International Journal of Computer Science and Applications,

Vol. 8, No. 2, pp. 102 – 117, 2011

SEMANTIC WEB REASONING ON THE INTERNET SCALE WITH LARGE KNOWLEDGE COLLIDER

ALEXEY CHEPTSOV

High Performance Computing Center Stuttgart (HLRS)
Nobelstr. 19, 70560 Stuttgart, Germany
cheptsov@hlrs.de
http://www.hlrs.de

The latest advances in the Semantic Web community have yielded a variety of reasoning methods used to process and exploit semantically annotated data. However most of those methods have only been approved for small, closed, trustworthy, consistent, and static domains. In the last years, Semantic Web has been facing significant challenges dealing with the emergence of Internet-scale data-intensive application use cases. Still, there is a deep mismatch between the requirements for reasoning on a Web scale and the existing efficient reasoning algorithms over restricted subsets. This paper discusses the Large Knowledge Collider (LarKC), a platform, which focuses on supporting large-scale reasoning over billions of structured data in heterogeneous data sets. The architecture of LarKC allows for an effective combination of techniques coming from different Semantic Web domains by following a service-oriented approach, supplied by sustainable infrastructure solutions.

Keywords: Semantic Web; incomplete reasoning; scalability; performance, LarKC.

1. Introduction

Semantic Web has become de-facto an indispensable aspect of the human's everyday life, offering the content of the Web to be exploited by means of the techniques from formal Knowledge Representation, as described in Broekstra et. al. (2001). The machine-processable data representation formats, such as the Resource Description Framework RDF)[a], allow for data interchange on the Web and consequently enable automated reasoning to support more intelligent support on the Web. Such machine-understandable data enables novel uses of the Web such as semantic search, data integration, personalization and others described in Daconta et. al. (2003).

However, most of the currently available Web content is only suitable for human consumption and is not amenable to intelligent processing by machines. Even the Web content that is generated automatically from databases is usually represented excluding the original structural information located in databases. Typical use of the Web today involves people seeking and combining information, or reviewing catalogues of on-line stores and ordering products by filling out forms. The users themselves, without machine support, must do the most of this work. For example, the search engines only compare the string values without understanding the concepts denoted by those strings, and hence

[a] http://www.w3.org/RDF/

suffer from homonym- and synonym-problems; information from different catalogues cannot be combined automatically (except by ad hoc tools such as the current shop-bots and price-comparison sites), and web pages cannot be personalized to match the user's interests and requirements, respectively.

Furthermore, with ever growing scale of the semantic data, standard methods of reasoning become unrealistic. It becomes even much worse when the semantic data are inconsistent or incoherent. The former refers to the state in which there exists no any semantic model to interpret the data, whereas the latter refers to the state in which there exist some unsatisfiable concepts in the data. The standard inference procedures always assume that the data are consistent and coherent. However, it has observed that inconsistency and incoherence occur easily when the scales of the semantic data are increasing as shown by Huang et. al. (2008). That appeals for robust, scalable, trustful and performable reasoning methods to deal with the "big data".

In the recent years, a tremendous increase of the structured data sets has been observed on the Web, in particular in the government domain, as for example promoted by the Linking Open Data (LOD)[b] project, but also in science and e-Commerce (e.g. Ontoprise[c]). The massive amount of data, annotated using such standard modes for data interchange as RDF, is a key challenge for the Semantic Web. As a reaction to this challenge, several software engines have been developed, allowing processing over the data on the Web and even inferring logical consequences from the facts asserted in those data, i.e. performing reasoning over them**,** e.g. as described in Sirin et. al. (2007).

The amount of the data, semantically structured and annotated on the Web in the recent years, has been tremendously grown as Bizer et. al. (2009) describe, following the adoption of specifications like RDF and OWL[d] ontologies, as well as for the SPARQL query language format. Despite a great variety of reasoning algorithms and applications implementing them, available for Semantic Web, they all have faced with a great challenge – ability to scale up to the requirements of the rapidly increasing amount of data, such as those coming from millions of sensors and mobile devices, terabytes of scientific data produced by automated experimentation, or big enterprise content. On the other hand, there is still a need to cope with heterogeneity in knowledge representation and with noisy and inconsistent data. Furthermore, most of the current semantic reasoning approaches are centralized, which limits scalability to the boundaries of the hardware where the application is running , as stated in Hench et. al. (2008). Some work has been done on distributed reasoning and distributed information retrieval, please find details in Fensel and van Harmelen (2007). Nevertheless the techniques explored still present some limitations with regards to performance.

This paper presents Large Knowledge Collider (LarKC) – a new emerging solution for the challenges of reasoning over the "big data" for the Semantic Web. In Section 2, we discuss the main ideas how the Semantic Web reasoning can be applied for the large data volumes and introduce the LarKC's main concepts and objectives. In Section 3, we

---

b http://linkeddata.org
c http://www.ontoprise.de/
d http://www.w3.org/TR/owl-features/

introduce some typical reasoning application scenarios that will benefit from the LarKC execution model, namely a data-integration platform for the urban planning and a parallel, large-scale, and computation demanding statistical semantics application. In Section 4, we discuss the architecture of LarKC, concentrating on the user, platform, and infrastructure domains. In Section 5, we discuss the conclusions and future research directions for LarKC.

