QROWD - Because Big Data Integration is Humanly Possible

Innovation Action

Grant agreement no.: 732194

D8.2 – Benchmarking registry, reporting, and crowdsourcing monitoring tools

Due Date	30 Nov 2018
Actual Delivery Date	30 Nov 2018
Document Author/s	Semih Yumuşak (AI4BD) Daniel Hladky (AI4BD) Eddy Maddalena (SOTON)
Version	0.5
Dissemination level	PU
Status	Final
Document approved by	Claus Stadler

TABLE OF CONTENT

History

Version	Date	Reason	Revised by
0.0	16.10.2017	Table of Contents	Semih Yumuşak (AI4BD)
0.1	18.10.2017	Virtual City Explorer Monitoring	Eddi Maddalena (SOTON)
0.2	23.10.2017	Benchmarking results and monitoring tools	Semih Yumuşak (AI4BD)
0.3	02.11.2017	Ready for Internal Review	Semih Yumuşak (AI4BD)
0.4	16.11.2017	Ready for final check	Semih Yumuşak (AI4BD)
0.5	26.11.2017	Finalized after review	Semih Yumuşak (AI4BD)

EXECUTIVE SUMMARY

This document is the second deliverable within WP8 and builds on top of the proposed architecture D8.1. In this deliverable, benchmarking tools and datasets for data science users and monitoring tools for the integrated platform users are provided. Several work packages contributed to datasets and benchmarking as part of the ongoing improvement of the use cases WP1 and WP2.

The D8.2 document explains the benchmark components and initial setup together with the monitoring tools provided. The datasets and components related to benchmarking are utilized to track the performance of the tools and report accordingly. Within Section 2, benchmarking datasets and tools collected from WP 3-7 are summarized and reported. Additionally, crowdsourcing monitoring tools related to virtual city explorer (D3.2) and the initial QROWD platform (D8.1) are demonstrated in Section 3.

1 INTRODUCTION

The components developed under the QROWD platform utilize and merge different data sources with human-in-the-loop processes. While dealing with these heterogeneous data sources, the performance of the tools used gains importance. In this deliverable, initial datasets provided by the QROWD project are described and the initial reports for the data storage tools improved during the project are created as a benchmarking report. Currently, there are two dataset definitions provided as; (1) Transportation modes, (2) Virtual city explorer dataset. Together with these datasets, initial reporting of the QUAD store and QROWD-DB performance based on predefined benchmarks are provided as a report.

Additionally, crowdsourcing monitoring tools provided by the initial QROWD platform are explained under three categories: (1) Virtual city explorer monitoring, (2) Data flow monitoring, (3) Streaming data crowdsourcing integration monitoring.

2 BENCHMARK DATASETS, TOOLS, AND REPORTS

The initial benchmarking registry is created as a repository, in which QROWD partners upload their applications and datasets available for benchmarking. The repository is currently available under the QROWD repository as QROWD Benchmarking Registry[1] and QROWD applications are listed in QROWD project space[2].

In this section, we list the benchmarking datasets, tools, and results provided by partners involved in WP4 to WP7. From those work packages, we collected tools and datasets for transportation modes (D6.1), virtual city explorer (D3.2), QROWD-DB (D7.3), and QUAD (D7.2). All of the provided benchmarks are supporting the innovative use cases in relation to increase on a continuous base the performance. D8.2 is a foundation and will be used after the end of the project by TomTom and Comune di Trento as a base in order to elaborate and improve the processes.

2.1 Transportation Modes dataset and tools

The Transportation Recognition (TR)[3] is the implementation of a system based on Machine Learning techniques to detect transportation modes using the accelerometer data of iLog users in QROWD project and reported as QROWD D6.1.

The ML techniques were used in combination with Data Mining strategies to pre-process the streaming data and evaluate the models to provide confident labels for specific trips like bike, bus, car, train and walk. The result is a model able to predict new data with high accuracy and high confidence level.

[1] http://ckan.qrowd.aksw.org/group/benchmarking-registry
[2] https://github.com/QROWD
[3] https://github.com/QROWD/TR

The Data Mining techniques used include Time Window operations and Signal Processing methods [1] to preprocess the data. In the ML level, classifiers like Adaboost, ANN, C4.5, CART, k-NN, Random Forest, SVM and XGBoost [2] are used. The simplest way to generate and evaluate the models with the datasets available is calling the evaluation function. The parameters are the window type (static or slide), the discrete transformation (dft, dwt or dsf) and the window size:

Rscript --vanilla run.r evaluation static dft 450

The output is the average performance of the models by user and the best model exported. The average performance is a matrix where the columns represent the classifiers available and the lines represent the evaluation measures. The output is listed in Table 1.

Table 1: The transportation recognition tool outputs performance indicators for several ML approaches

	Adaboost	**ANN**	**C4.5**	**CART**	**kNN**	**RF**	**SVM**	**XGBoost**
acc	0.788889	0.394444	0.688889	0.672222	0.711111	0.922222	0.85	0.75
kappa	0.745261	0.295005	0.625863	0.605322	0.651902	0.906009	0.818046	0.698324
f1	0.770294	NaN	0.668178	0.637719	0.685491	0.913814	0.837193	0.727626

The best model will be exported in the main folder with the name model.rds. To evaluate a new data, you can call the prediction function with the model and file path:

Rscript --vanilla run.r prediction model.rds test.txt

The output is a table called out.csv with the label and associated probability column.

Using TR with additional data and labels:

The datasets used in this project is a combination of public available accelerometer data [4]. and iLog data. You can add more data (from other users) including a new file in the subfolder datasets. The file needs to be a csv separated by comma with the user id, timestamp, accelerometer (x, y, and z) and the label columns. To add more labels, is important to guarantee that at least 2 users have the same label to avoid errors in the evaluation process.

As a result of the dataset creation process, there are 6 different datasets provided for training in the code repository[4]: bike, bus, car, still, train, walk.

[4] https://github.com/QROWD/TR/tree/master/datasets

2.2 Virtual City Explorer Benchmark

The Virtual City Explorer is a tool for using paid crowdworkers or volunteers to generate maps of Points of Interest (PoI) in the Google Street View representation of urban areas [3].

