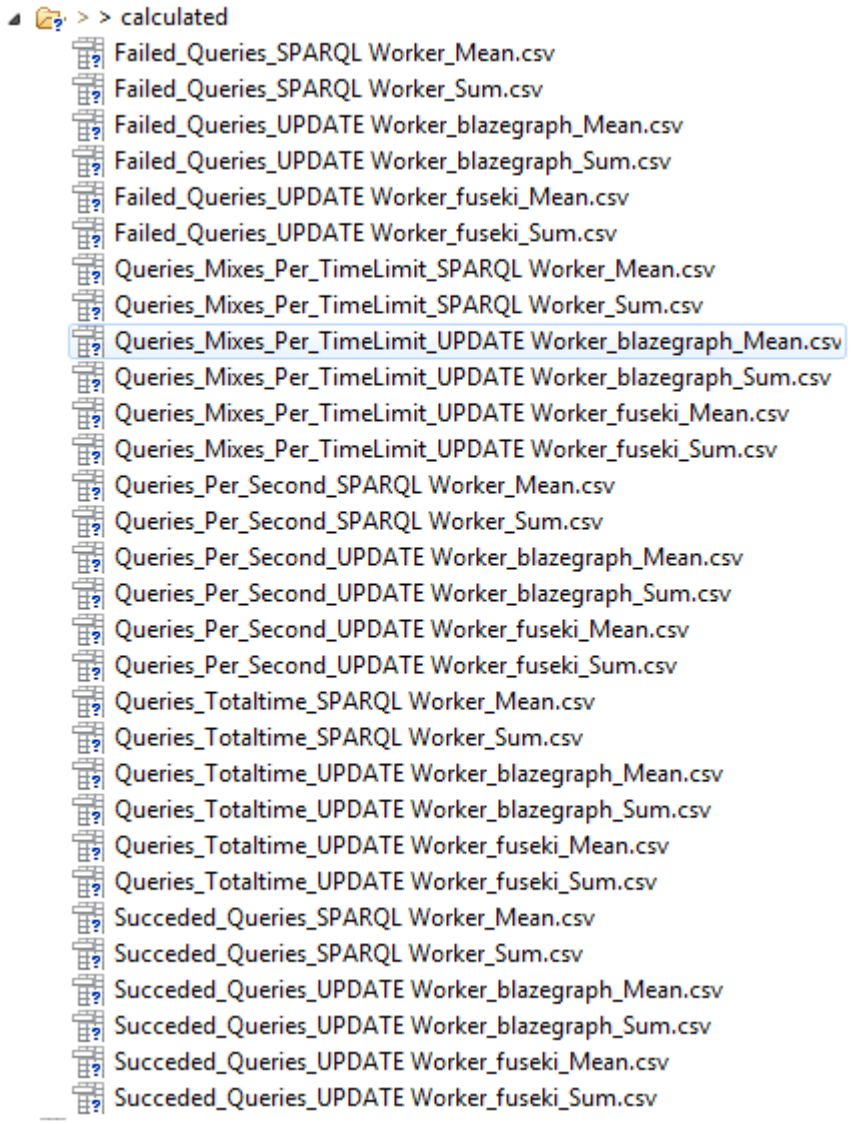

Figure 9: FederatedStressTest result folder overview

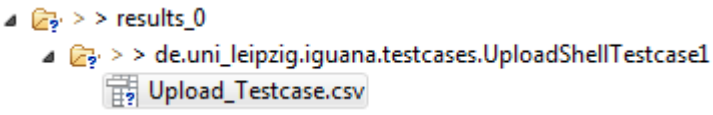

Figure 10: Upload result folder overview

```
Connection;1.0;2.0;0.5;0.2
blazegraph;190;55;60;38
virtuoso_4;131;67;75;112
fuseki;52;93;60;36

