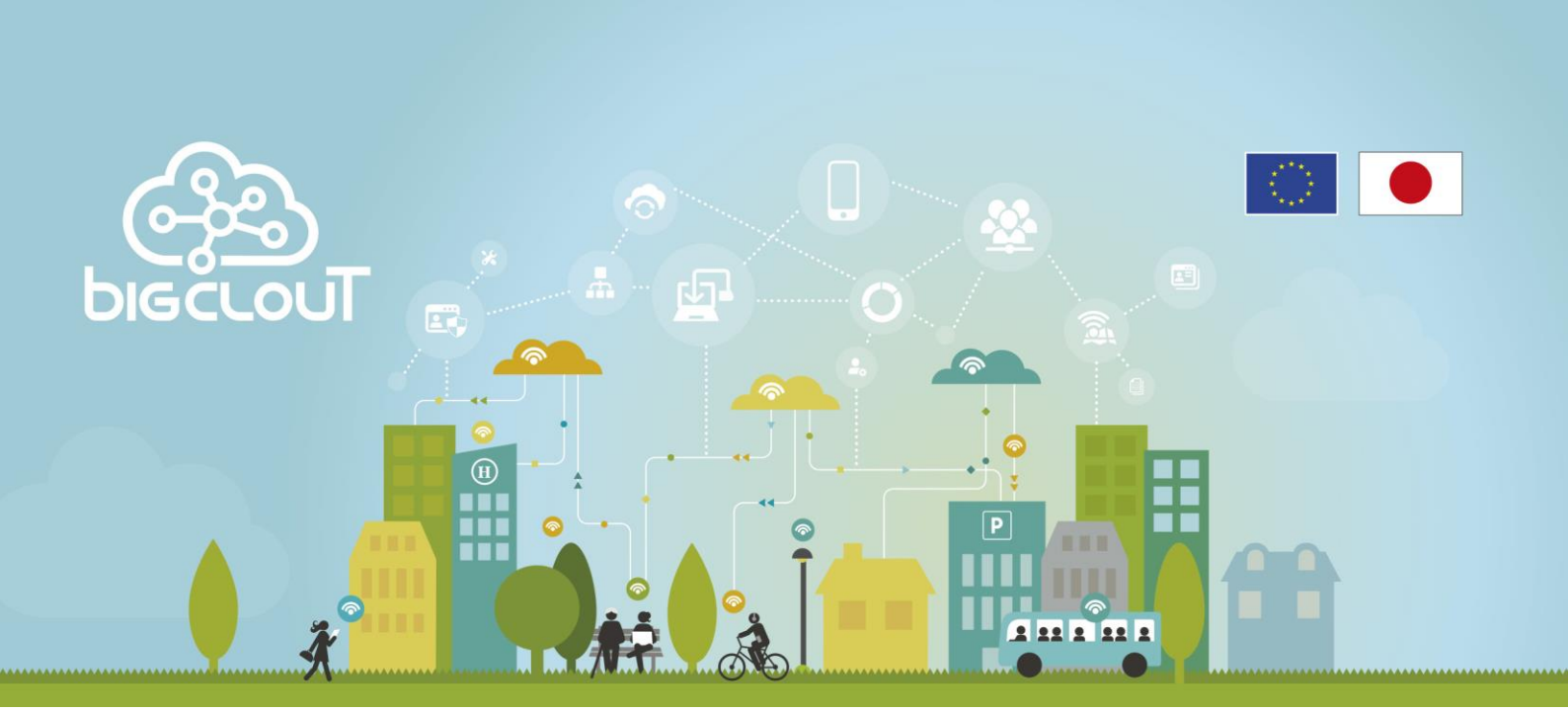


WP3 Extracting city knowledge for intelligent services
D3.4: Big Data Analytics Framework report - final release

Grant Agreement N°723139
NICT management number: 18301

BIGCLOUT

Big data meeting Cloud and IoT
for empowering the citizen ClouT in smart cities

H2020-EUJ-2016 EU-Japan Joint Call

EU Editor: **ICCS** JP Editor**: TSU**

Nature: Report

Dissemination: PU

Contractual delivery date: 2018-12-31 (M30)

Submission Date: 2019-03-20 (M33)

Co-funded by the EU H2020 GA. 723139 and NICT GA. 18301

ABSTRACT

This deliverable consists in a technical report of the final version of Big Data Analytics Framework of BigClouT that is identified in the general BigClouT architecture and presents in depth the functional subcomponents of the City Data Processing as well as the integration and implementation that took place through various use case scenarios.

Disclaimer

REVISION HISTORY

Revision	Date	Description	Author (Organisation)
V0.1	07.12.2018	ToC and Initial Content	Orfefs Voutyras (ICCS)
V0.2	19.12.2018	Section 3.1 (final contribution)	Savong Bou (TSU)
V0.3	14.01.2019	Sections 2, 3.2 (final contribution), Section 4 (initial contribution)	Guiseppe Ciulla (ENG)
V0.4	19.02.2019	Section 3.3 (final contribution)	Takuro Yonezawa (KEIO)
V0.5	22.02.2019	Section 3.4 (final contribution)	George Palaiokrassas (ICCS)
V0.6	08.03.2019	Sections 1, 4, 5 (final contribution)	Orfefs Voutyras (ICCS)
V0.7	18.03.2019	Review	Giuseppe Ciulla
V1.0	19.03.2019	Final version	Orfefs Voutyras (ICCS)
V1.0	20.03.2019	Submission	Levent Gürgen (CEA)

TABLE OF CONTENTS

LIST OF FIGURES

ACRONYMS

ACRONYM	DEFINITION
API	Application Programming Interface
CKAN	Comprehensive Knowledge Archive Network
CQ	Continuous Queries
CSaaS	City Software as a Service
CSV	Comma-Separated Values
DAG	Direct Acyclic Graph
DB	Data-Base
DoE	Deep on Edge
ESP	Edge Shift Padding Layer
GFP	Global Feature Padding Layer
GLCIC	Globally and Locally Consistent Image Completion
(G)UI	(Graphical) User Interface
HTTP	Hypertext Transfer Protocol
I/O	Input/Output
JSON	JavaScript Object Notation
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
OLAP	OnLine Analytical Processing
REST	REpresentational State Transfer
SSD	networks Single Shot Multibox Detector
TCP/IP	Transmission Control Protocol/Internet Protocol
UC	Use Case
URL	Uniform Resource Locator
WP	Work Package

Executive Summary

This deliverable consists in a technical report of the final version of Big Data Analytics Framework of BigClouT that is identified in the general BigClouT architecture and investigates in depth the functional subcomponents of the City Data Processing as well as their integration and implementation through various use case scenarios.

The document aims at presenting in depth the technical details of each and every functionality offered by the several assets that have been adopted or created under the framework of WP3. The goal behind following a detailed presentation of the elementary services offered by the several assets provided by the partners of the consortium was twofold:

- Facilitating the collaboration between technical partners of the project (from both WP2 and WP3), thus supporting the **integration** plans and actions between the various components already provided.
- Facilitating the discussions between the pilot partners and technical partners, thus enabling the **demonstration** of the several services provided by the project through various use case scenarios.

Section 2 presents the final WP3 architecture. Section 3 describes in detail all the WP3 elementary services in a manner that supports the two aforementioned goals (more details in Section 1.3). Section 4 presents the main integration points that were addressed by this work package and the corresponding use cases that are supported by them. Finally, Section 5 concludes this report.

1 Introduction

1.1 Objectives and goals of the task and deliverable

This deliverable (D3.4) aims at presenting in depth the technical details of each and every functionality offered by the several assets that have been adopted or created under the framework of WP3. The functionalities cover four different areas of interest, namely:

1) Data Event Processing,
2) Big Data Analysis,
3) Machine Learning, Predictive Modelling and Decision making,
4) and Visualisation

This categorisation, which is aligned to the tasks of WP3, focuses on the extraction of city knowledge for intelligent services by providing a common framework:

- T3.1 Big data analytics and business intelligence
- T3.2 Learning , predictive modelling and decision making
- T3.3 Distributed real-time data mining with event detection for actuation

1.2 Relation to other WPs and Tasks

This document is building on top the work presented in "**D3.1 Big Data Analytics Framework Architecture**" during the first year of the project and **"D3.2 Big Data Analytics Framework report"** and "**D3.3 Big Data Analytics Framework Prototype – Demonstration**" during the second year of the project. Finalising the general WP3 architecture, providing the latest technical details of the several components and focusing on the integration and demonstration of the WP3 services are three of the main differences between this deliverable and the previous ones.

This deliverable is closely related to the deliverables of WP4 (for example "**D4.3 Integrated use cases and first large-scale deployments and experimentation**") as well as the corresponding technical deliverables of WP2. As it will be made clear in Section 1.3, most of the subsections of Section 3 are structured in such a manner that they can present all the necessary details regarding the integration and demonstration activities that took place, while Section 4 provides an overall view of this implementation.

1.3 Methodology followed

Due to the wide range of services that can be provided under the four main modules of the WP (Data Event Processing, Big Data Analysis, Machine Learning, Predictive Modelling and Decision making, and Visualisation), each one of them includes more than one services provided by the assets of the project. Likewise, due to the wide range of functionalities of the assets and the modular approach we follow, an asset may appear in more than two categories/modules.

For this deliverable to present all the details related to integration and (integrated) demonstrations, the elementary services presented in Section 3 adopt the following structure:

- **Description of Functionalities:** A general description of the main functionalities of the service/tool is given, as an introduction to its capabilities.
- **Implementation details and Internal Architecture:** A presentation of the technical details of the component, sometimes represented by specific subcomponents.

- **Interfaces and Integration:** A presentation of all the UIs, APIs, etc. that can be used to give input to the module and access its output, thus presenting the possibilities of interaction with both end-users and other components (in WP2/3).
- **Performance, Evaluation and Stress-tests:** More technical characteristics, showing how the module performs when it comes to requirements as presented in past deliverables (big data, granularity of data, response time, number of nodes that can be managed at the same time, integration, etc.).

Finally, in Section 4, the final Use Cases that are supported by the WP3 components are presented, showcasing how the several components interact with each other in real scenarios and how they provide services of higher value through this collaboration.

2 Final WP3 architecture

This section provides information and details about the final version of the sub architecture of BigClouT City Data Processing module that represents the Big Data Analytics Framework. Definition of the final version of this sub architecture starts from the final architecture of BigClouT reported in "D1.4 Updated use cases, requirements and architecture" and from the two incremental versions reported in "D3.1 Big Data Analytics Framework Architecture" and in "D3.2 Big Data Analytics Framework report - first release".

Indeed, coherently with the general approach of BigClouT architecture, this sub architecture must be considered as a reference architecture characterised by flexibility and adaptability that can be adapted and customised to address specific needs and requirements.

The aim of the final version of this sub architecture is to consolidate the logical organisation of its macro modules and of the technological assets.

As reported in D3.1, the Big Data Analytics Framework matches with the logical module "City Data Processing" of the general logical BigClouT architecture (Figure 1) and its subcomponents, except for the subcomponent "Context Management & Self-Awareness" that is investigated in WP2. More information about the "Context Management & Self-Awareness" can be found in: "D2.1 Data collection tools and architecture", in "D2.3 Self-aware distributed city data platform-First Release" and in "D2.5 Self-aware programmable city platform - demonstration" WP2 deliverables.

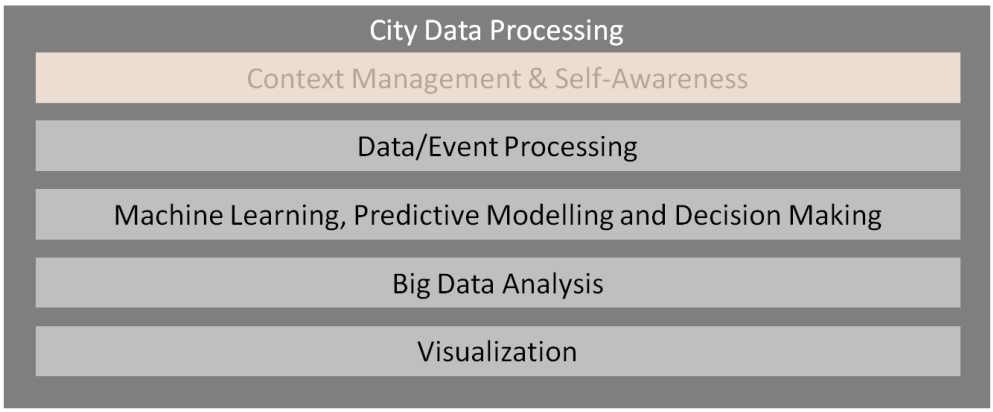


FIGURE 1: CITY DATA PROCESSING

Logical relations among these subcomponents and assets mapping are reported in Figure 2. As depicted, some assets are reported in more than one logical subcomponents. This is due to the fact that these assets provide different functionalities that fulfil different aims. For instance, KNOWAGE provides useful functionalities for both "Big Data Analysis" and "Visualisation" subcomponents.

In Figure 2, lines of different colours are used only to avoid confusion in the relations depicted in the figure.

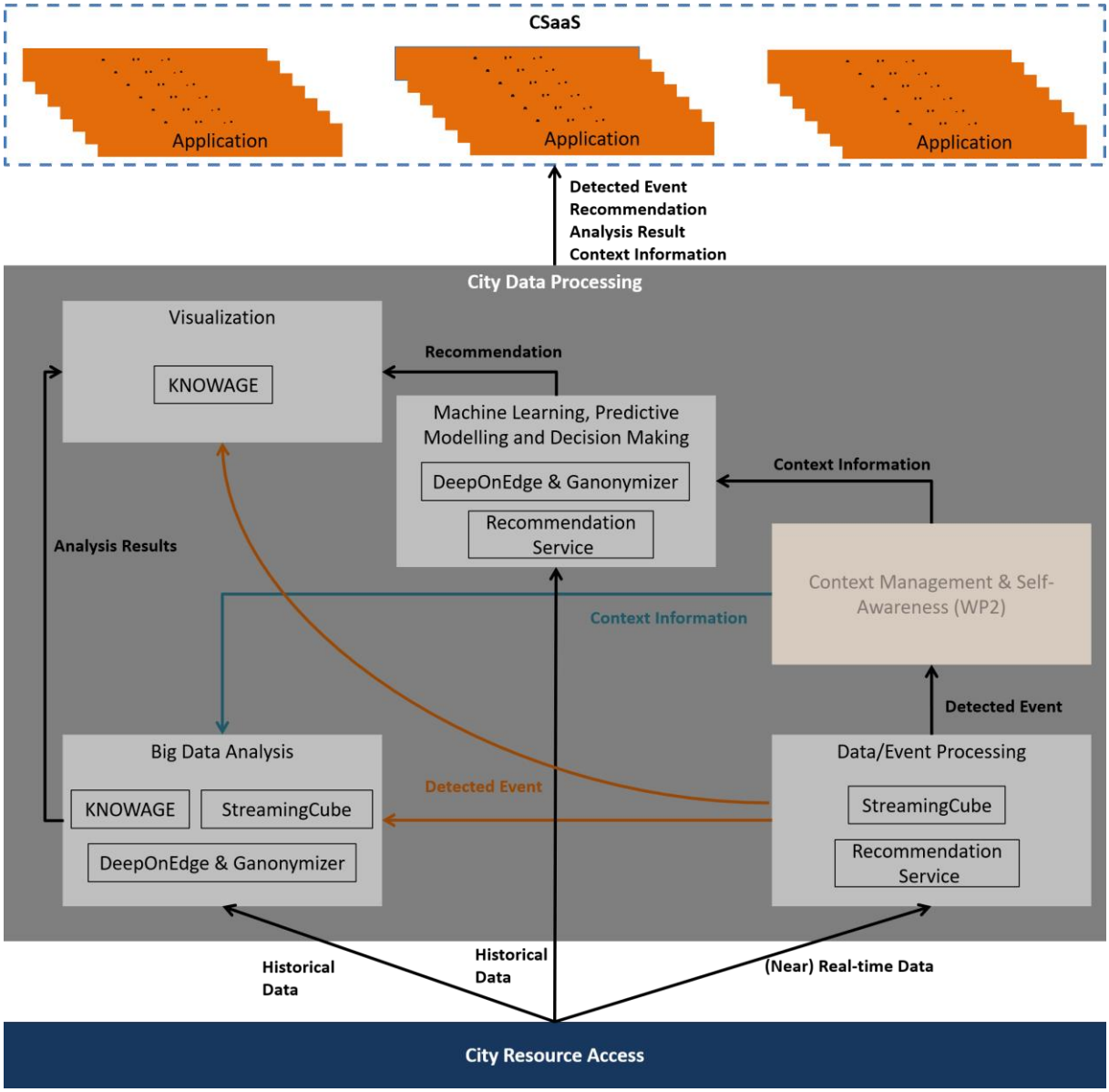


FIGURE 2: RELATIONS BETWEEN SUBCOMPONENTS OF CITY DATA PROCESSING AND ASSETS MAPPING

The interactions among City Data Processing assets are shown in Figure 3. In order to give some insight about the data sources, sensiNact, the BigClouT Data Lake and other entities that can provide data (such as Fujisawa's garbage trucks) are depicted in this diagram. These assets provide historical data or (near) real-time data that are used to produce recommendations, analysis and visualisations and to detect events. Historical data are gathered through BigClouT Data Lake taking advantage of its CKAN RESTful APIs. (Near) Real-time data are provided to the City Data Processing assets taking advantage of sensiNact RESTful APIs and thanks to IoT sensors deployed in the garbage trucks, in which a specific instance of DeepOnEdge and Ganonymiser analyses the data and provides its results to the rest of the City Data Processing components. DeepOnEdge and Ganonymiser outputs are stored in the BigClouT Data Lake and are also provided in forms of daily CSV files to be used by KNOWAGE and the Recommendation Service as data source. The recommendations produced by the Recommendation Service are provided to the rest of the platform thanks to its RESTful API. StreamingCube reads the (near) real-time data from sensiNact and provides its output to the rest of the platform thanks to its RESTful API and thanks to its GUI. KNOWAGE analysis results and dashboards are provided to the CSaaS module thanks to its GUI and RESTful APIs.

