Parallelization and Distribution Techniques for Ontology Matching in Urban Computing Environments

Axel Tenschert[1], Matthias Assel[1], Alexey Cheptsov[1], Georgina Gallizo[1]

[1] HLRS – High-Performance Computing Center Stuttgart, University of Stuttgart,
Nobelstraße 19,
70569 Stuttgart, Germany
{tenschert, assel, cheptsov, gallizo}@hlrs.de

Abstract. The usage of parallelization and distribution techniques in the field of ontology matching is of high interest for the semantic web community. This work presents an approach for managing the process of extending complex information structures as used in Urban Computing system by means of ontology matching considering parallelization and distribution techniques. Especially when thinking of matching large-scale ontologies coming from diverse resources, there is the need of considering an approach allowing for distributing the workload accordingly in order to address performance and scalability issues more efficiently. In this paper we present approaches for solving these issues by using parallelization and distribution techniques in order to improve scalability and performance of the ontology matching process.

Keywords: Ontology Matching, Semantic Content, High Performance Computing, Parallelization, Distribution, Urban Computing

1 Introduction

Through the use of information and distribution technologies a huge amount of disparate information are available and is continuously increasing. This advancement has entailed a growing number of resources collecting tons of heterogeneous information. Hence, there is the problem of managing this increasing amount of knowledge. With regard to this challenge distribution and parallelization techniques are encouraging concepts to cope with this situation. Furthermore, ontology matching methods from the field of semantic technologies have attracted significant attention in the recent years [1].

An ontology provides structured semantic vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. However, several ontologies provide different data and conceptual models, for example, sets of terms, classifications, database schemas, or fully axiomatized theories. Through this, the problem of heterogeneity has to be solved as well in order to make use of several ontologies.

A solution for the semantic heterogeneity problem is ontology matching. Through this, correspondences between semantically related entities of ontologies are exposed.

This is an important fact for various tasks, such as ontology merging, query answering, data translation, etc.

The Goal of this work is to improve existing ontology matching methods by using distribution and parallelization techniques. For this reason, ontologies, which are required for a specific use case, are portioned into several parts. Each part of the portioned ontology is matched with the concepts of another ontology in a distributed manner and/or in parallel. Both approaches are presented in this work. Through the distributed execution it is possible to distribute the matching process on several resources. Furthermore, the required resources are provided in a cluster with the aim to support the matching procedure with the required compute resources.

Therefore, this work presents approaches how to solve ontology matching for large- scale datasets by adapting parallelization and distribution techniques.

The reminder of the paper is structured as follows. Section 2 provides an introduction into a challenging field namely Urban Computing that could benefit from more effective ontology matching techniques anyway. Section 3 describes our approach in more detail. Finally, section 4 concludes the work.

2 Motivating Example

A relatively new field introduced as "Urban Computing" recently attracted attention to the research community. Urban computing basically deals with the usage and integration of ICT[1] for/into everyday urban settings and lifestyle [2] in order to face the challenges of modern cities such as intelligent traffic management, (re)development of neighbourhoods and business districts, cost planning etc.

In the following sections, we first of all present its basic concepts and show some few areas of application. Moreover, we investigate different requirements for applying semantic techniques in particular ontology matching approaches to process the huge amount of information coming from billions of sources (mobile devices, sensors, etc.) and being used to solve many of the abovementioned challenges, primarily traffic management.

2.1 Urban Computing

As previously introduced, Urban Computing enables the integration of computing, sensing, and actuation technologies into everyday urban settings and lifestyles.

Urban settings range from own cars, buses to public spaces such as streets and squares including semipublic ones 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 [3].

Some years ago, due the lack of data and high-speed networks, solving Urban Computing problems seemed to be an unrealizable dream. Nowadays, a large amount

[1] Information Communication Technologies

of required information is already been available on the Internet at almost no cost: maps with the 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 of mobile phone users [4], parking availabilities in specific parking areas, and so on.