The core problem can be summarised as follows: Given an urban area defined as a closed geographic polygon, locate all instances of a target PoI type, while maximizing completeness (all items found?) and coverage (number of points of the underlying street view polyline), then minimizing the number of contributors and cost (if using paid crowdworkers).

To benchmark our approach and to motivate further research in the field, we publish two benchmark areas and reference maps, using bike racks as the target PoI type. In this context, benchmarking refers to enabling the comparison of results of crowdsourcing experiments - i.e. a benchmark run involves human labor.

QROWD publishes many resources including metadata of crowdsourcing experiment setups to facilitate future comparison. For this purpose, we devised the QROWD Voc ontology[5].

The areas correspond to the limited traffic zone of Trento, and a section of the Penn Quarter of Washington D.C. They were chosen due to the availability of reference maps collected by employees or volunteers.

Trento-Area.json contains in the array ['geometry']
- ['coordinates']: array of coordinates with polygon definition
- ['nodes']: array of description of points that conform the Street View polyline inside the polygon
  - "lat": latitude
  - "lng": longitude
  - "pano" Panorama ID, this is the ID of the Street View (the 360 image) corresponding to a point (see: https://developers.google.com/maps/documentation/javascript/streetview)
  - "imageDate": date of the pano image in Street View
  - "description": reverse geocoding of lat and lng (long version)
  - "shortDescription" reverse geocoding of lat and lng (short version)
- ['edges']: array of segments of the polyline (useful for computing distance)
  - p1: first panorama ID of edge
  - p2: second panorama ID of edge
  - "description": reverse geocoding of p1 (street name)
  - "distance" : length of the segment

A sample record of Trento-area.json file is provided in Table 2.

[5] https://github.com/QROWD/crowd-voc

Table 2: Sample record from Trento-Area.json


```
á
  íëa\íÆ á
    íòga\íÆ íþY\ü]ûúúĂþ^Ă[ûýÿĂùY[ÿ\ÿüí
  âÄ
  ílqh]íÆ í&]Ylmj]íÄ
  íhjgh]jla]kíÆ á
    íYml`gjíÆ í%\\q -Y\\Yd]fYíÄ
    ífYe]íÆ í4j]flg Ý:4,þíÄ
    í\]k[jahlagfíÆ í-Yh g^ :gfY \a 4jY^^a[g ,aealYlg Ý:4,þ g^
4j]flgÇíÄ
    í\Yl]íÆ íûùúāÈùüÈúÿí
  âÄ
  í_]ge]ljqíÆ á
    ílqh]íÆ í0gdq_gfíÄ
    í[ggj\afYl]kíÆ ß
      ß
        úúÇúúāûĂûāùāþÄ
        ýÿÇùÿĂāĂÿùþāû
      àÄ ÇÇÇÇÇÇ
    àÄ
    ífg\]kíÆ ß
      á
        ídYlíÆ ýÿÇùÿāþāĀûāüýāûÿÿÄ
        ídf_íÆ úúÇúûûþāûþúÿüþúĀúüÄ
        íhYfgíÆ íý9ü"2*!ÿpi164e31jú_,ioíÄ
        íaeY_]$Yl]íÆ íûùúĀÈúùíÄ
        í\]k[jahlagfíÆ íþû 6aY /kk -YrrmjYfYÄ 4j]flgÄ
4j]flafgÈ3gml` 4qjgdíÄ
        ík`gjl$]k[jahlagfíÆ íþû 6aY /kk -YrrmjYfYí
      âÄ ÇÇÇÇÇÇ
    à
  â
â
```


For each area, there are two maps collected using physical presence methods.

For Trento
- Trento_Municipality.json: Collected by a municipality employee (collected July 2017)
- Trento_OSM.json: Bike racks from Open Street Maps (downloaded September 2017)

For Washington
- Washington_Rackspotter.json: Bike rack map from Rackspotter, a volunteered initiative (collected September 2017)
- Washington_OSM.json: Bike rack map from Open Street Maps (downloaded September 2017)

A sample record of Trento_Municipality.json file is provided in Table 3.

Table 3. Sample record from Trento_Municipality.json

```
á
  ílj]flgëleíÆ ß
      á
        ídYlíÆ ýÿÇùÿĂĀāÿûāüüùþüüÄ
        ídf_íÆ úúÇúûùúùþĀüüúÿþýÿā
      âÄ ÇÇÇÇÇ
    à
â
```

In order to compare one's results; we expect from a crowdsourcing expert to run the crowdsourcing approach (or an improvement over VCE) on the reference areas and record how many workers you used and how much you paid to them.

The output should be:

- A list of located bike rack coordinates (must be already cleaned of false positives)
- A dictionary {("lat","lng","pano") -> numberOfVisits} representing how many times workers visited a panorama ID.

By this way, the result will be comparable on; how many items found with respect to the physical presence approaches. It needs to be considered that the coordinates might not be exact, so, if the coordinate is at least at 10m distance of the reference coordinate it should be considered as the same. Additionally, the areas covered with at least one worker or at least n workers explored most streets can be compared.

2.3 QROWD-DB Big Data Storage Benchmarking Report

In this section, the benchmarking of Big Data Storage present in QROWD-DB that stores the personal big data collected from the citizens' smartphone is explained.

The benchmarking of QROWD-DB is applied by measuring database query read/write performance under a simulated workload. In QROWD-DB, we use cassandra[6] as the basis, for which the cassandra-stress[7] tool exists. The QROWD-DB is deployed in a cluster, which has a total of 12 nodes, each one being virtualized and distributed across three physical servers. The virtualization is configured through Kubernetes which creates 12 containers. Each physical server has the following characteristics: 48 cores CPU, 192GB RAM, 2 x 480GB SSD + 2TB HDD (storage disk)

QROWD-DB is configured with a replication factor of 3; this means that every data is persisted with three different write operations on three different nodes of the cluster.

