



LarKC

*The Large Knowledge Collider:
a platform for large scale integrated reasoning and Web-search*

FP7 – 215535

D6.9 – 4th periodic report on data and performances

Coordinator: Daniele Dell'Aglio (CEFRIEL)

With contributions from: Florian Steinke (Siemens), Irene Celino (CEFRIEL), Stanley Park and Tony Lee (Saltlux)

Quality Assessor: Ioan Toma (STI Innsbruck)

Quality Controller: Emanuele Della Valle (CEFRIEL)

Document Identifier:	LarKC/2008/D6.9/V0.1
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	Version 1.0
Date:	December 23, 2010
State:	Final
Distribution:	Public



EXECUTIVE SUMMARY

This document updates D6.7 – “3rd periodic report on data and performances” with the description of the tests we performed on the Traffic LarKC and on the Korean Road Sign Management applications. Both of them were developed in the previous months on the top of the LarKC platform and are described in D6.8 – Urban Computing environment v2.



DOCUMENT INFORMATION

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	6.9	Title	4 th periodic report on data and performances
Work Package	Number	6	Title	Urban Computing










Date of Delivery	Contractual	M33	Actual	M33
Status	Final		final ■	
Nature	prototype <input type="checkbox"/> report ■ dissemination <input type="checkbox"/>			
Dissemination level	public ■ consortium <input type="checkbox"/>			

Authors (Partner)	Florian Steinke (Siemens), Daniele Dell' Aglio and Irene Celino (CEFRIEL), Stanley Park and Tony Lee (Saltlux)			
Responsible Author	Name	Daniele Dell' Aglio	E-mail	daniele.dellaglio@cefriel.it
	Partner	CEFRIEL	Phone	+39-02-23954-243

Abstract (for dissemination)	This document is the third step of reporting about the collection of data sources and the evaluation results of the performance of the early Urban Computing demonstrators.
Keywords	data sets, measurements, tests, performances, stress tests, evaluation, periodic report, use case, urban computing

Version Log			
Issue Date	Rev. No.	Author	Change
November 4, 2010	1	Daniele	Creation of the document
November 22, 2010	2	Daniele	Initial description of the Traffic LarKC runtime tests
November 25, 2010	3	Daniele	Addition of the Traffic LarKC runtime tests results (tables and graphs)
November 29, 2010	5	Florian	Initial draft of the Traffic LarKC NN evaluation test
December 2, 2010	4	Stanley	Initial draft of the RSM LarKC evaluation test
December 9, 2010	6	Florian	Final draft of the Traffic LarKC NN evaluation test
December 9, 2010	7	Daniele	Final draft of the Traffic LarKC runtime tests
December 9, 2010	8	Daniele	Integration of Florian contribution
December 9, 2010	9	Daniele	Introduction
December 10, 2010	10	Daniele	Conclusions
December 13, 2010	11	Stanley	test result, consideration and conclusion

PROJECT CONSORTIUM INFORMATION

Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, University of Innsbruck	 	Prof. Dr. Dieter Fensel, Semantic Technology Institute (STI), universitaet Innsbruck, Innsbruck, Austria, E-mail: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle, CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA, Milano, Italy, Email: emanuele.dellavalle@cefriel.it
CYCROP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O.		Michael Witbrock, CYCROP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia, Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo, Höchstleistungsrechenzentrum, Universitaet Stuttgart, Stuttgart, Germany, Email: gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext Lab, Sirma Group Corp		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: atanas.kiryakov@sirma.bg
SALTLUX INC.		Tony Lee, SALTLUX INC, Seoul, Korea, Email: tony@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp, SIEMENS AKTIENGESELLSCHAFT, Muenchen, Germany, E-mail: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK, Email: h.cunningham@dcs.shef.ac.uk





<p>VRIJE UNIVERSITEIT AMSTERDAM</p>		<p>Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM, Amsterdam, Netherlands, Email: Frank.van.Harmelen@cs.vu.nl</p>
<p>THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY</p>		<p>Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE, Mabeshi, Japan, Email: zhong@maebashi-it.ac.jp</p>
<p>INTERNATIONAL AGENCY FOR RESEARCH ON CANCER</p>		<p>Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RESEARCH ON CANCER, Lyon, France, Email: brennan@iarc.fr</p>
<p>INFORMATION RETRIEVAL FACILITY</p>		<p>John Tait, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email : john.tait@ir-facility.org</p>



TABLE OF CONTENTS

1. INTRODUCTION	7
2. PERIODIC REPORT ON PERFORMANCE	8
2.1. TRAFFIC LARKC - NEURAL NETWORK EVALUATION	9
2.2. TRAFFIC LARKC - RUNTIME WORKFLOW STRESS TESTS (POLICIES COMPARISON)	10
2.3. TRAFFIC LARKC - RUNTIME WORKFLOW STRESS TESTS (WITHOUT CACHING).....	14
2.4. TRAFFIC LARKC - RUNTIME WORKFLOW STRESS TESTS (WITH CACHING)	15
2.5. RSM LARKC – EVALUATION (SALTLUX/VUA)... ..	18
3. CONCLUSIONS.....	22
4. REFERENCES	22



1. Introduction

This deliverable represents the fourth periodic report on data and performance of the Urban Computing use case in LarKC. It is based on the templates provided in D6.2 [2], it follows the early results presented in D6.4 [2] and it updates the tests described in D6.6 [6] and D6.7 [7]. In this deliverable we focus on the evaluation of two applications developed in the previous months by WP6: the Traffic LarKC and the Korean Road Sign Management (both described in the D6.8 [8]).

The Traffic LarKC¹ is a route planner for Milano that implements three different path finding policies: the shortest path, the fastest path and the fastest path with traffic predictions (done using historical traffic sensors data in Milano, weather and calendar information and Milano detailed topology). The application uses recurrent neural networks for traffic predictions, operational research algorithms for path finding and semantic web technologies for input data selection. This application implements two LarKC workflows: a batch-time one and a run-time one. The first works in order to estimate the traffic condition of Milano streets in the following time period, while the latter compute the most desirable path between two points. This workflow implements also a cache mechanism (at plug-in level) that stores the paths after their computation in order to re-use them (if possible).

The Korean Road Sign Management (RSM) LarKC² manages road signs information (in Seoul) by taking into account urban regulations and points of interest information. The considered data is road signs (from KICT), road topology (from OpenStreetMap) and a set of validation rules expressed in SPARQL.

In this deliverable we do not describe new data sources, due to the fact that data used in the two applications is described in previous deliverables D6.6 and D6.7.

The deliverable is structured as follows. Chapter 2 [6] presents our tests in terms of the adopted methodology, the tested demonstrators, the results of this testing and the interpretation we can give to those results. Finally, some conclusions are offered in Chapter 3.