2. Concepts and Objectives of LarKC

2.1. *Large-scale Data Reasoning*

The broad availability of data coupled with increased computation and storage capacities and decreased exploitation costs has led the Semantic Web community to rethink the approaches to efficiently handle the abundance of data, in particular by the reasoning applications. Data-intensive reasoning requires a fundamentally different set of principles than the traditional mainstream Semantic Web offers.

As we have discussed above, the standard reasoning methods are not valid in the existing setting of the Semantic Web. The essence of LarKC is to go beyond traditional notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide "useful" responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches.

Interleaving reasoning and selection is considered to be one of the non-standard approaches to deal with the "big data". The main idea of the interleaving approach is to introduce a selection procedure so that the reasoning processing can focus on a limited (but meaningful) part of the data. Therefore, the procedure of interleaving reasoning and selection consists of the following selection-reasoning-decision-loop: i) select a (consistent) subset of the data, ii) reasoning with the selected data to get answers, and iii) deciding whether or not the answers are satisfying. If so, it would stops the processing, otherwise it goes to the step i) and the step ii) in the loop, as described by Huang (2010).

2.2. *Objectives of LarKC*

Aiming at avoiding the above-mentioned limitations, a plan to build the Large Knowlegde Collider (LarKC) - a trend-new platform for massive distributed incomplete reasoning - was firstly introduced by Fensel et al. (2008). This idea found its realization and implementation in the ICT EU-FP7 project LarKC[e] started in April 2008. In this paper, we present one of the main outcomes of LarKC, a service-oriented platform that facilitates creation of large-scale Semantic Web applications, eliminating the currently available technically specific scalability barriers of most existing reasoning engines.

[e] http://www.larkc.eu/

The LarKC project aims at building an experimental platform for massive distributed incomplete reasoning, which will support large-scale inferencing over billions of structured data in heterogeneous data sets by removing the scalability barriers pertained to the currently available reasoning engines as discussed in the previous section. The major mission and vision of LarKC is to enable a distributed Semantic Web reasoning infrastructure to go far beyond current reasoning paradigms that are strictly based on standard logics in order to scale up to the size of the current and future Web, as well as the reasoning requirements of future Web applications. This is achieved by implementing a highly innovative reasoning approach promoted by LarKC, which combines interdisciplinary problem solving techniques (inductive, deductive, incomplete reasoning, etc.) with methods from diverse fields (information retrieval, machine learning, cognitive and social psychology, parallel and distributed computing, and so forth). In order to achieve the identified goals, the software architecture of LarKC is built up on effective combination of those techniques coming from different disciplines and domains by following a service-oriented approach (Fig. 1).

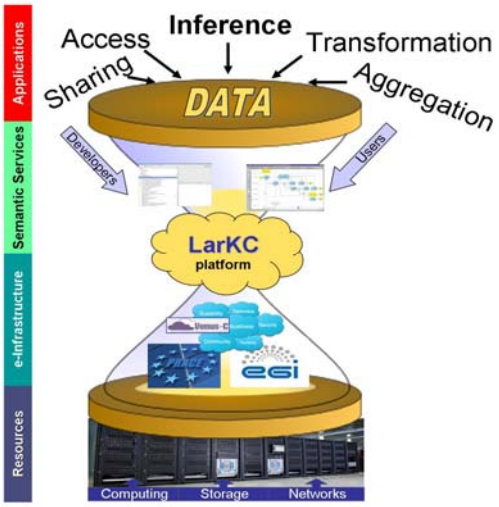


Fig. 1. LarKC as a platform for building enhanced semantic web applications.

3. Related Work

LarKC platform is primarily intended as a scalable and distributed infrastructure. Although distributed processing is not a complete solution for scalable reasoning, it is surely part of a solution. There are two opportunities for applying research results from distributed computing: in the plug-ins that implement particular forms components of reasoning, and in the platform that supports the execution of, and communication between, these plug-ins.

Peer-to-Peer systems are another type of a distributed system that is being explored in the development of the LarKC platform. Such systems are also most likely to be suitable for the implementation of distributed reasoning architectures and plug-ins with coarse computational granularity, either in terms of the duration of processing steps, or the degree to which applicable data can be segmented. While successful in supporting applications with small computation demands, the serial master/slave or peer-to-peer architectural approaches have begun to reach computational, operational and economic limits in the face of today's rapidly growing Interne-scale applications and data sets. On the other hand, the current mainstream parallel architectures such as clusters or high performance computers offer the power of several hundred thousands of CPU cores, Petabytes of storage, and Terabytes of RAM for performing the most challenging reasoning tasks. MPI and MapReduce are examples of parallelisation standards that enable high performance computing for conventional applications. Nevertheless, most production HPC infrastructures have a number of restrictions as for example a job-based application execution models, which causes the limited job's life time, or security limitations such as in- and outbound traffic restriction.

LarKC's goals were shared in part by the KAON project, see more details in Bozsak et. al (2002), which built an open-source ontology management infrastructure targeted for business applications. It produced a comprehensive tool suite allowing easy ontology creation and management and a framework for building ontology-based applications. An important focus of KAON was scalable and efficient reasoning with ontologies. LarKC extends beyond that aim by allowing the creation of highly complex workflows that integrate reasoning with ontologies with reasoning over massive data from the Semantic Web. It also extends the notion of reasoning to embrace techniques, such as the use of spreading activation for selection, that are not part of the standard formal reasoning toolkit.

