Innovatives Supercomputing in Deutschland
Towards High Performance Semantic Web - Experience of the LarKC Project

The essence of Semantic Web is the idea that the Web can exploit techniques from, e.g., formal Knowledge Representation, to make information available in a machine-processable format, so that a more intelligent user support can be achieved on the Web [1]. Such machine-understandable data formats, for instance the Resource Description Framework (RDF), enable novel uses of the Web such as semantic search, data integration, statistical analysis and others. Recent advantages in Semantic Web have forced Web applications to scale up to the requirements of the rapidly increasing amount of interconnected and distributed data such as observed in the Linked Open Data repository for data located across the Web or the Linked Life Data semantic integration platform for the biomedical domain, but also in e-Science and e-Commerce (e.g., Ontoprise).

The massive and tremendously growing amount of data requires effective exploitation and is hence of a great challenge for the modern IT platforms and infrastructures. Another big challenge for achieving the efficiency and web-scaling of Semantic Web applications is the heterogeneous nature of explored data on the Web, which results in data inconsistencies, incompleteness, but also redundancies due to varying methodologies used during the data generation and collection.

Given the large problem sizes addressed by Semantic Web and considering the complexity of some data exploration algorithms such as Random Indexing described below, it seems natural to explore the benefits of porting Semantic Web applications for running on High Performance Computing architectures.

Large Knowledge Collider

One of the major practical attempts to build a Semantic Web engine capable of processing billions of structured data, i.e., web-scale data, is performed in the EU FP7 project LarKC (www.larkc.eu). LarKC, which stands for the Large Knowledge Collider, builds an experimental platform for massive distributed incomplete reasoning (see Figure 1), which aims at removing the scalability barriers recognized for most of the currently available reasoning engines. This goal is achieved by means of a number of the original techniques adopted by LarKC, e.g., a highly innovative reasoning approach for merging the retrieval process and the reasoning by means of selection, identification, or transformation [2]. On the other hand, LarKC enables numerous novel IT infrastructure solutions to support those optimization techniques, such as High Performance or Grid Computing e-Infrastructures.

The optimal resource provisioning is of especial importance for ensuring the web-scale property of Semantic Web applications. However, since introducing a special e-Infrastructure for Semantic Web, as done in LarKC, processing vast amount of data is not a major bottleneck any more.

Nevertheless, leveraging those resources requires necessary adaptations in the traditional (serial) application codes, i.e., their parallelization. The parallelization becomes thus a major challenge for the next-generation Semantic Web applications executed in a context of e-Infrastructure.

Parallelization Strategies adopted by LarKC

In solving those issues related to the large-scale Semantic Web applications, LarKC allows a reasoning application to be built on top of numerous lightweight Semantic Web computational blocks (plug-ins, see the actual list on LarKC Market Place at http://www.larkc.eu/plug-in-marketplace), used for identification, selection, transformation, and actual reasoning. When combined in a common workflow, such as one shown in Figure 2, these plug-ins can be efficiently utilized for solving problems of the almost virtual dimensionality.

Figure 1: LarKC – a high performance Semantic Web reasoning platform

Figure 2: Parallelization in Semantic Web application workflows

Figure 4: Comparison of time (a) and bandwidth (b) of inter-node communication of different MPI libraries for Java (on the HLRS NEC Nehalem cluster with Ethernet and Infiniband interconnects).
To support this feature, the composition of the plug-ins in a workflow enables parallel execution of the plug-ins on the high performance resources. In terms of LarKC applications, the parallelization suggests the identification of the concurrent regions of the application data as well as instruction flow, with further mapping them to the independent processor units of a parallel system.

Among the most widely utilized and successful in Semantic Web parallelization approaches, such as Multithreading, Map-Reduce, as well as the Message-Passing Interface (MPI), the latter (MPI) is the most promising one in terms of the implementation efforts needed for a serial application as well as in terms of the provided scalability. There have been several initiatives striving to provide HPC support for Java, which is de-facto a default programming language in the LarKC Semantic Web community. One of the most successful MPI implementations for Java has proved to be mpiJava, chosen for adoption in LarKC (Figure 3). The mpiJava framework is currently developed and supported by HLRS.

**MPI Realization of Random Indexing**

Random Indexing is a novel approach for vector space modelling (3). 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 function of the given words’ context vectors (1).

\[ \forall x \in \mathcal{X}, \exists v = [f(x, c_j) \in m] \] (1)

where \( f(x) \) is a co-occurrence function between the word \( x \) from the word set \( \mathcal{X} \) and each of the contexts \( c_j \), \( m \) is a total number of the contexts, \( n \) is a total number of the words in all contexts. However, such popular Random Indexing implementation packages, such as Airhead [4], are increasingly ineffective when complexly addressing large data amounts, e.g. as collected by Linked Life Data. LarKC has examined the domain decomposition-based parallel implementation of Random Indexing, as depicted in Figure 4.

With regard to the aforementioned Airhead library (5), very promising performance characteristics were obtained for both pure MPI and hybrid MPI-JavaThreads implementations (see Figure 5). The document set based on a selection of the Wikipedia articles (1M high density entries, 16 GB disk space) was used for this performance benchmark. Detailed results for different input document sizes as well as cluster configurations are reported in [6].

**Outlook**

Recent advantages in the Semantic Web require the underlying (Java) applications to scale up to the requirements of the rapidly increasing amount of processed data, such as those coming from millions of sensors and mobile devices, or TB of data volumes conducted during scientific experiments using laboratory equipment. Introducing HPC in Semantic Web domain can greatly support this challenge.

Traditionally, the Semantic Web and the High Performance Computing community have been somewhat disjoint. However, as the needs and capabilities of these two communities continue to converge, it turns to be beneficial for both to leverage their respective technologies. Parallel realization of serial codes is a key enabler of high performance architectures and is therefore a great challenge for the majority of Semantic Web applications. LarKC aims at simplifying development of high performance, parallelized applications, and thus bridging the gap between Semantic Web and HPC.

**References**