¹ <http://larkc.cefriel.it/traffic-larkc/>

² <http://larkc.saltlux.kr/rsm/>



2. Periodic report on Performance

In this section we describe the tests we performed on the Traffic LarKC and on the RSM LarKC.

In the following sections we will describe each test we performed. The general structure of every test report is the following:

- Test goals: explain the reasons we considered to perform the test.;
- Considered workflow: a description of the workflow involved in the test;
- Methodology: an explanation of how we conducted the test;
- Environment: the hardware and the software used;
- Results: the results we obtained;
- Considerations: a brief analysis of the results obtained in the test.

Going in depth with the contents, the first tests will be related to the Traffic LarKC: in Section 2.1 there is the description of the tests batch-time workflow; Sections 2.2, 2.3 and 2.4 contain the description on the runtime part of the application.

2.1. Traffic LarKC - Neural network evaluation

2.1.1. Test goals

Here, we evaluate traffic predictions with respect to quality and performance.

2.1.2. Considered workflow

The Traffic LarKC batch-time workflow is considered.

2.1.3. Methodology

We compare the RNN traffic predictions with some standard competing prediction methods, and also check the plausibility of the results visually. The quality of the Bayesian semi-supervised learning to interpolate traffic predictions from sensors to all streets is checked visually. The whole workflow is timed, and the resulting times are assigned to different parts within the workflow.

2.1.4. Environment

LarKC 1.0³ is used in these tests. RNN prediction performance is tested with the Siemens proprietary software SENN. Computational performance tests are run on a 3GHz laptop PC.

2.1.5. Test results

The quality of the RNN traffic forecasts is examined in Figure 1. On the left traffic flow time series for some examples sensors are shown. The past 24h of known measurements are used to predict the next four hours. A numerical evaluation against other standard regression techniques, namely a feed forward neural network and linear regression, is presented on the right. The average relative error of the time delay RNNs is significantly lower than for the competing methods, and also shows a much smaller variance.

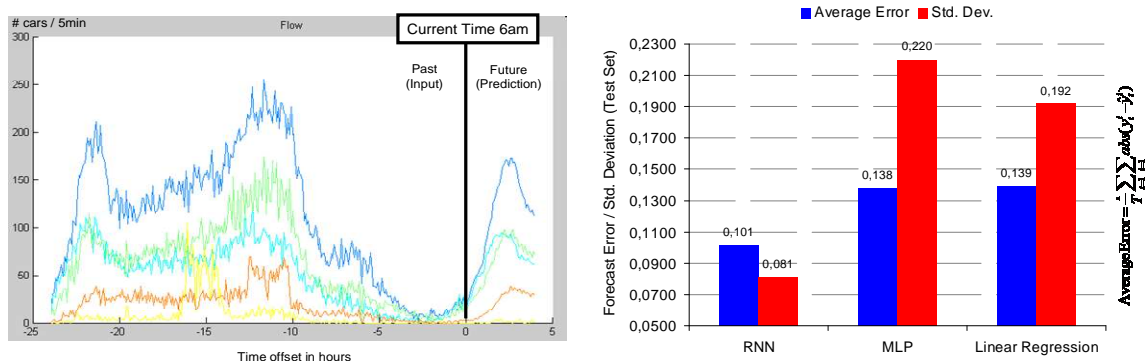


Figure 1 - RNN Traffic predictions. (Left) Time-series with some example sensors, (right) comparison of the time delay RNNs vs. feed-forward neural networks and linear regression

An example of the network-wide generalisation is shown in Figure 2. Numerical validation is problematic here, as no in-between-the-sensors information was available to us. However, the results are qualitatively plausible. They show connected areas of congestion around sensor locations with traffic distortions. Different road directions which are modelled with separate links may show different traffic situations as is often the case in reality.

³ LarKC 1.0 is the platform released in April 2010, available on the SourceForge SVN and described in D5.4.1 [1]. It is the version of the platform that we will consider in almost the tests.

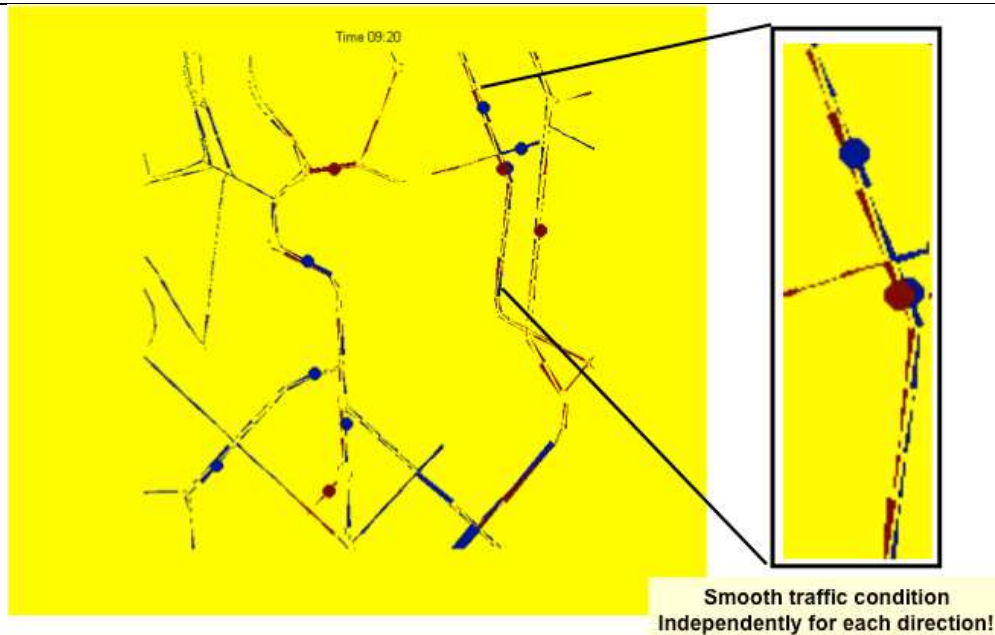


Figure 2 - Results of generalizing traffic condition predictions from sensor locations (dots) to all links of the road network via Bayesian Semi-Supervised Learning. Blue means normal condition, red congested

Regarding computational performance, note that the traffic prediction workflow is called only once an hour and then predicts the traffic for the next 4 hours on 30k links in 5 min intervals. This amounts to 600k triples that have to be updated in each run based on the newly available knowledge. While the whole runs needs on average acceptable 90 seconds, it should be noted that the interaction with the data layer alone, i.e. deleting old traffic predictions, reading the necessary data for the RNN predictions and rewriting the new predictions back into the data layer took a share of 81sec of the total, leaving only 9 sec for the actual algorithms.