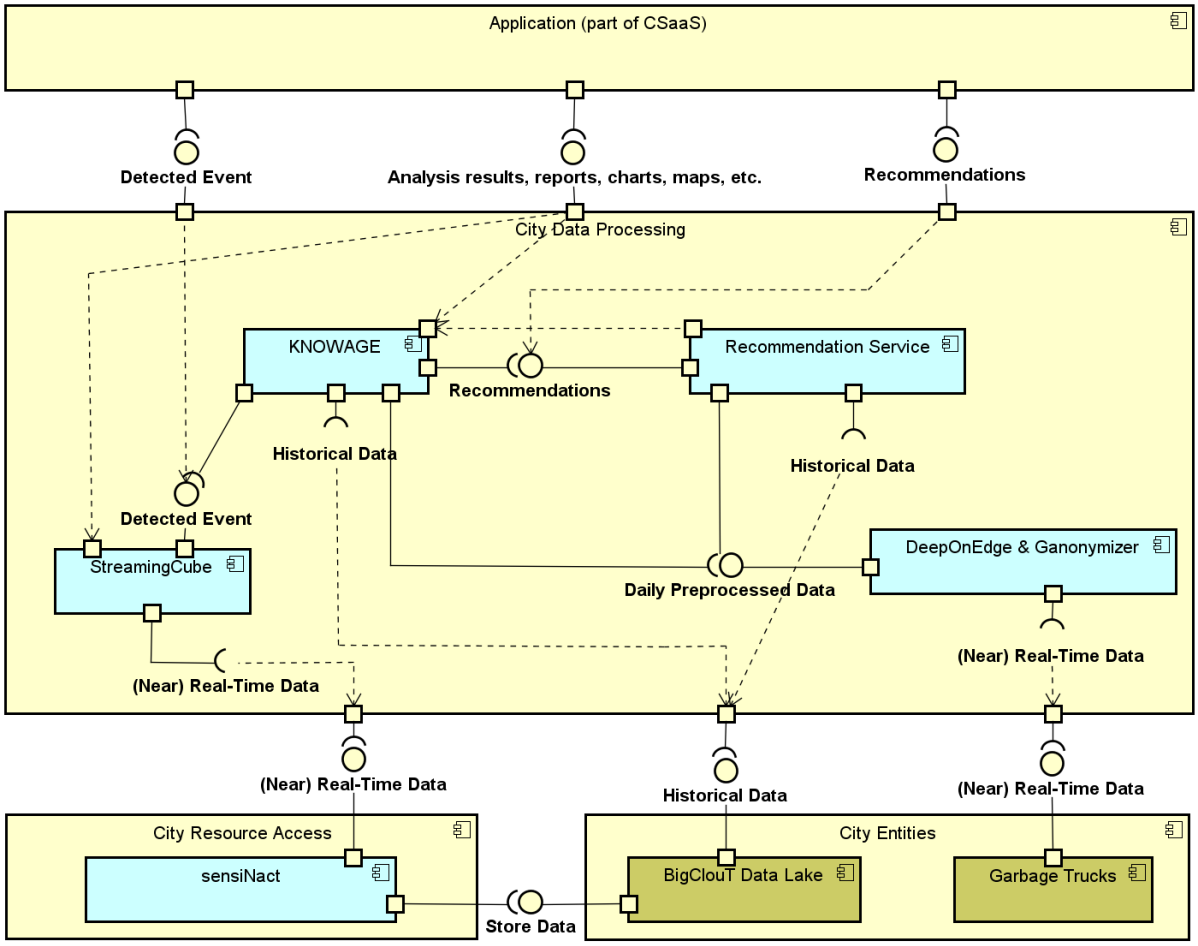


FIGURE 3: INTERACTIONS AMONG CITY DATA PROCESSING ASSETS

3 Assets and Components

This section presents the finalised versions of the several components that have been updated or developed in order to facilitate the several WP3 functionalities. For each asset, a description of its main functionalities, implementation and integration details as well as performance results are given to provide a complete overview of the final results of the WP, from a technical point of view.

3.1 StreamingCube

In most streaming applications, the data streams need to be analysed continuously to make instant decisions exploiting latest information. Data streams are often multidimensional and are at the low-level of abstraction, whereas analysts are interested in multi-level interactive analysis of data streams across several dimensions. On-line analytical processing (OLAP) is a proven technique for such analysis of static data and has also been studied by some researchers for data streams. Traditionally, this is achieved by coupling a stream-processing engine with an OLAP engine. We believe that coupling multiple systems is not an efficient solution as it results in lower performance (due to the transfer of data between multiple systems), resource wastage (due to replication of data for each coupled system) and increased complexity and maintenance cost. To this end, we present StreamingCube[1], a unified framework for both data stream processing and interactive OLAP analysis. StreamingCube possesses all the essential operators to process data streams and introduces a new operator; cubify, to maintain OLAP lattice nodes (materialised views) incrementally. The novelty of the cubify operator is the incremental maintenance of the materialised views. To demonstrate StreamingCube, a web-based GUI has been developed which enables users to register continuous queries (CQs). Once a CQ has been registered, users can perform different OLAP operations through the GUI for the interactive analysis. The results of the OLAP queries/operations are displayed in the form of tables, charts, and graphs.

3.1.1 Description of Functionalities

StreamingCube is a data processing framework that allows OLAP operation over data streams exploiting off-the-shell stream processing engine (JsSpinner[2]) combined with OLAP engine (StreamOLAP[3][4]). The overview of StreamingCube is shown in Figure 4. This system consists of two main processing components:

- **Stream processing component:** this component possesses all the essential operators of JsSpinner to process data streams. It is responsible for continuously generating aggregation results at some selected aggregation levels using multiple queries. This component is in charge of providing stream-processing features. Users can register queries in Jaql-like query language[5], thereby making it possible to continuously get filtered

1 S. A. Shaikh and H. Kitagawa, "StreamingCube: A Unified Framework for Stream Processing and OLAP Analysis", Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 2527-2530, Singapore, Singapore — November 06 - 10, 2017.

2 S. A. Shaikh, Y Watanabe, and H. Kitagawa, "Smart Query Execution for Event-driven Stream Processing", Proc. 2nd IEEE International Conference on Multimedia Big Data (IEEE BigMM 2016), pp.97-104, Taipei, Taiwan, April 20-22, 2016.

3 K. Nakabasami, T. Amagasa, S. Shaikh, F. Gass, and H. Kitagawa, "An Architecture for Stream OLAP Exploiting SPE and OLAP Engine", Proc. 2015 IEEE International Conference on Big Data (IEEE BigData 2015), pp. 319-326, Santa Clara-CA, USA, Oct.29-Nov.1, 2015.

4 S. A. Shaikh and H. Kitagawa, "Approximate OLAP on Sustained Data Streams", Proc. 22nd International Conference on Database Systems for Advanced Applications (DASFAA 2017) , pp. 102-118, Suzhou, China, March 27-30, 2017.

5 K. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Eltabakh, C.-C. Kanne, F. Ozcan and E. Shekita, "Jaql: A Scripting Language for Large Scale Semi-Structured Data Analysis," PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, vol. 4, no. 12, pp. 1272-1283, International conference on very large data bases; VLDB 2011.

streams. It accepts JSON streams as input and provides output JSON streams, which can be reused by other systems to get more useful information. Four main functionalities are provided by this component:
    - Data Filtering
    - Data Joining
    - Event Detecting
    - Data Aggregating
- **OLAP component:** this component possesses all the OLAP capabilities of StreamOLAP for insightful analysis. It uses the in-memory results from the stream-processing component to further compute the results at various aggregation levels that are not defined in the registered queries. Therefore, it can provide more insightful analysed information.

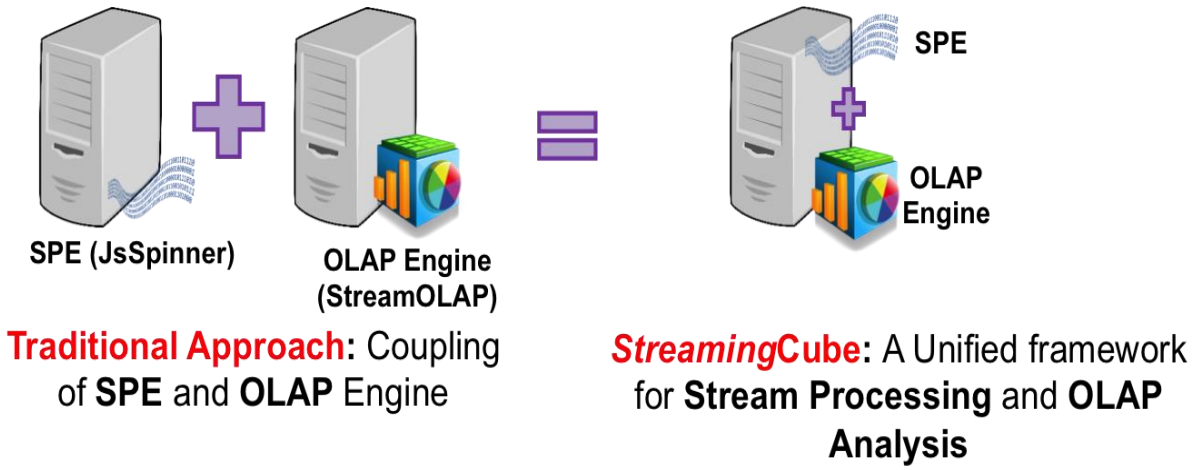


FIGURE 4 WHAT IS STREAMINGCUBE?

3.1.2 Implementation details and Internal Architecture

3.1.2.1.1 Architecture and Query Processing

StreamingCube is a unified framework capable of processing high-speed data streams and performing interactive OLAP analytics on it. In the following, we will briefly talk about its architecture and query processing.

Figure 5 depicts the main components of StreamingCube. Users can either register a simple stream processing query to perform ordinary streaming operations, i.e. selection, filtering, aggregation, join, etc. or a stream processing query with a cubify operator to create and maintain OLAP lattice in addition to ordinary stream processing to support different OLAP operations, i.e. roll-up, drill-down, etc. The presence of cubify operator in the query causes StreamingCube to generate a lattice structure where each node represents an OLAP query. A lattice may contain a large number of nodes especially when the number of dimensions is large. Materialisation of all the nodes requires huge memory and is expensive to compute, therefore only a few nodes are usually materialised while the queries related to non-materialised nodes are answered from the nearest materialised nodes. The node at the lowest granularity is always materialised to enable StreamingCube to answer all the OLAP queries. The dimensions information along with their hierarchies and the lattice nodes to materialize need to be provided by the end user as a configuration file, which is supplied as a parameter to the cubify operator.

The registered query is received by the Query Manager. The Query Parser translates the query into query intermediate representation and sends it back to the Query Manager. The Query

Manager forwards the intermediate representation to the Query Plan Manager, which generates a query execution plan. The query execution plan is in the form of DAG (Direct Acyclic Graph) of query operators, which is sent to the Operator Scheduler. The Operator Scheduler executes DAG by selecting one operator at a time. The Wrapper Manager with the help of Schema Interpreter parses the input data streams into elements (StreamingCube internal data representation). The operators accept data elements, process them, generate query results and send the results continuously to the end user and/or aggregate the results based on the nodes selected for materialization. Once a continuous query including cubify has been submitted, users can issue OLAP queries for interactive OLAP operations including roll-up and drill-down.

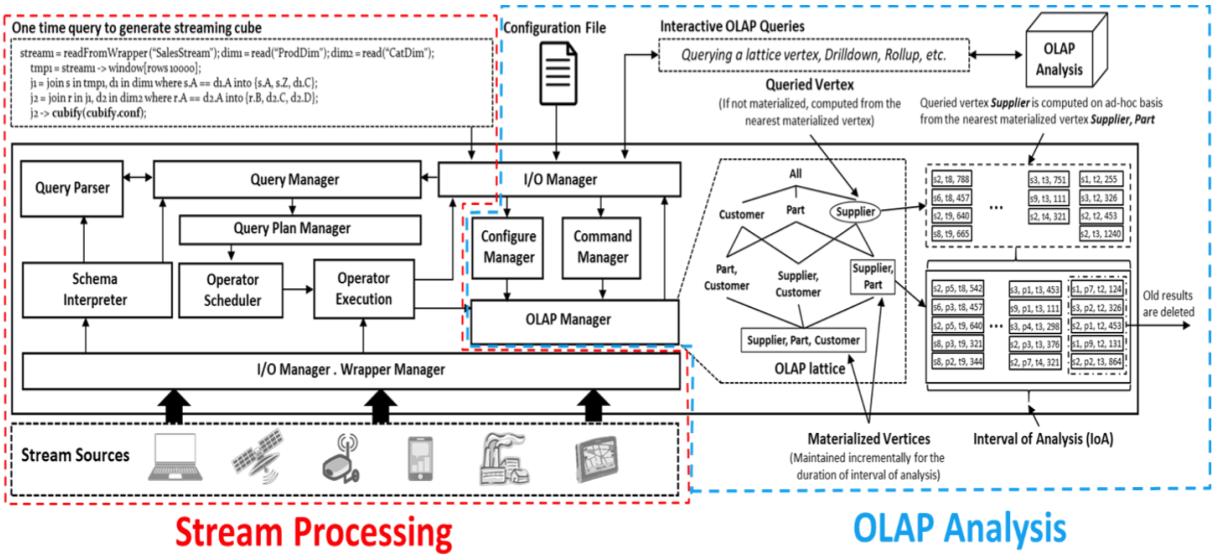


FIGURE 5 STREAMINGCUBE ARCHITECTURE

3.1.2.2 Incremental Computation and View Maintenance

Computation, which updates the output incrementally instead of re-computing everything from scratch for successive runs of a job with input changes, is called incremental. Similarly, algorithms that compute changes to a view in response to changes to the base relations are called incremental view maintenance algorithms[6].

To achieve incremental computation in StreamingCube, each stream tuple in the StreamingCube is additionally tagged as either an insertion (+) or deletion (-). In our unified framework, the tags are used to incrementally maintain operators' synopsis and to answer CQL, but also to maintain the OLAP materialised views incrementally, in addition to operators' synopsis.

For instance, when a new stream tuple s is read by StreamingCube it is appended with a timestamp t and a "+" tag, thus forming an element e of the form <s, t, +>. It inserts element e in the window operator's synopsis. On the other hand if an old element e' expires due to the window size, it is removed from the synopsis. The window then outputs elements <s, t, +> and <s', t, –>, which are sent to the downstream operators' synopsis to reflect the addition and deletion of elements e and e' respectively. In the similar fashion, OLAP materialised views are updated incrementally in StreamingCube.

