Large Knowledge Collider - a Service-Oriented Platform for Large-Scale Semantic Reasoning

Matthias Assel,
Alexey Cheptsov,
Georgina Gallizo
High Performance Computing Center Stuttgart
Nobelstr. 19, 70560 Stuttgart (Germany)
+49 -711-685-87269
assel@hlrs.de

Matthias Assel,
Alexey Cheptsov,
Georgina Gallizo
High Performance Computing Center Stuttgart
Nobelstr. 19, 70560 Stuttgart (Germany)
+49 -711-685-87269
assel@hlrs.de

Matthias Assel,
Alexey Cheptsov,
Georgina Gallizo
High Performance Computing Center Stuttgart
Nobelstr. 19, 70560 Stuttgart (Germany)
+49 -711-685-87269
assel@hlrs.de

Irene Celino,
Daniele Dell'Aglio
CEFRIEL - ICT Institute
Politecnico di Milano
Via Fucini 2, 20133 Milano (Italy)
+39-02-23954-266
irene.celino@cefriel.it

Luka Bradeško,
Michael Witbrock
Cycorp Europe
Teslova Cesta 30, Ljubljana,
SI-1000 Slovenia
+386-51-368-848
luka@cycorp.eu

Emanuele Della Valle
DEI – Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 Milano (Italy)
+39-02-23954-324
emanuele.dellavalle@polimi.it

ABSTRACT

Recent 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. 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 describes the pilot implementation of LarKC – the Large Knowledge Collider, 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.

Categories and Subject Descriptors

I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving – *inference engines.*

I.2.3 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *semantic networks.*

H.3.4 [**Information Storage and Retrieval**]: Systems and Software– *distributed systems, question-answering systems.*

General Terms

Algorithms, Management, Performance, Design, Reliability, Experimentation, Standardization, Languages, Theory.

Keywords

Semantic Web, Reasoning Infrastructure, Service-Oriented Platform, Large Knowledge Collider.

1. INTRODUCTION

The essence of the Semantic Web is the idea that the Web can exploit techniques from formal Knowledge Representation [1] to make information available in machine processable format, allowing 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 [2]. Semantic Web has become de-facto an indispensable aspect of the human's everyday life, offering a content of the Web to be utilized from within the users' applications.

Still, most of the current Web content is only suitable for human consumption and is not amenable to intelligent processing by machines. Even Web content that is generated automatically from databases is usually presented without the original structural information found 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 much of this work. Search-engines only search for strings without understanding the concepts denoted by these strings, and hence 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.

In the recent years, a tremendous increase of 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)[1] project, but also in science and e-Commerce (e.g. Ontoprise[2]). The massive amount of data, in particular described by RDF[3] (Resource Description Framework) – a standard model for data interchange on the Web – is a key challenge for 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. [3]):

The amount of the data, semantically structured and annotated on the Web in the recent years, has been tremendously grown [4], following the adoption of specifications like RDF and OWL[4] 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 [5]. Some work has been done on distributed reasoning and distributed information retrieval [6]. Nevertheless the techniques explored still present some limitations with regards to performance.

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. [7]. This idea found its realization and implementation in the ICT EU-FP7 project LarKC[5] 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. The remainder of this document is structured as follows:

Section 2 introduces LarKC, its main objectives and tasks to be supported. Section 3 gives a short summary of related projects and initiatives, and describes their cross-fertilization with LarKC. Section 4 presents one of the pilot scenarios in LarKC – Urban Computing, pointing out the advantages of service-oriented and LarKC-supported execution for a typical application in this area. Section 5 introduces the architecture of LarKC and describes in details, its main components and subsystems. Section 6 summarizes the work done and provides an outlook about further actions to be performed within the LarKC project.

2. CONCEPT AND OBJECTIVES OF LARKC

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 [5], as well as the reasoning requirements of future Web applications. This is achieved by implementing a highly innovative reasoning approach [7] 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 (Figure 1).