However, current technologies are still not able to fully solve Urban Computing problems: this requires combining a huge amount of static knowledge about the city (i.e., urbanistic, social and cultural one) with an even larger set of data (originating in real time from heterogeneous and noisy data sources) and reasoning above the resulting time-varying information in order to retrieve reasonable knowledge. Nevertheless, there are already a couple of applications available that use parts of this variety of information in order to provide added-value services to citizens.

CitySense [5] is a mobile application for local nightlife discovery and social navigation, answering the question, "Where is everybody?" CitySense shows the overall activity level of the city, top activity hotspots, and places with unexpectedly high activity, all in real-time. Then it links to Yelp and Google to show what venues are operating at those locations. Using a billion points of GPS and Wi-Fi positioning data from the last few years – plus real-time feeds – CitySense sees San Francisco from above and puts the top live hotspots onto mobile devices.

In London, people can use the so-called Journey Planer [6] for requesting any information held by the Transport for London Company in order to find the optimal route. People either can use the website that utilizes maps, timetables, and route information or using different mobile services, which assist them in planning their travels as well as finding particular places more easily.

Another example is the Helsinki Bus Map [7], which tracks the positions of transport vehicles in Helsinki and displays their movements in Google Maps in real time.

2.2 Requirements on Semantic Techniques

In this section, we summarize requirements of traffic management (claiming that they can be extended to other areas of Urban Computing as well). As argued before, we are particularly interested in the reasoning requirements for ontology-based semantic techniques including ontology matching approaches.

Basically, we can identify three main issues that demand the application of semantic techniques in Urban Computing environments.

Data Heterogeneity

Dealing with heterogeneous data has been appealed for long time in many areas in computer science and engineering, which include database systems, multimedia applications, network systems, and artificial intelligence. Here, we would like to propose a comprehensive notion of heterogeneity processing for semantic technologies. We distinguish the following levels of heterogeneity, namely *Representative Heterogeneity* and *Semantic Heterogeneity* [8], although there could also be a system’s default heterogeneity, which is referred to as the system’s support for various specification defaults of semantic data.

Representational Heterogeneity means that using different specification languages represents semantic data. Systems supporting representational heterogeneity would allow for semantic data, which are specified by multiple semantic languages, rather than using a single metadata or ontology language. However, different representations of semantic data do not necessarily mean that they have different semantics. Urban Computing-related data can come from different and independent data sources, which can be developed with traditional technologies and modeling methods (e.g., relational DBMS) or expressed with semantic descriptions and languages (e.g., RDF/S, OWL, WSML).

Semantic Heterogeneity means that the systems allow for multiple paradigms of reasoners. For instance, many applications of Urban Computing may need different reasoners for temporal reasoning, spatial reasoning, and causal reasoning. However, it does not necessarily mean that we have to develop a single but powerful reasoner, which can cover all of those reasoning tasks. A system, which supports semantic heterogeneity, would find a way to allow multiple single-paradigm-based reasoner to achieve the result of semantic heterogeneity.

Scale of Data

The advent of Pervasive Computing and Web 2.0 technologies led to a constantly increase of data about urban environments, like information coming from diverse sensors (traffic detectors, public transportation, pollution monitors, etc.) as well as from citizens' observations [9] (black points, commercial activities' ratings, events organization, etc.). The result, however, is that the amount of data available to be used and integrated is not manageable by existing technologies and tools and a severe focus on scalability issues must be taken into account. For example, intelligent methods for data sampling or selection should be adopted before employing traditional reasoning techniques.

Noisy, Uncertain and Inconsistent Data

Due to the amount of heterogeneous sources (devices, sensors, cameras, etc.) together with time-dependency of certain information (e.g., event-driven or periodically streams of information), data sometimes becomes biased prior to arriving at the corresponding requestor. Generally, we distinguish between three cases which can affect data and thus bias its meaning.

