Applying High Performance Computing Techniques for Advanced Semantic Reasoning

Matthias ASSEL, Alexey CHEPTSOV, Georgina GALLIZO, Katharina BENKERT, Axel TENSCHERT
High Performance Computing Center Stuttgart (HLRS)
Nobelstr. 19, 70569, Stuttgart, Germany
Tel: +49 711 68562515, Fax: + 49 711 68565832
Email: {assel,cheptsov,gallizo,benkert,tenschert}@hlrs.de

Abstract: Recent advances in the Semantic Web community have enabled processing and exploitation of semantically annotated data by a variety of modern reasoning methods. However, there is still a deep mismatch between the requirements for reasoning on a Web scale and the existing efficient reasoning algorithms over restricted subsets. Fulfilment of those requirements can be greatly achieved by applying special techniques, which enable high-performance computing architectures to be included as a part of a sustainable infrastructure for large-scale semantic reasoning, as set up within the LarKC project that aims at providing large-scale reasoning over billions of structured data in heterogeneous data sets. In this paper, the mainstream strategies for overcoming the limitations of today’s reasoning systems’ scalability and performance by means of applying high-performance computing are discussed.

Keywords: Semantic Web, Reasoning Systems, High-Performance Computing, Distributed Computing, Parallelisation

Introduction

The Semantic Web is the evolving development of the World Wide Web in which information and services have been extended with explicit meanings (semantics) in order to make Web contents understandable and processable by machines. However, such machine-accessible data poses novel challenges to the use of the Web such as semantic search, semantic reasoning, data integration and others.

Currently, most Web content is still not amenable to intelligent processing by machines. Typical uses of the Web today involve people collecting and combining information, or searching through online catalogues and ordering data (e.g. products). Much of this work is done manually, without any machine support: search engines only search for keywords without understanding the meanings denoted by these strings and web-pages cannot be easily personalised to match the users’ preferences.

Another big problem today is that Semantic Web reasoning systems do not scale to the requirements of the rapidly growing amount of data, such as those coming from millions of sensors and (mobile) devices or the terabytes of scientific data produced by automated simulations. Although the amount of semantically structured data available on the Web is continuously growing, there is still a challenge to cope with heterogeneity in knowledge representation and with noisy and inconsistent data. Furthermore, most of the current semantic reasoning approaches are centralised, e.g. Linking Open Data, which limits scalability to the boundaries of the hardware where the application is running. Some work has been done on distributed reasoning and distributed information retrieval but the concepts explored still present some limitations with regards to performance.
This paper presents the approach followed by the ICT FP7 project LarKC (The Large Knowledge Collider) for the design and development of Semantic Web infrastructure for large-scale semantic reasoning. This platform shall address the limitations on current semantic reasoning engines by explicitly involving concepts from the High Performance Computing (HPC) domain like massive job/workflow parallelisation and distribution.

**The LarKC Platform**

The main assumption of LarKC is that the newly developed distributed Semantic Web reasoning infrastructure has to go beyond current reasoning paradigms in order to scale to the size of the current and future Web.

Hence, the LarKC Platform architecture allows an effective combination of techniques coming from different interdisciplinary fields such as distributed and high performance computing, information retrieval, machine learning etc. In this context, a complex problem may be split into smaller pieces (the LarKC plug-ins), which can be combined in an appropriate manner into an execution workflow.

The core of the platform architecture aims to bridge the gap between the users, Semantic Web services and diverse hardware architectures. The platform facilitates combination, configuration, deployment, monitoring and management of semantic Web services, providing functions such as data access and transactions, event handling, service repository, flow control, metadata management, and many more. Moreover, the platform provides a mechanism for intelligent service and application deployment on virtually unlimited resource pool, allowing LarKC users to abstract from the various complex access and job submission mechanisms of the deployment.

![Diagram of the LarKC architecture](image)

**Figure 1: The high-level LarKC architecture**

The abovementioned semantic services are integrated within the LarKC Platform in form of plug-ins, through a well-defined API, and combined effectively making use of semantic descriptions, which model the behaviour and features of each of them. The LarKC plug-ins are classified in the following categories:

- **Identify**: Identify resources that may be useful for the given query/search request;
• **Transform**: Convert from one formalism to another, e.g. between different RDF representations or from natural language to RDF;
• **Select**: Take a subset of all possible data using heuristics, statistics or any other strategy;
• **Reason**: Perform inference to materialise implicit knowledge;
• **Decide**: Control plug-in behaviour to enhance the efficient functioning of the workflow and decide when to terminate.