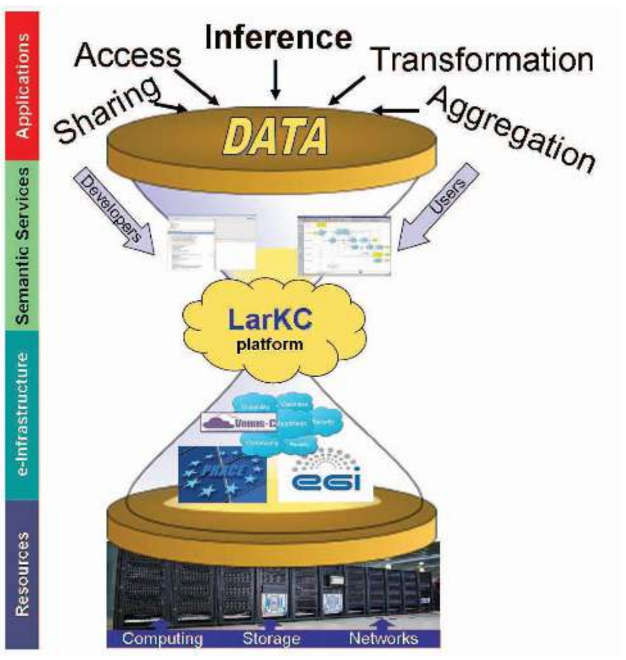


Figure 1. LarKC as a platform for building enhanced semantic reasoning applications

The two main question solved by LarKC are how to develop a reasoning application, combining solutions and techniques coming from diverse domains of the Semantic Web as well as other Computer Science disciplines (e.g. High Performance Computing), and how to ensure the requested quality-of-services requirements, in particular by targeting the modern e-Infrastructures such as Grids and Cloud environments.

In solving those questions, the LarKC approach follows the divide-and-conquer principle [8], applying it for complex reasoning problems. According to this approach, a reasoning algorithm can be built by coupling lightweight Semantic Web computational blocks for identification, selection, transformation, and actual reasoning in a common workflow. This approach allows blocks to be separate and lightweight, and ensures high flexibility and easy reusability of the blocks when building an application on top on them.

Considering the blocks as services (or "plug-ins" in LarKC terminology), LarKC offers a platform for development of highly flexible reasoning applications based on them. The plug-ins, performing a subtask related to data access and reasoning functionality, can be used as both standalone components or within an application/workflow with more complex functionality, depending

[1] http://linkeddata.org

[2] http://www.ontoprise.de/

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

[4] http://www.w3.org/TR/owl-features/

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

on the task to be solved. Plug-ins, developed by geographically distributed teams of researchers and practitioners, constitute a marketplace[6], which paves the way towards "true" interoperability of the components as well as exchangeability of plug-ins or entire workflows. Wrapping an existing component/algorithm into a LarKC plug-in ensures its interoperability and usage within a big family of the already developed plug-ins that can be combined into more advanced workflows.

Further to providing a framework for development of plug-ins and workflows, the LarKC platform serves a deployment environment for their execution. The distinctive feature of the LarKC platform is that from the very beginning it was designed for support of distributed computing environments. This means that the workflows will take the advantage of executing their parts at the distributed resource pool. Among the resources supported are web servers (e.g. Tomcat), available over HTTP, as well as any network-connected computer, accessible by means of standard protocols, such as SSH. When running on a distributed system, the workflows benefit not only from parallel execution of concurrent operations on different resources, i.e. realizing a SETI@home scenario[7], but also from shipping the code closer to the data processed (as in case of the web server solution) or even running parts of the computationally expansive algorithms on high performance computers in the context of so-called Computational Grids [9].

3. RELATED WORK

The 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 (compare with Figure 7).

Some previous work in distributed computing is particularly relevant to the development of plug-ins for reasoning and information retrieval. Previous approaches include [10], [11] and [12]. In the same way, distributed computing techniques can be applied to scale data storage [13], for example, proposes a distributed RDF storage and retrieval engine. Technologies such as MapReduce [14], which enables processing of large data sets, and BigTable [15], a distributed storage system for structured data, are being actively explored for support in the platform, where they can offer organizing principles for the replicated execution of data-reduction and reasoning, and storage and retrieval plug-ins, respectively.

Peer-to-Peer systems [16] 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.

LarKC's goals were shared in part by the KAON project [17], 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 [18]. 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 [19]. 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 Kepler [20], Taverna [21], and others.

4. APPLICATION SCENARIO – URBAN COMPUTING

The success and application of the service-oriented platform is being be demonstrated, within the LarKC project itself, in three end-user case studies. The first case study is from the Urban Computing sector. 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.

Throughout the rest of the paper, we concentrate on the first case study - real-time reasoning over large volumes of data in order to provide more responsive, dynamic and efficient urban environments. In this section, we first give some brief background information about this scenario.