2.1.6. Considerations

The quality of the RNN traffic predictions outperforms standard prediction methods. Complex statistical learning algorithms can be well integrated into LarKC.

Regarding computational performance, there is a large overhead for reading, writing and deleting triples from the data layer. However, this overhead allows flexibly working with the predictions in other plug-ins in the LarKC platform. These then do not have to know about the internals of the traffic prediction workflow.

Nevertheless, improvements in the semantic representation to efficiently deal with large amounts of low-structured, sequential data like time-series would be beneficial.

2.2. Traffic LarKC - Runtime workflow stress tests (policies comparison)

2.2.1. Test goals

The Traffic LarKC is able to find paths following several policies:

- shortest: finds the path with the minimum length;
- quickest: finds a path where the minimized dimension is the nominal travel time (the travel time in an ideal traffic condition);
- quickest considering the traffic conditions, evaluated by the batch-time workflow.

The goal of this test is to check how the different policies influence the path computation.

2.2.2. Considered workflow

Figure 3 shows the workflow we considered in this test.

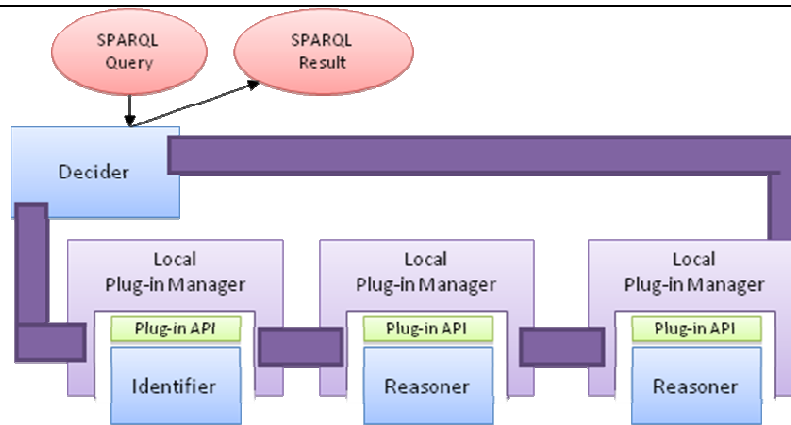


Figure 3 - Runtime Traffic LarKC Workflow

It is composed by four LarKC plug-ins:

- *UrbanNavigationDecider*, the decider of the application that build the workflow and executes it;
- *PathFindingPreparationIdentifier*, an identifier that analyzes the input query identifying the proper path finding policy to consider and prepares the data on which the path has to be computed;
- *DijkstraPathFindingReasoner*, a reasoner to compute a path minimizing a weight feature (length, travel time, etc.) and to store the results into the Data Layer;
- *PathCompletionReasoner*, the reasoner that adds information about the computed path (if missing) and answers to the input query.

The runtime workflow implements a cache mechanism at plug-in level, in order to compute the path only if it is missing. In other words if a path has been previously stored into the Data Layer, the Traffic LarKC answers to the query without computing it again.

The input query of this workflow should contain the following data:

- the starting point;
- the goal point;
- the policy that should be used in the computation;
 - o if the policy is the quickest path taking into account the traffic condition, the query should also contain time constraints (to specify what is the time to consider for the traffic predictions).

2.2.3. Methodology

The methodology is similar to the one described in D6.6, Section 4.5.3 [6] and reported in the following. In order to submit requests to LarKC we wrote a program (named *LarKC Stresser*): it requires an initial set of requests that can be provided via XML file, allowing an easy scheduling of the tests; alternatively it's possible to select one of several default requests generators that we built. A *request* is a triple composed by:

- an identification code;
- a SPARQL query;
- a submission time (in order to allow delayed requests).

The application can send *concurrent requests* (set of one or more requests sent at the same time) to the Traffic LarKC recording the response time for each of them. The act of sending a multiple request (in other words the submission a set of queries to LarKC at the same time) is an *experiment*.

This test has been structured as follows:

- for each multiple request (characterized by a different number of requests):
 - o we executed ten experiments and for each of them we calculated the variances and the average times;
 - o we calculated the mean of the average time values (this mean is the *average response time* of the multiple request) and the average variance (the mean of the variances of the ten experiments).

In this case the stress test increasing the number of concurrent requests for each one of the three policies.

2.2.4. Environment

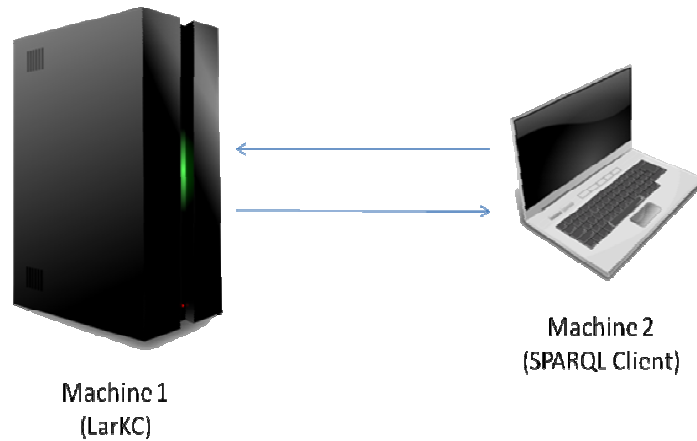


Figure 4 - Environment used in the path finding stress test

Hardware

Machine 1 (LarKC):

Processor: Six-core AMD Opteron Processor 2431 (2.4 GHz) – only one core was assigned to the virtual machine we used

RAM: 8 GB

Operating system: Ubuntu 10.04 64 bit

Location: Ontotext

Machine 2 (SPARQL client):

Processor: Intel Core 2 (2.16 GHz)

RAM: 2 GB

Operating system: Microsoft Windows XP Professional (Service Pack 3)

Location: CEFRIEL

Software

LarKC 1.0 platform

2.2.5. Test results

We performed the test increasing the number of concurrent requests from 1 to 20. All the responses returned by the Traffic LarKC were correct, but we didn't increase the number anymore, due to the fact that the average response time was too high. In Table 1 the average response time of the tests performed with the three different policies are reported.

