Benchmarking RDF Storage Solutions with Iguana

Felix Conrads[1,4], Jens Lehmann[2], Muhammad Saleem[1,4], and Axel-Cyrille Ngonga Ngomo[1,4]

[1] University of Leipzig, AKSW, Germany
email: {lastname}@informatik.uni-leipzig.de
[2] University of Bonn and Fraunhofer IAIS
email: jens.lehmann@cs.uni-bonn.de, jens.lehmann@iais.fraunhofer.de
[3] System and Network Engineering Group, University of Amsterdam
email: m.morsey@uva.nl
[4] Department of Computer Science, University of Paderborn

Abstract. Choosing the right RDF storage storage is of central importance when developing any data-driven Semantic Web solution. In this demonstration paper, we present the configuration and use of the Iguana benchmarking framework. This framework addresses a crucial drawback of state-of-the-art benchmarks: While several benchmarks have been proposed that assess the performance of triple stores, an integrated benchmark-independent execution framework for these benchmarks was yet to be made available. Iguana addresses this research by providing an integrated and highly configurable environment for the execution of SPARQL benchmarks. Our framework complements benchmarks by providing an execution environment which can measure the performance of triple stores during data loading, data updates as well as under different loads and parallel requests. Moreover, it allows a uniform comparison of results on different benchmarks. During the demonstration, we will execute the DBPSB benchmark using the Iguana framework and show how our framework measures the performance of popular triple stores under updates and parallel user requests. Iguana is open-source and can be found at http://iguana-benchmark.eu/.

Keywords: Benchmarking, Triple Stores, SPARQL, RDF, Log Analysis

1 Introduction

The number of size of datasets available as RDF grows constantly.[5] Consequently, many applications that consume and manipulate RDF data rely on storage solutions to manage the data they need to manipulate. Iguana (Integrated Suite for Benchmarking SPARQL) [1][6] is a robust benchmarking framework designed for the execution of SPARQL benchmarks. Our framework is complementary to the large number of benchmarks already available (e.g., [3,5]) in that it provides an environment within which SPARQL benchmarks can be executed so provide comparable results. Therewith, Iguana is (1) able to pinpoint the strengths and weaknesses of the triple store under test. This in turn allows,

[5] http://lodstats.aksw.org
[6] Main paper to be presented at ISWC 2017

(2) the evaluation of the suitability of a specific triple store to the application under which it is to operate, and the proposal of the best candidate triple stores for that application. In addition, benchmarking triple stores can also help (3) identifying the best running conditions for each triple store (e.g., the best memory configuration) as well as (4) providing developers with insights pertaining to how to improve their frameworks.

The methodology implemented by IGUANA follows the four key requirements for domain-specific benchmarks that are postulated in the Benchmark Handbook [2], i.e., it is (1) *relevant*, as it allows the testing of typical operations within the specific domain, (2) *portable* as it can be executed on different platforms and using different benchmarks and datasets, (3) *scalable* as it is possible to run benchmarks on both small and large data sets with variable rates of updates and concurrent users and (4) *understandable* as it returns results using standard measures that have been used across literature for more than a decade. Links to all information pertaining to IGUANA (including its source code, GPLv3) can be found at `http://iguana-benchmark.eu`. A guide on how to get started is at `http://iguana-benchmark.eu/gettingstarted.html`.

2 The IGUANA Framework

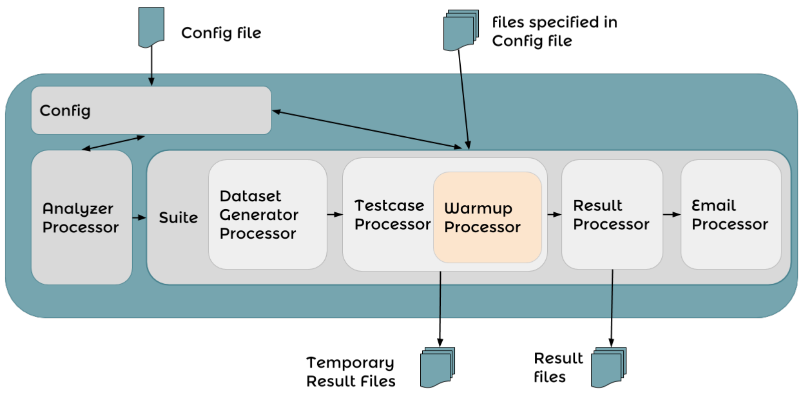


Fig. 1: Overview of the IGUANA benchmarking components.

Figure 1 shows the core components of the IGUANA framework. The input for the framework is a *configuration file* (see Example in Listing 1.1), which contains (1) a description of the storage solutionn to benchmark, (2) instructions that orchestrate how the benchmark queries are to be generated, processed and issued as well as (3) a specification of the benchmarking process and (4) the external data sources to be used during this process. In our example, the configuration file describes two storage solutions which are to be benchmarked: OWLIM (see Lines 4–7) and Fuseki (Lines 8–11). A benchmarking suite describes the points (2) to (4) aforementioned. The DBpedia knowledge base is to be used for the evaluation (see Line 16). No data is to be dropped before the evaluation (ergo, we assume and make sure that the triple stores are empty before the beginning of the benchmarking process). The benchmark in itself is a stress test of the two triple stores OWLIM and Fuseki (see Line 22). Here, 16 query users and 4 update users emulate working in parallel on the underlying triple store.