3.1.3 Interfaces and Integration

6 A. Gupta, I. S. Mumick and V. S. Subrahmanian, "Maintaining views incrementally", Proceedings of the 1993 ACM SIGMOD international conference on Management of data, p.157-166, May 25-28, 1993, Washington, D.C., USA

3.1.3.1 *Interfaces*

A Web user interface of this system is provided in order to allow users to submit the queries and extract results at various aggregation levels (e.g.: roll up, drill down, sum, min, max, etc.). The results are in form of tables, charts, and graphs. In the context of BigClouT and smart city management, it can provide insightful analysis of city data so that administrators of a smart city can monitor current city status more efficiently.

A Web user interface of this system is provided so that users can submit queries to StreamingCube system as shown in Figure 6.

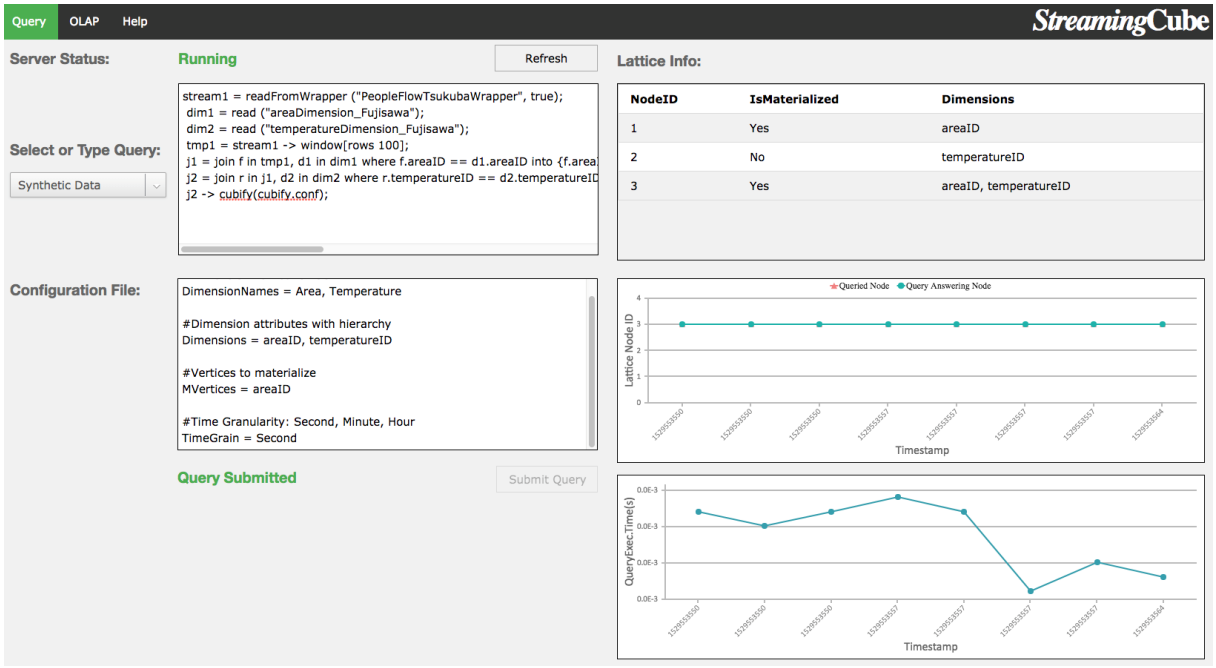


FIGURE 6: INPUT QUERY

After queries are submitted to the system, the Web user interface is used to extract results at various aggregation levels. The results are in the forms of tables, charts, graphs, and interactive city map.

The overview of the analytical result visualisation is shown in Figure 7. The results can be rolled-up and drilled-down by respectively increase or decrease the number of dimensions. Similarly for the time dimension, by respectively select hour, minute, and second, the results are aggregated and updated in real time per hour, minute, and second respectively.

The table in Figure 7 (top left) shows the analysis results in real time of the selected dimensions (e.g., area) w.r.t. the time (e.g. second). The chart on the top right visualises the results based on both the selection dimensions (e.g., area) and time. It shows the percentage of the combined dimensions of all valid data that has been received so far. To meet different needs of the users, StreamingCube provides a wide range of charts for interpreting the results. All available charts (Bar, Column, Area, Spline, Pie, and Doughnut) are shown in Figure 8. The bar chart on the bottom left (Figure 7) shows the analysis results w.r.t. the time dimension regardless of other dimensions.

The interactive map (bottom right in Figure 7) visualises the results to an interactive Google map, which is explained in Figure 9.

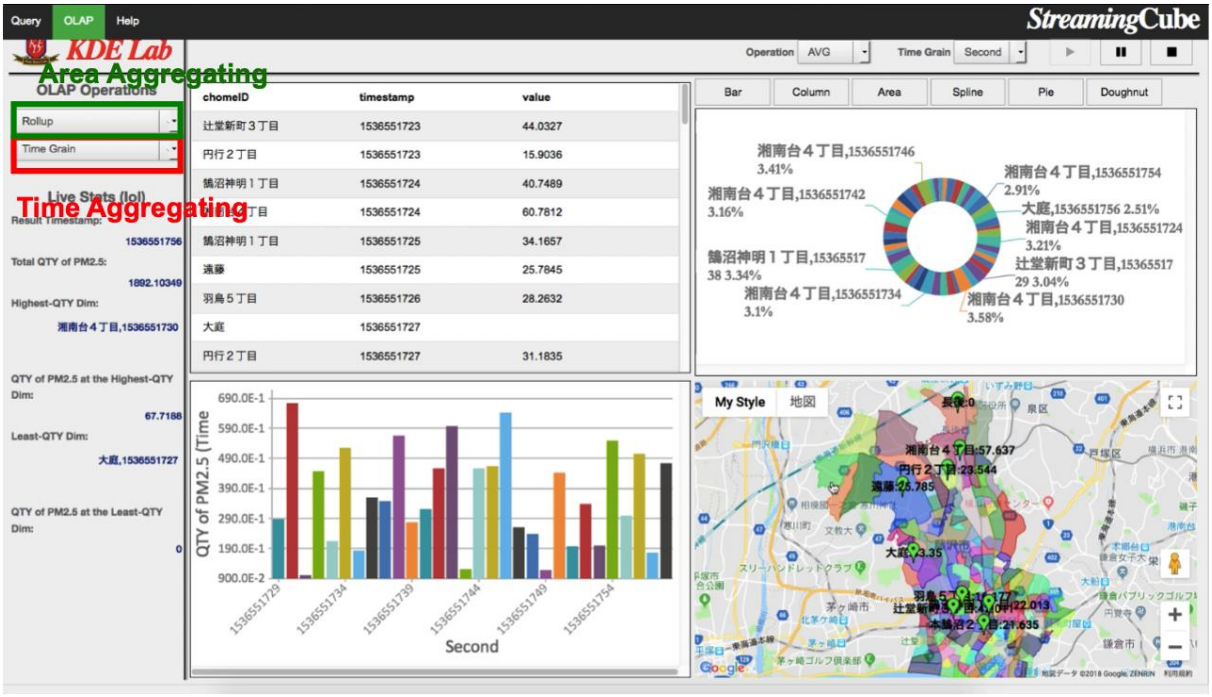
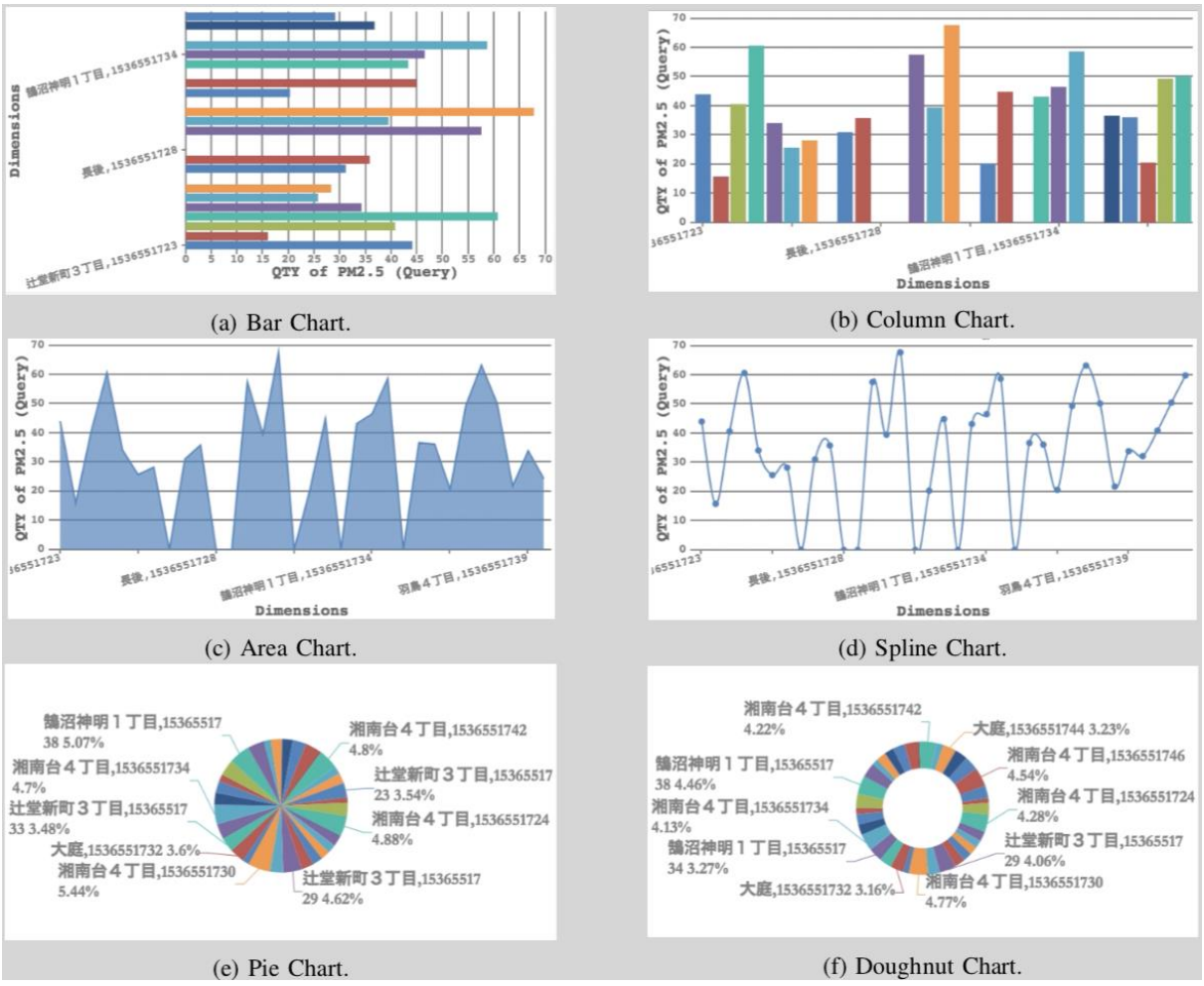


FIGURE 7: OUTPUT SHOWING THE AMOUNT OF PM2.5 WITH RESPECT TO DIFFERENT AREAS



(a) Bar Chart.

(b) Column Chart.

(c) Area Chart.

(d) Spline Chart.

(e) Pie Chart.

(f) Doughnut Chart.

FIGURE 8 VISUALISATION OF AVERAGE PM2.5 BY CHARTS

Figure 9 shows the results on the interactive city map with respect to the chosen dimensions (e.g., area). Note that, different areas are highlighted in different colours. The areas with location markers represent the active areas with analysis results. The labels of location markers on the map consist of two pieces of information, which are separated by colon: (1) Area name (left-hand side of colon), and (2) Analysis result values (right-hand side of colon). In addition, the results can be drilled down further at different timestamps by clicking on the location marker. This visualisation of analysed results is useful because it is easy to understand and interpret by the average city citizens, and the detailed aggregating results can be viewed very conveniently.

This application can also track the real-time movements of moving objects, such as people, cars, etc. (right hand-side in Figure 9). Different colour-lines represent the routes of different moving objects. The icon is placed on the current locations.

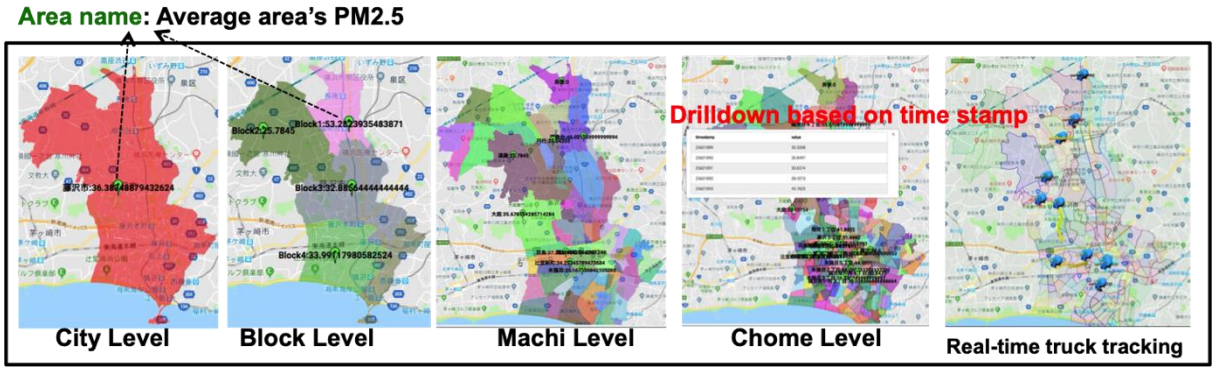


FIGURE 9 RESULTS IN DIFFERENT AREA AGGREGATING

3.1.3.2 Integration

I/O manager of StreamingCube is responsible for dealing with and accepting/sending data streams from the City Resource Access module of the BigClouT architecture. City data from the City Resource Access module is accessible by either HTTP request or REST API. Thanks to the well-defined I/O manager, StreamingCube can connect with City Resource Access module with minimum customisation to the I/O manager by either:

1. **Direct Connector:** Directly add functions to the I/O manager of StreamingCube to read/access city data from the City Resource Access module.
2. **Socket Stream Connector:** The City Resource Access module directly sends the city data to StreamingCube.

Both approaches can be done either in a synchronous way (client/server mode) or in an asynchronous way (publish/subscribe mode) by requesting or subscribing for city data from the City Resource Access module.

Notice that, the above methods require the registration of schema of the city data to the I/O manager of StreamingCube. The schema is in JSON format textual document. In addition, I/O manager is also responsible for dispatching the output of StreamingCube to other logical subcomponents of the City Data Processing.

3.1.4 Performance, Evaluation and Stress-tests

StreamingCube can efficiently process large-scale city data. Stress tests were performed and the results are shown in Figure 10. The tested queries involve joining two different streams. As it can be seen, StreamingCube (Smart Scheme), two execution modes of which are shown in the figure,

can efficiently process data streams with more than 100k tuples/s, which is good enough for real-life big data processing for BigClouT.

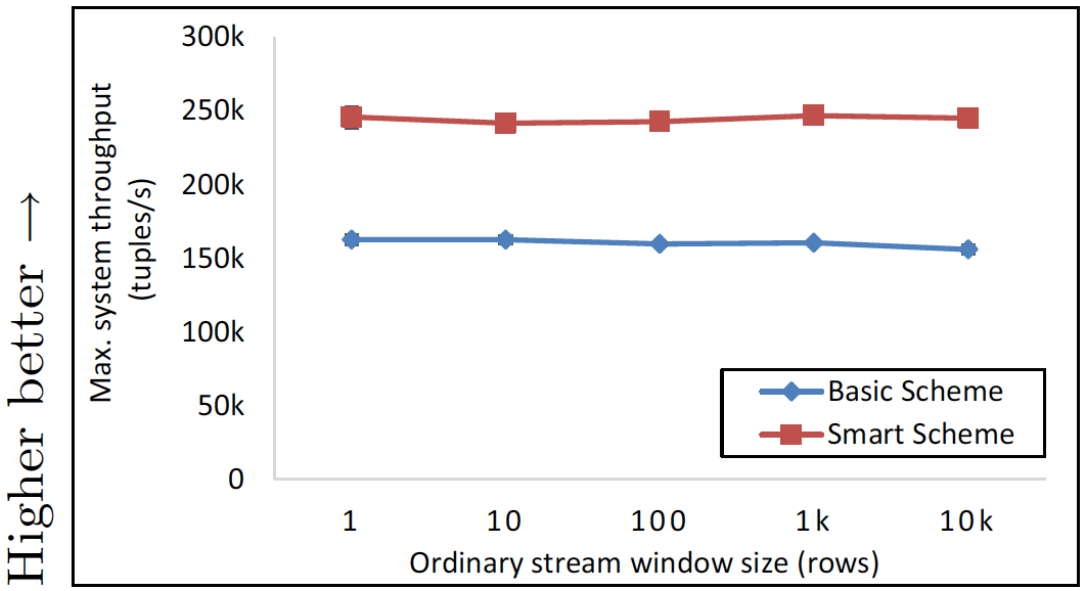


FIGURE 10 MAXIMUM SYSTEM THROUGHPUT EVALUATION

3.2 KNOWAGE

KNOWAGE[7] is a suite for Business analytics, it is freely available by downloading the Community Edition (KNOWAGE CE) or it is possible to exploit its advanced features by subscribing to the Enterprise Edition (KNOWAGE EE). This section of the document refers to KNOWAGE CE version.