Number of requests	Length (seconds)	Nominal Travel Time (seconds)	Estimated Travel Time (seconds)
1	3.396306	3.878974	4.196588
2	5.658513	5.930618	6.478235
3	8.596779	8.766359	10.39879
4	10.68979	11.8675	11.72309
5	13.39739	13.51399	15.05786
6	15.37011	15.38114	15.69981
7	17.81706	18.55391	19.72041
8	19.88775	20.1478	20.1966
9	22.04418	22.93375	22.95205
10	24.20643	24.7205	26.21817
11	26.18254	25.72379	27.76557
12	28.01248	27.72995	28.6728



13	30.88111	31.72211	31.90275
14	32.79946	34.08347	33.50838
15	34.08457	34.86292	35.99028
16	37.1661	38.72222	38.20015
17	38.63446	40.70494	40.72029
18	39.9037	41.12991	41.02473
19	43.11506	43.00682	43.79107
20	45.52799	46.38334	47.4226

Table 1 - Results of the Traffic LarKC stress test (policies comparison)

Figure 5 shows the average response time of the considered policies in a graphical way.

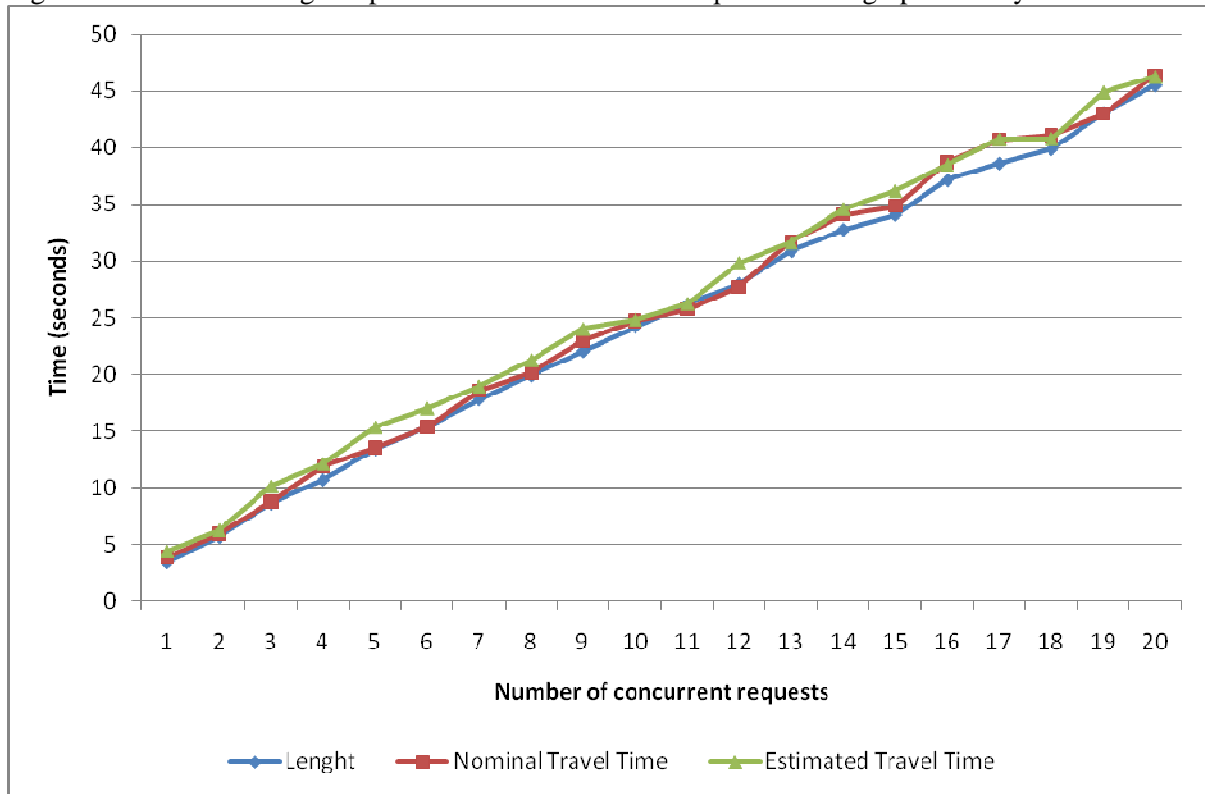


Figure 5 - Average response time for each policy

2.2.6. Considerations

The results we obtained show that the time performance of the Traffic LarKC runtime workflow is only imperceptibly influenced by the application of the different policies. The slight differences are given by the following facts:

- the shortest path policy is the quickest of the three because it works on the minimal amount of data (only the topology of Milano);
- the quickest path policy is a bit slower than the shortest path one; the main reason is that in this case LarKC works with a larger amount of data (the topology of Milano and the nominal travel time values);
- the quickest path policy taking into account the traffic prediction is the slowest one; this happens for two reasons:
 - o as in the quickest path policy LarKC works on a larger amount of data than the shortest path one;
 - o the input query requires an additional computation effort in order to process the time constraints.

Nevertheless, the difference of the three results is very limited and we can consider that the three policies have the same behaviour. We can conclude that the time performance of the runtime workflow is not influenced by the kind of policy considered.



2.3. Traffic LarKC - Runtime workflow stress tests (without caching)

2.3.1. Test goals

In the following we describe the stress test we made on the Traffic LarKC runtime workflow without using paths stored into the cache. The goal of this test is to get information about how the behaviour of the Traffic LarKC is influenced when a multiple set of users interact with it.

This test will simulate the worst execution case, where the cache is never read: it means that the application stores the paths into the Data Layer but it never re-use them, and for each query a new path will be computed.

2.3.2. Considered workflow

The workflow is described in Section 2.2.2 and represented in Figure 3.

2.3.3. Methodology

We executed this test following a methodology similar to the one described in Section 2.3.3. As query we consider a set of random generated query using the same policy (the shortest one); as showed in the previous test, it does not influence noticeably the results

2.3.4. Environment

The environment is the one described in Section 2.2.4.

2.3.5. Test results

We executed the test increasing the number of concurrent requests from 1 to 20. The results of the test are available in Table 2.

Number of requests	Mean (seconds)	Variance (seconds ²)
1	3.396306	
2	5.658513	1.363916
3	8.596779	1.566848
4	10.68979	1.12128
5	13.39739	0.94881
6	15.37011	0.819417
7	17.81706	0.757411
8	19.88775	0.660556
9	22.04418	0.56381
10	24.20643	0.703442
11	26.18254	0.800088
12	28.01248	0.576834
13	30.88111	0.490797
14	32.79946	0.555761
15	34.08457	0.276032
16	37.1661	0.371074
17	38.63446	0.330282
18	39.9037	0.383602
19	43.11506	0.436443
20	45.52799	0.339761

Table 2 - Results of the Traffic LarKC stress test (without caching)

Additionally a graphical representation of the average response time is available in Figure 6.