Noisy data are often completely useless or semantically meaningless. Inconsistent data could be logically self-contradictory, or without any semantic relationship. Finally, the semantics of uncertain data could be partial, incomplete, or they are conceptually arranged into a range with multiple possibilities. Traffic data and especially floating car data [10] are very good examples of such data

Reviewing the above mentioned issues, ontology matching techniques could help to cope with these challenges while addressing these issues nearly simultaneously. First of all with respect to the problem of data heterogeneity and data scale respectively, ontologies usually should assist people in guessing the meaning of the underlying raw data more easily. In case of very large data sets – in size and structure (deep and multiple relationships) – an immediate interpretation is no longer achievable. Here, ontology matching could be applied and provide a promising solution. Instead of looking up very large ontologies every time while searching for relevant information, ontology matching could be used to analyze only subsets of the entire ontology (analysis is parallelizable) in order to look for matches with one's own priority ontology and thus, identifying the most appropriate information faster.

Secondly, ontology matching improves the handling of incomplete or inconsistent data. Assuming that one is searching for a specific subset of information, the entire ontology does not necessarily be complete, only relevant parts have to. However, if the dedicated subsets appear to be incomplete or inconsistent with one's priority ontology, the matching could identify some new subsets, which are of potential interest for the requestor. Nevertheless, this new information could be inadequate or inconsistent as well but the probability that the new information is valuable is essential higher.

Ontology matching techniques are beneficial for receiving semantic data structures from several other ontologies. Therefore, we will take a detailed look at techniques for improving ontology matching.

3 Ontology Matching

3.1 Local Ontology Matching

At present there are several approaches for ontology matching in order to merge ontologies [11]. Furthermore, a combination of several ontology matching techniques has to be considered, too [12]. These well-proofed approaches are beneficial for ontology matching but there is still a challenge to combine different ontology matching techniques together [13]. Among this, an adequate algorithm is required to ensure a useful merging. These approaches require a high amount of compute resources with the aim to meet the requirements of the matching and merging methods. Hence, several issues have to be considered such as the selection of a suitable ontology, scalability and robustness, matching sequence and identification of the ontology repositories.

In order to solve the mentioned issues a beneficial matching algorithm as well as the usage of the required compute resources is crucial. Approaches for dividing selected ontologies with the aim to execute matching processes independently from other parts of the ontology are considered to solve this challenge. However, a local ontology matching is a risk for these approaches because of scalability and performance issues. When thinking of a separation of ontologies the usage of distribution methods as well as parallelization techniques approaches for local ontology matching should be extended.

3.2 Distributed Ontology Matching

Distributing Method

Currently there are lots of ontology matching strategies and merging tools available. When thinking of approaches for ontology matching systems in open distributed systems there are several approaches available such as GLUE [18], Edamok [19], KAON [20] or Matchmaker [21] which focus on different techniques for ontology matching. However, matching ontologies entails the problem of processing the matching in a scalable way with respect to the performance. For this, distribution methods and parallelization techniques are used to increase the performance and ensure scalability by executing several matching processes at the same time on dispersed compute resources, e.g. inside cluster environment.

With regard to existing approaches for ontology matching and distribution of processes for matching ontologies, it is of high interest to consider existing approaches and adapt and improve these approaches for this work. Therefore, we will consider the LarKC project[2] in which new techniques for processing large datasets in the research field of the semantic web are investigated and developed. In addition, distribution methods and parallelization techniques are used to execute matching processes at same time in parallel on distributed compute resources.

One approach for matching ontologies is to select one priority ontology, which is extended with matching concepts of other ontologies from a known set. However, the matching procedure is executed in parallel for many concepts on distributed resources. For this, a cluster environment is applicable. Furthermore, when thinking of matching several parts of the priority ontology in parallel in a cluster environment, the matching processes are partitioned. As presented in figure 1 one priority ontology is divided in several parts which are matched with concepts of ontologies taken from a known set. Afterwards, the parts of the priority ontology are merged together again and the new extended priority ontology is generated.