KNOWAGE allows to interact with traditional data sources and Big Data sources in order to perform analysis such as data extraction and correlation. The visualisation of an analytical process is performed by defining a so-called Analytical Document. Analytical Documents group under a common concept the different types of documents that can be developed with KNOWAGE. An example of Analytical Document is a Cockpit which is used to build dynamic dashboards. Every Analytical Document has its own peculiarities and must be created and configured following a specific wizard. More details can be found in KNOWAGE CE's manual[8].

3.2.1 Description of Functionalities

The aim of this subsection is to provide an overview of the main functionalities offered by KNOWAGE. In particular, subsection 3.2.1.1 describes the main analytical functionalities such as Data Source Definition, Dataset Definition and the definition of Filters. Subsection 3.2.1.2 describes the visualisation functionalities giving some insight about the cockpit document definition.

3.2.1.1 Analytical Functionalities

In order to perform analysis, KNOWAGE mainly relies on the concepts of Data sources and Datasets.

A Data Source is a connection to a system, such as a DB, used by KNOWAGE to retrieve datasets. KNOWAGE supports SQL and NoSQL DB and Big Data sources (for instance: MySQL[9], MongoDB[10] and Apache Hive[11]). One of the advanced Data Sources features provided by the suite is the possibility to set a data source as an internal cache that is used to store the intermediate analytical results in order to ease the visualisation or further analysis on the data.

A Dataset is the portion of data used to perform analysis. Datasets can be retrieved by querying the previously defined Data Sources following the specific query syntax or from external data providers, for instance by uploading a CSV file or by retrieving the data from a RESTful API. Advanced features related to datasets are the explicit persistence of their data in the cache or the application of the pivoting transformation. Moreover, KNOWAGE allows to create a Federation of Datasets combining two or more heterogeneous datasets, coming from different data sources or external data providers, in a common model. Specific Dataset Types supports the definition of parameters. This functionality gives the chance to guide the analysis by filtering, for example, the data used in an Analytical Document dynamically.

Further details are provided in "D3.2 Big Data Analytics Framework report –first release" and "D3.3 Big Data Analytics Framework Prototype – Demonstration".

3.2.1.2 Visualisation Functionalities

[7] https://www.knowage-suite.com
[8] https://knowage-suite.readthedocs.io
[9] https://www.mysql.com/
[10] https://docs.mongodb.com/
[11] https://cwiki.apache.org/confluence/display/Hive/Home

This section is intended to give a general overview of the Cockpit document describing its main features and the widget it provides.

A Cockpit document allows a user to build complex, dynamic and multi-sheet visualisation of data by simply creating and configuring different widgets, defining the associations among them so that by clicking on a widget, other widgets will dynamically be updated. In order to start building the cockpit the user should choose one or more datasets to be used by the widgets and manage the association among their fields, if needed. Among the several widgets it provides, one of the most important is the Chart, KNOWAGE provides several charts in particular: Bar, Chord, Gauge, Heatmap, Line, Parallel, Pie, Radar, Scatter, Sunburst, Treemap and Wordcloud. Every chart can be customised following the specific needs of the analysis.

The Map widget allows to visualise data in a map (Figure 11). In order to be used in a map widget, it is mandatory that the Dataset has a field configured as Spatial Attribute. This can be a comma separated string with latitude and longitude of a point or a complex GeoJSON feature.

FIGURE 11: KNOWAGE - MAP WIDGET

The other datasets fields are used to display measures in the map, if a field is configured as a Measure, or to show the details of the feature in a popup window, if a field is configured as an Attribute or a Measure. An example of a complex Map widget is depicted in Figure 12, where a heatmap is built taking advantage of KNOWAGE visualisation functionality.

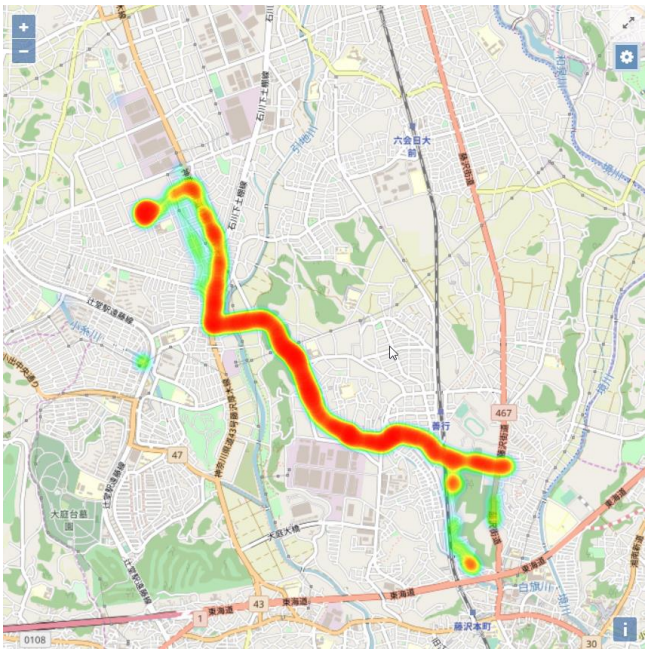


FIGURE 12: KNOWAGE - MAP WIDGET WITH DATA

3.2.2 Implementation details and Internal Architecture

KNOWAGE logical architecture (Figure 13) is layered on three main levels:

- **Delivery Layer:** which, manages the access to KNOWAGE's functionalities through its GUI or its RESTful APIs.
- **Analytical Layer:** which provides analytical features and capabilities, managing the access to the platform in a role-based mode.
- **Data Layer:** which manages the access to data through the definition of Data Sources and Dataset, leveraging the access to SQL, NoSQL and Big Data data sources.

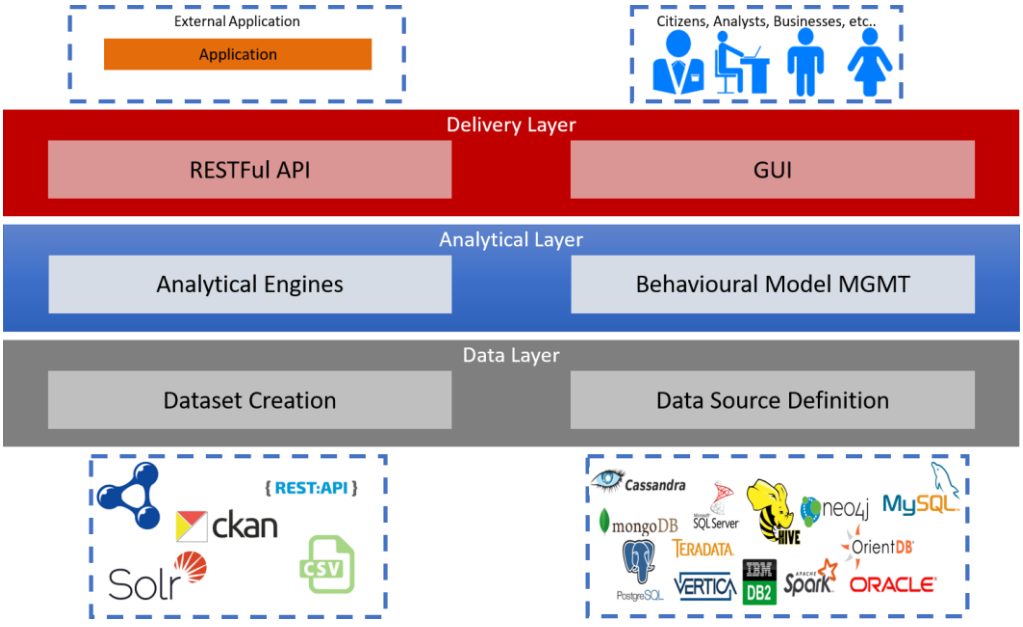


FIGURE 13: KNOWAGE LOGICAL ARCHITECTURE

The framework is developed with Java and its front-end application takes advantage of AngularJS[12] framework. Connection with Data Sources is managed with Hibernate[13] framework. The framework used to produce charts is Highcharts JS[14] or Chart.js[15], depending on the library chosen during the installation. Resulting charts could differ depending on the library.

3.2.3 Interfaces and Integration

KNOWAGE's functionalities can be accessed through its GUI. Depending on the role of the user and the authorization assigned by the administrator of the platform, the logged user would be able to manage or use different KNOWAGE's module and process. The administrator of the platform will be able to manage the entire platform managing, for instance, its Behavioural Model.

The Behavioural Model is one of the most important topics in every KNOWAGE instance. It is based on four main concepts:

- **User profile:** which allows to define the user's roles and attributes;
- **Repository rights:** which allows to define the user's rights configuring the accessibility to documents or functionalities of the platform;
- **Analytical drivers:** which allows to define the portion of data of a document can be shown to a user depending on his/her role in the platform;
- **Presentation environment settings:** which allows to define how a user can reach and execute his/her documents.

KNOWAGE gives the chance to an administrator to define several roles in the platform, configuring a specific Behavioural Model for each role. For instance, Figure 14 shows, in the left panel, the defined roles in a demo instance and, in the right panel, the specific authorization assigned to the role.

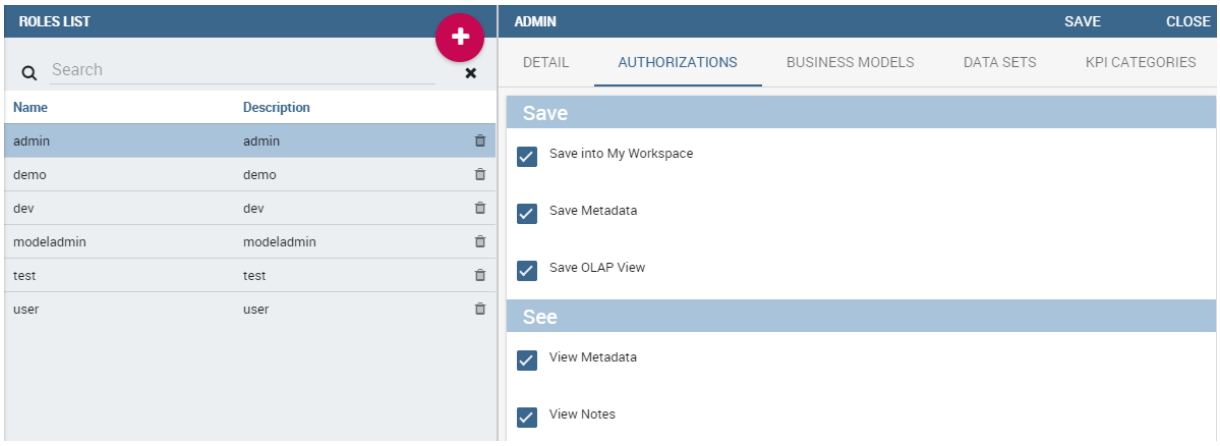


FIGURE 14: KNOWAGE - ROLES AUTHORIZATIONS

Every custom defined role must belong to a specific role type. A role type is a high-level categorization used by the platform to map specific functionalities of the platform to the custom defined role.

[12] https://angularjs.org/
[13] http://hibernate.org/
[14] https://www.highcharts.com/
[15] https://www.chartjs.org/

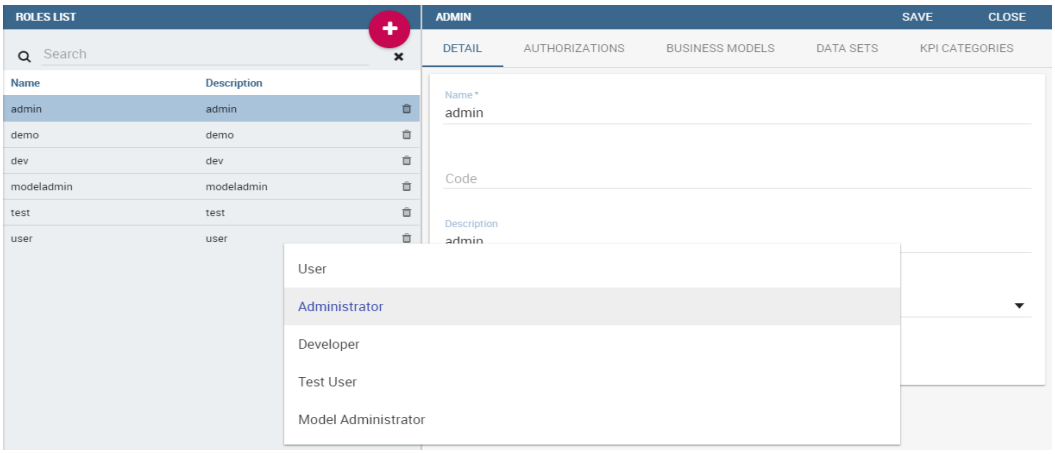


FIGURE 15: KNOWAGE - PREDEFINED ROLE TYPES

Any additional details about Behavioural Model are provided in the official documentation[16].

KNOWAGE exposes also RESTful APIs in order to, for example, create, update or list dataset and documents. The full list of the available APIs and their details are accessible in a dedicated Apiary[17].

3.2.4 Compliance Tests

This section of the document covers the conformance and integration tests performed using both WP2 and WP3 assets. Thanks to the several Dataset types, KNOWAGE is able to connect with:

- **BigClouT's Data Lake**: this component is part of WP2 and, to use its data for analysis and visualisation in KNOWAGE the CKAN connector is used;
- **Edge Storage**: this component is part of the WP2 and the connection to this asset is performed taking advantage of its CDMI RESTful interfaces. Indeed, KNOWAGE has a specific REST connector able to retrieve data from web services;
- **DeepOnEdge**: this component is part of WP3. The integration with this asset is performed taking as input the CVS file it produces, load this file into KNOWAGE thanks to its FILE connector and use this file for further analysis and visualisations;
- **Recommendation Service**: this asset is part of WP3. As for the Edge Storage component, the Recommendation Service exposes its functionalities through RESTful APIs. By using KNOWAGE's REST connector it is possible to query the Recommendation Service and to retrieve the data.

In order to speed up analysis and visualisation, KNOWAGE caches the Datasets data and metadata avoiding retrieving data in real time. This feature is helpful for large dataset, as the ones retrieved from the BigClouT's Data Lake. KNOWAGE's cache cleaning operation is temporized and the time interval can be configured through a configuration parameter. This feature deletes every cached version of a dataset older than the time interval, so if a user would use the cleaned dataset, KNOWAGE would have to reload the dataset from the data source. In order to avoid this behaviour one of the advanced Dataset's feature is the persistence of the dataset in a specific database configured for this purpose. By persisting the dataset, the framework will use its persisted version speeding up the analysis and the visualisations.

[16] https://knowage-suite.readthedocs.io/en/latest/administrator-guide/behavioural-model.html
[17] https://knowage.docs.apiary.io

FIGURE 16: KNOWAGE - RECOMMENDATION SERVICE DASHBOARD

Figure 16 shows an example of dashboard built taking advantage of the dataset retrieved through the Recommendation Service. This example scenario depicts a hypothetical dashboard to check information about energy consumption in houses. The right panel helps to choose the value of several parameters: the user, the day and the hour. By clicking on the execute button KNOWAGE executes a real time query on the Recommendation Service component retrieving the requested data and visualising the results.

The CSV daily files produced by DeepOnEdge are analysed and used to create a dynamic dashboard that helps Fujisawa municipality officers to check the most damaged areas in the city in order to schedule the maintenance operations. The punctual measures are aggregated, classified and used to create the dynamic dashboard depicted in Figure 17.