[6] http://cassandra.apache.org/

[7] https://docs.datastax.com/en/cassandra/3.0/cassandra/tools/toolsCStress.html

The main storage of each QROWD-DB node is on a network bay directly connected to the physical servers through a LAN connection. The bay contains multiple 2TB disks one used by each node.
Since we are using a distributed database, the network connection can influence the performances and consequently the benchmarking. For this reason, in this section, we describe how the different nodes and physical servers are connected to each other.

The distribution of the 12 QROWD-DB nodes across three physical clusters implies, that on machine can host multiple nodes (2-4 depending on the machines' load). Those nodes on the same machine are connected to a virtual network inside each machine while the physical machines are connected with dedicated 1gbit LAN links. Similarly, the network storage bay is connected with the same 1gbit link to the physical servers.

The tool used to generate the benchmark is called cassandra-stress and is the official tool made available by Apache. It is a Java-based stress testing utility for basic benchmarking and load testing for any Cassandra cluster. We decided to use this tool since is something available to everyone that will facilitate the comparison of our results with others.

We decided to go with a configuration illustrated in Figure 1, so that to generate results that should be close to the real usage scenarios we are dealing with in the project, instead of analyzing an ideal situation that would have produced much better results not attached to reality.

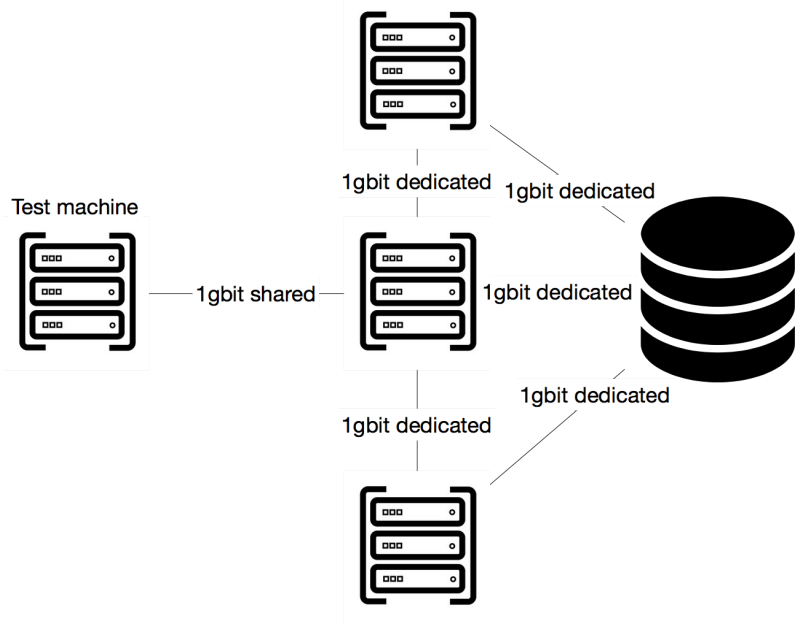

Figure 1: Benchmarking configuration

The stress test for the QROWD-DB was performed by using the number of samples to be written and the number of concurrent threads that simultaneously perform operations on the database. The total number of samples was selected as 1,000,000 and total number of concurrent threads are assigned as 1000.

The results for the QROWD-DB benchmarking report are listed for writing, reading and mixed (writing/reading) performances.

The write performance diagram is visualized in Figure 2 with test details in Table 4, the reading performance diagram is visualized in Figure 3 with test details in Table 5, and mixed performance is visualized in Figure 4 with test details in Table 6.

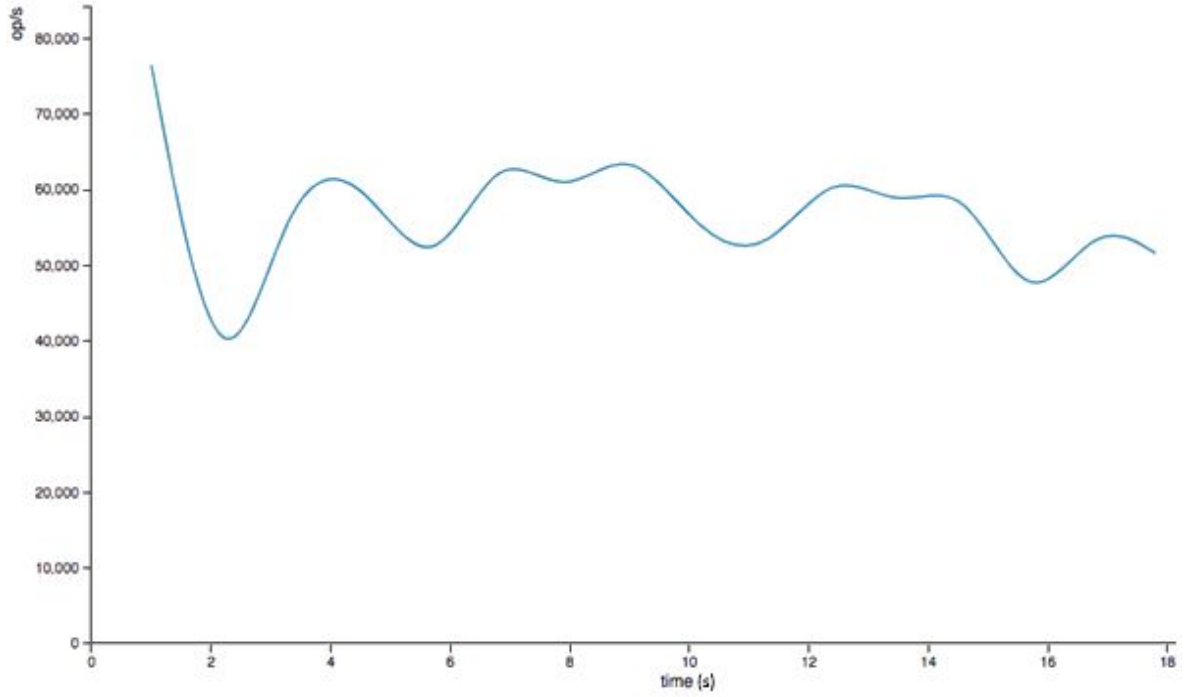

Figure 2: Writing performances - multiple concurrent writes

Table 4: Benchmarking result details for writing performance

Op rate : 56232 [Write:56232] Partition rate: 56232 [Write:56232] Row rate: 56232 [Write:56232] Latency mean: 18.9 Latency median: 11.7	Latency 95th percentile: 50.2 Latency 99th percentile: 91.5 Latency 99.99th percentile: 212.5 Latency max: 907.1 Total operation time: 00:00:17

