Geo-coding and Smart Space Platforms Integration Agent Performance Testing and Analysis

Kirill Yudenok, ETU, OSLL

Geo-coded Smart Space (GCSS)

GCSS is an integration mediator between smart space and geo-coding platforms, which provides the ability to use geographical information of real world objects in the smart space.

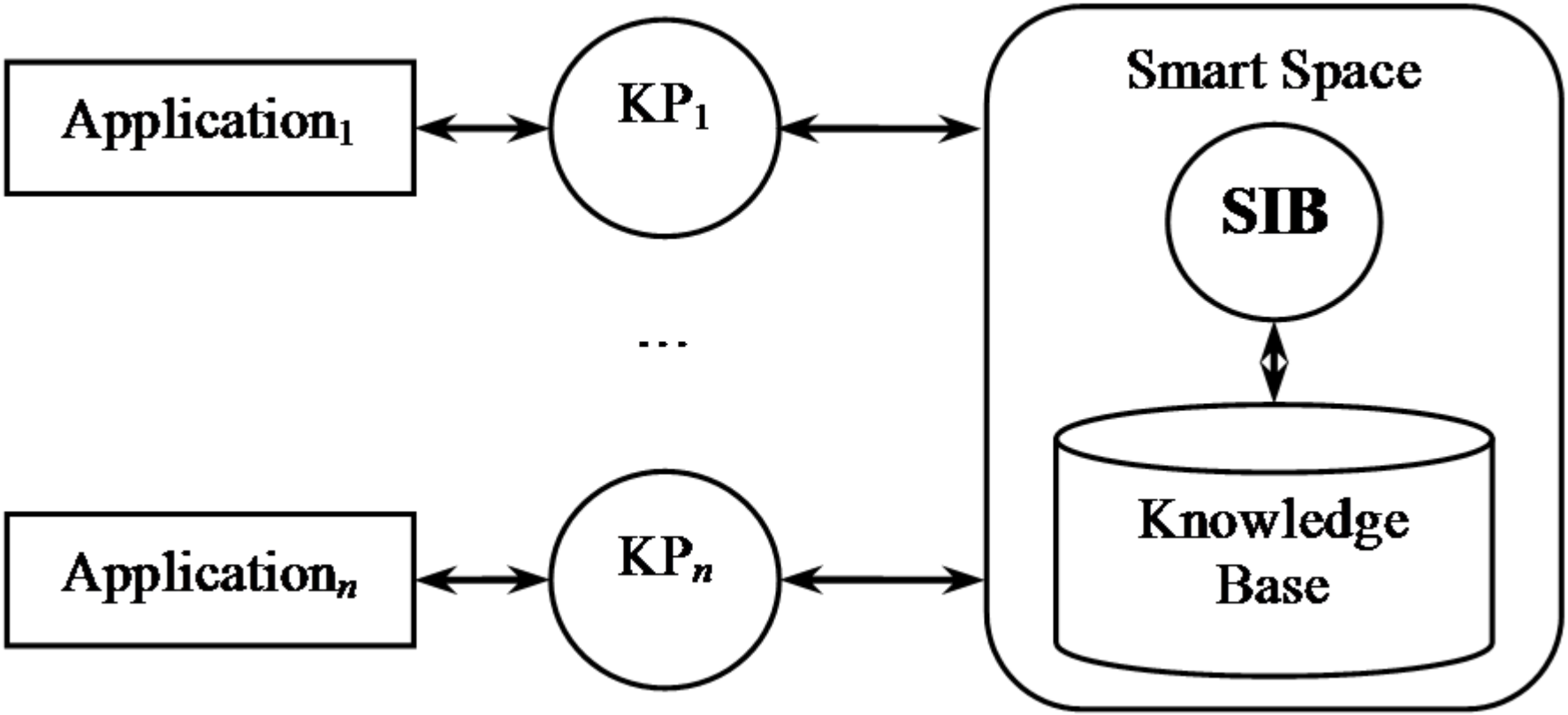
- **The main difference from previous works –** common platforms for the knowledge and geographical data processing;
- **New property –** location in space and time, that give the additional abilities for a searching smart space subjects in a various described spaces.