```

Figure 11: Results of UploadShellTestcase

```
</iguana>
```

3.1 databases

The databases element consists out of several database elements. It must exist 1 and only 1 time in the config
The database elements consist of two attributes, the id and the type. The type should always be *impl* It consists out of a required endpoint element and optional update endpoint, user and password. The parent is the root tag.

```
...
<!-- mininmal occurs: 1 -->
<!-- maximal occurs: 1 -->
<databases main="">
       <!-- mininmal occurs: 1 -->
       <!-- maximal occurs: n -->
       <database id="sparqlendpoint1" type="impl">
              <!-- required -->
              <endpoint uri="http://localhost:8080/sparql" />
              <!-- optional -->
              <update-endpoint uri="http://localhost:8080/update"
                   />
              <!-- optional -->
              <user value="dba" />
              <!-- optional -->
              <pwd value="dba" />
       </database>
       ...
<databases>
...
```

If you want to user your own implemented Connection you can do this like following:

```
...
       <database id="sparqlendpoint1" type="lib">

       //IMPORTANT: the type must be "lib" !
```

```
        <database id="id1" type="lib">
                /* Now you can specify any
                 * Parameters you want.
                 * Important is that there needs to be the
                     attribute value
                 * Every other attribute will be ignored!
                 */
                <parameter1 value="asdasd"/>
                ...
                <par2232 value="Whatever you whish"/>
                ...
                //Nevertheless you need th following element!
                <class value="class.which.is.your.connection.class"
                    />
        </database>
...
```

The implemented Connection will then get a Map with the element name as key (for example "parameter1") and the value as the value (for example "asdasd"). The class element will tell IGUANA which class it should load.This is the most important thing which must be correct!
Be aware that you should not use the following element names: *connectionType*

3.2 logclustering

The LogClustering element needs 3 attributes, the class name, the path or the name of the log file and the name of the output queries file. Also it consists out of several properties with the attributes name and value. the parent is the root tag

```
...
        <!-- mininmal occurs: 1 -->
        <!-- maximal occurs: 1 -->
        <log-clustering class="org.example.clusterer.Clusterer"
           path="logFile" output-file="queries.txt">
                <property name="ask" value="false"/>
                <property name="union" value="false"/>
                <property name="clause-filter"
                        value="(OPTIONAL AND DISTINCT) OR (UNION)"/>
                <property name="feature-filter"
                        value="ResultSize &gt;= 5 AND ResultSize
                          &lt;= 100 AND BGPs &gt;= 2 AND
                          TriplePatterns &lt;=10"/>
        </log-clustering>
...
```

3.3 suite

The *suite*s elements are the elements who contains all the information for your specific benchmark. You have several elements in the suite element. The number-of-triples element tells IGUANA if files which will be uploaded and removed should be splitted into files with the specified number of triples in it. These will be uploaded seperated. Be aware that IGUANA will only measure the time of each file to upload/remove and not the splitting etc. also. The graph-uri element tells IGUANA in which graph datasets should be loaded in. The warmup Tag tells you how many minutes the warmup should take and the file name with the SPARQL Queries to warmup the connection. You can also specify updates in the warmup element
Further on you have the test-db tag which will be explained later on as well as the testcases tag which will be also explained later on.
the parent is the root tag

```
...
	<!-- mininmal occurs: 1 -->
	<!-- maximal occurs: n -->
	<suite>
		<!-- mininmal occurs: 0 -->
		<!-- maximal occurs: 1 -->
		<number-of-triples value="9"/>
		<!-- mininmal occurs: 1 -->
		<!-- maximal occurs: 1 -->
		<graph-uri name="http://dbpedia.org" />
		<!-- mininmal occurs: 1 -->
		<!-- maximal occurs: 1 -->
		<warmup time="5" file-name="warmup.txt" update-path
		    ="optional"/>
		<!-- mininmal occurs: 1 -->
		<!-- maximal occurs: 1 -->
		<random-function ...>
			...
		</random-function>
		<!-- mininmal occurs: 1 -->
		<!-- maximal occurs: 1 -->
		<test-db ...>
			...
		</test-db>
		<!-- mininmal occurs: 1 -->
		<!-- maximal occurs: 1 -->
		<testcases>
			...
		</testcases>
	</suite>
...
```