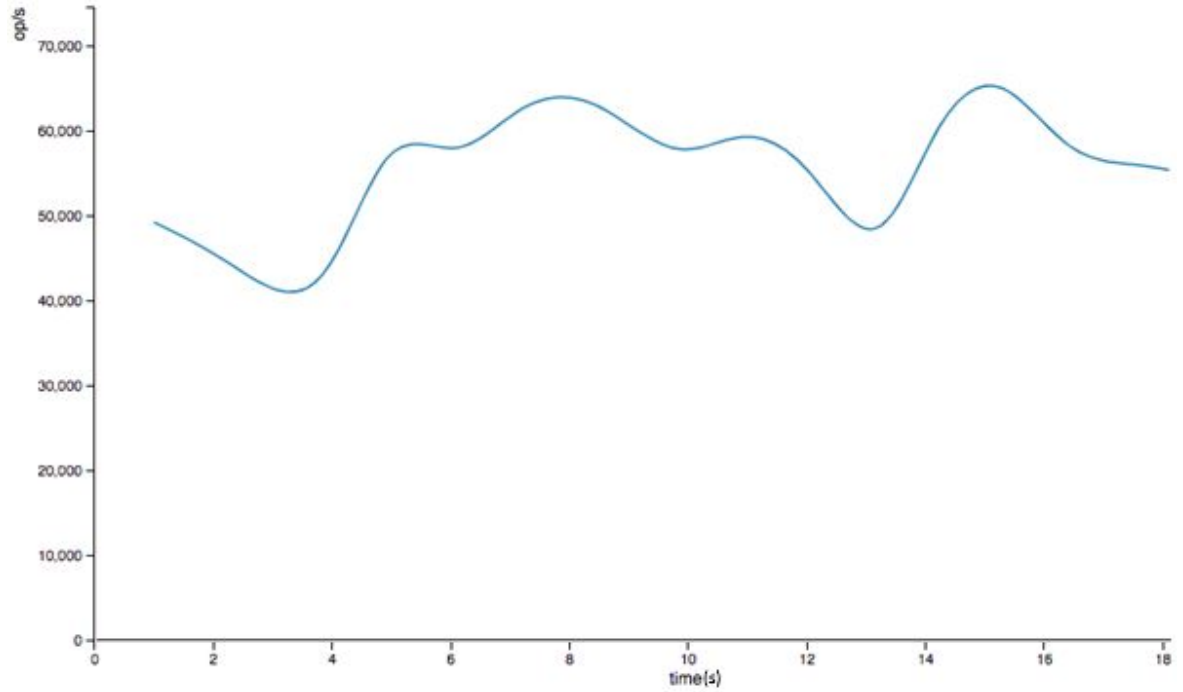

Figure 3: Reading performances - multiple concurrent reads

Table 5: Benchmarking result details for writing performance

Op rate : 55245 [Read:55245] Partition rate: 55245 [Read:55245] Row rate: 55245 [Read:55245] Latency mean: 18.5 Latency median: 11.3	Latency 95th percentile: 48.3 Latency 99th percentile: 74.9 Latency 99.99th percentile: 207.2 Latency max: 587.3 Total operation time: 00:00:18

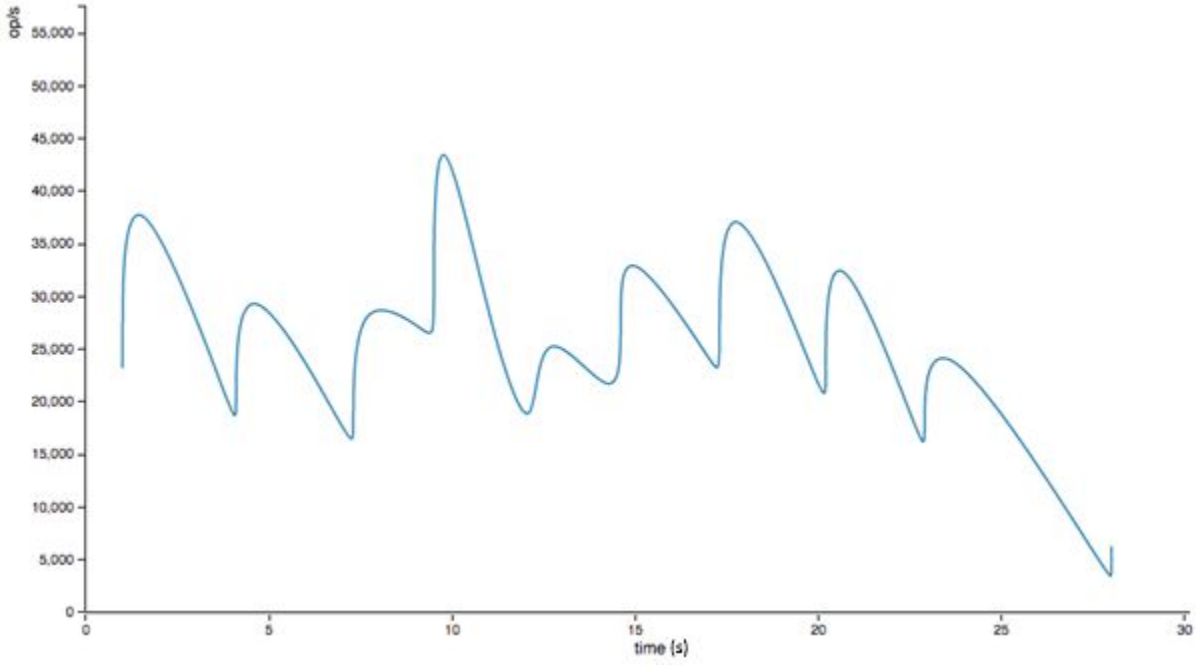

Figure 4: Mixed writing-reading performances

Table 6: Benchmarking result details for mixed (writing-reading) performance

Op rate : 33379 [Read:16712, Write:16667] Partition rate: 33379 [Read:16712, Write:16667] Row rate: 33379 [Read:16712, Write:16667] Latency mean: 27.1 [Read:27.3, Write:26.8] Latency median: 14.2 [Read:14.2, Write:14.0] Latency 95th percentile: 66.8 [Read:66.2, Write:67.5] Latency 99th percentile: 167.5 [Read:168.8, Write:167.2] Latency 99.99th percentile: 710.6 [Read:456.0, Write:1184.0] Latency max: 1257.1 [Read:1257.1, Write:1255.5] Total operation time: 00:00:28

Based on the benchmarking results for the current QROWD-DB setup, no deadlock, overload, or error condition was encountered. However, due to the high load after second 25, the operations per second decreased without creating a deadlock.