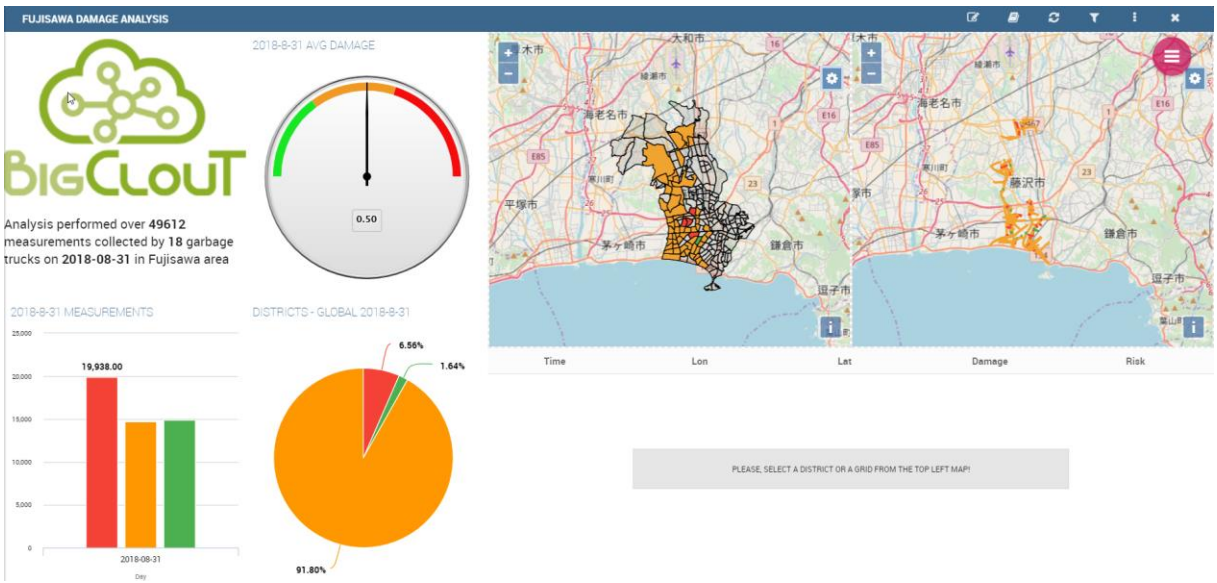


FIGURE 17: KNOWAGE - FUJISAWA DAMAGE ANALYSIS DASHBOARD

3.3 DeepOnEdge & GAnonymiser

DeepOnEdge is a tool that enables edge devices to analyse road damage condition with low computational resources. The specific component has been presented with a lot of detail in "D3.2 Big Data Analytics Framework report" as well as "D3.3 Big Data Analytics Framework Prototype – Demonstration".

In addition to DeepOnEdge, in this version of the document we present GAnonymiser, a component that removes privacy-related objects from captured images on edge devices. This section presents in detail GAnonymiser and how it is related to DeepOnEdge.

3.3.1 Description of Functionalities

GAnonymiser automatically detects and removes objects related to privacy (such as persons and cars) from an image. GAnonymiser composes of two machine learning networks: a privacy detection network and an object removal network. The privacy detection network detects objects related to privacy such as persons and cars from an image. The object removal network removes the detected objects naturally as if they did not exist in the first place. To combine these two networks, we developed an Edge Shift Padding Layer (ESP) and a Global Feature Padding Layer (GFP) between the two networks in order to adapt GAnonymiser to the characteristics of the images of the city and enable GAnonymiser to remove objects more "naturally". ESP helps GAnonymiser to remove objects placed at the edge of an image and GFP helps GAnonymiser to remove large objects in an image. In our experiment, we adapt GAnonymiser to anonymise videos (5246 images) taken from a car running in Fujisawa city, Japan. The implemented GAnonymiser detects and removes persons, cars, buses, bikes, and bicycles as the targets of objects related to privacy based on the features of the video. We confirmed that our method contributes to removing privacy-related objects.

Background of GAnonymiser

By analysing a large dataset of urban video images, we can understand context of the city. For example, that is the case with DeepOnEdge (as described in previous deliverables) where it becomes possible to **automatically** detect blurred lines and signs on the road. DeepOnEdge also proved that leveraging video images of dashcams on public city garbage trucks can cover the entire area of city. Moreover, the method of detecting damage on a road through smartphone has also been proposed. The method used the light model MobileNet SSD as an object detection network and trained the model with the dataset annotated with damages on a road. Furthermore, the method proposed the smartphone application that automatically detects damages on a road using its camera in real time and also opened the dataset used in the model training to the public. These methods of analysing urban images enable us to automatically monitor urban infrastructure. Not only urban infrastructure context, but also seasonal changes can be analysed by urban video images. For example, Morishita et al.[18] proposed SakuraSensor which extracts and shares information of flowering cherries along roads, by using smartphone cameras on cars. These existing works presented usefulness of urban video images - we can extract various city context by sharing and analysing urban video from many vehicles.

Although we can analyse video images on edge side and share only the result of the analysis, some application scenarios such as efficient road damage management require to share video images themselves. After asking city officers of the road management section in Fujisawa city, Japan, about how automatic road damage detection technique could be useful for actual road management, it was realised that automatic road damage indexing of the whole city is very useful,

[18] https://dl.acm.org/citation.cfm?id=2804273

but it is still necessary to check actual road images by human eyes for deciding which road should be repaired. Thus, we still have necessity to share urban video images. Moreover, access to this kind images video image datasets could be valuable for various researchers that want to evaluate various image analysis methods.

When sharing urban video images, the main obstacle is the privacy concern. Many companies and cities own urban video data, and the data must be useful for opening/sharing to analyse city infrastructure and seasonal change of the environment. However, since urban video contains privacy data such as persons and cars, it is difficult to open/share the data widely. In addition, in Europe, the General Data Protection Regulation (GDPR) recently came into force. Therefore, a privacy protection mechanism for urban video images is crucial to enable video sharing.

There are two main steps to protect privacy related data included in a video. The first step is to remove privacy-concern data in the video. This can be achieved by removing video frames which contain privacy data, or by anonymising privacy data such as masking with mosaic pixels. The second step is to secure the video sharing process. This can be achieved by using encrypted communication and enhancing traceability. In the following subsections we focus on the first step - how we can remove privacy data from urban video images. This step is considered the most important one, since it reduces privacy invasion risk as much as possible: even if the video data are stolen in the video sharing process, the actual privacy related data are removed.

Anonymisation Level

To remove privacy data from the urban video images, we considered the anonymisation method by referencing the definition of Chinomi et al[19]. Chinomi et al. defined the anonymisation of images as the process of reducing the concreteness of objects related to privacy. They also defined several anonymisation levels such as cases where it is possible to mosaic or black out objects which are related to privacy data. In their definition, the abstract degree should be carefully selected, considering the opinion of the user of images and the subjects of images. However, urban images include an unspecified large number of subjects, and as such, we cannot consider the opinions of all of them. Moreover, in the case of monitoring the city's infrastructure, objects related to privacy such as people and cars are not required. Meanwhile, according to the definition of Chinomi et al, removing objects from the images is the most anonymised level (transparency). Therefore, based on their definition, our goal is to make objects related to privacy transparent by removing them naturally.

Urban Video Anonymisation Task

Simple anonymisation method cannot be adopted to mobile urban video images. One of the major anonymisation methods is utilising background subtraction. By subtracting the current video frame from the previous video frame, we can acquire an area where something changed and where nothing changed. Then, the changed area is replaced with the same area from another frame where nothing appears. While the background subtraction method suits for the video from a fixed-point camera, it does not work at mobile urban videos. This is due to the characteristic of the video being taken by a mobile camera, which differs from that of a video taken by a fixed-point camera: the background is not always the same. Thus, in order to anonymise the various urban videos including mobile cameras, we have to adopt the way which uses not multiple frames but only one frame.

Additionally, there is another characteristic of the urban videos: the kind, the size and the position of the object which appears in it might be very diverse. This characteristic makes it difficult to

[19] Chinomi, K., Nitta, N., Ito, Y., Babaguchi, N.: Prisurv: privacy protected videosurveillance system using adaptive visual abstraction. In: International Conferenceon Multimedia Modeling. pp. 144–154. Springer (2008)

apply the way to detect and remove the target objects such as the pattern/template matching method. Therefore, we have to adopt the method which is flexible to detect and remove various privacy-related objects. After detecting and removing the objects, we have to embed an appropriate image to be natural in the area where the objects have appeared.

In summary, we treat the urban video anonymisation as an object removal and background completion task.

3.3.2 Implementation details and Internal Architecture

In order to tackle the privacy-related object removal and background completion task, we propose an anonymisation method, called GAnonymiser, which consists of two parts of neural networks. In this section, we explain the architecture of GAnonymiser and the new layers that we propose to complete the background to be more natural.

Network Architecture

The architecture of GAnonymiser is shown in Fig. 22. In order to detect the target objects from the input image (which might violate the privacy) we adopt the deep neural networks Single Shot Multibox Detector (SSD)[20]. SSD is one of the popular models that can detect the object with high accuracy. More specifically, we select SSD512 which is the variant SSD model and performs better than any other. Since the target objects are general and those are contained in the PascalVOC dataset, we use the model weights which are trained by PascalVOC.

After the target objects are detected, GAnonymiser replaces the area where the target objects is. Although there are a lot of completion methods using computer vision technology, such as PriSurv and PatchMatch, the result images are not realistic and are unnatural. On the other hand, the in-painting methods which are adopted by the deep neural networks succeed in generating more realistic and natural images. For GAnonymiser, we adopt the Globally and Locally Consistent Image Completion (GLCIC)[21] model, which is one of the most successful models in image completion.

GLCIC is based on generative adversarial networks (GAN) and consists of three networks: the completion network, a local discriminator network, and a global discriminator network. Since GLCIC requires an image and a corresponding binary mask for its input, GAnonymiser creates the mask based on the bounding boxes which are the outputs from SSD512. Then, GLCIC reconstructs the mask part of the input image based on the whole image and is trained by the procedure of GAN. The local discriminator assesses the quality of the mask part of the image which is completed by the completion network. Simultaneously, the global discriminator assesses the quality of the entire image which is completed by the completion network. The training is terminated when the discriminator networks cannot distinguish between the original input image and the image which is reconstructed by the completion network, that is, the completion network becomes able to reconstruct the mask part of the input image realistically and naturally.

In terms of object removal, it is significant to naturally reconstruct masks based on the various backgrounds of the images. Hence, for our GLCIC, we apply the model trained with the places dataset, which contains the pictures of the various places, so that it can reconstruct the mask more naturally.

[20] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
[21] Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) **36**(4), 107 (2017)

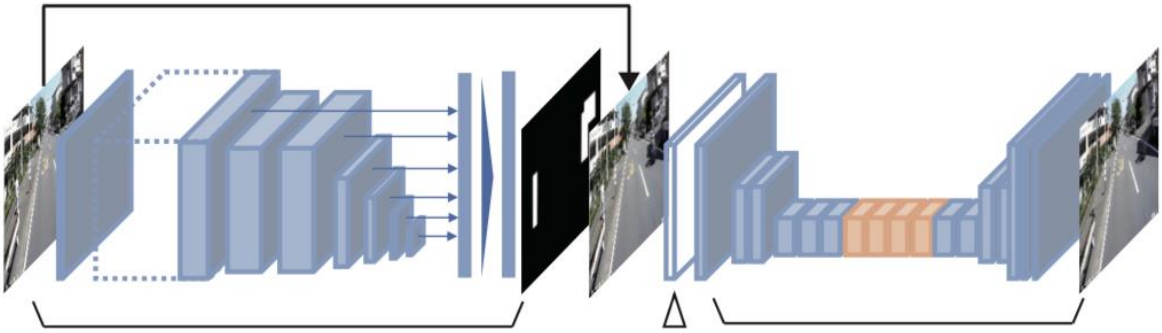


FIGURE 18: THE ARCHITECTURE OF GANONYMIZER

Auxiliary Context Layer

With the architecture of GAnonymiser, most of the urban videos are anonymised effectively. Nevertheless, there are some cases that fail to anonymise well; failure cases occur when the size of the mask is too large or it is placed at the edge of the image. The more the objects approach to the camera, the larger the size of the mask becomes. It is difficult to reconstruct these large masks since GLCIC reconstructs them by using the context of the areas surrounding the mask via convolutional operations; there is no information about the centre of the large mask for re-construction. In order to solve these problems, we propose the information padding layers, edge shift padding layer and global feature padding layer that are inserted between SSD512 and GLCIC.

Edge Shift Padding Layer

To reconstruct the mask which is placed at the edge of the image, we define the edge shift padding layer (ESP). ESP copies several pixel rows to the outside of an image. When the edge of the mask is not adjacent to the edge of the image, the pixel rows at the edge of the image are copied. By applying ESP, the input image is extended to the edge side, so that the mask can receive the information from all direction.

Global Feature Padding Layer

In order to cope with the large mask, we also propose a global feature padding layer (GFP), which has two steps. The first step is to resize the input image into small images so that the size of mask is enough to be able to reconstruct directly and naturally, and then reconstructed. In the next step, the part of a reconstructed mask is extracted as a mesh pattern to use for the information to be completed. GFP makes it easy for GLCIC to reconstruct the large mask thanks to the mesh pattern.

3.3.3 Interfaces and Integration

We implemented GAnonymiser as command-based executor with python language. With the Distributed Node-RED integration in BigClouT project, we leverage our implementation with Python enabler of Node-RED. Thus, we integrated GAnonymiser within Node-RED environment.

3.3.4 Performance, Evaluation and Stress-tests

In order to determine some parameters of pre-processing and present its effect, we conducted the preliminary experiments. In the preliminary study, we use a road image extracted from the dataset and a pure white image for clearly visualising the result. The size of both images is 400x400.

Edge Shift Padding. In order for ESP to effectively assist the generation of natural images from edge masks, we verify the following two points. First, we look for enough pixels on the edge for the GLCIC to reconstruct the edge mask. Next, based on that, we find the optimal number of pixels for edge padding In order to verify the above two points, we conducted two experiments. First, we performed GLCIC to the mask the distance of which is from 0px to 10px. When the distance is 2px or less, the reconstruction result of the edge mask is not natural due to the lack of surrounding area's information, while the reconstruction result is natural when the distance is 3px or more. Then, we performed each ESP in which the padding pixels are random, random pixels in an image, the pixel at the edge of an image, and the pixels on opposite side across the mask. ESP is applied to the mask the distance of which is 3px or less and we adopt 4px for the pixel of the edge padding so that we can generate a natural image from the edge mask.

Global Feature Padding. We need to examine the size of the mask where GL- CIC can directly generate naturally so that GFP can help GLCIC generate a natural image from a large mask. In this experiment, we directly reconstruct the large mask the size of which is from 50px to 200px at intervals of 10px using GL- CIC. The result was realistic enough when the size was 120px or less, while it was not when the size was 130px or more. Meanwhile, regarding the way of dividing the mask, when the method that finely divides the mask (in e.g. 16 divisions) is used, the influence of the lattice between divided small masks is too large. When the method coarsely divides the mask (e.g. in 4 divisions) the influence of the lattice is too small. Therefore, the GFP applied to the mask the size of which is 130 px or more and we adopted the 9 divisions to divide it, so that we could improve the reconstruction of a large mask.