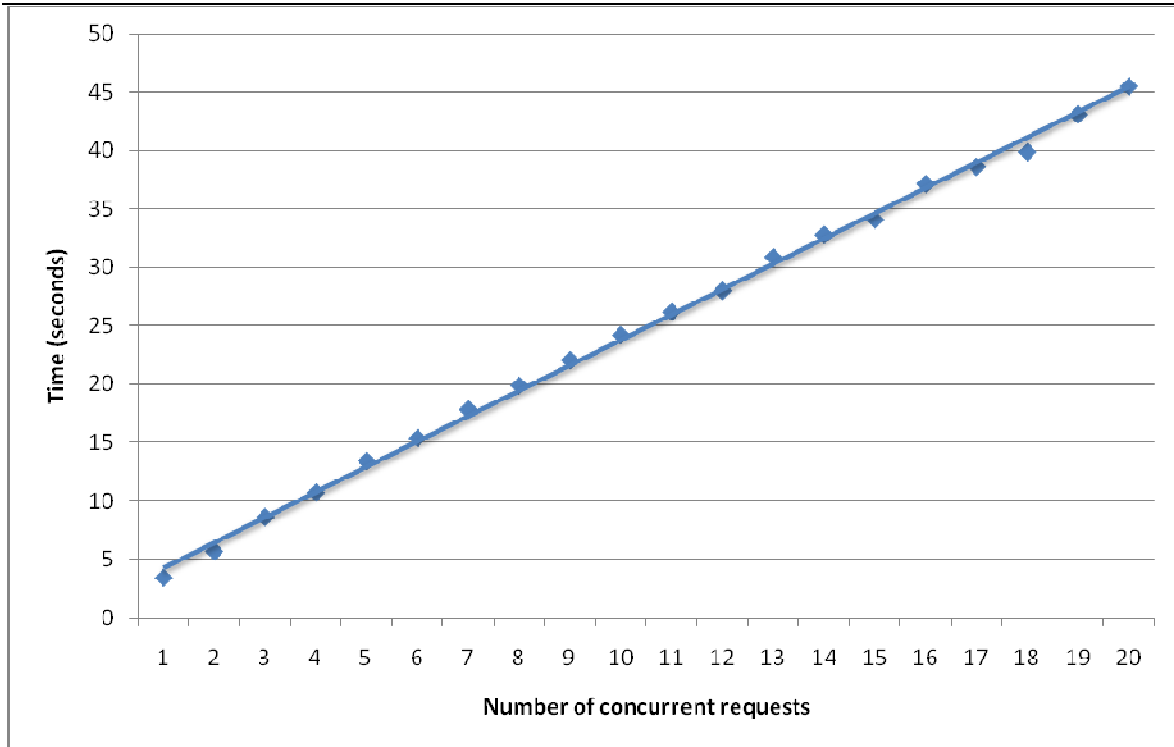


Figure 6 - Average response time of the Traffic LarKC stress test (without caching)

2.3.6. Considerations

As it's possible to observe in Figure 6, the response time follows a linear trend, with a function structured as following:

$$\text{Average response time} = x * \text{Number of requests}$$

This result is similar to the one we obtained evaluating the Alpha Urban LarKC path finding workflow in previous deliverables [6][7]. In this case the gradient value is about 2.3: the reasons for this behaviour are the following:

- we replaced the *OpResPathFinderReasoner* plug-in (used by the Alpha Urban LarKC) with the *DijkstraPathFindingReasoner*, which is a more generic plug-in (it can work with graphs modelled with different vocabularies); this, however, implies an overhead in the workflow execution time;
- the testing machine running the Traffic LarKC has only one available processor; the use of a multi-processor machine would improve the performance when there are concurrent requests as explained in [7];
- the plug-in level caching approach requires that paths should be stored into the Data Layer, so it also affects the overall response time;
- since this test considers the worst case, the cached paths are never read; the positive effect of caching will be evident in the next test.

2.4. Traffic LarKC - Runtime workflow stress tests (with caching)

2.4.1. Test goals

This test aims to understand the behaviour of the Traffic LarKC runtime performance when the cache is enabled and used. While the test described in Section 2.3 allowed us to collect data in the worst case, this one is executed in the best case: each query ask for a path that has been computed in the past and has been stored into the Data Layer.

2.4.2. Considered workflow

The workflow is described in Section 2.2.2 and represented in Figure 3.

2.4.3. Methodology

In order to execute this test we followed the methodology described in Section 2.3.3.



2.4.4. Environment

The environment is the one described in Section 2.2.4.

2.4.5. Test results

The Table 3 shows the results we collected in this test. In this case we increased the number of concurrent requests from 1 to 50.

Number of requests	Mean (seconds)	Variance (seconds ²)	Number of requests	Mean (seconds)	Variance (seconds ²)
1	1.015879		26	3.080249	0.25668
2	1.748479	2.675367	27	3.482006	0.231557
3	2.096188	1.781946	28	3.255622	0.202123
4	2.300356	1.341897	29	3.766665	0.311754
5	2.412738	1.064107	30	3.152489	0.203279
6	2.583681	0.800358	31	3.848838	0.20903
7	2.557554	0.729024	32	3.09945	0.184333
8	2.676942	0.588175	33	3.13617	0.188554
9	2.690218	0.502092	34	3.659233	0.167
10	2.729265	0.475613	35	3.201531	0.401233
11	2.760827	0.394879	36	3.567288	0.147445
12	2.785714	0.416309	37	3.179791	0.193984
13	2.842455	0.371107	38	3.202686	0.162384
14	2.826623	0.344051	39	3.175248	0.187847
15	2.836313	0.31217	40	3.362348	0.159924
16	2.882051	0.3023	41	3.717721	0.144431
17	2.96473	0.294904	42	3.778034	0.179752
18	2.976682	0.308846	43	3.684726	0.154724
19	2.952794	0.275875	44	3.312683	0.173125
20	3.202395	0.429032	45	3.320965	0.160381
21	3.167274	0.566558	46	3.389756	0.1666
22	3.156875	0.298375	47	3.79136	0.172582
23	3.350731	0.190447	48	3.42417	0.135591
24	3.021559	0.215549	49	3.516181	0.362504
25	3.561562	0.191655	50	3.992146	0.154067

Table 3 - Results of the Traffic LarKC stress test (with caching)

A graphical representation of the average response time is showed in Figure 7.