2.4 QUAD Benchmarking Report

QUAD [1] is a graph store, which is based on a vector database schema for Quadruples and it is realized by facilitating various index data structures. QUAD also comprises approaches to optimize the SPARQL query execution plan by using heuristic transformations. In the QROWD project, QUAD is the RDF store utilized in D4.4 and D7.2.

In order to test the QUAD performance, the service was tested by using ODIN Benchmark Version 1 (used for RDF data ingestion) in the Hobbit Benchmarking Platform[8]. ODIN (StOrage and Data Insertion beNchmark) is a benchmark to run and test triple/quad stores on their storing and retrieving performance, whose details are explained in [2].

In brief, the aim of the Mighty Storage Challenge (MOCHA)[9] at ESWC 2018 was to test the performance of solutions for SPARQL processing in aspects that are relevant for modern applications. These include ingesting data, answering queries on large datasets and serving as backend for applications driven by Linked Data. The challenge tested the systems against data derived from real applications and with realistic loads. An emphasis was put on dealing with data in form of streams or updates. The QUAD store was tested using this method.

The comparison of the performance of QUAD with other applications are visualized in Figures [5]-[7]. Based on these results, QUAD store performs better for micro-average-precision, which is described in [2] as:

[8] https://master.project-hobbit.eu
[9] https://link.springer.com/chapter/10.1007/978-3-030-00072-1_1

$$Micro-Average-Precision = \frac{\sum_{i=1}^{\lambda} |\{relevant\ result_i\} \bigcap \{retrieved\ results_i\}|}{\sum_{i=1}^{\lambda} |\{retrieved\ \ result_i\}} \quad \textbf{(1)}$$

Other metrics used in this benchmarking report is listed in equations 2,3 and 4.

$$Micro-Average-\ Recall = \frac{\sum_{i=1}^{\lambda} |\{relevant\ result_i\} \bigcap \{retrieved\ results_i\}|}{\sum_{i=1}^{\lambda} |\{relevant\ result_i\}} \quad \textbf{(2)}$$

$$Micro-Average-Precision = \frac{\sum_{i=1}^{\lambda} Precision_i}{\lambda} \quad \textbf{(3)}$$

$$Macro-Average-Recall = \frac{\sum_{i=1}^{\lambda} Recall_i}{\lambda} \quad \textbf{(4)}$$

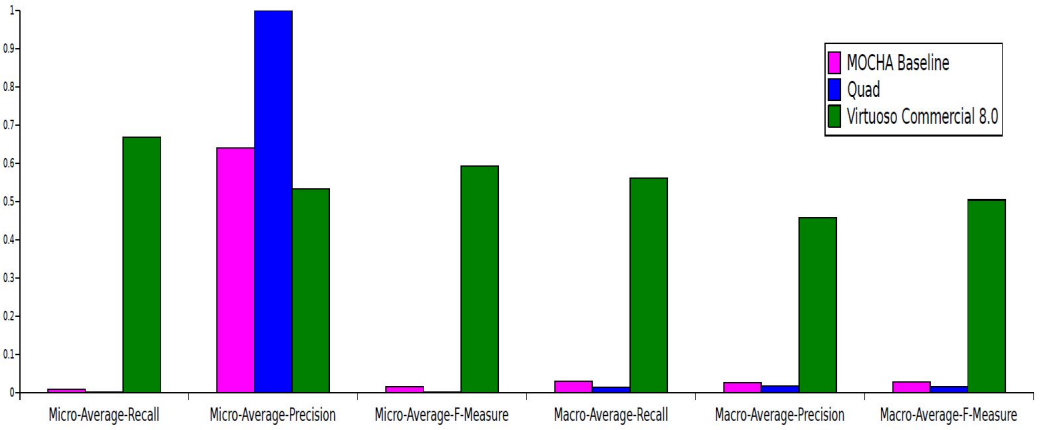

Figure 5: [2] Micro-Average-Recall, Micro-Average-Precision, Micro-Average-F-Measure, Macro-Average-Recall, Macro-Average-Precision, Macro-Average-F-Measure of MOCHA Baseline, QUAD and Virtuoso Commercial 8.0 for MOCHA2017 [2]

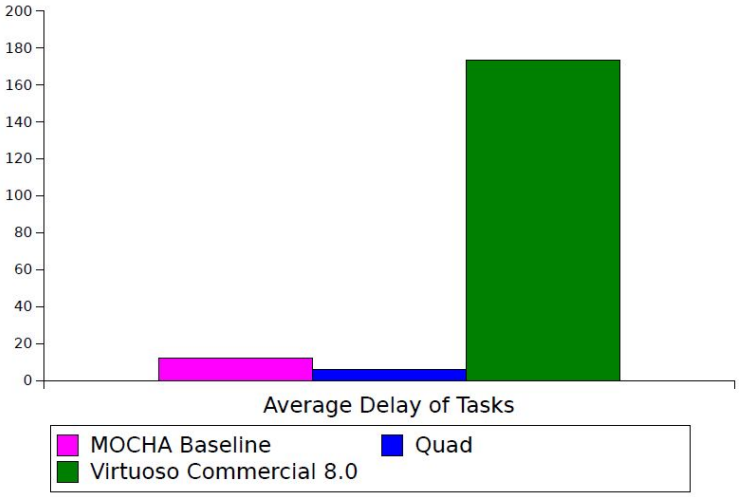

Figure 6: Average Delay of tasks of MOCHA Baseline, QUAD and Virtuoso Commercial 8.0 for MOCHA2017 [2]