In order to provide a flexible and modular environment where users and developers are able to build their own workflows and plug-ins respectively in an easy and straightforward manner, the LarKC platform is being designed following the latest concepts of modern Service-Oriented Architectures (SOA), including the semantic description of services and their inputs, outputs, and of meta-data such as Quality-of-Service (QoS) requirements. To this end, LarKC is a collaborator with and consumer of the work products of another European Research Project, named SOA4All, which aims at realizing a world where billions of parties are exposing and consuming services via advanced Web technology, see in Domingue et. al. (2008). The main objective of this project is to provide a comprehensive framework that integrates complementary and evolutionary technical advances (i.e., SOA, context management, Web principles, Web 2.0 and semantic technologies) into a coherent and domain-independent service delivery platform. However, LarKC foremost deals with the scalable reasoning over hundreds of thousands of RDF triples through a pluggable service platform while SOA4All, on the other hand, tries to build a service infrastructure allowing people to use different kinds of services more effectively.

Problem-solving environments or so-called virtual laboratories have been the subject of research and development for many years. LarKC is meant to be both a platform for executing Web-scale reasoning, and an experimental apparatus on which the techniques for web scale reasoning can be developed and evaluated. The plug-in architecture supports the preparation of experimental apparatus (new modes of knowledge preparation, knowledge selection, and reasoning) and their testing within a platform intended to vastly reduce experimental overhead. This architecture has been motivated in part by previous research on research infrastructures as well as workflow systems, such as famous Kepler, Taverna, and others.

4. Application Scenarios

In this section we will describe two typical application scenarios that will benefit from LarKC at most, one allowing for integration of heterogeneous data sets (urban computing) and another one that adopts large-scale data parallelism (statistic semantics).

4.1. *Urban Computing*

Urban Computing is a research branch that enjoys especial attention in Semantic Web. It aims at real-time aggregation and analysis of information about a city's population, events, and services location in order to regulate city infrastructure functions such as public transport and to provide context-sensitive navigation information. The other two case studies are in the life-sciences domain, related respectively to drug discovery and carcinogenesis research. Both life-sciences cases require large-scale data integration and analysis of scientific data and literature. All these use cases involve reasoning in some depth about data at a scale beyond what is possible with current Semantic Web infrastructure. Urban Computing enables the integration of computing, sensing, and actuation technologies into everyday urban settings and lifestyles in order to face the challenges of modern cities such as intelligent traffic management, (re)development of neighbourhoods and business districts, cost planning etc. as described by Kindberg et. al. (2007).

Urban settings range from personal cars and city buses to fixed public spaces such as streets and squares including semi-public environments like cafés, pubs or tourist attractions. Urban lifestyles are even broader and include people living, working, visiting and having fun in those settings. Not surprisingly, people constantly enter and leave urban spaces, occupying them with highly variable densities and even changing their usage patterns between day and night, as in Readers et. al. (2007).

Some years ago, due the lack of data and high-speed networks, solving Urban Computing problems, making urban settings individually responsive to their inhabitants and users, seemed to be an unrealizable dream. Recently, and quite suddenly, a large amount of the required information has been being made available on the Internet at almost no cost: maps with commercial activities and meeting places (e.g. Google Maps), events scheduled in the city and their locations, positions and speed information of public

transportation vehicles and, in some cases, of mobile phone users (Kane et. al.) , parking availability in specific parking areas, and so on.

However, current technologies are still not able to fully solve Urban Computing problems: doing so requires combining a huge amount of static knowledge about the city (including urban, legal, architectural, social and cultural knowledge) with an even larger set of data (originating in real time from heterogeneous and noisy data sources) and reasoning over the resulting time-varying information in order to retrieve and create useful knowledge.

For these reasons, Urban Computing serves as a challenging use case for large-scale semantic reasoning infrastructure, which involves heterogeneous data sources. With the particular use cases in the project, LarKC aims at providing added-value services to citizens in order to support them in coping with daily urban obstacles, which serves as a proxy of the more general case of serving the needs of billions of internet users by individual, custom application of reasoning over all the world's knowledge. More concretely, the requirements of Urban Computing with respect to semantic reasoning techniques allow us to identify the main issues relevant to building a sustainable reasoning platform, as in Della Valle et. al. (2008).

4.2. *Semantic Statistics based on Random Indexing*

One of the newest advances in semantic statistics, which enjoys the global data stores for the information retrieval, is Random Indexing. Random indexing is a distributional statistic technique used for extracting semantically similar words from the word co-occurrence statistics in the text data, based on high-dimensional vector spaces. It is a novel approach for word space modelling, founded on the distributional hypothesis described in Sahlgren (2005), according to which the two words that tend to constantly co-occur in several contexts have the similar meaning. Typically, a context refers to a multi-word segment, i.e. document. In the context of the data repository, that comprises a set of contexts C^m of size m, which are built upon a set of words X^n occurring in those contexts (obviously, $m \leq n$), it is possible, for each word x, belonging to the word set X^n, to build a so-called context vector (1), whose elements are defined by means of the co-occurrence function f_q:

$$\forall x \in X^n, \exists\, v=[\, f_q(x,c_j)] \tag{A.1}$$

where f_q is a co-occurrence function between the word x and each of the contexts, m is a total number of the contexts, n is a total number of the words in all contexts $c_j \in C^m$, $j = \overline{1..m}$.