3.3.1 randomfunction

As the random functionas attribute type and generate are specified for the DataGenerator we simply tell you following: Let type be RandomTriple and generate be false and everything is okay. The containing elements percent will tell IGUANA which percantage size (as double, 100%=1.0) the dataset with the file-name has. If you use the UploadShellTestcase you should tell in the file-name the real filename. Otherwise you can simply put something in it, it will be ignored.
the parent is the suite tag

```
...
       <!-- mininmal occurs: 1 -->
       <!-- maximal occurs: 1 -->
       <random-function type="RandomTriple" generate="false">
              <!-- mininmal occurs: 1 -->
              <!-- maximal occurs: n -->
              <percent value="1.0" file-name="dbpedia2/" />
              ...
       </random-function>
...
```

3.3.2 test-db

The test-db element tells you firstly with the type if only *choose*n connections of the defined connections should be tested or *all*. Be aware that IGUANA will remove the *reference* connection out of the test! Even if you'll define it as choose. If you still want to test the connection which is the reference connection you can simply add a new defined connection which is exactly the same connection but with a different id. The reference connection is declared with the id in the defined connections.
The containing element db has simply the id of the connection which should be tested if the type is *choose*.
the parent is the suite tag

```
...
       <!-- mininmal occurs: 1 -->
       <!-- maximal occurs: 1 -->
       <test-db type="choose" reference="ref">
              <!-- mininmal occurs: 1 -->
              <!-- maximal occurs: n -->
              <db id="owlim" />
              ...
       </test-db>
```

```
...
```

3.3.3 testcases

The testcases element has 2 attributes called testcase-pre and testcase-post which are optional and can declare shell scripts which will be executed before and after the whole testcases incl warmups. The shell scripts can have following parameters which will be exchanged automatically: .%DBID%, %PERCENT% and %TESTCASEID%. The latter one tells the number of the order of the testcase. The testcases element contains several testcase elements which has as an attribute the class of the testcase which should be tested. The testcase element contains several element called property which have name and value as attributes.
the parent is the suite tag

```
...
       <!-- mininmal occurs: 1 -->
       <!-- maximal occurs: 1 -->
       <testcases testcase-pre="pre.sh" testcase-post="post.sh">
              ...
              <!-- mininmal occurs: 1 -->
              <!-- maximal occurs: n -->
              <testcase class="de.uni_leipzig.iguana.testcases.
                  StressTestcase">
                     <property name="xyz"
                            value="abc"/>
                     ...
              </testcase>
              ...
       </testcases>
...
```

3.4 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<iguana xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <databases main="dbpedia">
    <database id="owlim" type="impl">
      <endpoint
        uri="localhost:8080/openrdf-workbench/repositories/owlim-
            lite/query" />
      <update-endpoint
        uri="localhost:8080/openrdf-workbench/repositories/owlim-
            lite/update" />
    </database>
```

```
  <database id="fuseki" type="impl">
    <endpoint uri="localhost:3030/tdb/sparql" />
    <update-endpoint uri="localhost:3030/tdb/update" />
  </database>
  <database id="blazegraph" type="impl">
    <endpoint uri="localhost:9999/bigdata/sparql" />
  </database>
  <database id="virtuoso" type="impl">
    <endpoint uri="localhost:8890/sparql-auth" />
    <user value="dba" />
    <pwd value="dba" />
  </database>
  <database id="ref" type="impl">
    <endpoint uri="dbpedia.org/sparql" />
  </database>
</databases>

<suite>
  <drop-db value="false" />
  <query-diversity value="2000" />
  <graph-uri name="http://dbpedia.org" />
  <random-function type="RandomTriple" generate="false">
    <percent value="1.0" file-name="dbpedia2/" />
  </random-function>
  <warmup time="20" file-name="warmup.txt" update-path="updates/
      "/>
  <test-db type="choose" reference="ref">
    <db id="owlim" />
    <db id="blazegraph" />
    <db id="virtuoso" />
    <db id="fuseki" />
  </test-db>
  <testcases testcase-pre="./testcasePre.sh %DBID% %PERCENT% %
      TESTCASEID%"
    testcase-post="./testcasePost.sh %DBID% %PERCENT% %
        TESTCASEID%">
    <testcase class="de.uni_leipzig.iguana.testcases.
        StressTestcase">
      <property name="sparql-user" value="1" />
      <property name="update-user" value="0" />
      <property name="latency-amount0" value="20" />
      <property name="latency-strategy0" value="VARIABLE" />
      <property name="queries-path" value="queries-175.txt" />
      <property name="is-pattern" value="false" />
      <property name="timelimit" value="3600000" />
    </testcase>
```