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

[6] http://www.larkc.eu/plug-in-marketplace/

[7] http://setiathome.berkeley.edu/

management, (re)development of neighborhoods and business districts, cost planning etc [22].

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 [23].

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 [24], 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. 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 [25].

5. ARCHITECTURE OF LARKC

5.1 Overview

From the system organization's standpoint, LarKC is a software infrastructure that offers:

- a framework for development of decoupled components (plug-ins) as well as for constructing workflows based on those plug-ins;

- a run-time environment for launching and executing the workflows on a dynamic resource pool of heterogeneous and distributed systems, ranging from web servers hosting the plug-in to parallel and supercomputing systems: The latter ensure optimal performance and scalability characteristics for computationally intensive applications.

LarKC strives to beneficially apply the divide-and-conquer approach [8] for operational support of highly-flexible, lightweight Semantic Web computational blocks and, considering them as services, to allow loosely coupled, reliable and asynchronous interactions among the blocks as well their composition to build powerful applications.

The actual edition of the LarKC platform, which reuses a substantial code base donated by OpenCyc[8], has been designed as a flexible, pluggable and extensible architecture, which enables design and testing of new reasoning techniques at vastly reduced costs (i.e., different techniques could be easily combined, compared and evaluated within a single workflow). Therefore it aims to be open, dynamic and scalable but, at the same time, to fulfill the requirements imposed by applications (refer to Section 4). The techniques and thus the design of the LarKC architecture allows for a tradeoff between flexibility and performance, in order to achieve a good balance between generality and usability of workflows and applications, respectively.

The architecture of LarKC can be conceptualized as instantiating this service-oriented approach. Following the middle-out approach, the architecture serves three main subsystems/domains: the user domain, platform domain, and infrastructure domain (see Figure 2).

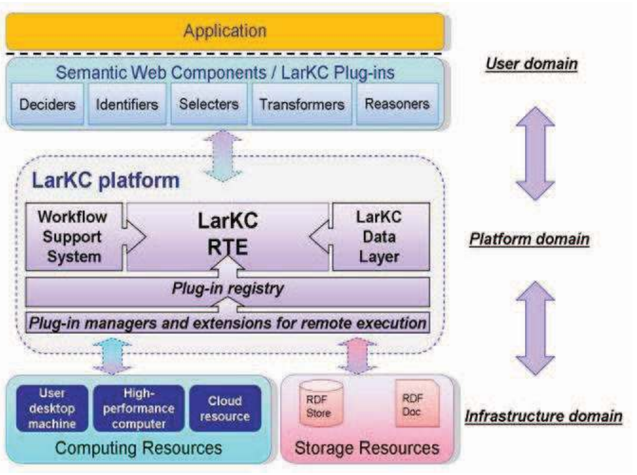


Figure 2. The High-level LarKC Architecture.

From the user's perspective, the architecture enables gluing together diverse operational blocks and provides a framework for interconnecting those within complex applications/workflows in a way that ensures the components' scalable deployment and applications' maximal performance. The use of LarKC basically involves three categories of user groups:

- *Plug-in developers* who design and implement single plug-ins and deploy them in the platform or publish them on the marketplace[9] for future use;

- *Workflow designers* who select existing plug-ins and combine them in the appropriate way to solve a certain task, either by implementing a scripted Decider (see Decider definition below);

- *End-users* who use the services provided by the platform, i.e. a particular workflow configuration to execute SPARQL queries. This interaction pattern includes both interaction with existing application interfaces and designers that implement applications that need to use the services provided by the platform.

A single LarKC module implementation is a *Plug-in*. Plug-ins are grouped into categories corresponding to the common elements of functionality recognized in the Semantic Web communities and targeted by LarKC:

[8] http://www.opencyc.org

[9] http://www.larkc.eu/plug-in-marketplace/

- *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 that enables construction, management, control and execution of the plug-ins within the workflows. Coupled together into a common workflow, workflows of LarKC plug-ins are intended to efficiently handle diverse large-scale semantic reasoning tasks. Such workflows are often complex, branched and dynamic compositions of the basic components identified above. A decider is responsible for building and maintaining such a workflow, acting as a main entry point for interfaces of semantic reasoning applications, represented in Figure 2 as "Application" rectangle. Deciders can be implemented in two different ways: 1) In form of a simple, linear decider (so-called "scripted deciders") which statically follow a predefined structure of tasks or 2) as "intelligent" components that can dynamically influence or even re-enact workflows according to sudden changes. What is more, complex workflows may delegate responsibility to subordinate deciders.