The co-occurrence function is in the simplest case a frequency counter, specifying how many times the word occurs in the corresponding context, normalized and weighted in order to reduce the effects of high frequency words, or compensate for differences in document sizes. Therefore, the vector (1) represents the context in which the word occurs in the document collection. In the scope of the document base, the full word co-

occurrence characteristic is collected by combining the context vectors for each words resulting in the vector space – a two-dimensional co-occurrence matrix of size $m_x n$.

The vector space represents the distributional profile of the words in relation to the considered contexts/documents. The main methodical value of this profile is that it enables calculation of the semantic similarity between the words in scope of the document collection (text corpus), based on the cosine similarity of the given words' context vectors. Cosine similarity is a measure of similarity between the two vectors of m dimensions, equals to the cosine of the angle between them.

This context vector property has found a wide application in Semantic Web applications. Prominent examples of algorithms based on semantic spaces are query expansion and subsetting, described in Damljanovic et. al. (2010) and Guyon and Elisseeff (2003), accordingly. Query expansion is extensively used in Information Retrieval with the aim to expand the document collection (see Fig. 2a), which is returned as a result to a query thus covering the larger portion of the documents. Subsetting (also known as selection), on the contrary, deprecates the unnecessary items from a data set (see Fig. 2b), in order to achieve faster processing. Hence, the query expansion and subsetting applications are complementary, and are used to change properties of the query process to best adapt it to the search needs.

Random Indexing becomes very computation expensive when applied for Internet-scale data such as aggregated from Wikipedia or LOD. As such, the use of high performance computers becomes the key approach to achieve the desired scalability. Currently, only a few experiments with Random Indexing have been performed over the "big data". The reason for that is twofold. On the one hand, the code parallelisation requires quite a lot of development efforts and is often avoided by the application providers. However, since emerging such parallelisation techniques as for example MapReduce the main drawback become porting to a HPC environment. Such an application as Random Indexing is not easy to be ported to an HPC infrastructure as it is often a part of complex worklows that cannot be completely run in security-restricted HPC environments, for example when the access to any public database is needed. However, the component-based application realization as proposed by LarKC in conjunction with the distributed execution techniques allows these limitations to be easily overcome.

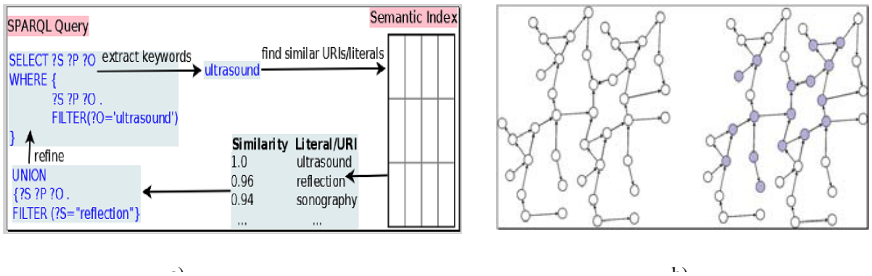


Fig. 2. Examples of Random Indexing applications: a) Query expansion, b) RDF subsetting.

5. Architecture of LarKC

5.1. *Overview*

The LarKC's design has been guided by the primarily requirement to be a scalable platform for distributed high performance reasoning. Fig. 3 shows a conceptual view of the LarKC architecture. The architecture consists of three major domains - the user domain, the platform domain, and the infrastructure domain. Each domain has its specific features and requirements. Only complexly addressing those requirements and ensuring the needed level of proliferation and collaboration among these domains, a trade-off between flexibility and performance can be ensured, in order to achieve a good balance between generality and usability of workflows and applications. Below we describe some of those key requirements and discuss the LarKC's architectural solutions for them.