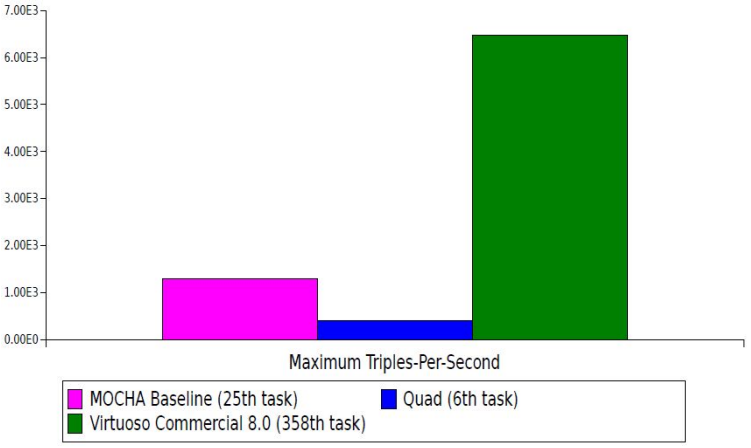

Figure 7: Maximum Triples-per-Second of MOCHA Baseline, QUAD and Virtuoso Commercial 8.0 for MOCHA2017 [2]

Based on these results, although QUAD performs well based on micro-average-precision, Virtuoso Commercial 8.0 performs better in other tasks. More information about the results can be found in [1] and [2].

3 CROWDSOURCING MONITORING TOOLS

In this section, the monitoring tools provided by the initial QROWD platform are demonstrated. First, crowdsourcing monitoring tool for Virtual City Explorer is explained in Section 3.1. Then, the data flow monitoring tools for QROWD platform are demonstrated in Section 3.2. Lastly, the crowdsourcing data flow monitoring tool is explained in Section 3.3.

3.1 Virtual City Explorer Monitoring Tools

The Virtual City Explorer (VCE) introduced in D3.2 is a standalone crowdsourcing tool to support the Completing Mobility Infrastructure use case. Crowd workers are required to virtually explore a constrained area of a city and identify within it some static items that are relevant for mobility purposes, such as bike racks, disable parking sports, or road signs. The exploration is done in a panel that integrates the Google Streetview[10] navigation service. Multiple crowdworkers that are typically recruited on traditional crowdsourcing platforms can begin the job simultaneously. This parallelism that in particular valuable since it allows fast results can lead to contraindications. For example, some small areas or the city could be not explored enough exhaustively, and the task requester might wish to bring slight changes to the task configuration to improve the exploration coverage of such areas. To the task, requester being able to supervise the work progress, we supplied the VCE with a monitor that allows observing detections collected on a given task. Also, the interface shows the list of the taboo items, computed through spatial clustering of the row detections coordinates. A taboo item is defined as a coordinate where three or more crowdworkers agree on the location of a single item.

[10] Google Streetview (https://www.google.com/streetview/)

