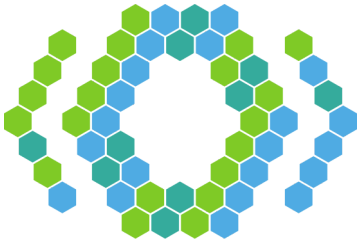

OPAL
OPEN DATA PORTAL

Deliverable D2.3
Benchmark-Spezifikation und Ergebnisse des ersten Crawlers

Autoren: Geraldo de Souza Jr
Reviewer: Adrian Wilke

Veröffentlichung	Öffentlich
Fälligkeitsdatum	30.06.2019
Fertigstellung	31.07.2019, Version 1.0 / 20.08.2019, Version 1.1
Arbeitspaket	AP 2
Typ	Bericht
Status	final
Version	1.1

Kurzfassung:
Zur Evaluation der ersten Version des OPAL Crawlers wurde ein Benchmark spezifiziert und ausgeführt. Dazu wurden entsprechende Wissensgraphen generiert und der in OPAL erweiterte Squirrel Crawler mit der Softwarelösung LDSpider verglichen. Zur Evaluation der extrahierten Daten wurden die erwarteten zu extrahierenden Daten mit den tatsächlich extrahierten Daten verglichen. Der Squirrel Crawler lieferte sowohl für HTTP Daten als auch für RDF Daten bessere Ergebnisse.

Schlagworte:
Crawler, Crawling, Benchmark, Squirrel, Evaluation

mFUND
Das Startkapital für die Mobilität 4.0

Bundesministerium
für Verkehr und
digitale Infrastruktur

Inhalt

1 Introduction

To evaluate the performance of Squirrel crawler, a benchmark comparison with LDSpider was developed. The benchmark is developed using the Hobbit Platform , allowing us to compute comparable results on standardised hardware.

The goal of the benchmark is to evaluate the following Key Performance Indicators (KPI's) :

- Recall : Between 0 and 1 (Triples that were successfully retrieved)
- Time
- Memory Consumption (Not available yet)
- CPU Load (Not available yet)

2 Benchmark Architecture

The benchmark is composed by a set of Hobbit components. These components are:

- **Benchmark Controller**: The central component of an experiment. It creates and controls the data generators, task generators, evaluation storage and evaluation module.
- **Data Generator:** Generates the data needed for the evaluation. It creates a graph containing linked data based on parameters provided by the user in the front end.
- **System Adapter**: Starts the system (Crawler) selected by the user, initializing all the dependencies and services necessary for that system execution.
- **Evaluation Module**: Evaluates the output of both crawlers, comparing with the expected output (generated data).

To simulate a properly environment for linked data, we create a custom LOD cloud, containing three different types of nodes: HTTP, CKAN and SPARQL. Each one of this nodes receives the graph from the Data Generator and adds the graph triple to each one of its type.
It is very unlike that a RDF file will reference a Sparql or Ckan endpoints and vice-versa. Because of this characteristic, only the HTTP node reference its own graphs.