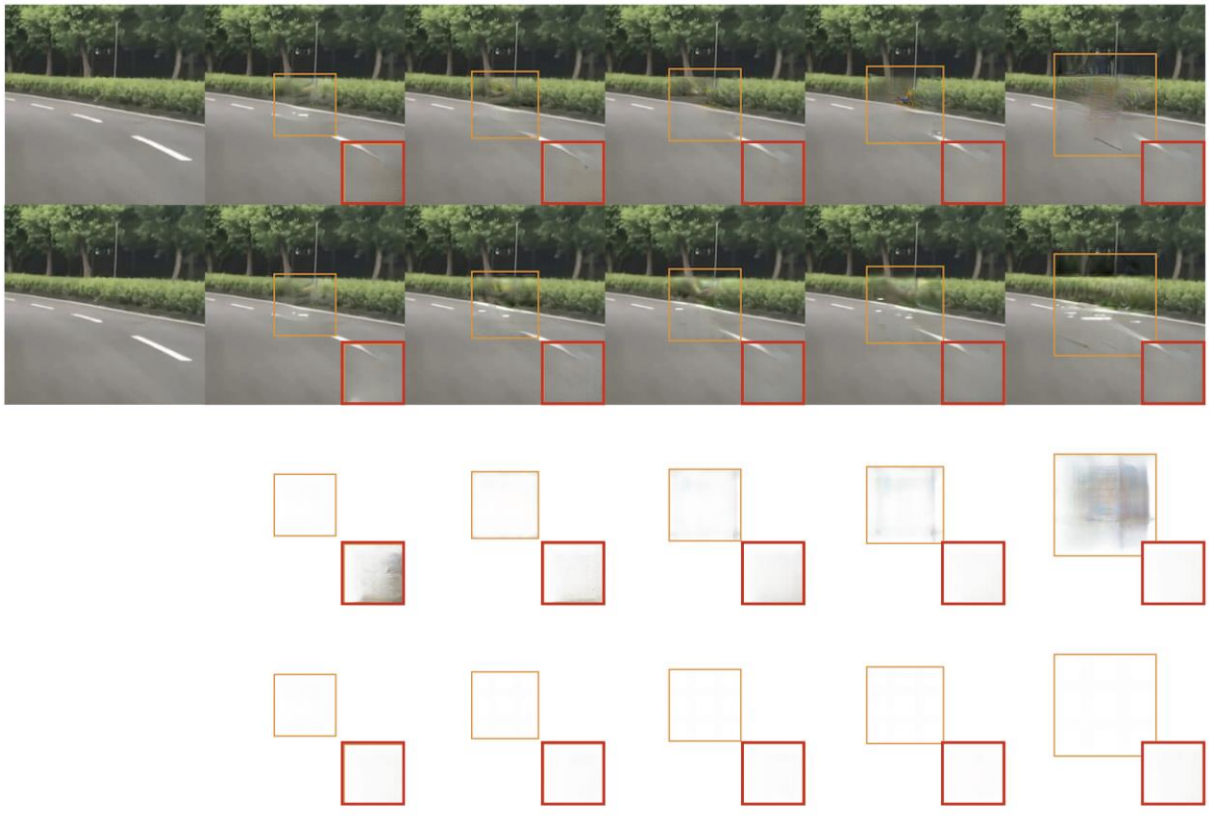


FIGURE 19: PRELIMINARY STUDY RESULT. FROM TOP ROW, RECONSTRUCTION WITHOUT ESP AND WITH ESP REPEATEDLY. THE LEFTMOST IMAGES ARE THE ORIGINAL IMAGE. THE IMAGE INCLUDES THE LARGE MASK (ORANGE) AND THE EDGE MASK (RED). FROM LEFT TO RIGHT, THE SIZE OF THE ORANGE MASK IS {120, 130, 140, 150, 200} AND THE DISTANCE BETWEEN IMAGE EDGE AND RED MASK EDGE OF {0, 1, 2, 3, 4}

Images for Anonymisation

In our experiments, the data are the images taken from an iPhone7 on a car running in the Fujisawa city, Kanagawa, Japan. The total number of frames used in the experiment is 5246, and the size of each image is 1080 x 1920 without resizing. We use the images taken during daytime and the evening, when the field is clear and there is risk of privacy invasion. We select persons, cars, buses, bicycles, and bikes as the objects related to privacy, considering the features of the urban images.

Results

Figure 20 and Figure 21 show the results of applying GAnonymiser to urban images during daytime and the evening. The results in Figure 20 show that GAnonymiser succeeded in removing persons and cars naturally from the image. ESP removed the objects appearing at the edge. Moreover, the GFP improved the reconstruction of the large mask. However, as shown in the left of Figure 21, the detection was difficult in the case that the object appears too large, or the object appears small. Although the GFP tackled the problem of reconstructing a large mask, the result of removing a large object is still blurry and coarse compared to the result of removing small objects. We could remove objects related to privacy, while there are some bad results where some objects are not detected or are not naturally removed. The experiment was conducted on an Intel Xeon CPU E 5-1680 v 3 @ 3.20 GHz.

FIGURE 20: THE RESULT OF APPLYING GANONYMISER TO THE URBAN IMAGES. TWO LEFT IMAGES IS IN DAYTIME AND TWO RIGHT IMAGES IS IN EVENING. THE UPPER ROW IS INPUT IMAGES AND THE LOWER ROW IS OUTPUT IMAGES.

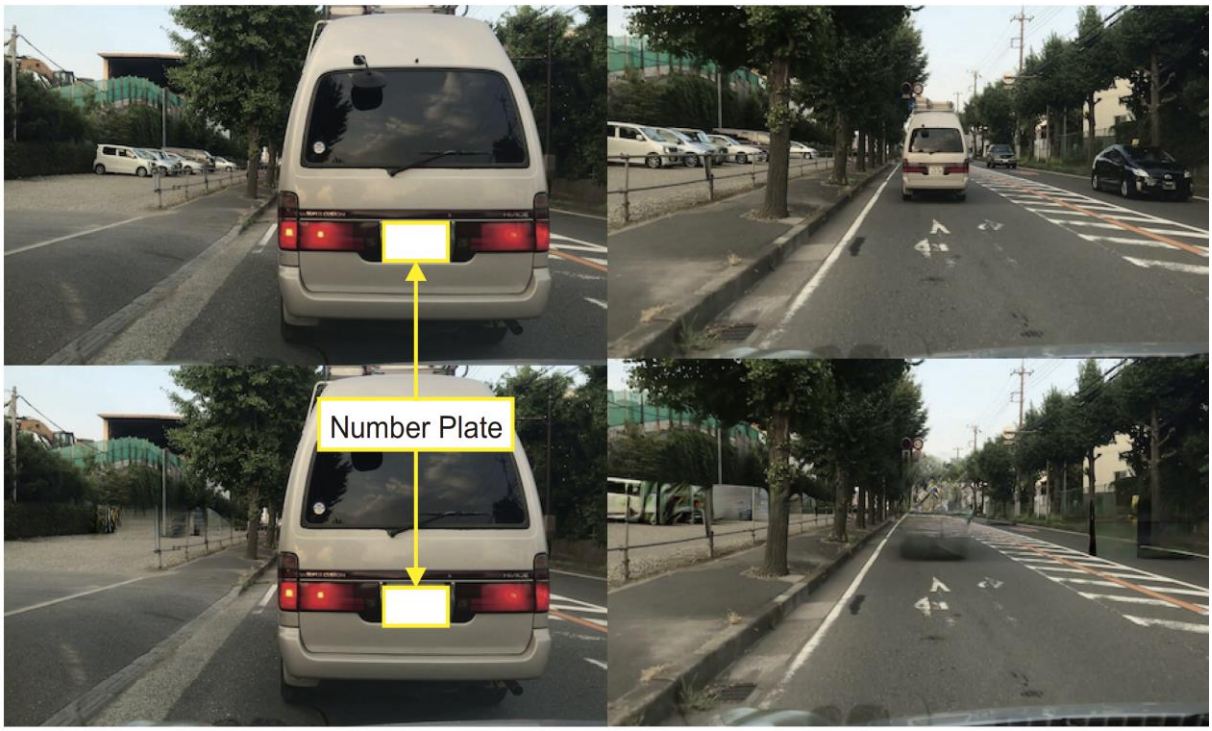


FIGURE 21: FAILURE EXAMPLES. (TOP) THE OBJECT WAS TOO LARGE, WHICH LEADS TO NOT DETECTING THE OBJECT. (BOTTOM) THE RECONSTRUCTION OF THE BIG CAR IN THE CENTRE IS MORE BLURRY AND COARSE THAN THAT OF THE SMALL CARS.

Discussion

In terms of the privacy-related objects detection, we can effectively detect multiscale objects related to privacy by using the highest accuracy SSD model. However, in the case that a privacy object is extremely close or far, it was difficult to detect objects related to privacy. Figure 21 presents such examples of such misses. When the object is far, detection failure is not a problem because the object does not appear clearly. On the other hand, when the object is close, privacy problems happen by failing detection. However, detecting a too close object is difficult, especially when the entire figure of the object is not visible. There is a possibility that adopting the segmentation method can solve this problem. Segmentation categorises the pixels into several classes so that it is more likely to partially detect objects. Therefore, it is likely to improve the detection accuracy by using segmentation instead of object detection or combining them, so integrating it to GAnonymiser is one of the future works.

In terms of the object removal, we could reconstruct most masks naturally except for too large masks. Moreover, the two auxiliary context layers help GL- CIC to reconstruct difficult masks naturally. Although reconstructing a large mask was improved by the GFP, the reconstruction results of the large mask parts are still blur and coarse compared to that of the small mask parts, as shown Figure 21. However, the models based on GAN have the limitation of the size of natural reconstruction. Kingma et al. have proposed Glow that can generate high resolution images[22]. Therefore, it is also part of future work to apply Glow image reconstruction in order to naturally reconstruct masks of any size.

[22] Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. arXiv preprint arXiv:1807.03039 (2018)

3.4 Recommendation Service

3.4.1 Description of Functionalities

This service provides recommendations to end users/citizens in several application settings such as transportation management, energy consumption, public safety, supply chain management, tourism services, air and sound pollution management and many others.

3.4.2 Implementation details and Internal Architecture

Recommendation Service can be applied in different domains. To better explain its implementation and its internal architecture, a scenario about a *smart-mobility* is taken into count.

As shown in Figure 22, the Recommendation Service consists of three independent components:

i) the Recommender Application with a Front-End and Back-End which handles the interactions with the user and delivers routing suggestions to the user i.e. shortest path and green path;

ii) the Neo4j Graph Database;

iii) the IoT Node-Red Flows component which wires together the different hardware devices, APIs and web services, connecting the distributed components, sensors into a common IoT application. The latter component mainly contains the two flows presented below.

It is important to underline that all three components are independent; this represents an important feature for possible extensions and scalability.

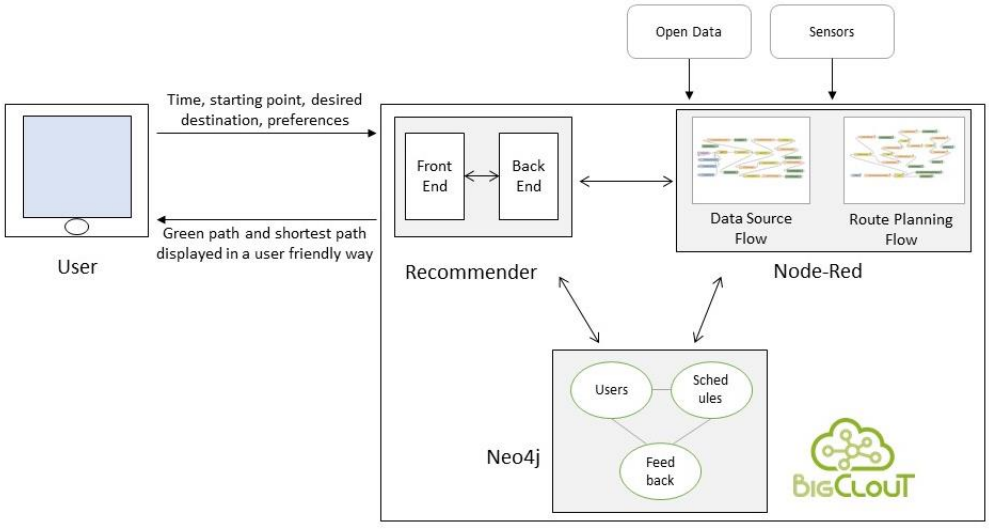


FIGURE 22 OVERVIEW OF THE PROPOSED ARCHITECTURE

Data Source Flow

The first Node-Red flow, called the "Data Source Flow", handles the input of the data from different sources. In order to effectively transmit data that are being exchanged between the sensors as well as the main system, the MQTT protocol[23] is utilised. MQTT is a lightweight publish and subscribe messaging protocol for use on top of the TCP/IP protocol, and is ideal for use with low power sensors with limited resources. MQTT is based on the principle of publishing messages and subscribing to topics. More specifically, in our scenario, sensors behave as clients who connect to

[23] http://mqtt.org/

a broker and publish messages to topics, while the broker enables the connection, acting as a common interface. The data in the MQTT broker are transmitted using a simple JSON format.

An example of a JSON message regarding air quality used in a smart-mobility scenario, its format and included information is presented below:

```
{
 fields:
 {
  datestart: "2019-01-01T00:00:00+00:00",
  no: 39.22,
  geo_point_2d:
  [
   double latitude,
   double longitude
  ],
  siteid: integer,
  location: "String description of the location of interest",
  nox: 132.83,
  no2: 65.15
 },
 geometry:
 {
  type: "Point",
  coordinates:
  [
   double coordinate1,
   double coordinate2
  ]
 },
 record_timestamp: "2019-01-01T00:00:00+00:00"

}
```

Different kinds of devices are being used in this implementation such as sensors.

We have implemented a Node-Red MQTT node to effectively gather, process and republish data to the connected services or brokers when required. Before actually forwarding them, the Data Source Flow performs a first level processing of the data in order to calculate specific failsafes for the data.

More specifically, a Generic Data failsafe is being calculated, counting the number of measurements we receive from each sensor in a specific timeframe, ensuring that they are above a certain numerical threshold, thus ensuring the validity of their frequency. In addition, a series of failsafes concerning the temperature data are being calculated, such as failsafes to locate measurements below the absolute zero or above a selected high temperature (70 ^{o}C), which would indicate a malfunction in the sensor.

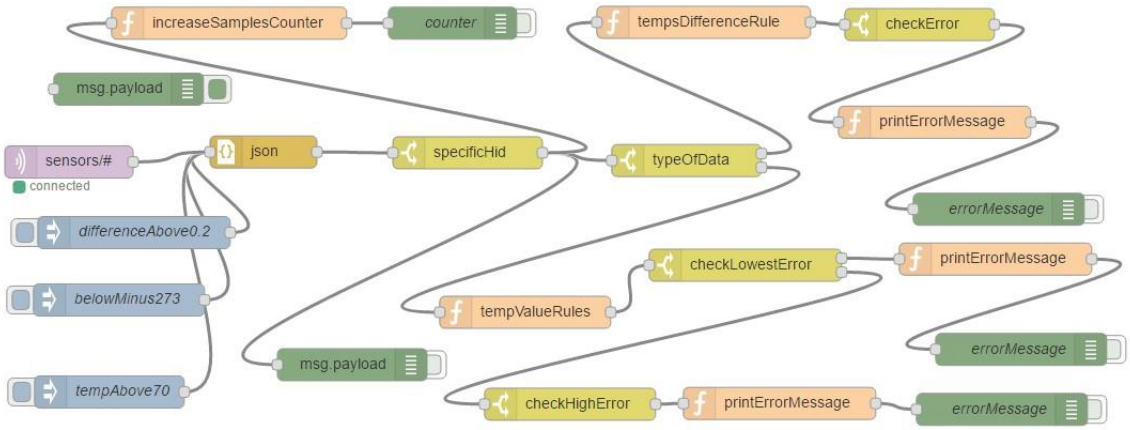


FIGURE 23: DATA SOURCE NODE-RED FLOW HANDLING INPUT FROM SENSORS AND OPEN DATA, COMPUTING AND MONITORING SPECIFIC FAILSAFES AND ALERTING THE USER ACCORDINGLY

Graph database Modelling - Heating schedule management

The implementation is based on a state-of-the-art smart city application, which is based on open data captured by the city IoT infrastructure and user generated content in terms of smart mobility and route planning and user's profile. Such an application is regarded as a smart city application for a particular city context.

The open data that are being deployed refer to:

i) weather conditions as collected by the weather stations (measuring wind speed, rainfall, humidity and temperature)
ii) weather forecast for a certain period
iii) air quality measurements such as NO, NO_x, NO_2
iv) road disruption data such as traffic jams in specific parts of the city in the form of a timestamp, some text describing the problem, geographical coordinates and severity
v) user preferences

The user provides details regarding the starting point and the desired destination. The system, taking into consideration these features, makes complex queries to a graph database to collect related information and produces personalized recommendations for the specific user, producing routes.

One of the most remarkable features of a graph database, which is also crucial for our implementation, is its effectiveness into handling big volumes of information with real-time

response to queries. Our system, based on a highly scalable graph database, processes in real-time simple and complex questions such as "What is the temperature on a specific day/time?", "What is the level of humidity and rainfall on a specific timestamp?" by rapidly traversing the graph while reading and processing values in nodes, relations and properties. Similarly, it handles other questions related to the historical data of users regarding past route recommendations and environmental conditions. Questions about environmental conditions are effectively handled, by traversing the graph as shown in Figure 24, where the ratings of users are depicted.