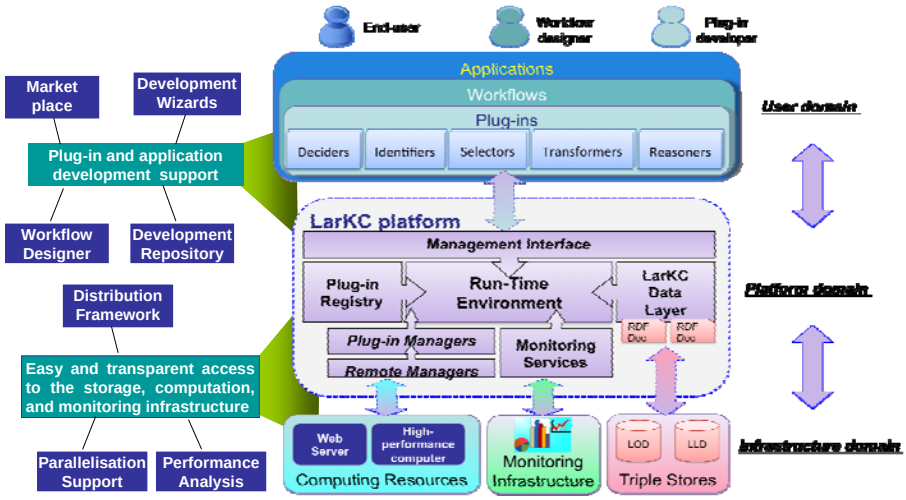


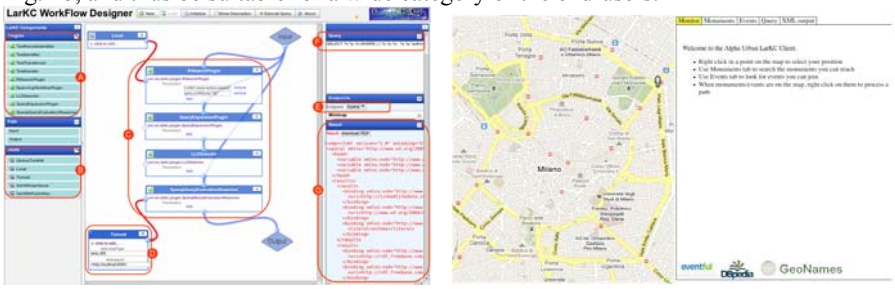
Fig. 3. The architecture of LarKC.

5.2. *User domain*

This domain serves as a consolidation layer for several categories of the users, namely Semantic Web services' developers, applications' designers, and applications' end-users who are not the Semantic Web experts.

The core of any LarKC applications is constituted by decoupled, lightweight, and flexible modules/services that programmatically implement some specific part of a reasoning logic as discussed in Section 2.1, whether it is selection, identification, transformation, or reasoning algorithm. Those services are called "plug-ins" in the LarKC terminology as can easily be plugged into a workflow.

In terms of LarKC the workflow is a reasoning application that is constructed of the plug-ins. The workflow's topology is characterised by the plug-ins included in the workflow as well as the connections between these plug-ins. The ready-to-use plug-ins are published on the Marketplace[f], which is a web-enabled repository where the plug-ins are available for downloading and using with the user's workflows. The workflow designers get access to the Marketplace in order to construct a workflow from the available plug-ins, combined to solve a certain task. Workflows are already standalone applications, which can be submitted to the platform and executed by means of such tools as Workflow Designer, shown in Fig. 4a. However, workflows can also be equipped with more powerful endpoints, such as one for the urban computing application, as shown in Fig. 4b, and thus be suitable for a wide category of the end-users.



a) b)

Fig. 4. The front-end interfaces of LarKC: a) Workflow Designer – the application that allows construction of workflow from the available plug-ins b) Urban Computing GUI – an end-user application

Plug-ins are grouped into categories corresponding to the common elements of functionality:

- *IDENTIFIER* - used for narrowing the scope of a reasoning task from all available information sets to only those potentially relevant to answering the query;
- *TRANSFORMER* - transforming data from one representation to another, these transformations can be simple, or arbitrarily complex;
- *SELECTER* - is responsible for selecting which identified input statements should be used for reasoning toward an answer to the user's query;
- *REASONER* - intended for applying different kinds of reasoning (deductive, inductive, etc.) to the data identified/selected/transformed before;
- *DECIDER* – a supervising component used to manage, control, and if necessary spawn new workflows at run-time. The latter is achieved by means of so called smart desiders, which use special platform services to instantiate new workflows on the fly if required by the reasoning logic; for such engines as Cyc can adjust the workflow configuration at time of the reasoning process, depending on the retrieved results.

For implementation of plug-ins, LarKC offers special API that allows the plug-in to be integrated into workflow and interact with the other plug-ins; see more info on the LarKC plug-in API in Assel et. al. (2011). Within the workflow, plug-ins communicate with

[f] See the LarKC Plug-in Marketplace at http://www.larkc.eu/plug-in-marketplace/

each other by means of messages containing RDF statements, i.e. each plug-in has an input and an output parameters of the RDF data type. The topology of the dataflow is defined by the workflow configuration, specified in RDF.

The plug-in development aspect is also in focus of LarKC. For even more user convenience, several infrastructure and collaborative software services are provided by LarKC for plug-in development. Among these services are such important as a Subversion-based repository for source code collaborative development and version control, or plug-in wizard that leads the developer through all steps of plug-in creation and generates a plug-in skeleton.

Although the plug-ins are lightweight and easy to develop, they are to a large extent useless if they don't collaborate with other plug-ins or don't benefit from the infrastructure resource layer. All these issues, including interconnection, instantiation, and even logging are covered by the platform domain, which we will further discuss.

5.3. *Platform domain*

The platform is the core module of LarKC that consolidates a few key services that enable plug-ins' and workflows' features that are intrinsically associated with the LarKC data and execution model. In particular, services for execution of plug-in-based workflows, centralised workflow and plug-in registration, management of the workflow execution (execution flow control), data management inside and over the plug-ins (data flow control), remote execution of plug-ins, etc. are provided. On the other hand, the platform allows the plug-in based applications to leverage diverse peer-to-peer infrastructure resources for distributed execution of their services. In the infrastructure domain, the focus lies on the proliferation of high performance computers into the resource layer of the applications, which is a major enabler of achieving the Internet-scale. A persistent storage based on the OWLIM RDF database engine for data integration over the plug-ins is served by the platform as well. Moreover, the applications can leverage a monitoring infrastructure for performance analysis issues.