The abovementioned blocks are deployed within the platform according to the module-based approach, following the main SOA principles [26]: pieces of self-contained functionality, functional interoperability, using different infrastructures and technologies, and loose coupling by reducing dependencies.

Figure 3a shows an example of a workflow for the urban scenario, introduced previously in Section 4. The workflow is used for the identification and selection of interesting monuments in Milan based on the user actual position and comprises of the following plug-ins:

- a Decider, selecting the correct workflow based on the input query, i.e. the decider catches the SPARQL queries forwarded from the overarching application and executes different workflows according to a given user request (the urban scenario comprises three subworkflows);

- a Transformer, analyzing a SPARQL query to get the triple patterns that are passed to the Identifier;

- an Identifier, querying Sindice [27] search engine to get possibly-relevant RDF documents;

- a Selecter, filtering the documents to extract information about relevant monuments;

- a Reasoner, that identifies and combines content from the RDF documents to answer the user's query.

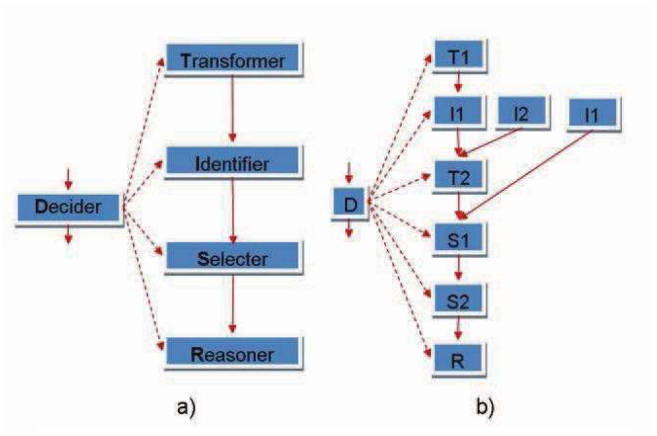


Figure 3. Examples of simple LarKC workflows a) the Urban LarKC Milan Monument Scenario b) a hypothetical branched workflow

A more complex workflow is presented in Figure 3b. The workload in this workflow is shared out among several plug-ins or several instances of the same plug-in. Even these straightforward workflows require that the LarKC plug-in architecture and the implemented platform support data and control flow management among those plug-ins.

5.2 The Platform Domain

The platform's main goal is to provide full operational support for several categories of users (identified user groups are explained in the see section above). This is achieved by means of a complete run-time environment (RTE) provided by the platform as well as a service set, which is loosely coupled with the RTE. The LarKC RTE is based on the OpenCyc engine, which is most visible via the platform-provided semantic web endpoint for remote processing the SPARQL queries (accessible through HTTP) as well as offers basic reasoning functionalities that can be used to identify suitable plug-ins for a particular workflow (i.e., users can ask LarKC to find a plug-in that fulfills their requirements, e.g. a plug-in that is able to retrieve triples from Sindice).

The main services provided by the platform (taking advantage of the elements of OpenCyc and Ontotext OWLIM [28] included) are:

- Plug-in Registry

- Plug-in Management System

- Workflow Support System

- Data Layer

The *Plug-in Registry* is intended to store, register and retrieve plug-ins in the internal knowledge base. The LarKC platform's knowledge base is designed in a way that makes the plug-ins cacheable, easy to manipulate, fast to access and retrieve, and prepared for reasoning tasks with almost no additional overhead. The compliance of the plug-ins within the registry knowledge base is ensured through the Plug-in API elaborated in LarKC (Figure 4).