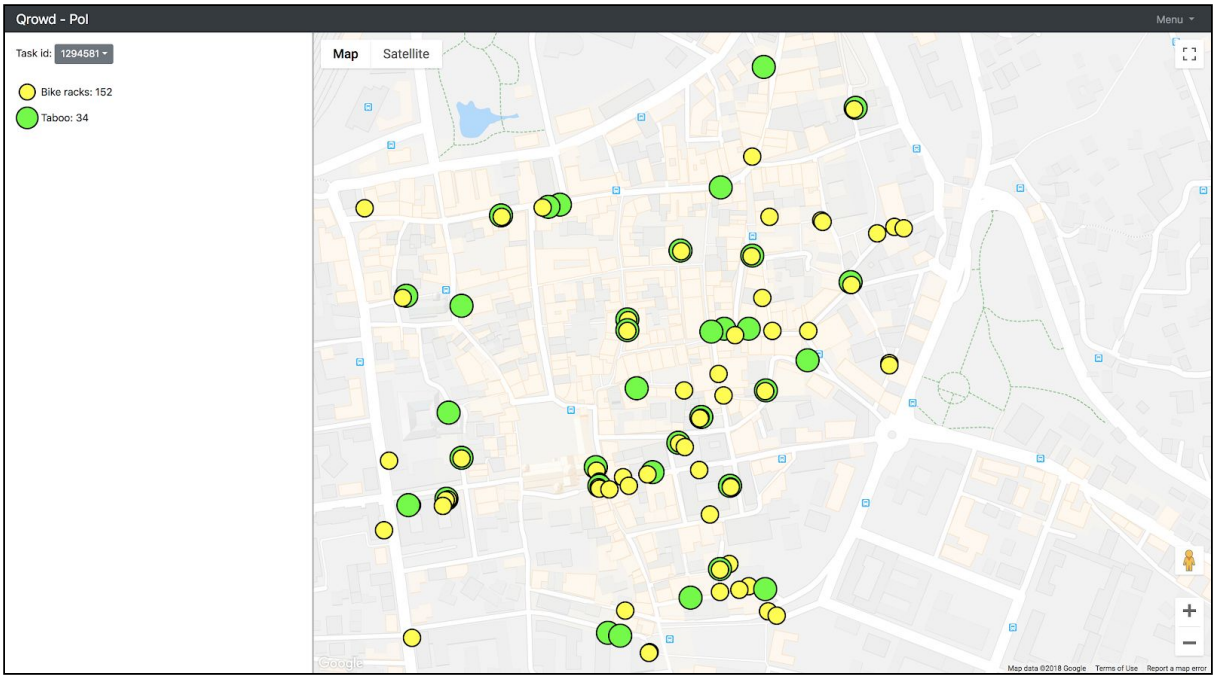

Figure 8: VCE monitor showing the map of item detected

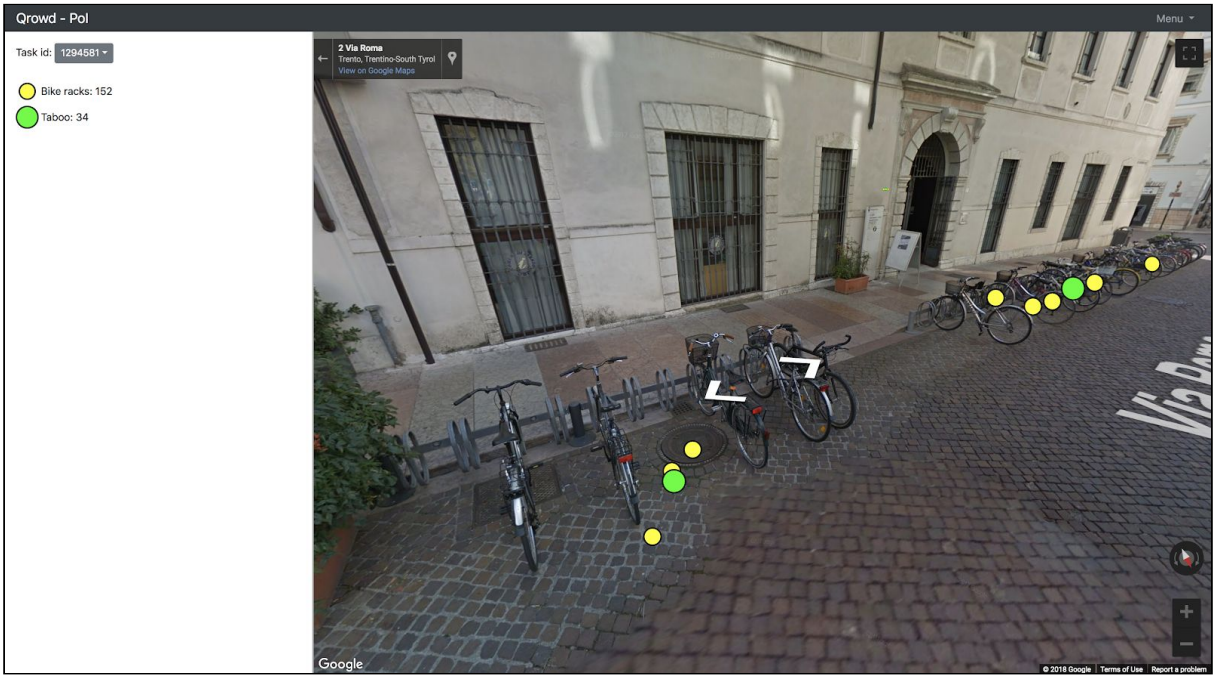

Figure 9: VCE monitor showing worker detections in a Google Streetview 3D perspective

The VCE monitor allows tho different views of such data: (1) Figure 8 map view from the top, and (2) Figure 9 from a Google Streetview 3D perspective.

3.2 Data Flow Monitoring Tools

The initial QROWD platform, which is based on Apache NiFi has the capability to allow data science users to monitor the data flow between different processes. As explained more detailed in D4.2, monitoring tools related to NiFi are initiated in the platform. By using these monitoring functionalities, every process can be monitored by using the Figure 10 and every integration can be monitored as the data flow between two processes as shown in Figure 11.