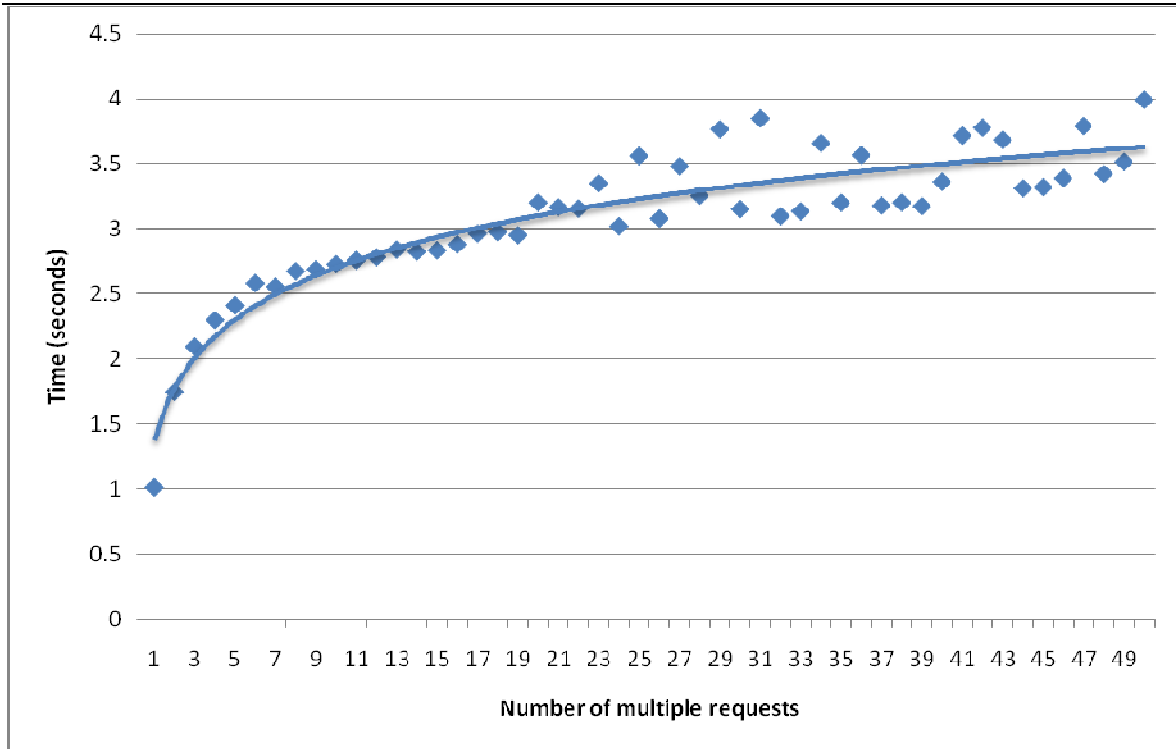


Figure 7 - Average response time of the Traffic LarKC stress test (with caching)

2.4.6. Considerations

The results we obtained in this test are positive: as it's possible to observe in the graphical representation in Figure 7, the trend followed by the average response time is sub-linear. The cache mechanism we developed allows users to invoke concurrently the system having a good response time.

Table 4 shows the evolution of the caching mechanism in the different path finding workflows we developed. While in the first version (the Alpha Urban LarKC evaluated in D6.6), there was no caching mechanism at all, we introduced the caching of the RDF graph describing the map in the second version (the Alpha Urban LarKC evaluated in D6.7). Additionally, in the current version of the path finding workflow developed for the Traffic LarKC, we introduced a caching mechanism not only for the map, but also for the computed paths.

	alpha Urban LarKC D6.6	alpha Urban LarKC D6.7	Traffic LarKC D6.9
Map	×	○	○
Path	×	×	○

Table 4 - Status of the cache in the developed path finding workflows

Figure 8 shows the average response time trends of the three path finding workflows we have just described.

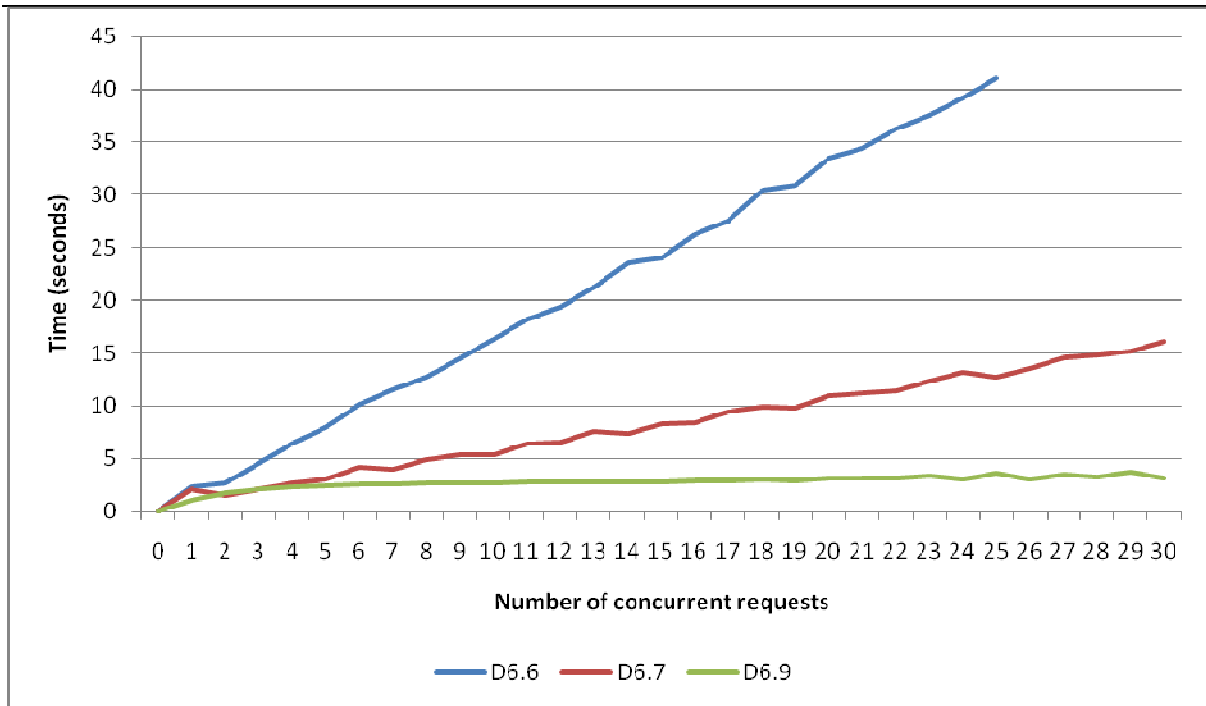


Figure 8 - Comparison of the path finding workflows (with cache)

Even if several optimizations have been introduced in the different versions of the path finding workflows, the cache is the feature that influences most the time performance. The figure shows that the first cache version (evaluated in D6.7) decreased drastically the response time even if the trend continued to be linear as in the path finding workflow without caching (D6.6). The best case so far is the cache implemented in the Traffic LarKC that follows a sub-linear trend (more visible in Figure 7).

2.5. RSM LarKC – evaluation (Saltlux/VUA)...

2.5.1. Test goals

Road Sign Management based on LarKC platform is evaluated two view of qualities that are efficiency and effectiveness.