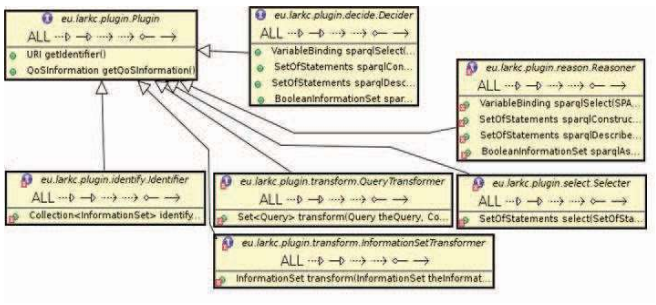

eu.larkc.plugin.Plugin
ALL
URI getIdentifier()
QoSInformation getQoSInformation()
eu.larkc.plugin.decide.Decider
VariableBinding sparqlSelect(...
SetOfStatements sparqlCon...
SetOfStatements sparqlDesc...
BooleanInformationSet spar...
eu.larkc.plugin.reason.Reasoner
VariableBinding sparqlSelect(SPA...
SetOfStatements sparqlConstruc...
SetOfStatements sparqlDescribe...
BooleanInformationSet sparqlAs...
eu.larkc.plugin.identify.Identifier
Collection<InformationSet> identify...
eu.larkc.plugin.transform.QueryTransformer
Set<Query> transform(Query theQuery, Co...
eu.larkc.plugin.select.Selecter
SetOfStatements select(SetOfSta...
eu.larkc.plugin.transform.InformationSetTransformer
InformationSet transform(InformationSet theInformat...

Figure 4. Basic-level LarKC Plug-in API

Plug-ins that conform to the interfaces in Figure 4 can be automatically registered with the platform for composition into workflows.

In addition, each plug-in is supplied with a semantic annotation, as part of the component that allows the platform to find and manage them in an efficient way. The semantic description includes information about the plug-in type, its functional and non-functional properties [29], and the input and output data processed respectively produced by this plug-in.

The coupling of the plug-ins and their interoperability is achieved by means of the *Plug-in Managers*, which serve a high-level container that enables data and control flow between the plug-in and platform and among the plug-ins. By exchanging relevant information during runtime, the plug-in manager communicate with the respective plug-in and control the state at any time. Hence, deciders can directly interact with the managers and check the actual behavior of one particular plug-in, so as to react on any problems. Furthermore, plug-in managers support plug-in execution in the runtime environment served by the platform (details about remote execution are provided in the next section).

Thanks to these managers, the plug-ins can easily be integrated into a common workflow. The integration is supported by a collection of platform tools and utilities that compose the *Workflow Support System*. This system helps the decider to find the appropriate plug-ins in the registry and dynamically construct, execute and manage the workflow in the runtime environment set up by the platform. The following services are enabled by the system:

- *Queuing mechanisms*

- *Flow Control mechanisms*

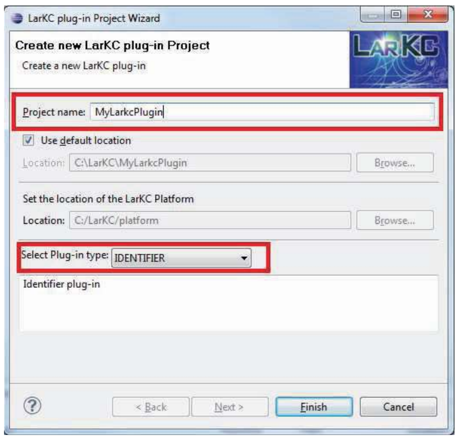
- *Logging capabilities*

The queues and multiplexers as the core part of the Workflow Support System provide the mechanism for connecting different plug-ins together within a workflow. Queues provide a simple mechanism allowing plug-ins to send and receive data or control messages to and from the plug-ins immediately before or after them in the workflow. The multiplexers provide a mechanism for splitting, joining, or distributing messages, for example where one plug-in sends out a single messages that must be sent on to multiple other plug-ins or where the output of multiple plug-ins must be combined before sending this data on to the next plug-in in the workflow. The queues and multiplexers thus make it easy for decider plug-in writers to build the plumbing between different plug-ins, created through the plug-in factory, in the workflow.

Additionally, the Workflow Support System provides a unified logging system for the use of plug-ins such that plug-in writers do need to create their own solutions. The infrastructure provides facilities for raising logging messages of different sorts, e.g. ERROR, WARNING, DEBUG, and INFO. Decider plug-in writers can thus use a single logging point for debugging and checking their workflows, rather than having to navigate individual log files from the different plug-ins that make up the workflow.