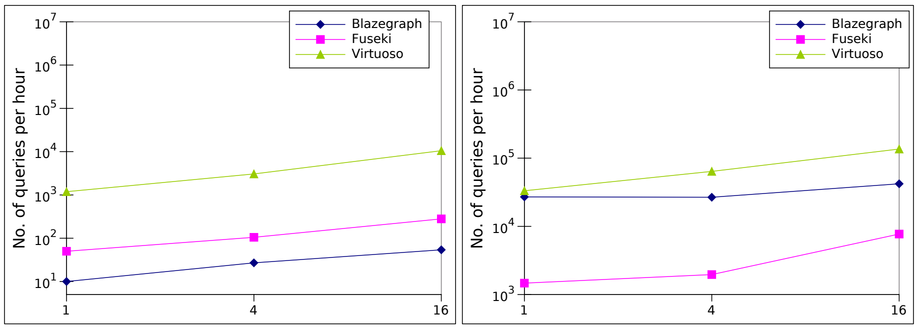
A representation of the parsed config file is stored internally in a *configuration object* (short: config object). If the config object points to a query log, then the *analyzer processor* analyzes this query log file and generates benchmark queries (e.g., using the FEASIBLE [5] approach). Iguana also supports the benchmark queries being directly provided to the framework. The *dataset generator process* creates a fraction (e.g., 10% of DBpedia) of dataset, thus enabling it to test the scalability of the triple stores with varying sizes of the same dataset. Note that this generator is an interface which can be used to integrate data generators (e.g., the DBPSB [4] generator) that can create datasets of varying size. The *warmup processor* allows the execution of a set of test queries before the start of the actual stress testing. The *testcase processor* then performs the benchmarking by means of stress tests according to the parameters specified in the config file. Finally, the *result processor* generates the results which can be emailed by the *email processor*. Details on the core components of Iguana can be found in [1].

Listing 1.1: Example configuration for Iguana

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <iguana xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3  <databases main="dbpedia">
4   <database id="owlim" type="impl">
5    <endpoint uri="http://localhost:8080/openrdf/lite/query"/>
6    <update-endpoint uri="http://localhost:8080/openrdf/lite/up"/>
7   </database>
8   <database id="fuseki" type="impl">
9    <endpoint uri="http://localhost:3030/tdb/sparql" />
10   <update-endpoint uri="http://localhost:3030/tdb/update" />
11  </database>
12  <database id="ref" type="impl">
13   <endpoint uri="http://dbpedia.org/sparql" />
14  </database></databases>
15 <suite>
16  <graph-uri name="http://dbpedia.org" />
17  <random-function type="RandomTriple" generate="false">
18   <percent value="1.0" file-name="dbpedia2/" />
19  </random-function>
20  <warmup time="20" file-name="warmup.txt" />
21  <test-db type="choose" reference="ref">
22   <db id="owlim" /> <db id="fuseki" />
23  </test-db>
24  <testcases testcase-pre="./testcasePre.sh %DBID% %PERCENT% %TESTCASEID%"
25        testcase-post="./testcasePost.sh %DBID% %PERCENT% %TESTCASEID%">
26   <testcase class="org.aksw.iguana.testcases.StressTestcase">
27    <property name="sparql-user" value="16" />
28    <property name="update-user" value="4" />
```

```
    <property name="latency-amount0" value="20" />
    <property name="latency-strategy0" value="VARIABLE" />
    <property name="queries-path" value="qs.txt" />
    <property name="is-pattern" value="true" />
    <property name="timelimit" value="3600000" />
   </testcase> </testcases> </suite>
</iguana>
```

During the demo, we will show how to build and use a configuration akin to that in Listing 1.1. Typical results generated by Iguana based on such configurations are shown in Figure 2. The framework allows the evaluation of triple stores across different datasets and user types and configuration. For example, it shows clearly that the performance of modern storage solutions increases with the number of workers as the systems make good use of parallel processing.



(a) DBpedia (b) SWDF

Fig. 2: Number of queries per hour achieved by three triple stores on two different datasets. The x-axis shows the number of query users.

References

1. F. Conrads, J. Lehmann, M. Saleem, M. Morsey, and A.-C. Ngonga Ngomo. IGUANA: A generic framework for benchmarking the read-write performance of triple stores. In *Proceedings of International Semantic Web Conference (ISWC 2017)*, 2017.
2. J. Gray, editor. *The Benchmark Handbook for Database and Transaction Systems (1st Edition)*. Morgan Kaufmann, 1991.
3. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *International Semantic Web Conference (ISWC)*, 2011.
4. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. Usage-Centric Benchmarking of RDF Triple Stores. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI 2012)*, 2012.
5. M. Saleem, Q. Mehmood, and A.-C. Ngonga Ngomo. FEASIBLE: A featured-based sparql benchmark generation framework. In *International Semantic Web Conference (ISWC)*, 2015.