An intelligent combination of different types of plug-ins in an application workflow enables users to run their own experiments, and should allow scaling well beyond what is currently possible. Thanks to the platform support for distributed and remote execution, interoperability is enabled between distributed heterogeneous systems, using different platforms and technologies, permitting workflows to take advantage of all available resources (hardware and software) to achieve the highest scalability. Finally, through the application of parallelisation strategies on different levels (i.e., parallelisation could be applied within a particular plug-in or on a broader level between certain plug-ins) the performance of compute-intensive tasks shall be improved to satisfy the reasoning requirements of future Web applications.

**Related Works**

The LarKC Platform is primarily intended as a scalable and distributed infrastructure. Hence, some previous work in distributed computing is particularly relevant to the development of plug-ins for reasoning and information retrieval. Previous approaches include and , amongst others. In the same way, distributed computing techniques can be applied to scale data storage. , for example, proposes a distributed RDF storage and retrieval engine. Technologies such as MapReduce , which enables processing of large data sets, and BigTable , a distributed storage system for structured data, are being actively explored for support in the platform, where they can offer organising principles for the replicated execution of data-reduction and reasoning, and storage and retrieval plug-ins, respectively. 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.

LarKC’s goals were shared in part by the KAON project , which built an opensource ontology management infrastructure targeted for business applications. However, 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. Furthermore, the architecture of the LarKC Platform has been motivated in part by previous research on research infrastructures and workflow systems respectively. Projects such as Kepler or Taverna have shown the ease of use of building complex workflows within different scientific communities. Apart from these two popular systems, there are many other solutions that assist users during experiment / workflow creation and execution - for instance Geodise , LEAD , ViroLab etc. Nevertheless, an extension of expressiveness of current programming models, interoperability, scalability and performance, and access to different computing resources are still a subject of research.

**Implementation**

The platform acts not only as development environment for Semantic Web applications, but also serves a complete deployment test-bed for the execution of LarKC plug-ins and applications built on top of those. Deployment options can include combinations of generic remote web servers, flexible desktop and service grid systems as well as cloud environment
resources in general. In LarKC, this flexibility is achieved by means of plug-in manager architecture (see Figure 1) which enables plug-in deployment on a remote resource.

Performance impact on the Semantic Web applications obtained from deployment of their entire workflow or parts of the workflow on remote resources is twofold. First of all, positive performance effect is conditioned by characteristics of the remote resource, which allows the single component to fully exploit the benefits of the target architecture (for example, applying shared or distributed memory parallelisation for the plug-in algorithms when running on parallel and high-performance computing architecture). On the other hand, distributed execution enables sharing of the workflow load over the resource pool, which might be highly beneficial for complex workflows described through loops, branches and conditional operations among the single plug-ins.

Remote execution is a core functionality of the LarKC decentralised architecture. The typical scenario of using the distribution model is to be able to execute LarKC plug-ins on the resource where the Platform cannot be deployed on (such as public web hosts providing access to underlying web services via http protocol, or firewall-protected high-performance resources). The main motivation for applying distribution to deployment of the LarKC components is to:

- make use of all resources available (which may be distributed in local and remote locations);
- use data that cannot be shipped (either because of the size or because of security restrictions), by moving the computational tasks as close as possible towards the data.

As long as the executing components within a workflow (plug-in instances) are independent of one another, they can be distributed so that they run simultaneously and remotely on multiple machines or at multiple network locations. If they are not fully independent, the extent to which this distribution can be achieved is more limited. For example, in Figure 2 remote instantiation and execution of a single plug-in is shown which is the most required type by the majority of the LarKC use cases.

![Diagram](https://via.placeholder.com/150)

**Figure 2: Remote Plug-In Invocation Schema**

In order to be executed remotely, plug-ins might be previously deployed on the remote resource, which should provide access to the hosted components from within the Platform. Among the main host types the following are of especial interest for LarKC components:

- **any remote computer** which is able to host the plug-ins (please find description of the main hosting strategies below) but not the platform;
• a public web server (which provides the services wrapping the LarKC component’s functionality);
• a high-performance computer accessible via a dedicated front-end node (for complex and computation-intensive components);
• a grid/cloud site which is accessible through a resource broker.

With regard to the specifics of high-performance computing (such as limited administrative possibilities, special access protocols), the Grid Access Toolkit (GAT) technology is of especial interest for adopting by LarKC. Java implementation of Grid Application Toolkit (JavaGAT) offers a set of coordinated, generic API for remote accessing resources of several types. The flexibility of GAT is achieved by means of Adaptors - generic mechanism which allows the user to perform all the operations on the remote resource in a unified way while adaptation to specific of the concrete resource is done in the corresponding Adaptor.