Efficiency evaluation is performed by comparing query time with different scale of data and hardware under same LarKC platform.

Effectiveness such as functionality and usability evaluation is identified by comparing Road Sign Management LarKC with Road Sign Management of KICT (Korea Institute of Construction Technology) running on http://218.49.22.28:8000/main_form.asp.

2.5.2. Considered workflow

The workflow for Road Sign Management is shown as following.

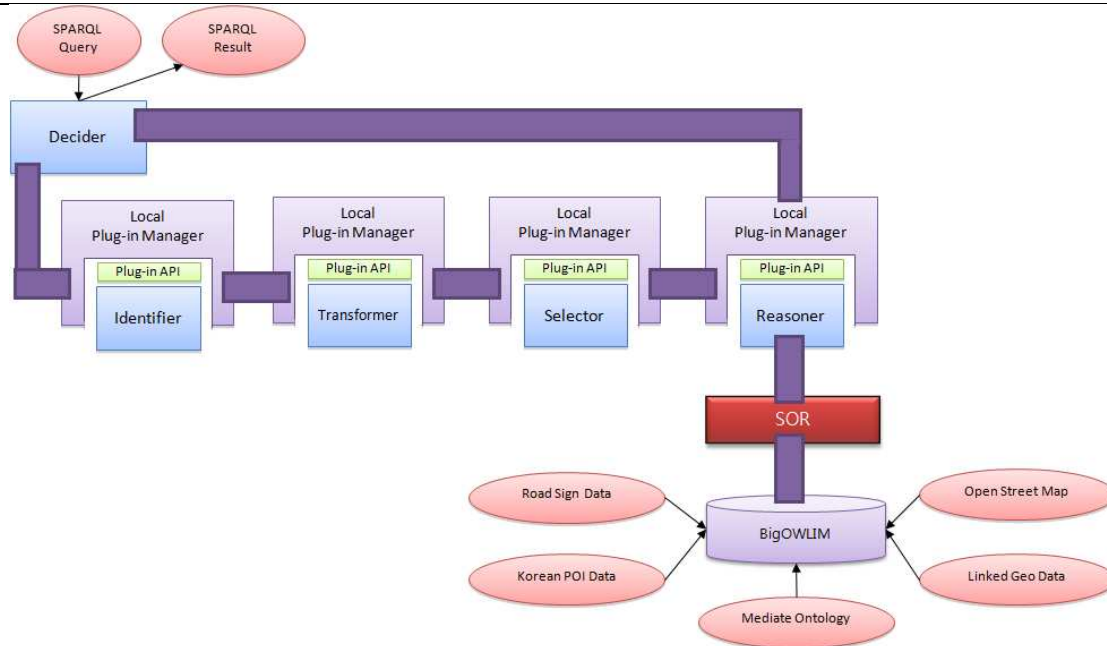


Figure . Road Sign Management Workflow

Road Sign Management workflow that is generated by RSM Decider is composed in linear order of RSM identifier, RSM Transformer, RSM Selector, RSM Reasoner.

RSM Decider creates a RSM workflow with a SPARQL query received as a parameter to start from RSM Identifier.

RSM Identifier generate temporary subject, predicate, object with default namespace that is <http://www.saltlux.com/rsm#> to return InformationSet result to RSM Trasformer as input parameter.

RSM Transformer returns result without processing received parameter from RSM Identifier to RSM Selector.

RSM Selector returns it result to RSM Reasoner in a same way of RSM Transformer.

RSM Reasoner which received the result of RSM Selector and SPARQL query from RSM Decider while creating workflow sends SPARQL query only to remote BigOWLIM of SOR.

SOR is semantic object retrieval framework of Saltlux that integrating semantic repositories and inference engines.

The result of remote BigOWLIM of SOR which process querying and reasoning on stored all dataset returns to RSM Decider as VariableBinding Object type.

A feature of this workflow is a extendable remote platform functions of plug-ins:

RSM Reasoner plug-in processes selection and reasoning with SPARQL queries of RSM by APIs of remote 3th party semantic application framework.

Transformer has room for transforming noisy dataset regarding road data that contains incomplete, uncertain and duplicated data in Road Sign Management that make wrong results of reasoning into clean dataset that is purified by remote platform.

2.5.3. Methodology

Road Sign Management based LarKC running in Eclipse waits to receive SPARQL query for retrieving information or reasoning as input parameter from web application passing parameters of UI web browser.

SPARQL query for performance test in scenario is about finding all road signs containing target Point of Interest.

This query is need to process of RDFS++ reasoning and retrieving triples.

RDFS++ reasoning is performed while all triple datasets of Road Sign Management are loaded as a forward chaining.

Retrieving triples is performed while BigOWLIM process SPARQL queries.

We are check performance of loading and retrieval time in different test environment.

Test query is as following.



Name	Query for retrieving road signs indicating target and its coordinates.
SPARQL	<pre> PREFIX rsm: <http://www.saltlux.com/rsm#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX g2r: <http://www.saltlux.com/g2r/functions#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> SELECT distinct ?id ?long ?lat ?poiName WHERE { rsm:kpoi_24892 rsm:indicatedBy ?id . rsm:kpoi_24892 rsm:name ?poiName . ?id rdf:type rsm:RoadSignElement . ?id geo:lat ?lat . ?id geo:long ?long . } </pre>
Property	owl:inverseOf, rdfs:subPropertyOf, rdfs:subClassOf

rsm:kpoi_24892 KPOI instance is domain value of indicateBy property which is generated from materialized values of indicate property of RoadSignElement by reasoning.
 rsm:kpoi_24892 KPOI instance that a road sign indicate is a value of one of subsequent direction properties of indicate property of RoadSignElement class.
 The value of direction property also is generated from further subsequent property of direction property which is distance property of direction by reasoning.
 Values of distance property of direction are materialized from dataset.
 RoadSignElement class instance that meet indicate rsm:kpoi_24892 KPOI instance is subsequent class of NodeElement class.
 RoadSignElement class instance can get WGS84 coordinates with property of parent class.

Queries for road sign valid checking are selecting target, finding road signs related target, tracking road for target indicating road sign to be check.

2.5.4. Environment

There are two environments of test whose conditions are scale of dataset, hardware performance and version of remote framework of reasoner plug-in.
 Specifications of each environment that is named Server environment and PC environment are as following.