```
<testcase class="de.uni_leipzig.iguana.testcases.
    StressTestcase">
  <property name="sparql-user" value="2" />
  <property name="update-user" value="0" />
  <property name="latency-amount0" value="20" />
  <property name="latency-strategy0" value="VARIABLE" />
  <property name="queries-path" value="queries-175.txt" />
  <property name="is-pattern" value="false" />
  <property name="timelimit" value="3600000" />
</testcase>
<testcase class="de.uni_leipzig.iguana.testcases.
    StressTestcase">
  <property name="sparql-user" value="4" />
  <property name="update-user" value="0" />
  <property name="latency-amount0" value="20" />
  <property name="latency-strategy0" value="VARIABLE" />
  <property name="queries-path" value="queries-175.txt" />
  <property name="is-pattern" value="false" />
  <property name="timelimit" value="3600000" />
</testcase>
<testcase class="de.uni_leipzig.iguana.testcases.
    StressTestcase">
  <property name="sparql-user" value="8" />
  <property name="update-user" value="0" />
  <property name="latency-amount0" value="20" />
  <property name="latency-strategy0" value="VARIABLE" />
  <property name="queries-path" value="queries-175.txt" />
  <property name="is-pattern" value="false" />
  <property name="timelimit" value="3600000" />
</testcase>
<testcase class="de.uni_leipzig.iguana.testcases.
    StressTestcase">
  <property name="sparql-user" value="16" />
  <property name="update-user" value="0" />
  <property name="latency-amount0" value="20" />
  <property name="latency-strategy0" value="VARIABLE" />
  <property name="queries-path" value="queries-175.txt" />
  <property name="is-pattern" value="false" />
  <property name="timelimit" value="3600000" />
</testcase>
<testcase class="de.uni_leipzig.iguana.testcases.
    StressTestcase">
  <property name="sparql-user" value="1" />
  <property name="update-user" value="1" />
  <property name="latency-amount0" value="20" />
  <property name="latency-strategy0" value="VARIABLE" />
```

```
    <property name="queries-path" value="queries-175.txt" />
    <property name="is-pattern" value="false" />
    <property name="linking-strategy" value="ID" />
    <property name="update-path" value="ld" />
    <property name="worker-strategy0" value="ADDED" />
    <property name="update-strategy" value="VARIABLE" />
    <property name="timelimit" value="3600000" />
  </testcase>
  <testcase class="de.uni_leipzig.iguana.testcases.
      StressTestcase">
    <property name="sparql-user" value="2" />
    <property name="update-user" value="1" />
    <property name="latency-amount0" value="20" />
    <property name="latency-strategy0" value="VARIABLE" />
    <property name="queries-path" value="queries-175.txt" />
    <property name="is-pattern" value="false" />
    <property name="linking-strategy" value="ID" />
    <property name="update-path" value="ld" />
    <property name="worker-strategy0" value="ADDED" />
    <property name="update-strategy" value="VARIABLE" />
    <property name="timelimit" value="3600000" />
  </testcase>
  <testcase class="de.uni_leipzig.iguana.testcases.
      StressTestcase">
    <property name="sparql-user" value="4" />
    <property name="update-user" value="1" />
    <property name="latency-amount0" value="20" />
    <property name="latency-strategy0" value="VARIABLE" />
    <property name="queries-path" value="queries-175.txt" />
    <property name="is-pattern" value="false" />
    <property name="linking-strategy" value="ID" />
    <property name="update-path" value="ld" />
    <property name="worker-strategy0" value="ADDED" />
    <property name="update-strategy" value="VARIABLE" />
    <property name="timelimit" value="3600000" />
  </testcase>
  <testcase class="de.uni_leipzig.iguana.testcases.
      StressTestcase">
    <property name="sparql-user" value="8" />
    <property name="update-user" value="1" />
    <property name="latency-amount0" value="20" />
    <property name="latency-strategy0" value="VARIABLE" />
    <property name="queries-path" value="queries-175.txt" />
    <property name="is-pattern" value="false" />
    <property name="linking-strategy" value="ID" />
    <property name="update-path" value="ld" />
```