Agenda

- Geo-Coded Smart Space (GCSS);
- Smart-M3 and Geo2Tag platforms;
- GCSS conceptual model and layered architecture;
- GCSS high-level requirements;
- GCSS testing methodology;
- Functional testing – Unit-testing;
- Performance testing – Profiler;
- Performance analysis;
- GCSS optimization recommendations;
- Conclusions and future steps.

Smart-M3 platform

Smart-M3 is an open source software platform that aims to provide Semantic Web information sharing infrastructure between software entities and various types of devices.



Geo2Tag platform

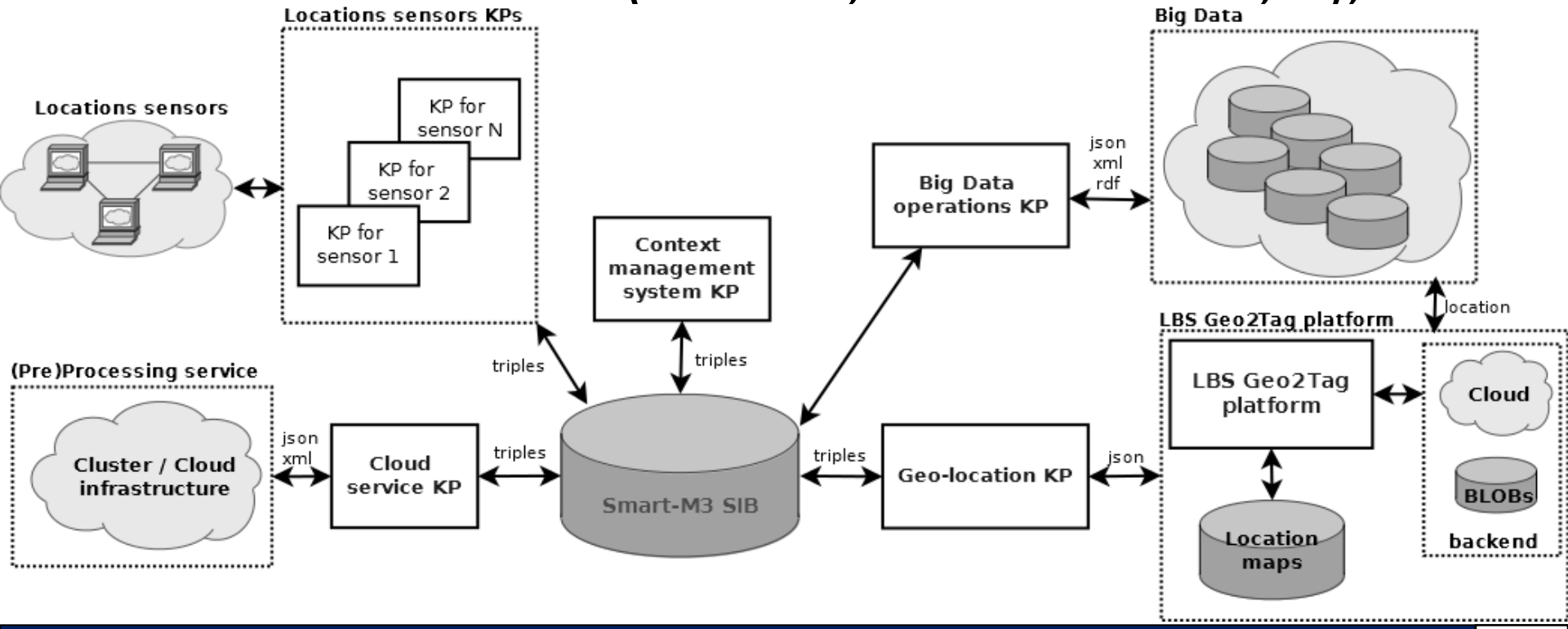
Geo2Tag platform is the centralized high performance geo-coding database with a dedicated server, which is provided RESTful API for an access to the geo-tags.