The following main services are provided by the platform:

- The LarKC Runtime Environment (RTE) is the "control centre" of the LarKC Platform and responsible for the initialisation and invocation of workows. Its core components are the Executor, which is responsible for workflow invocation and control, and a set of Endpoints, which are user interfaces to send queries and retrieve results in the format needed by the workflow such as SPARQL. The executor also controls the workflow, managing the control and data dependencies over the plug-ins.
- The Management Interface is the platform's central front-end. It enables the users and also other services to retrieve the registered plug-ins, submit workflows, communication with the workflows (i.e. instantiate, start, stop, retrieve results etc.) .Some more advanced application such as the above-discussed Workflow Designer can make use of this service as well.
- The Plug-in Registry is a service that allows the platform to load the plug-ins as well as the external libraries needed by them to the internal plug-in knowledge base,

where the plug-ins can be instantiated from when constructing and executing the workflow.

- The Plug-in Managers facilitate the integration of plug-ins within a workflow and the management of their execution. Hence, they allow a plug-in to be executed either locally or in a distributed fashion. The latter is enabled with the help of the corresponding extensions for remote execution (i.e., adapters to connect to various remote resources, e.g. SSH or GT4 adapter).
- The Data Layer is an OWLIM realization that supports the plug-ins and applications with respect to storage, retrieval (including streaming), and lightweight inference on top of large volumes of RDF data. In particular, the Data Layer is used for storing the data passed between the plug-ins, so that only a reference is passed; this reveals the plug-ins from the need of handling the RDF data and hence make them applicable for large data volumes.

5.4. *Infrastructure domain*

The main mission of the platform is not only to provide means for creating plug-ins and developing workflows based on them, but also to serve a runtime environment for executing the workflows and supporting the applications relying on them. The runtime environment relies on distributed system approach, facilitating meeting the flexibility, QoS, or performance requirements by the plug-ins. Those are achieved by:

- splitting up the big dataset into subsets with further parallel processing each subset by a separate plug-in instantiated at a remote machine (data-level parallelism in Fig. 5);
- executing the plug-in on the resource hosting the data to be processed;
- executing computationally intensive plug-ins on the resource which ensures better performance characteristics as the one where the platform is deployed (instruction-level parallelism in Fig. 5).

The latter approach is especially beneficial to be applied for parallel systems when plug-ins utilize one of the parallelization approaches applied within LarKC, e.g. multithreading, MPI, or Map-Reduce.

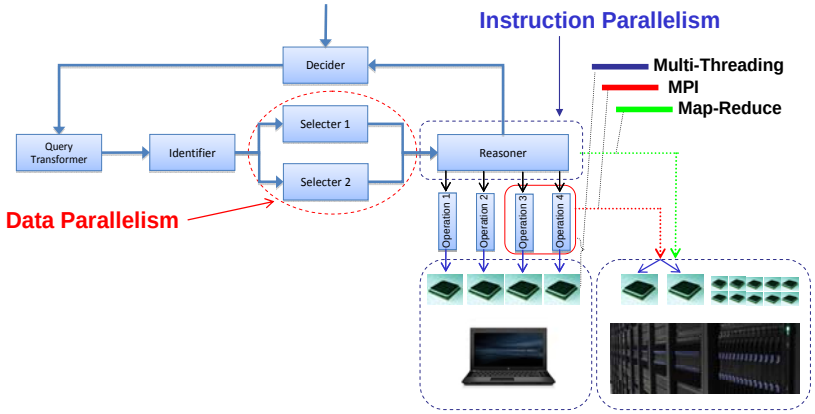


Fig. 5. Potential scenarios for distributed and parallel workflow execution.

The platform architecture design is flexible enough to enable a virtually unlimited variety of resource configuration for plug-in deployment and execution. Deployment options include combinations of generic remote web servers, desktop and service grids as well as cloud environment and others. Unless utilizing special features for instruction-level parallelism, there is no need in the adaptation of the plug-in to be at the remote resource. This is achieved by means of special platform manager extensions, such as shown in Fig. 6.

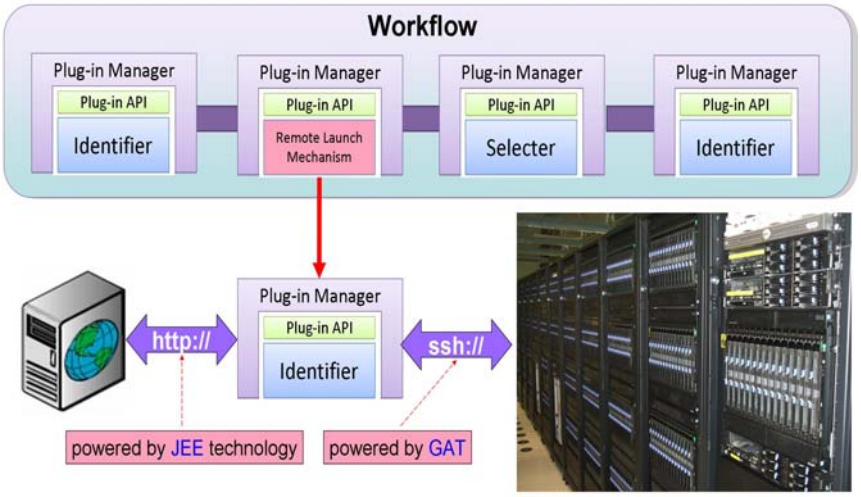


Fig. 6. Distributed plug-in execution in LarKC.