Data access, storage and management are facilitated in the platform by means of the *Data Layer*. The Data Layer supports the plug-ins and applications with respect to storage, retrieval, and lightweight inference on top of large volumes of RDF data. The layer relies heavily on OWLIM [28] and implements a range of data access modalities, including a direct Java API, SPARQL endpoint and linked data publication. Among the data layer's main features are: persistent storage to RDF data; a reference implementation of the ORDI data model; facilities for passing data either by value or reference; resolvable RDF data identifiers; optional forward-chaining reasoning, under RDF semantics, within the data layer itself; interfaces that allow streaming data processing and utility methods for queries to remote data or multiple datasets, since often RDF data is published on the web without support for remote querying.

For developer convenience, the LarKC development activities are carried out in public view on the SourceForge[10] development hosting service, which offers, among many other services, a centralized software code repository for maintaining all the plug-ins. At time of this paper's writing, more than forty LarKC plug-ins have been implemented, some designed from scratch, and some by wrapping with the LarKC APIs. Plug-in development is facilitated by the provision of templates and masters or through a graphical user interface called the Plugin-Wizard (see Figure 5), and support is provided for several development environments (including the popular Eclipse and NetBeans IDEs).


LarKC plug-in Project Wizard

Create new LarKC plug-in Project

Create a new LarKC plug-in

Project name: MyLarkcPlugin

☑ Use default location

Location: C:\LarKC\MyLarkcPlugin

Set the location of the LarKC Platform

Location: C:/LarKC/platform

Select Plug-in type: IDENTIFIER

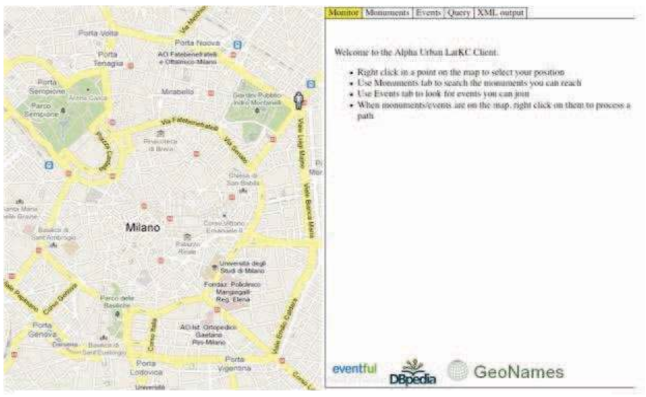
Identifier plug-in

Figure 5: The LarKC Plug-in Wizard for easy plug-in development

Workflows built on top of the plug-ins can easily be integrated into high-level user applications. For example, based on the Alpha Urban LarKC workflow presented and discussed above, CEFRIEL has created an application[11] for web planning of visits to the monuments in Milan (Figure 6).

[10] LarKC@SourceForge: http://larkc.sourceforge.net/

[11] http://seip.cefriel.it/alpha-Urban-LarKC/



Monitor | Monuments | Events | Query | XML output

Welcome to the Alpha Urban LarKC Client.

- Right click in a point on the map to select your position
- Use Monuments tab to search the monuments you can reach
- Use Events tab to look for events you can join
- When monuments/events are on the map, right click on them to process a path

Figure 6. The Alpha Urban LarKC application view.

This application shown in Figure 6 is backed by a LarKC Urban Computing workflows and allows users to plan paths through the Milan city, taking into account current web content providing information about local monuments and events.

5.3 The Infrastructure Domain and Distributed Execution

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 Figure 7);

- 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 Figure 7).

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.

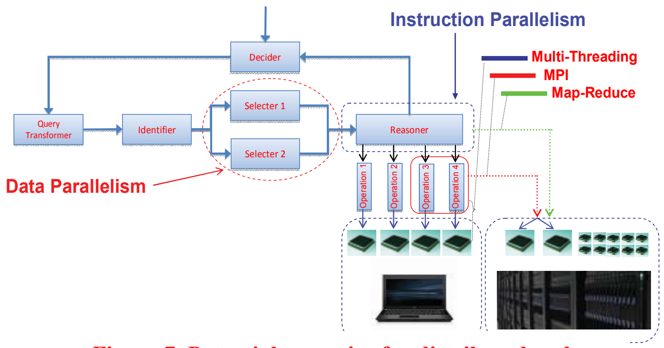


Figure 7. Potential scenarios for distributed and parallel workflow execution.

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 Figure 8.