Geo2Tag features:

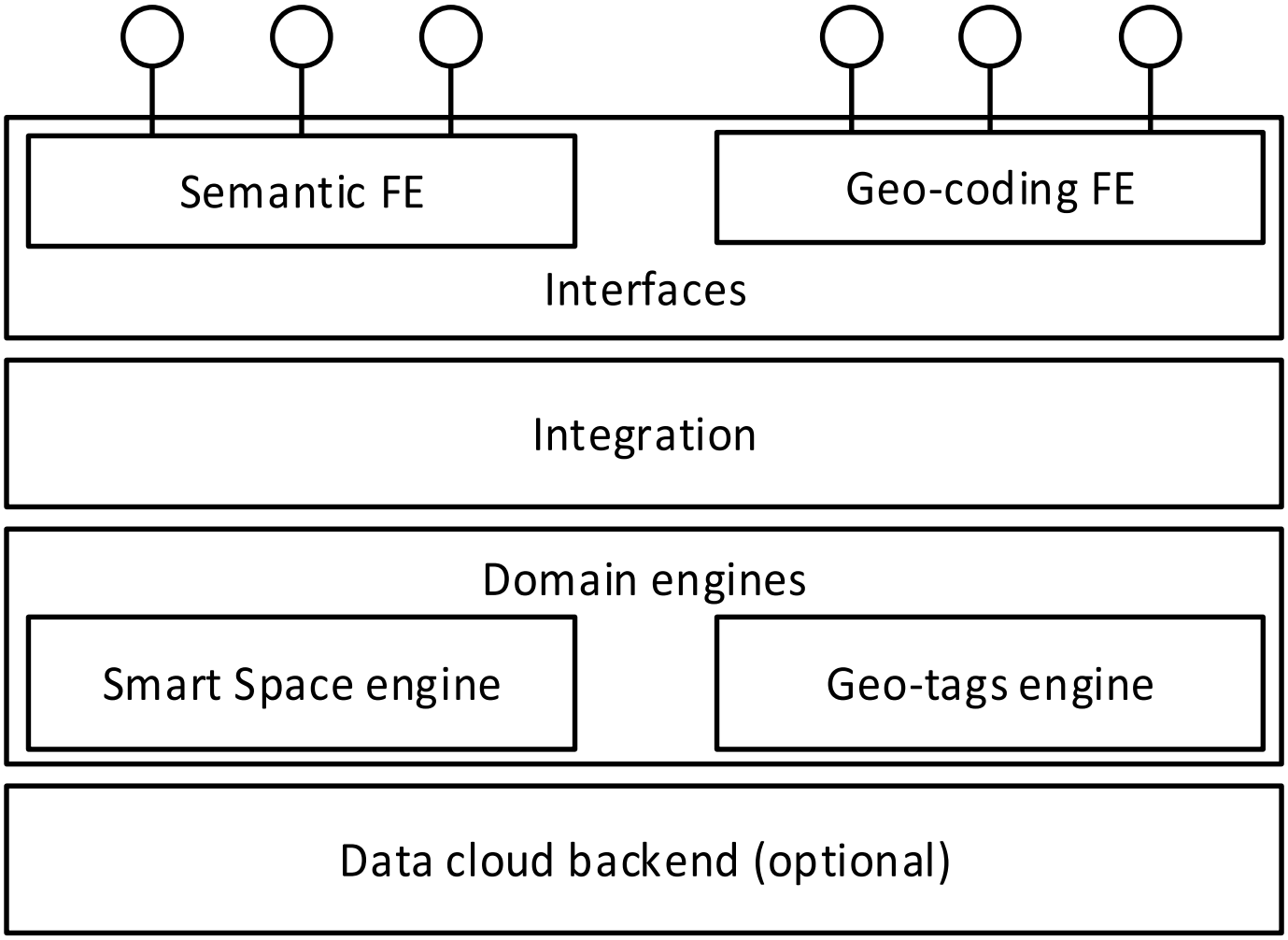
- users management: registration, login, logout, sessions;
- channels management: subscription/unsubscription;
- RESTful API for getting access to geo-data;
- spatial and temporal geo-data filtration;
- integration with 3rd party maps services.

GCSS conceptual model

- Cloud-service KP (High-performance cloud system);
- Context management system;
- Big Data KP (Data collections for the geo-space);
- Location sensors KPs (sensors, mobile devices, …);



GCSS layered architecture



GCSS high-level requirements

- interfaces for an access to the semantic and geo-data;
- distributed storage for semantic and geo. information;
- interfaces for association semantic objects with geo-tags;
- spatial and temporal filtration between platforms.

Non-functional requirements:

- Performance (cloud based massive offline processing and local context indexing/caching);
- Compatibility (legacy interfaces: SSAP, REST);

GCSS testing methodology

Two testing types:

1. functional testing of an integration agent mechanisms;
2. integration agent load and stress performance testing.