```
      <property name="worker-strategy0" value="ADDED" />
      <property name="update-strategy" value="VARIABLE" />
      <property name="timelimit" value="3600000" />
    </testcase>
    <testcase class="de.uni_leipzig.iguana.testcases.
        StressTestcase">
      <property name="sparql-user" value="16" />
      <property name="update-user" value="1" />
      <property name="latency-amount0" value="20" />
      <property name="latency-strategy0" value="VARIABLE" />
      <property name="queries-path" value="queries-175.txt" />
      <property name="is-pattern" value="false" />
      <property name="linking-strategy" value="ID" />
      <property name="update-path" value="ld" />
      <property name="worker-strategy0" value="ADDED" />
      <property name="update-strategy" value="VARIABLE" />
      <property name="timelimit" value="3600000" />
    </testcase>
  </testcases>
 </suite>
</iguana>
```

3.5 How to start IGUANA?

To start IGUANA you can either use the start scripts provided or write following in the commandline:

```
java -cp "lib/*" de.uni_leipzig.iguana.benchmark.Main config.xml
```

This is assuming you're in the folder with the extraced lib folder and your config file is called config.xml. If you want to have more information about the benchmark (be aware that this can lead to biiiiiiig files (400G log files) !!!) you can set a parameter debug

```
java -cp "lib/*" de.uni_leipzig.iguana.benchmark.Main config.xml
    debug=true
```

Have fun! :)

4 Write your own ...

In this section we'll describe how you can write additional Modules in IGUANA without changing the code itself.

4.1 Testcase

We'll show you how you can write your own Testcase. The following testcase will test one in the properties given query 1000 times and measures the time. It will calculate the mean of the time of the queries as well as the total time.

```
import org.bio_gene.wookie.connection.Connection;
import de.uni_leipzig.iguana.utils.ResultSet;

public class OneQueryThousandTimesTestcase implements Testcase{

  private Collection<ResultSet> results = new LinkedList<ResultSet>();
  private String conName;
  private Connection con;
  private String percent;
  private String query;
  private int count = 1000;

  public void start() throws IOException{
    long time=0L;
    for(int i=0;i<count;i++){
      time+=con.selectTime(query);
    }
    String[] prefixes = new String[4];
    prefixes[0] = "This";
    prefixes[1] = "will";
    prefixes[2] = "be";
    prefixes[3] = "folders";
    ResultSet total = new ResultSet();
    ResultSet mean = new ResultSet();
    total.setFileName("Total_time_of_query");
    total.setTitle("Total time of query");
    total.setyAxis("time in ms");
    total.setxAxis("query");
    total.setPrefixes(prefixes);
    mean.setFileName("Mean_time_of_query");
    mean.setTitle("Mean time of query");
    mean.setyAxis("time in ms");
    mean.setxAxis("query");
    mean.setPrefixes(prefixes);

    List<String> header = new LinkedList<String>();
    header.add("Connection");
    header.add("query");
    total.setHeader(header);
    mean.setHeader(header);

    List<Object> rowT = new LinkedList<Object>();
    rowT.add(this.conName);
```

```
    rowT.add(time);
    total.addRow(rowT);
    List<Object> rowM = new LinkedList<Object>();
    rowM.add(this.conName);
    rowM.add(time/1000.0);
    mean.addRow(rowM);
    if(results.size()==0){
      results.add(mean);
      results.add(total);
    }
    else{
      Collection<ResultSet> col = new LinkedList<ResultSet>();
      col.add(mean);
      col.add(total);
      addCurrentResults(col);
    }
  }