Item	Server Environment	PC environment
RDF set	- Open Street Map, Korean POI, Road Sign Data, Mediate Ontology, Linked Geo Data, - explicit triple count : 1.1Billion	- Open Street Map, Korean POI, Road Sign Data, Mediate Ontology - explicit triple count : 150 Mega
Hardware	CPU : Intel Xeon 2.33GHz RAM : 64GB OS : Linux 64bit	CPU : Intel Pentium4 3.06GHz RAM : 4GB OS : Window server 2003 servicepack 2, 32bit
Software	SOR version : 3.5.2 BigOWLIM : 3.3	SOR Version : 3.0.0 BigOWLIM : 3.1
Common	LarKC Platform 1.0 Jetty Web server	

RDF set of server side contains full Linked Geo Data as large dataset and Road sign management related triples and mediate ontology compared to PC side for compact dataset that is essential for running Road Sign Management. Full dataset are larger 667 times than compact dataset.

As values of CPU benchmark with CPUMark application that are Xeon CPU is 3036 and PC CPU is 457, Server side is faster 6.7 times than PC side.

2.5.5. Test results

There were two comparable test results of environment between large scale triples and compact triples for Road Sign Management as following.

Response time

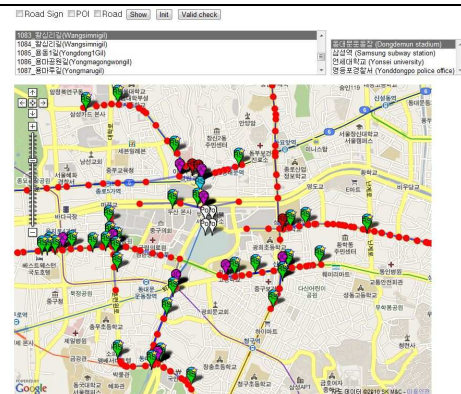
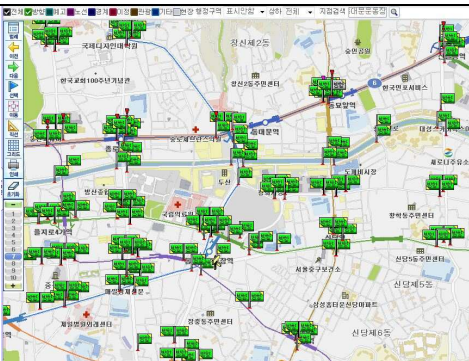
Item	Server Environment	PC environment
Loading	<ul style="list-style-type: none"> - total triples counts by inference : 2.57 Billion - loading time of explicit triple : 231076578ms (64hours) - indexing time : 2485657ms(40minutes) 	<ul style="list-style-type: none"> - total triples counts by inference : 300 Million - loading time of explicit triple : 41187ms(41seconds) - indexing time : 150422(2minutes)
Querying	<ul style="list-style-type: none"> -Query for retrieving road signs indicating target and its coordinates : average 230ms All queries for one road sign validation : average 3850ms 	<ul style="list-style-type: none"> -Query for retrieving road signs indicating target and its coordinates : average 313ms All queries for one road sign validation : average 2689ms

After loading explicit triples, OWLIM as repository of reasoner contains twice size triples by materialized for forward chaining inference with mediate ontology on each case. During loading time on server side, triple sets that are divided by 100mega byte per 1 triple set are committed every 200 triple set for speeding loading time up. Query processing time of two cases that are subsumption reasoning and retrieving is not different for person sense.

2.5.6. Considerations

Road Sing Management of LarKC can be extendable for the RDF data set supporting retrieving and reasoning with large dataset using LarKC Platform. But Road sign management of KICT is static for using relational database data to retrieving status of current road signs.

Effectiveness evaluation

characteristics	RSM LarKC	RSM of KICT
Image		
Purpose	Checking validation of road signs	Checking location of road signs
Functionality	Showing indication of targets of Road sign Showing roads with nodes.	Showing administrative divisions Searching by name.



	Showing main Point Of Interest in road sign Showing all the road sign, road, POI within view.	
Usability	Checking which road signs are invalid. Integrating different geo datasets to not only retrieving but also reasoning.	Identifying all kinds of road sign.

3. Conclusions

In this deliverable we continued the series of evaluation tests on our Urban LarKC applications. We focused on the last two applications we developed: the Traffic LarKC and the Korean Road Sign Management LarKC. We collected data about the performance of the LarKC workflows involved in these applications, drawing some conclusions and lessons learned. We also compared the results we obtained with other LarKC workflows we evaluated in the past in order to verify that the performance of the current platform and plug-ins in solving similar tasks are improved.

We demonstrated that, since LarKC can execute multiple workflows, we can split a complex processing task into different workflows and orchestrate their execution. Specifically we were able to divide the business logic of the Traffic LarKC into two workflows: a batch-time one whose execution is temporally scheduled and a runtime one invoked by users' input. In this way, the runtime performance in term of response time is not influenced by the traffic prediction computation. Furthermore, we improved the caching mechanism, thus improving the overall system response time, with the result of supporting a larger number of concurrent requests and users. In the forthcoming LarKC 2.0 release the caching mechanism will be supported at platform level, whereas in this deliverable we described our cache implementation at plug-in level.

In Road Sign Management, LarKC Platform act as core platform for expanding its plug-ins such that reasoner delegates its functions to remote another platform that is integrated reasoner and RDF repository for large data. As hardware and software performance are high, the result of reasoning performance is less time consuming comparing to optimised reasoning test. Another plug-in in Road sign management except reasoner could be used for another platform for data purification processing that using another type of reasoner.

We will continue to repeat the performed tests and, in case, to add more evaluations in the last data and performance deliverable, the D6.11 [9], in order to compare the indicators related to the various releases of the LarKC platform.

4. References

- [1] D5.4.1 – Initial release of the LarKC platform
- [2] D6.2 – Templates of periodic report on data and performances – Kono Kim, Irene Celino, Emanuele Della Valle, Daniele Dell'Aglio, Yi Huang, Werner Hauptmann
- [3] D6.3 – Urban Computing environment specification – Emanuele Della Valle, Daniele Dell'Aglio, Irene Celino, Kono Kim
- [4] D6.4 – 1st periodic report on data and performances – Emanuele Della Valle, Daniele Dell'Aglio, Irene Celino
- [5] D6.5 – Urban Computing environment v1 – Irene Celino, Daniele Dell'Aglio, Emanuele Della Valle, Kono Kim, Stanley Park, Florian Steinke, Ralph Grothmann, Werner Hauptmann
- [6] D6.6 – 2nd periodic report on data and performances – Daniele Dell'Aglio, Irene Celino, Emanuele Della Valle, Kono Kim, Stanley Park, Florian Steinke
- [7] D6.7 – 3rd periodic report on data and performances – Irene Celino, Daniele Dell'Aglio, Stanley Park, Tony Lee, Florian Steinke, Alexey Cheptsov, Matthias Assel
- [8] D6.8 – Urban Computing environment v2
- [9] D6.11 – 5th periodic report on data and performances