GCSS testing stand:

- CPU – Intel i7, 3.4 Mhz, 8 cores; RAM – 8 Gb;
- OS – Ubuntu 14.04 LTS;
- Geo2Tag – 0.31 version (Qt API 4.8);
- Smart-M3 – 0.9.01 (redland-1.0.16-unibo (Virtuoso), redsibd-0.9.01_time, sib-tcp 0.81, Libwhiteboard Qt API).

GCSS functional testing

Integration mechanisms unit-testing:

- loading geo-data from the Geo2Tag platform (basic filter by radius);
- triples filtering through the Geo2Tag platform;
- geo-data filtering through the Smart-M3 platform;
- convertion tags to triples and vice versa;
- insert data to the Smart-M3 platform;
- query data from the Smart-M3 platform.

GCSS performance testing 1/2

Performance metrics:

- query (operation) execution time;
- the number of operations performed in 1 second;
- the amount of consumed CPU and memory.

Performance testing tool:

- ***Profiler*** – Geo2Tag platform performance testing tool, that allow to identify some program functionality for the performance testing.

GCSS performance testing 2/2

Test case	Mean value (ms)	Standard deviation (ms)
Load tags from the Geo2Tag platform by radius	538	79.59
Triples filtering through the Geo2Tag platform	133	31.17
Geo-data filtering through the Smart-M3 platform	1621	424.97
Conversion 1000 triples to the tags	31,01	32.72
Conversion 1000 tags to the triples	27,24	15.73
Insert triples to the Smart-M3 platform	3015,4	173.16
Query triples from the Smart-M3 platform	1302.19	90.16

Main interesting test cases for further analysis:

1. geo-data filtering through the Smart-M3 platform (1-2);
2. insert triples to the Smart-M3 platform (3-4 seconds);
3. query triples from the Smart-M3 platform (1-1.5 sec.).

Performance testing and analysis 1/2

Profiling tools:

- Valgring + callgrind + kcachegrind;
- Intel Inspector XE 2013 (Intel Threads Profiler);

Two types of major problems:

- Multi-threaded data processing problems in the Smart-M3 platform components;
- Processing and parsing steps of obtained results for the basic Smart-M3 operations (insert, update, query) in the **Smart-M3 Qt API**.

Performance testing and analysis 2/2

- Smart-M3 multi-threaded data processing:
  - **redsibd** and **sib-tcp** components multi-threading problems (Intel Threads Profiles);
- **Libwhiteboard Qt API** processing and parsing part of the basic Smart-M3 operations (insert, update, query).
  - XML/DOM parser problems => **Binary protocol**;
- Tags-triples conversion mechanism bug (1 sec to 50 ms);
- **Smart-M3 + Virtuoso engine** provides 50% performance boost for the basic Smart-M3 operations;
- Smart-M3 KP API – **Python** or **Qt** or **C#** or **Java** or ... ???

GCSS optimization recommendations

- multi-threading errors correction for the Smart-M3 platform components – **redsibd**, **sib-tcp**;
- replacing current Smart-M3 platform protocol (SSAP) to the binary protocol, for example, KSP;
- use Smart-M3 platform with a Virtuoso engine (increase productivity of Smart-M3 platform operations ~ 50%);
- use Smart-M3 Python API or optimize basic platform operations (insert, update, query) in the Smart-M3 Qt API;
- use SparQL queries instead of the Template (WQL) queries (Libwhiteboard Qt API does not support SparQL queries);

Results and future steps

- performed functional and load testing of the platforms integration agent and it's basic integration mechanisms;
- identified weaknesses in the used platforms of the integration agent;
- provided some optimization recommendations for the platforms integration agent;

Future steps:

- performance of the whole system (platforms optimization);
- data cloud backend integration;
- context monitoring system.

Q&A

Kirill Yudenok {kirill.yudenok@gmail.com}
FRUCT & ETU OSLL {http://osll.fruct.org/}

Related works

- ***Pervasive Computing Research Group*** (indoor Location Based Services and coding real world objects);

- ***[K. Kolomvatsos et al'07]***: spacial ontology, ontology driven map annotation, GIS-based ontology population and navigation algorithms;

- ***[A. Dearle et al'03]***: tree-based region distribution of semantic information in global space (like geographical fractal structure of smart space data);