  public Collection<ResultSet> getResults(){
    return this.results;
  }

  public void addCurrentResults(Collection<ResultSet> currentResults){
    Iterator<ResultSet> it = results.iterator();
    for(ResultSet res : currentResults){
      ResultSet res2 = it.next();
      while(res.hasNext()){
        res2.addRow(res.next());
      }
    }
  }

  public void setProperties(Properties p){
    String query = p.getProperty("query");
  }

  public void setConnection(Connection con)[
    this.con = con;
  }

  public void setCurrentDBName(String name){
    this.conName = name;
  }

  public void setCurrentPercent(String percent){
    this.percent = percent;
  }

  public Boolean isOneTest(){
    return false;
  }
}
```

4.2 LogClusterer

We'll show you how you can write your own Log Clusterer with the example of BorderFlow [http://borderflow.sourceforge.net/]

```
public class BorderFlowClusterer implements Clusterer{

  private String harden;
  private Double threshold;
  private Boolean testOne;
  private Boolean heuristic;
  private Boolean caching;
  private Integer minNodes;

  @Override
  public String cluster(String logPath, String queriesFile){

      String clusterOutput = "cluster.log";

      String sortedFreqFile = getSortedFrequence(logPath);
      String simFile = getSimilarities(sortedFreqFile);

      borderFlow(harden,
        threshold,
        testOne,
        heuristic,
        caching,
        minNodes,
        sortedFreqFile,
        simFile,
        clusterOutput,
        queriesFile);

      return queresFile;
  }

  @Override
  public void setProperties(Properties p){
    harden = p.getProperty("harden");
    threshold = Double.valueOf(p.getProperty("threshold"));
    testOne = Boolean.valueOf(p.getProperty("test-one"));
    heuristic = Boolean.valueOf(p.getProperty("heuristic"));
    caching = Boolean.valueOf(p.getProperty("caching"));
    minNodes = Integer.valueOf(p.getProperty("min-nodes"));
  }

  private void borderFlow(String clusterHarden, double connThreshold,
        boolean testOne, boolean heuristic,
        boolean caching, Integer minNodes,
        String inputQueries, String input,
        String clusterOutput, String output)
        throws IOException{

    //This is the main class of the borderflow jar
    Main.borderFlowDemo(input,
          clusterOutput,
          connThreshold,
```

```
        testOne,
        heuristic,
        caching,
        HardenStrategy.valueOf(clusterHarden));
    rankAndChoose(inputQueries, clusterOutput, output, minNodes);
  }

  private void rankAndChoose(String input, String cluster,
        String output, String minNodes){

    /*
     * DO: rank the given input queries with their cluster bigger
     * than minNodes as you wish and write the choosen final
     * queries in the output file.
     */

  }

  private String getSortedFrequence(String logPath){

    /*
     * DO: Calculate Frequences of the queries in the logPath
     *  Then sort them after their frequence
     */

  }

  private String getSimilarities(String sortedFreqFile){

    /*
     * DO: Calculate Similarity between the given queries
     *  in the sortedFreqFile
     */

  }

}
```

5 Further information

5.1 Links

Beware that these following information can be outdated

website	https://aksw.github.io/IGUANA/
distribution	https://github.com/AKSW/IGUANA/tree/master
javadoc	https://aksw.github.io/IGUANA/javadoc
source code	https://github.com/AKSW/IGUANA
issue tracker	https://github.com/AKSW/IGUANA/issues
contact information	mai12cpg@studserv.uni-leipzig.de
Border Flow (Clustering algorithm)	http://borderflow.sourceforge.net
DBpedia SPARQL Benchmark	http://aksw.org/Projects/DBPSB.html
adjusted wookieConnection	https://github.com/AKSW/IGUANA/blob/develop/de.uni_leipzig.iguana/src/main/resources/lib/connection-0.0.1-SNAPSHOT.jar

5.2 License

IGUANA itself is licensed under LGPL [A], but be aware that there are libraries we use which are licensed under different licenses.

References

[1] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: A comparison of rdf benchmarks and real rdf datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 145–156, New York, NY, USA, 2011. ACM.

A LGPL

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license doc-

ument.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions

will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.