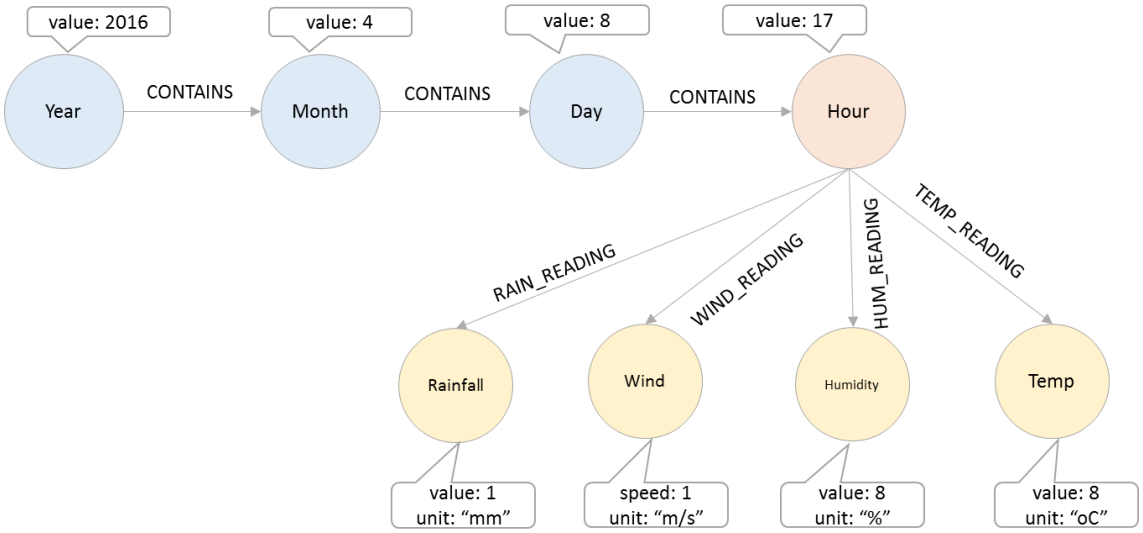


FIGURE 24: GRAPH DATABASE MODELING USING TIME TREES AND HANDLING BIG VOLUMES OF OPEN DATA COMING FROM THE CITY

Graph database Modelling - Overall modelling approach

A significant advantage of graph models is that they depict entities and relations in the same way as we think of them. Moreover, these models are forming a view on the queries we would like to implement in the graph database. The modelling process and approach in graph databases can be regarded as equivalent to the approach of creating graph structures that reflect the queries we would like to answer. "Users" or "Ratings" for example comprise the nodes of the graph, "names" are the attributes of the nodes, verbs such as "likes" depict the relations that link the nodes, and whatever refers to such verbs is regarded as the attributes of the relations. An interesting way to model time in Neo4j is through the use of time trees. In this approach, nodes represent years, months, and days, etc. while every node contains an attribute "value".

By forming the time trees, we can link particular events or other measured data on these trees. In our example, we link measurements of weather conditions (comprising of temperature, wind speed, rainfall and humidity).

The data have been fed into our graph database and linked together in an efficient and constructive way allowing the user to ask a multitude of questions such as: "what is the temperature", "what are the weather conditions", etc. At the same time users have been added in the database along with their preferences.

We begin the implementation of our system using Neo4j's query language Cypher, a declarative, SQL-inspired language for describing patterns in graphs visually. It allows us to state what we want to select matching a specific pattern, insert, update or delete data from our graph without

requiring us to describe exactly how to do it[24,25]. Firstly, we create the time trees, which constitute the keystone of our model. In order to achieve faster data retrieval and improved performance, we use indexes, a feature provided by Cypher, created once over a property for all nodes that have a label. Cypher automatically manages the update by the database whenever the graph is changed. Then, we populate our model by importing the temperature and weather data into the graph database linked to the suitable time nodes, adding additional integration layers when required e.g. an additional layer is required for wind level and speed. We should note that Neo4j offers great data visualisation options allowing us to view all the stored nodes, relations and information, giving real-time insights into how data nodes are related, facilitating the development and evaluation process.

The system provides a user-friendly way for users' registration in order to deliver path recommendation services. When a user is registered to the system, a new unique node is created with the user characteristics stored as properties. Afterwards, the user inserts preferences and historical data, and the corresponding nodes and relations are created in the graph database.

Recommendations

Our proposed recommendation service exploits similarity metrics among vector representations, users' preferences, graph analysis algorithms to deliver in real-time recommendations in the form of path recommendation.

The user receives the top path recommendations and chooses one to follow. For example, a resident of Bristol wishes to receive a path recommendation to move from his house to a point of interest. Our service provides the most suitable schedules after tracking the closest users, historical data, the number these schedules were applied and past evaluations. The user chooses the best one and afterwards optionally provides feedback in the form of a rating or suggestion for alteration if required.

Recommendation systems produce suggestions through various approaches, while they often take into consideration the dynamic context to establish relations among users and target objects. The production of effective recommendations is based on the comprehension of these relations and their quality as well. Graphs are perhaps the most suitable structure to represent dense connected data structures. By storing and studying these data using graph databases, an application is allowed to exploit and demonstrate in real-time the impact of users' actions and not just take advantage of predefined results of pre-existing data. A widely applied technique in collaborative filtering recommendation systems is identifying the closeness among different users with similarity metrics.

By executing queries in the graph database, path recommendations to reach the desired destination are returned:

i) the shortest path,
ii) the "green" path i.e. a path through the areas which have the lowest air pollution etc.
iii) a combination of the previously two mentioned approaches

The recommendation service exploits these data and delivers the top schedules for the particular destination.

[24] https://neo4j.com/developer/cypher-query-language/
[25] I. Robinson, J. Webber, and E. Eifrem. Graph databases: new opportunities for connected data. "O'Reilly Media, Inc.", 2015.

3.4.3 *Interfaces and Integration*

The Recommendation Service exposes a RESTful API, which allows the integration with other tools and services. The API manages the users' requests for recommendations and the connection with data sources, and handles user generated data. More specifically, through the API, some of the actions handled are:

- A user requests a recommendation.
- A user provides feedback after applying a received recommendation.
- A new user is registered.
- The information about a user is updated or deleted.
- Sensors to the data source flows are connected.
- Open data (for example, regarding weather conditions) are added/updated.
- Historical data (for example, regarding weather) conditions are received.
- Users requests to retrieve all recommendations they have received and the corresponding feedback they have provided.

3.4.4 *Performance, Evaluation and Stress-tests*

It is important for a recommendation service to maintain a reasonable response time and particularly the suggestion calculation to have a limited running time. It is very important, for large scale IoT deployments, especially in the context of smart city, to allow for an increased level of user experience. We have extensively tested the efficiency of our system for increasing number of users, data and complex queries and it returns high speed results. Exploiting the features of graph databases, it achieves for the big majority of queries responses in less than 100 msec. In order to achieve real-time operation features in a highly demanding smart city application domain, the suggested solution has been deployed in a high availability cluster environment, as the performance measurements have to be taken in realistic conditions, especially as the dynamic recommendation service, depending on time constraints, has to offer a user friendly experience to the user in terms of high availability and performance issue.

In order to achieve this, the Neo4j enterprise edition 3.2.3 has been selected because it allows the clustering approach. In the Neo4j High Availability (HA) architecture, the cluster is typically fronted by a load balancer HAProxy[26]. HAProxy has to be configured with two open ports, one for routing write operations to the master and one for load balancing read operations over slaves. Each application will have two driver instances, one connected to the master port for performing writes and one connected to the slave port for performing reads. The implemented cluster comprised of one server node acting as master node and 5 slave virtual machine nodes.

The goal of our distributed implementation is to provide high throughput. Each node processes a subset of the overall queries, ensuring scalability and performance in high-demand situations, reducing latency and providing continuous availability, even after a failure occurs to a machine. The proposed approach has been evaluated on the basis of read queries as to be able to determine the validity in demanding smart city applications with thousands of concurrent users requesting information. As depicted in Figure 25, the relationship between the response time of a single server implementation and the number of users/requests is almost linear.

[26] www.haproxy.org

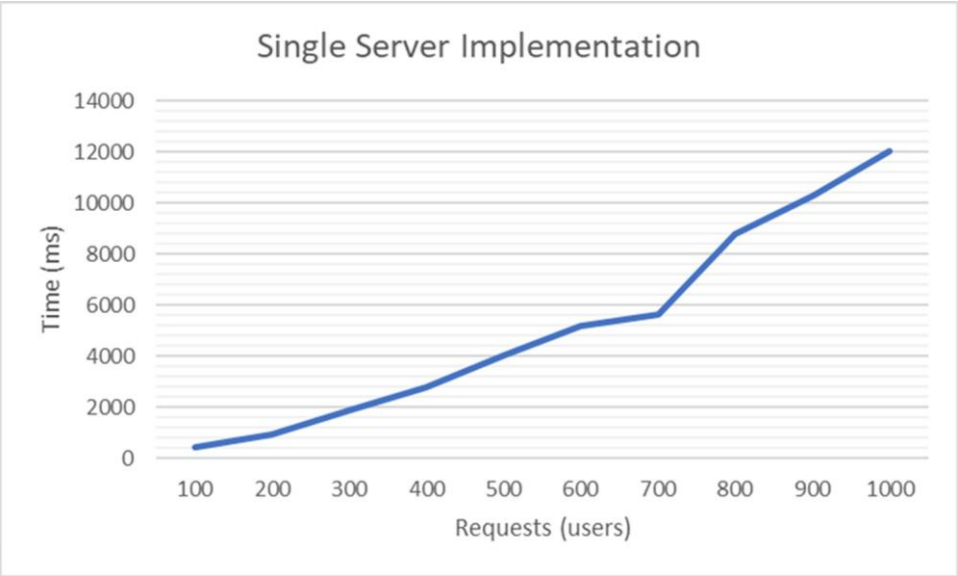


FIGURE 25: EXAMPLE OF 100 TO1000 REQUESTS – RESPONSE TIME ON A SINGLE SERVER

In order to test the scalability of our system, we examined its response time for an increasing number of requests from 100 to 1000. The process time is significantly reduced as the requests are allocated to more servers. In the following indicative figure, the total process time of 1000 requests is reduced from 12.02 seconds to 2.18 seconds after increasing the size of our cluster (Figure 26).

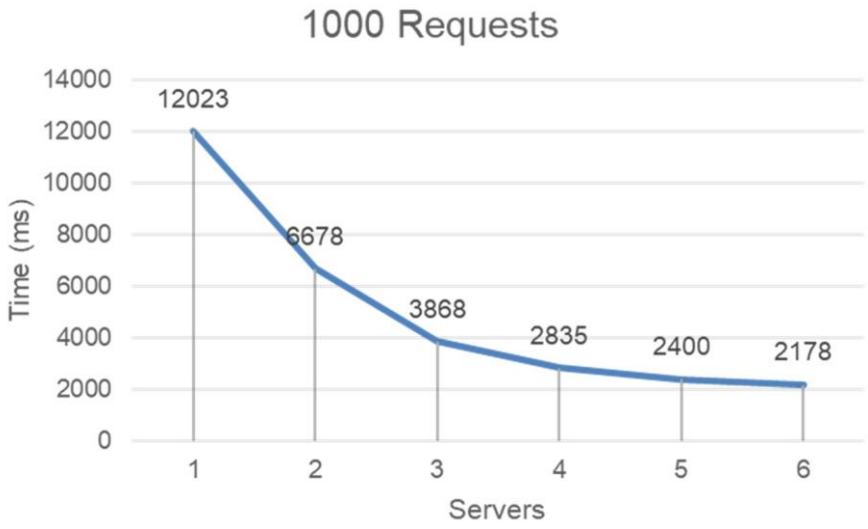


FIGURE 26: EXAMPLE OF 1000 REQUESTS – RESPONSE TIME ON UP TO 6 SERVERS-CLUSTER IMPLEMENTATION ON A HIGH AVALAIBILITY NEO4J CLUSTER

We have extensively evaluated the performance of our Recommendation Service based on High Availability Neo4j cluster in different settings. In the following representative diagram, we can see the reduction of total response time as we add more nodes (Figure 27).

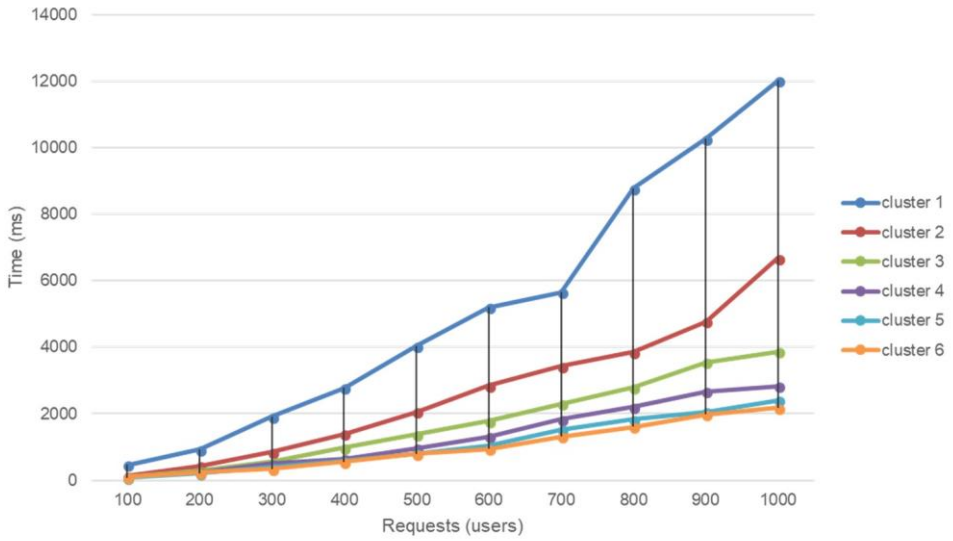


FIGURE 27: EXAMPLE OF 100 TO 1000 REQUESTS – RESPONSE TIME ON VARIETY OF CLUSTER NODE IMPLEMENTATIONS ON HIGH AVAILABILITY NEO4J CLUSTER IMPLEMENTATION

The overall improved performance of the High Availability Neo4j cluster compared to the performance of the single node Neo4j was improved in all cases. Even for experiments with a small number of requests (e.g. 300), the total response time was halved after adding one more node to the cluster (Figure 28).

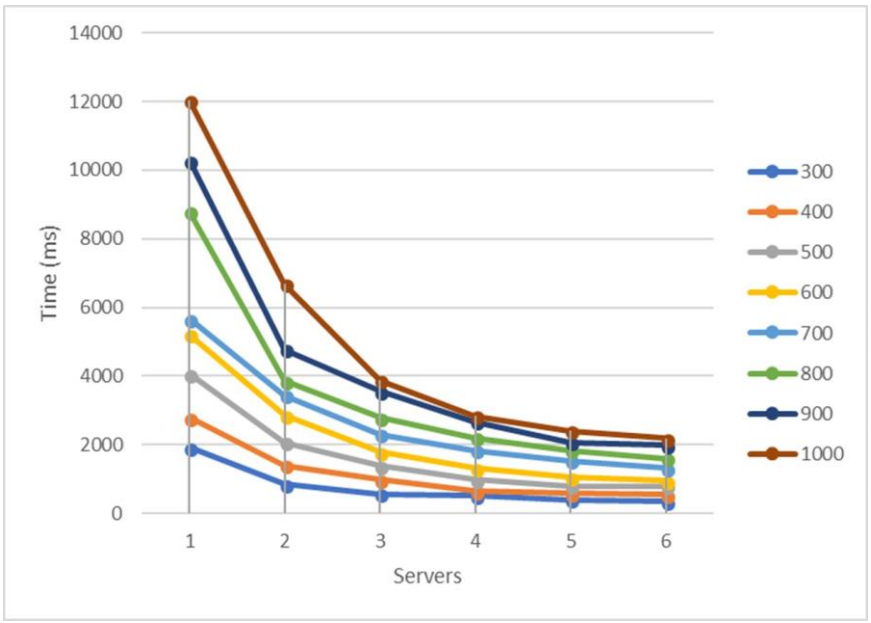


FIGURE 28: EXAMPLE OF 300 TO 1000 REQUESTS – RESPONSE TIME ON UP TO 6 SERVERS – CLUSTER IMPLEMENTATIONS ON HIGH AVAILABILITY NEO4J CLUSTER

As demonstrated previously, adding more nodes to the cluster improves the performance and reduces the overall response time. However, the reduction percentage is reduced, while we add more nodes. For example, in the case of 400 queries, the transition from single server to a two-nodes High Availability (HA) cluster reduces the overall required time by 49.8%, whereas the transition from a cluster with five nodes to six reduces the required time by 5.1% (Figure 29).