The LarKC architecture makes remote plug-in execution as simple as specification of a type of the resource as well as some basic properties (e.g. URI of the servlet the plug-in is packaged in) in the workflow properties, such as shown in Fig. 7.

```
1       <rdf:Description rdf:about="urn:tomcat">
2               <schema:resourceID rdf:resource="urn:my_webapp" />
3               <schema:resourceType rdf:resource="urn:Jee" />
4               <schema:resourceJeeUri rdf:resource="urn:http://localhost:8080" />
5               </schema:resourceProperties>
6       </rdf:Description>
```

Fig. 7. Exemplarily remote plug-in description properties for servlet-based hosting

Two main host types, recognized by the platform, are a web server (the plug-in is packaged in a servlet and provided as a web application) and a remote hosting machine (the plug-in is uploaded to the local file system and placed together with the other plug-ins). Both remote execution scenarios cover the following remote access protocols:

- HTTP, implemented with Java Servlet (JEE) technology,
- SSH/SCP[FTP], GSISSH/GSISCP, implemented with JavaGAT.

In case of HTTP, the plug-in is executed in a dynamic servlet container, e.g. Tomcat, and exposed as a web application. This requires the plug-in to be packaged into a servlet, in order to be able to respond an external HTTP request. However this packaging process is fully supported by the LarKC build tools. Accessing the web application with the plug-in is performed transparently by the LarKC platform.

Ensuring the minimum communication overhead for remote plug-in instantiation and synchronization with the locally running platform, this strategy can be applied for a wide range of hosts, in particular in a frame of a global grid (e.g. SETI@Home) scenario. To the main benefits of accessing the plug-in over HTTP can be referred very broad user community thanks to the widespread of the web technology.

Alternative to "a plug-in as a web application" approach is plug-in deployment on a remote machine accessible via a standard Security Shell (SSH) protocol, or special grid middleware, e.g. Globus Toolkit (GSISSH). The plug-in is accessed by means of the Java Grid Access Toolkit[g] (JavaGAT) technology. The plug-in is deployed as it is provided locally in the corresponding platform's folder. This approach helps overcome main disadvantages of the web applications – a need of a web server to be installed on the hosting machine that might be prohibited for some security-sensitive infrastructures, such as high-performance computers. Besides that, JavaGAT enables the following benefits when accessing the plug-ins: highly secure and trustful plug-in access and easy resource reservation and management for parallelized plug-ins. Nevertheless, the plug-ins running with JavaGAT suffer from the following drawbacks: job-based mode of the plug-in execution prohibits streaming operations and no unauthorized access to the plug-in is supported, resulting in decreasing the range of the users who may potentially access the plug-ins, unlike in case of a web application.

Both described technologies, Servlet and JavaGAT based, are complementary and offer LarKC plug-ins a broad spectrum of remote execution scenarios. The choice of the concrete strategy depends on the application use case, addressed user communities as well as available resources for plug-in deployment.

6. Conclusions and Future Directions

LarKC platform architecture has been designed with the main goals of openness, flexibility and scalability. At the same time, it must satisfy the requirements imposed by the emerging Semantic Web use cases, which will be its first large-scale users. LarKC allows the applications to reach a trade-off between flexibility (loosely coupled components, flexibility and dynamicity) and performance and scalability (if necessary, through a tailored solution to the concrete use case).

The platform already provides a high level of flexibility, performance, and scalability. Users are able to create own plug-ins and workflows or reuse existing components (plug-ins) or examples (workflows) for reasoning over large-scale semantic data sets. Through the component-based and service-oriented design, the platform allows for easy and automatic deployment and execution of certain plug-ins on multiple host infrastructures.

[g] http://gforge.cs.vu.nl/gf/project/javagat/

This use of more powerful resources can improve the performance for certain compute-intensive tasks like full closure computation of large-scale RDF data sets. In addition to its performance benefits, scalability is also enhanced through this distribution of plug-ins, which can be specified in the workflow construction phase.

The distribution of the workflow's load paves the way towards the parallel execution of the identified parallel branches and loops, that can be done both on multiple distributed hosts (e.g. with MPI or MapReduce) or even on a single local machine (e.g. by means of multithreading). The latter can be beneficial for the use cases running on real or near real time scale, whereby distributed processing of data is increasingly ineffective, such as in case of the discussed Urban Computing application. For the tested applications, applying of the parallelization techniques even on a single machine allows the application to increase the performance in several times, as achieved for the discussed Random Indexing application. In particular, especial interest arises around applying the distributed parallelisation techniques, such as MPI and MapReduce, as well as hybrid techniques combining several strategies (e.g. multithreading with MPI). We'll also keep on investigating and publishing on this topic.

One of our main future directions will be promotion of LarKC to a wider variety of application communities and performing more challenging experiments with the "big data".