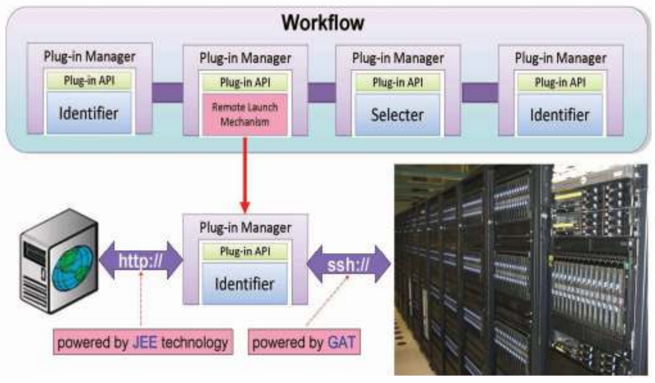


Figure 8. Distributed plug-in execution in LarKC.

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 Figure 9.

```
    <rdf:Description rdf:about="urn:tomcat">
            <schema:resourceID rdf:resource="urn:my_webapp" />
            <schema:resourceType rdf:resource="urn:Jee" />
            <schema:resourceJeeUri rdf:resource="urn:http://localhost:8080" />
            </schema:resourceProperties>
    </rdf:Description>
```

Figure 9. 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[12] (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. CONCLUSION AND OUTLOOK

The 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 LarKC use cases, which will be its first large-scale users. Therefore, a tradeoff must be reached between flexibility (loosely coupled components, flexibility and dynamicity) and performance and scalability (if necessary, through a tailored solution to the concrete use case).

In its current implementation state, 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 service-oriented design, the platform allows for easy and automatic deployment and execution of certain plug-ins on multiple host infrastructures. 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 Urban Computing application presented. For the tested applications, applying of the parallelization techniques even on a single machine allows the application to increase the performance in several times; more details about performance evaluation for some time critical use case can be found in our previous publications [30],[31],[32]. 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.

Besides, during the remaining project period, work will continue on the development of a distributed data layer, whose architecture has been driven by use-case performance needs, and our experience in working with the currently available platform. We will also further investigate the application of parallelization techniques for e.g. ontology matching [30] and the support for remote execution on a wider variety of resources and under a wider range of parallelization models. Research will continue on the definition, documentation and support of design patterns, providing best practices and guidelines for LarKC users (mainly workflow designers and plug-in developers) to take the maximum advantage of the LarKC platform features, achieving the needed level of performance for their applications. Finally, all LarKC project use cases as well as traditional and newly emerged reasoning techniques (i.e., logic-based, probabilistic reasoning versus incomplete, cognitively inspired reasoning) shall be migrated as rapidly as possible to the LarKC specification in order to make use of the advanced capabilities provided by the platform, and to maximize the rate at which those capabilities evolve towards supporting true Web-scale reasoning.

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

7. ACKNOWLEDGMENTS

This work has been supported by the LarKC project (http://www.larkc.eu), partly funded by the European Commission's IST activity of the 7th Framework Programme (ICT-FP7-215535). This work expresses only the opinions of the authors.

8. REFERENCES

[1] 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.

[2] 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.*.

[3] Sirin, E., Parsia, B., Grau, BC., Kalyanpur, A., Katz, Y. Pellet. 2007. A practical OWL-DL reasoner. *Web Semantics*. 5,2 51-53.

[4] 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.

[5] 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.

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

[7] 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.

[8] Bao, J., Honavar, V. 2006. Divide and Conquer Semantic Web with Modular Ontologies - A Brief Review of Modular Ontology Language Formalisms. In: *Haase, P., Honavar, V., Kutz, O., Sure, Y., Tamilin, A. (eds.) Proceedings of the 1st International Workshop on Modular Ontologies (WoMO'06)*. 1-14, CEUR-WS.org.

[9] Foster, I., Kesselman, C (eds). 1998. Computational Grids: The Future of High Performance Distributed Computing. *Morgan Kaufmann*.

[10] Feier, C. 2007. A framework for distributed reasoning on the semantic web based on open answer set programming. In: *Proceedings of the KWEPSY 2007 Knowledge Web PhD Symposium*.

[11] Serafini, L., Tamilin, A. 2005. DRAGO: Distributed reasoning architecture for the semantic web. In *Gomez-Perez, A., Euzenat, J. (eds.). The Semantic Web: Research and Applications.* LNCS, 3532, 361-376, Springer.