For the LarKC components, JavaGAT was applied to these functional operations:

• Execution of a plug-in deployed on the remote resource. This includes setting up a run-time environment, starting specified plug-in for execution and polling execution status of the plug-in. All of those operations are performed from the local host.
• File upload from the local host to the remote resource. This is used for uploading both files containing the data needed as input for the remotely launched plug-in (for example generated by a previously executed component in the workflow) or libraries required by the component (in case if it has not been pre-deployed to the remote host before).
• File download from the remote resource to the local host. This is used for collecting back the data produced during the remote component execution (and eventually submitting this data in the plug-ins output queue).

Along with GAT, several other technologies are of potential interest for adopting by LarKC. Among them are such famous as SAGA or COMPSs. Thanks to the flexible LarKC Platform design, the current remote plug-in manager architecture can be easily extended and enhanced for support of those technologies as well. The flexible LarKC Platform design is foundation for further investigations of those technologies as well.

Efficiency of the component’s distribution, especially in case if parallel high-performance architectures are involved, can be greatly supported by combining the distribution techniques with exploiting component nested parallelism, as described in the next subsection.

**Business Benefits**

The LarKC solution is being validated by three different applications coming from real-life scenarios: urban computing, early clinical development and carcinogenesis research. However, LarKC is independent of application and therefore a wide range of application fields can potentially benefit from the LarKC solution.

As mentioned before, the LarKC architecture combines techniques coming from different interdisciplinary fields. For this reason, also a wide variety of stakeholders is involved in the LarKC value chain. Each of them is benefiting from their individual participation - independently whether they are an academic or an industry partner - in order to reach a common goal, achieving large-scale semantic reasoning overcoming the current limitations. The main roles that can be distinguished in the LarKC value chain are: platform provider, platform developer, plug-in developer, plug-in provider, resource providers (in the sense of computing and storage resources), application developer and application provider.

Regarding the availability of the proposed solution, an open licensing model has been applied, in order to maximize the benefits of all partners involved (and future ones), both
academic and industrial. The LarKC Platform and its source code has been released under Apache License, Version 2.0. Regarding the plug-ins, given the potential high amount of them and the heterogeneous field of application, the plug-in developer is free to choose his/her preferred license, but always ensuring the compatibility with the platform license. For the applications data sets, it is recommended to use a copy-left or “some rights reserved” licensing such as Creative Commons or any similar license which does not limit or impact in any form the platform license. This licensing schema allows the application of different business models between the abovementioned stakeholders (e.g. the platform provider may charge plug-in providers to make them available through the LarKC marketplace, so that they can be re-used as part of different applications by application providers).

The open licensing model, together with the open and modular architecture of LarKC, contributes to ensure the sustainability of the project results. External developers are welcome to join the LarKC team and test their own components within the overall architecture through the opensource repository at sourceforge, which also offers user and developer support, through mailing lists and discussion fora. Furthermore, external plug-in developers are also encouraged to publish and advertise their plug-ins in the LarKC marketplace, a repository of plug-ins which are interoperable with the platform and may be combined in different ways in order to solve concrete problems in different fields.

In what respects to the opportunities of LarKC in the current market of semantic solutions, the SWOT (Strengths, Weaknesses, Opportunities, Threads) analysis presented in, is positioning LarKC among existing products and services. One of the major opportunities identified for the LarKC platform to enter the current semantic market is the improvement of the performance and scalability through the application of HPC and distributed computing techniques, which have not been effectively combined with semantic technologies before.

Conclusions

The current implementation of the LarKC 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 for reasoning over large-scale semantic data sets. The platform also allows the deployment and execution of certain plug-ins on powerful hardware resources like HPC clusters in order to improve the performance of certain compute-intensive tasks. In addition, scalability is being addressed through the distribution of plug-ins among heterogeneous resources, too.

The success and application of these features are being demonstrated, within the LarKC project itself, in three different case studies. The first case study is from the urban infrastructure sector. It aims at real-time aggregation and analysis of information about a city’s population, events, and services 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 require large-scale data integration and analysis of scientific data and literature. First evaluation results from the urban computing use case, in particular on the performance of the different urban computing scenarios (i.e., route planning and path finding), are presented in.

During the final 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 parallelisation techniques for e.g. ontology matching and the support for remote execution on a wider variety of resources. 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.

Acknowledgments

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

References


http://www.apache.org/licenses/LICENSE-2.0.html
http://creativecommons.org
http://larkc.sourceforge.net
http://sourceforge.net/projects/larkc/support