Acknowledgments

This research is partially supported in part by the LarKC EU co-funded project (ICT-FP7-215535).

References

Assel, M.; Cheptsov, A.; Gallizo, G.; Celino, I.; Dell'Aglio, D.; Bradeško, L.; Witbrock, M.; Della Valle. E. (2011): Large knowledge collider: a service-oriented platform for large-scale semantic reasoning, http://dl.acm.org/citation.cfm?id=1988737&bnc=1. Proceedings of the International Conference on Web Intelligence, Mining and Semantics (WIMS'2011)

Berry, M. W.; Dumais, S. T.; O'Brien, G. W. (1995): Using linear algebra for intelligent information retrieval, SIAM Review, **37**, pp. 573-595.

Bizer, C.; Heath, T.; Berners-Lee, T. (2009): Linked data - the story so far. *International Journal on Semantic Web and Information Systems.* 5(3), 1-22.

Bozsak, E., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Stumme, G., Sure, Y., Tane, J., Volz, R., Zacharias, V. (2002): KAON - Towards a Large Scale Semantic Web. In *Tjoa, AM., Quirchmayr, G., Bauknecht K. (eds.) Proceedings of the Third international Conference on E-Commerce and Web Technologies.* LNCS, 2455, 304-313 Springer.

Broekstra, J.; Klein, M.; Decker, S.; Fensel, D.; van Harmelen, F.; Horrocks, I. (2001): Enabling knowledge representation on the Web by extending RDF schema. In: *Proceedings of the 10th international conference on World Wide Web (WWW '01)*. 467-478, ACM.

Daconta, M.C.; Smith, K.T.; Obrst, L.J. (2003): The Semantic Web: a Guide to the Future of Xml, Web Services, and Knowledge Management. *John Wiley & Sons, Inc.*.

Della Valle, E.; Celino, I.; Dell'Aglio, D.; Kim, K., Huang, Z.; Tresp, V.; Hauptmann, W.; Huang, Y.; and Grothmann, R. (2008): Urban Computing: a challenging problem for Semantic

Technologies. In Proceedings of the Second International Workshop New forms of reasoning for the Semantic Web: scalable, tolerant and dynamic, Co-located with the 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand, December 2008.

Domingue, J., Fensel, D., Gonzalez-Cabero, R. (2008): SOA4All, Enabling the SOA Revolution on a World Wide Scale. In: *ICSC 08: Proceedings of the 2008 IEEE International Conference on Semantic Computing*, 530-537, IEEE Press.

Fensel, D.; van Harmelen, F. (2007): Unifying Reasoning and Search to Web Scale. *IEEE Internet Computing*. 11(2), 96-95.

Fensel, D.; van Harmelen, F.; Andersson, B.; Brennan, P.; Cunningham, H.; Della Valle, E.; Fischer, F.; Huang, Z.; Kiryakov, A.; Lee, T. K.; Schooler, L.; Tresp, V.; Wesner, S.; Witbrock, M.; Zhong, N. (2008): Towards LarKC: A Platform for Web-Scale Reasoning. In: *Proceedings of the 2008 IEEE international Conference on Semantic Computing ICSC*. 524-529, IEEE Computer Society.

Hartig, O.; Zhao, J. (2009): Using web data provenance for quality assessment. In: Freire, J., Missier, P., Sahoo, S.S. (eds.) Proceedings of the First International Workshop on the role of Semantic Web in Provenance Management (SWPM 2009), CEUR-WS.org

Hench, G., Simperl, E., Wahler, A., and Fensel, D. (2008): A Conceptual Roadmap for Scalable Semantic Computing. In: *Proceedings of the 2008 IEEE international Conference on Semantic Computing ICSC*. 562-568, IEEE Computer Society.

Huang, Z.; van Harmelen, F.; and ten Teije, A. (2005): Reasoning with inconsistent ontologies. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'05, pages 454-459, 2005.

Huang, Z.; and van Harmelen, F. (2008): Using semantic distances for reasoning with inconsistent ontologies. In:Proceedings of the International Semantic Web Conference 2008 (ISWC08), Lecture Notes in Computer Science 5318, pages 178-194. Springer.

Huang, Z. (2010): Interleaving Reasoning and Selection with Semantic Data, Proceedings of the 4th International Workshop on Ontology Dynamics (IWOD-10), ISWC2010 Workshop.

Kane, L.; Verma, B.; Jain, S.: Vehicle tracking in public transport domain and associated spatio-temporal query processing. Computer Communications. 31, 2862-2869.

Kindberg, T.; Chalmers, M.; Paulos E. (2007): Introduction: Urban Computing. IEEE Pervasive Computing. 6, 18-20.

McKendrick, J. (2010): Size of the data universe: 1.2 zettabytes and growing fast, ZDNet.

Reades, J.; Calabrese, F.; Sevtsuk, A.; Ratti, C. (2007): Cellular Census: Explorations in Urban Data Collection. IEEE Pervasive Computing. 6, 30-38.

Sahlgren, M. (2005): An introduction to random indexing. Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering TKE 2005, pp. 1-9.

Sirin, E.; Parsia, B.; Grau, BC.; Kalyanpur, A.; Katz, Y., Pellet (2007): A practical OWL-DL reasoner. *Web Semantics*. 5,2 51-53