3 Benchmark Workflow

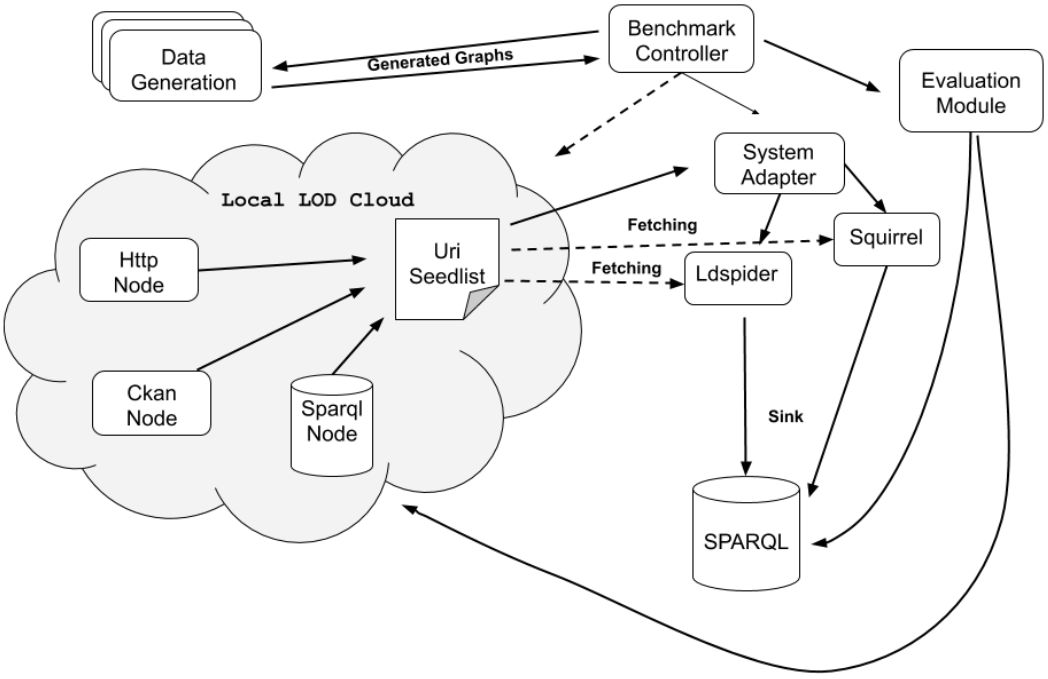

Figure 1: Benchmark Overall Schema

Figure 1 illustrates the overall schema of the benchmark, with the LOD Cloud interaction with Hobbit components. The benchmark is started by invoking the Benchmark Controller, with the parameters configured on the launch screen. These parameters are:

- System : Squirrel or LDSpider
- Number of Nodes
- Weight of CKAN node occurence (Between 0 and 1)
- Weight of SPARQL node occurence (Between 0 and 1)
- Weight of HTTP node occurence (Between 0 and 1)
- Average RDF Graph Degree
- Triples per Node
- Average Node Delay
- Seed

The Benchmark Controller invokes the **Data Generation Module**, which will create the graphs according to the parameters specified before. In the Sequence, the LOD Cloud Nodes will be created and the graphs will be given for each one of then.
The System Adapter is responsible for initializing the crawlers. It will configure the environment and start the necessary dependencies that each one of then requires. When the System Adapter finishes the initialization, will receive the **URI Seed List** and will give it to the crawlers.
Both Crawlers stores the crawled data on a Sparql Database. The **Evaluation Module** finally, analyzes the crawled data and compares with the expected data exposed by the LOD Cloud.

4 Evaluation

The goal of this Benchmark is to measure the reliability of crawled data extraction by both systems. The evaluation is done by comparing the expected graph, generated by the Data Generation Module against the stored data.
The **Recall** measure states the rate of triples that were expected and successfully recovered. Normalized, the result can vary from 0 to 1 (0 indicating that nothing was crawled and 1 that everything was recovered).
The Weighted Nodes (Ckan, Sparql and Http) can interfere in the Recall. The Node that has 0 node, will not be started by the Benchmark Controller, consequently will not have any data from that node kind. If not zero, the Node type relevance on the final result will be equivalent to the weight configured.

5 Results

The first results were achieved by running on two different scenarios. The first scenario consist of using only the HTTP Nodes in the LOD Cloud, in order to analyze how both crawlers deals with pure RDF data. The second scenario includes CKAN and Sparql resources as well, attributing a weight of 1.0 to all the three nodes. The generated graph contains 1000 triples on both scenarios.

	Squirrel	LdSpider
Average node graph degree	4	4
Average rdf graph degree	4	4
Number of Nodes	5	5
Triples evaluated	5009	5009
Triples per Node	1000	1000
HttpNode Weight	1.0	1.0
CkanNode Weight	0	0
SparqlNode Weight	0	0
Recall	**0.998**	**0.893**

Table 1. Http node only Scenario.

As described on **Table 1**, the first scenario with Http Nodes only give us a recall of 0.998 and 0.893 to Squirrel and LdSpider, respectively. Despite the Squirrel clear advantage, it was expected a value 1.0, where all the triples were successfully crawled. We will investigate the crawler to see what it is avoiding some triples to not be crawled.

	Squirrel	LdSpider
Average node graph degree	2	2
Average rdf graph degree	5	5
Number of Nodes	3	3
Triples evaluated	3003	3003
Triples per Node	1000	1000
HttpNode Weight	1.0	1.0
CkanNode Weight	1.0	1.0
SparqlNode Weight	1.0	1.0
Recall	**1.0**	**0.5**

Table 2. Multi nodes run scenario.

The difference between two crawlers really appears when is include other nodes, as show on **Table 2**. Squirrel reaches 1.0 and LdSpider gets 0.5, meaning that LdSpider found only the data from the Http Node. Considering the first scenario, is possible to guess that the 1.0 value is rounded, with minor missing triples now interfering in the final result of this run.
In future experiment runs, we will test different weights, rdf and graph degrees, triple count and number of nodes to see in more details the evaluation of both crawlers.