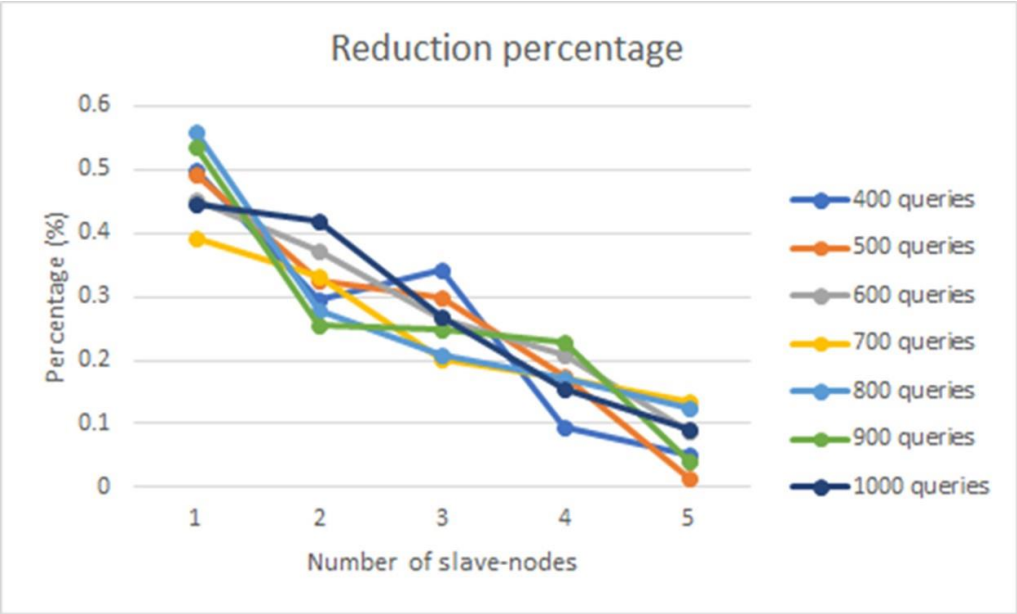
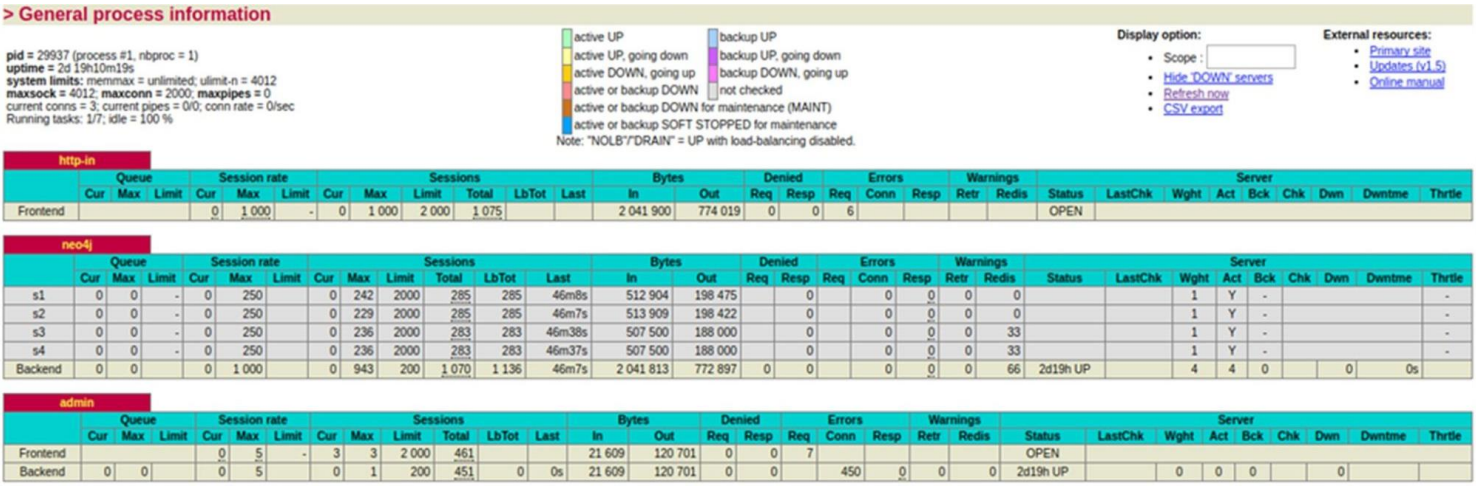


FIGURE 29: TIME REDUCTION PERCENTAGE ON A CONFIGURATION OF 400 TO 1000 REQUESTS ON UP TO 6 SERVERS - CLUSTER IMPLEMENTATIONS ON HIGH AVAILABILITY NEO4J CLUSTER



> General process information

pid = 29937 (process #1, nbproc = 1)
uptime = 2d 19h10m19s
system limits: memmax = unlimited; ulimit-n = 4012
maxsock = 4012; **maxconn** = 2000; **maxpipes** = 0
current conns = 3; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/7; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

http-in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1 000	-	0	1 000	2 000	1 075			2 041 900	774 019	0	0	6					OPEN								

neo4j

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	250		0	242	2000	285	285	46m8s	512 904	198 475		0		0	0	0	0			1	Y	-				-
s2	0	0	-	0	250		0	229	2000	285	285	46m7s	513 909	198 422		0		0	0	0	0			1	Y	-				-
s3	0	0	-	0	250		0	236	2000	283	283	46m38s	507 500	188 000		0		0	0	0	33			1	Y	-				-
s4	0	0	-	0	250		0	236	2000	283	283	46m37s	507 500	188 000		0		0	0	0	33			1	Y	-				-
Backend	0	0		0	1 000		0	943	200	1 070	1 136	46m7s	2 041 813	772 897	0	0		0	0	0	66	2d19h UP		4	4	0		0	0s	

admin

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	5	-	3	3	2 000	461			21 609	120 701	0	0	7					OPEN								
Backend	0	0		0	5		0	1	200	451	0	0s	21 609	120 701	0	0		450	0	0	0	2d19h UP		0	0	0		0		

FIGURE 30: HAPROXY DISTRIBUTES REQUESTS ACROSS MULTIPLE NEO4J SERVERS, OPTIMISING RESOURCE USE, MAXIMISING THROUGHPUT, MINIMISING RESPONSE TIME AND AVOIDING OVERLOAD

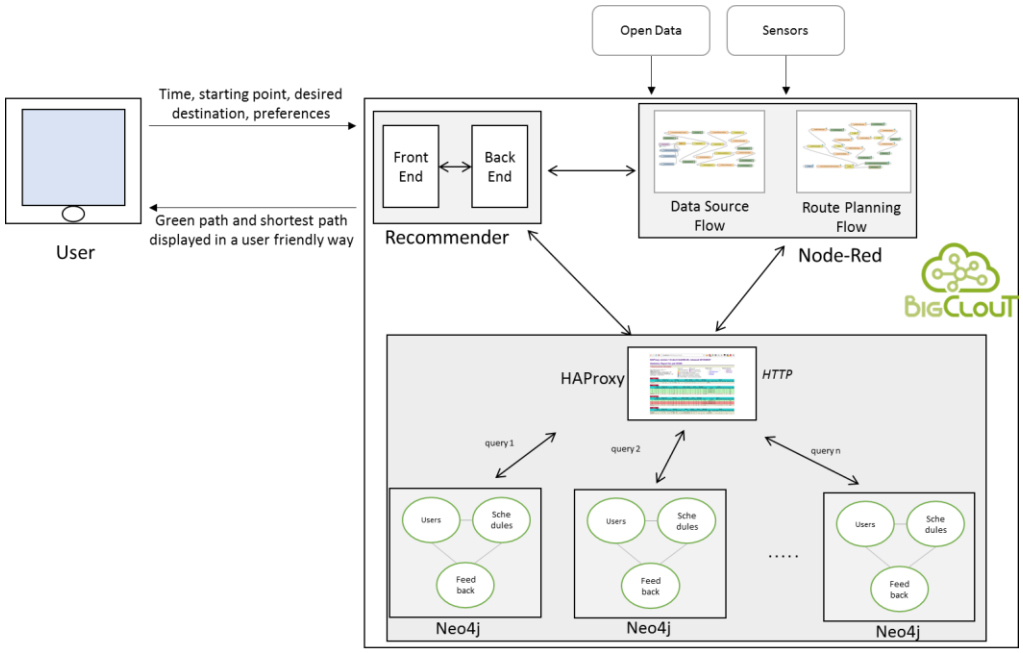


FIGURE 31: UPDATED ARCHITECTURE OVERVIEW INCLUDING HAPROXY AND DISTRIBUTED DATABASE

4 Integration Points & Use-Cases support

4.1 Fujisawa Trial 2: Fine-grained city infrastructure management

This use-case has been extensively described in the corresponding subsections where the DeepOnEdge as well as GAnonymizer have been presented. However, going beyond this implementation, services of even higher-level are offered in this scenario by using other components of WP3 (namely KNOWAGE and Recommendation Service).

DeepOnEdge outputs are provided to KNOWAGE as daily CSV files and are stored into the BigClouT Data Lake (WP2) as open data. The analytical process (performed by taking advantage of KNOWAGE's functionalities) consists of the:

1. Aggregation of the punctual measure by their geographical localisation, taking into account three types of features:
    - Fujisawa's Districts;
    - Fujisawa's Road;
    - Grid mesh
2. Classification of the aggregated measures by their level of damage (High, Medium and Low) using two different approaches:
    - The average damage value of the measurements that falls into a category
    - The majority of measurements that falls into a category

The high priority sorted output of the aggregation and classification processes is provided through JSON or CSV files to Fujisawa Municipality and the Recommendation Service in order to, respectively, schedule the proper maintenance services and to build a prediction model by linking these data to Fujisawa traffic and weather information. Moreover, these analysis results are used to build an interactive dashboard, depicted in Figure 17, which will be used by Fujisawa municipality officers to visually understand the city road damage status. Thanks to its RESTful API, recommendation produced by the Recommendation Service will be used in KNOWAGE as input for visualisation.

At the time this document is written, ENG and ICCS are working on giving the chance to manage the granularity of the features used to aggregate the punctual measures in order to find the optimal dimension for the prediction operation described above. In order to clarify this point, it is important to underline that the Fujisawa's Road and the Grid mesh used to aggregate features have currently a predefined dimension. Managing the granularity of these features will give deeper detail to Fujisawa municipality officers allowing to schedule more specific control and management to Fujisawa's road and, also, it will help the prediction operations providing larger dataset to work on. For instance, with this approach the user will have to decide dynamically the dimension of the grid mesh or of the road segments that will be used in the analysis.

4.2 Bristol Trial 1: Smart Mobility - Walkability and Air Quality trial

The integration between KNOWAGE and the Recommendation Service is used to provide to citizens of Bristol and Tsukuba insights about mobility in the city. This integration in both use cases is performed taking advantage of the RESTful APIs provided by the assets.

For the Bristol use case, the expected output is the correlation of air quality and traffic data to suggest the greenest path to pedestrians moving around the city. Air pollution is measured on 6 location of the city and the traffic and weather data is provided by Bristol City Council.

The details of this scenario are extensively presented in subsection 3.4.

These data are used by the Recommendation Service to provide recommendations about the greenest path the user should follow. These recommendations will be provided to citizens exploiting KNOWAGE's visualisation functionalities. Indeed, through a dynamic dashboard the recommended path will be visualised in a map together with a table and charts with details about the punctual measures provided thanks to the deployed sensors.

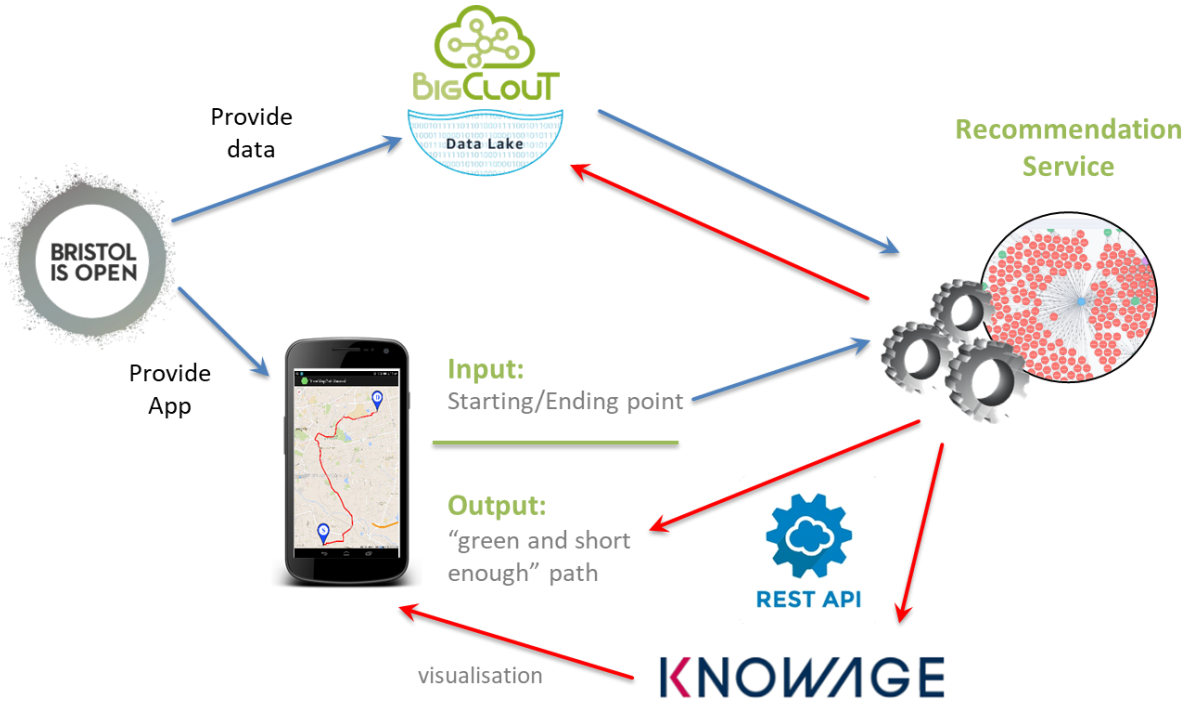


FIGURE 32: SMART MOBILITY USE CASE

4.3 Grenoble Trial 1: Business Events / Tsukuba Trial 1: Provide information in real time to visitors

This section describes the integration between KNOWAGE and Recommendation Service to provide visitor in Tsukuba and Grenoble with information. The data source of both the use cases is a mobile application which is used by visitors to:

- send notifications about problems experienced in Tsukuba City in terms of text and photo along with the user location, using a mobile application called HukuRepo.
- suggest events, restaurants or food trucks to visitors in (near) on the basis of the user location and registered preferences.

In the Grenoble use case, the recommendation about restaurants and events in the city will be produced by the Recommendation Service and provided to visitors in a friendly map that will be embedded in the mobile application. The map will be built exploiting KNOWAGE visualisation functionalities. The communication between the two assets will be accomplished taking advantage of the RESTful APIs provided by the two assets. Moreover, KNOWAGE will use the data collected by the mobile application to analyse the impact and information about the usage of the mobile application and the results of this analysis will be visualised in a dynamic dashboard.

For the Tsukuba use case, the expected outcome is the provision of concierge services for foreign visitors in Tsukuba city. The data will be provided to BigClouT platform thanks to a dedicated mobile application that will be in charge to gather from visitors their location and a timestamp together with the details of the specific user. These data will be used by the Recommendation Service to create specific recommendations for the user to suggest, for instance, a restaurant near their position. StreamingCube will be used in this use case to make spatial analysis of visitors. Moreover, KNOWAGE will be used to perform historical analysis over the collected data in order to give feedbacks to Tsukuba City officers about the activities of visitors.

The conceptual view of these use cases is similar to this presented in Figure 32.

5 Conclusions

This deliverable presents in great depth the Big Data Analytics Framework of BigClouT project and its architecture. Due to the detailed and structured presentation of the several elementary services and functionalities the WP3 modules and components provide, this report has been used as a reference document and a roadmap for all actions regarding the corresponding integration and (integrated) demonstrations.

The input provided by all WP3 partners in all previous deliverables of the WP has been used to define and finalise the implementation, integration and demonstration plan of the Big Data Analytics Framework that is responsible to provide these new functionalities in BigClouT.

Information provided in this document, and in general results produced by WP3, in terms of use cases, sequence diagrams and technical details about integration of the different assets composing the City Data Processing will be taken as input by the final activities of the project in order to continue the work towards the final results of BigClouT.