[12] Straccia, U., Troncy, R. 2006. Towards distributed information retrieval in the semantic web: Query reformulation using the omap framework. In: *Sure, Y., Domingue, J. (eds.) Proceedings of the 3rd European Semantic Web Conference.* LNCS, 4011, 378-392, Springer.

[13] Adamku, G., Stuckenschmidt, H. 2005. Implementation and evaluation of a distributed rdf storage and retrieval system. In: *WI 2005: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence.* 393-396, IEEE Press.

[14] Urbani, J., Kotoulas, S., Oren, E., van Harmelen, F. 2009. Scalable Distributed Reasoning Using MapReduce. In: *Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009.* LNCS, 5823, 634-649, Springer.

[15] Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.A.. 2008. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.* 26,2.

[16] Adjiman, P., Chatalic, P., Goasdoue, F., Rousset, M.C., Simon, L. 2006. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research*. 25, 269-314.

[17] 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.

[18] 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.

[19] Rycerz, K., Bubak, M., Sloot, P., Getov, V. 2006. Problem solving environment for distributed interactive simulations. In *Gorlatch, S., Bubak, M., Priol, T. (eds.) Achievements in European Reseach on Grid Systems. CoreGRID Integration Workshop 2006 (Selected Papers)*, 55-66, Springer.

[20] Altintas, I., Barney, O., Cheng, Z., Critchlow, T., Ludaescher, B., Parker, S., Shoshani, A., Vouk, M. 2006. Accelerating the scientific exploration process with scientific workflows. *J. Phys.: Conf.* 46, 1 468+.

[21] Oinn, T., Greenwood, M., Addis, M., Nedim Alpdemir, M., Ferris, J., Glover, K., Goble, C.A., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M. R., Senger, M., Stevens, R., Wipat, A., Wroe, C. 2006. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience.* 1067-1100, Wiley.

[22] Kindberg, T., Chalmers, M., Paulos E. 2007. Introduction: Urban Computing. *IEEE Pervasive Computing.* 6, 18-20

[23] Reades, J., Calabrese, F., Sevtsuk, A., Ratti, C. 2007. Cellular Census: Explorations in Urban Data Collection. *IEEE Pervasive Computing*. 6, 30-38.

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

[25] Emanuele Della Valle, Irene Celino, Daniele Dell'Aglio, Kono Kim, Zhisheng Huang, Volker Tresp, Werner Hauptmann, Yi Huang, and Ralph Grothmann. 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.*

[26] Josuttis, N. 2007. SOA in practice – The art of distributed system design. *O´Reilly*.

[27] Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G. 2008. Sindice.com: a document-oriented lookup index for open linked data, International Journal of Metadata. *Semantics and Ontologies.* 3(1), 37-52.

[28] Kiryakov, A., Ognyanov, D. and Manov, D. 2005. OWLIM - a pragmatic semantic repository for owl. In: *Dean, M., Guo, Y., Jun, W., Kaschek, R., Krishnaswamy, S., Pan, Z., Sheng, Q.Z. (eds.) Proceedings of Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005).* LNCS, 3807, 182-192, Springer.

[29] Roman, D., Bishop, B., Toma, I., Gallizo, G., Fortuna, B. 2009. LarKC Plug-in Annotation Language. In: *Proceedings of The First International Conferences on Advanced Service Computing – Service Computation, 2009.*

[30] Tenschert, A., Assel, M., Cheptsov, A., Gallizo, G., Della Valle, E., Celino, I. 2009. Parallelization and Distribution Techniques for Ontology Matching in Urban Computing Environments. In: *Shvaiko, P., Euzenat, J., Giunchiglia, F., Stuckenschmidt, H., Fridman Noy, N., Rosenthal, A. (eds.) Proceedings of the 4th International Workshop on Ontology Matching (OM-2009).*

[31] Matthias Assel, Alexey Cheptsov, Georgina Gallizo, Katharina Benkert, Axel Tenschert. 2010. Applying High Performance Computing Techniques for Advanced Semantic Reasoning. In: *Proceedings of the eChallenges e-2010 Conference. Paul Cunningham and Miriam Cunningham (Eds).* IIMC International Information Management Corporation. ISBN: 978-1-905824-20-5.

[32] Alexey Cheptsov, Matthias Assel, Georgina Gallizo. 2010. Enabling High-Performance Computing Resources For Efficient Semantic Web Reasoning. In: *Proceedings of the 7th Extended Semantic Web Conference (ESWC).*