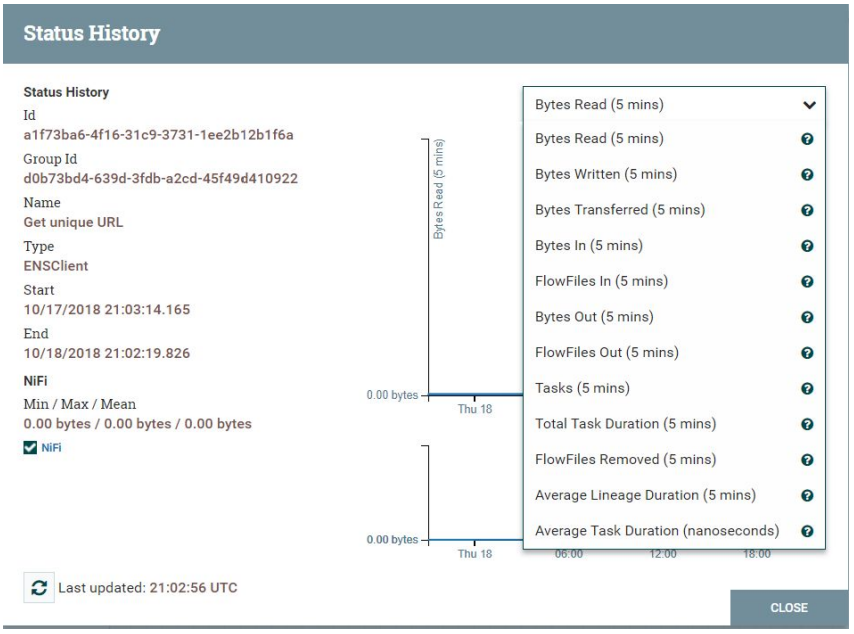

Figure 10: Monitoring windows for a processor

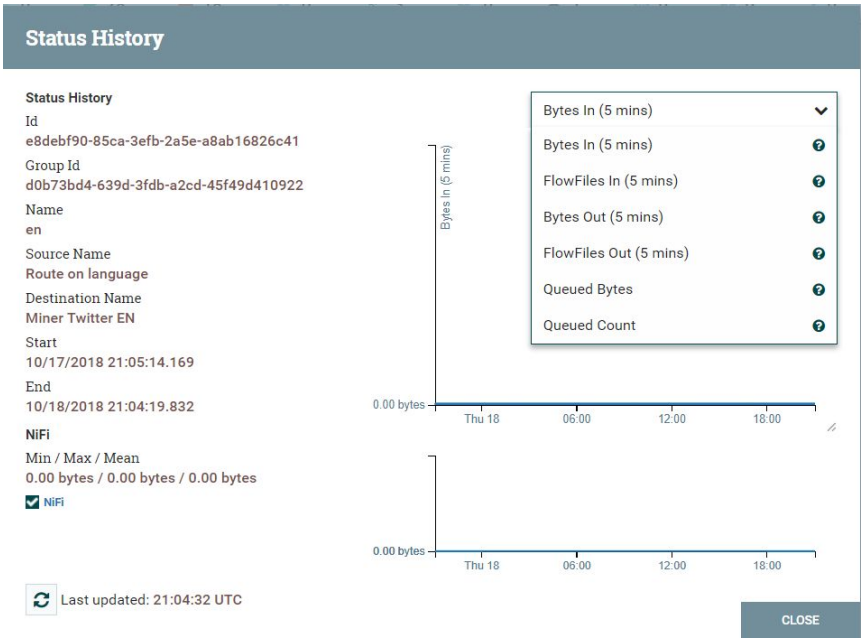

Figure 11: Monitoring window for an integration pattern

By using these data flow monitoring windows, the data flow within the integration platform from any processor can be monitored within the streaming data flow window. The crowdsourcing integration and monitoring processor is explained more in detail in the following section.

3.3 Streaming Data Crowdsourcing Integration and Monitoring

Crowdsourcing integration component is a processor mainly developed for integrating different processors with external crowdsourcing platforms. Currently, FigureEight[11] (former QrowdFlower) platform is integrated. The processor gets 4 main properties: Crowdsourcing platform, API URL, API Key, and corresponding Job ID which can be inputted from the configuration window shown in Figure 12. Job ID corresponds to the id of the created job in the external crowdsourcing platform.

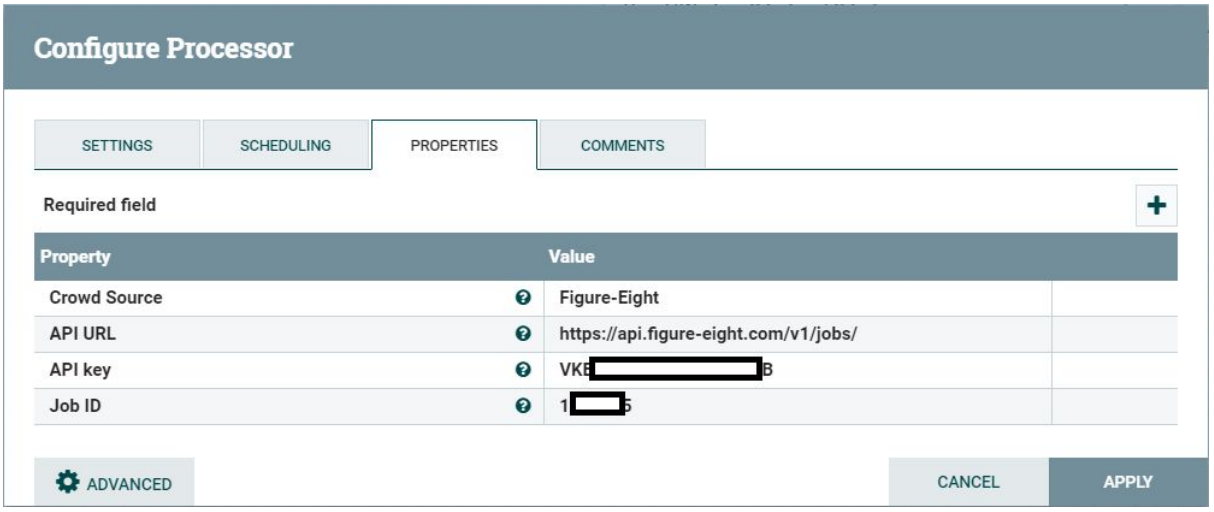

Figure 12: Crowdsourcing integration processor configuration window

By using the configuration input provided in Figure 12, the crowdsourcing integration processor can push HITs (Human Intelligence Tasks) to a specific job created before. The crowdsourcing integration processor mainly uses three properties from a flow file which are named as: CROWD_URL, CROWD_DESCRIPTION, CROWD_TITLE. As an initiator processor creates an HIT by giving values to these three properties, the crowdsourcing integration processor gets those values and pushes them to the predefined external crowdsourcing platform.

The task of pushing new HIT to an external crowdsourcing platform can be monitored by using another window, called “Advanced UI”. In the UI, as shown in Figure 13, a job can be created in the external crowdsourcing platform by using “Create Job” button. The identification number of the remote crowdsourcing job is retrieved back and saved as the Job ID, which is shown in Figure 12. In Figure 13, a sample job monitor for a tweet collections about Trento is shown.

[11] https://www.figure-eight.com/

Figure 13: Advanced UI designed for the crowdsourcing integration processor

CONCLUSION

In this document, the initial benchmarking tools and datasets are demonstrated together with the crowdsourcing monitoring tools provided by the initial QROWD platform. For the benchmarking, tools and datasets for transportation modes (D6.1), virtual city explorer (D3.2), QROWD-DB (D7.3), and QUAD (D7.2) are listed. QROWD publishes many resources including metadata of crowdsourcing experiment setups to facilitate future comparison. For this purpose, we devised the QROWD Voc ontology. Additionally, the crowdsourcing and data flow monitoring functionalities for the initial QROWD platform is demonstrated and made available for the use of other work packages.

The tools explained in this deliverable will be used within the project to improve and monitor the performance of the tools. All these tools will be presented in D8.3 QROWD platform as the final integrated platform, together with the business case feedbacks and implemented data flow integrations.

REFERENCES

[1] Potocki, A., Hladky, D., & Voigt, M. (2017, May). Challenge accepted: QUAD meets MOCHA2017. In ***Semantic Web Evaluation Challenge*** (pp. 16-20). Springer, Cham.

[2] Georgala, K., Spasić, M., Jovanovik, M., Petzka, H., Röder, M., & Ngomo, A. C. N. (2017, May). MOCHA2017: The Mighty Storage Challenge at ESWC 2017. In Semantic Web Evaluation Challenge (pp. 3-15). Springer, Cham.

[3] Maddalena, Eddy, Ibáñez, Luis-Daniel, QROWD Project Deliverable D3.2 - Crowdsourcing services, 31.05.2018