[2] LarKC (abbr. The Large Knowledge Collider): http://www.larkc.eu/

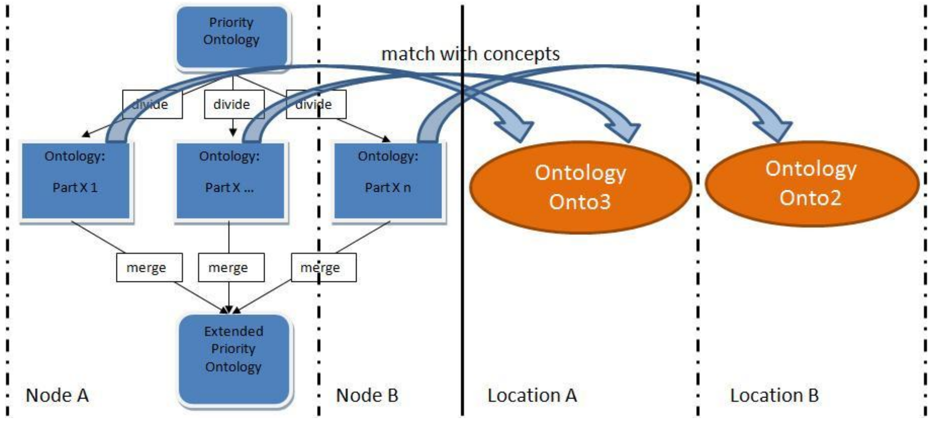


Figure 1: matching with several ontologies on different nodes

Figure 1 presents how to match the concepts of the priority ontology with several ontologies from a known set in parallel on several nodes. To avoid misunderstandings it is important to know that the matching processes are executed on the nodes. However, the ontologies are stored on the hard drives of the cluster, not on the nodes.

When executing a job in the cluster the number of required nodes is selected and every job is executed on the its allocated node. The selected nodes are set individually by the user considering his/her specific requirements in terms of compute resources. The user selects a number of nodes on which the jobs are executed.

Table 1. Parallel ontology matching within a cluster.

Matching Part	Node
Ontology Part 1	Node A
Ontology Part …	Node B
Ontology Part n	Node n

However, the assignment of ontology parts, which are matched to the nodes is just an example. The assignment depends on the size of data and of the jobs respectively, it is possible to execute several jobs on one node.

Distributing Workload and Parallelization Techniques

The ontology matching algorithms often perform computation intensive operations and thus being considerably time consuming. That poses a number of challenges towards their practical applicability for real complexity tasks (such as one presented for the Urban Computing scenario) and efficient utilization of the computing architectures that best fit the requirements of getting maximal performance and scalability of the performed operations. Distributed ontology matching enables the use of diverse computing resources, from users' desktop computers to powerful High Performance machines, as well as resources of heterogeneous Grid/Cloud infrastructures. Parallelization is the main approach for the effective ontology matching, especially when time characteristics are settled to the point and allows for the optimal execution of the ontology matching workflow. Parallelization might be beneficial when one of the following applies for a workflow (Figure 1):

1. Nested source code decomposition (when source code's regions can be identified with clear control/data dependencies between them, for example in form of a graph).
2. Nested data decomposition (when data can be portioned and processed independently).
3. Inherited parallelism (parallel execution of several processes at the same time).

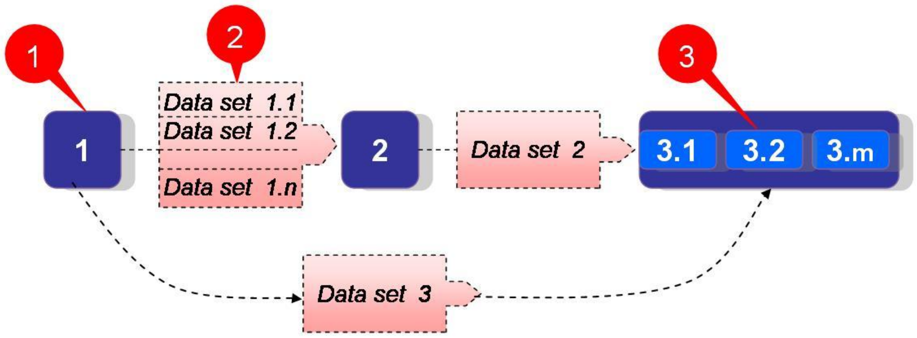


Figure 1: Some examples of the parallelism in an ontology matching workflow

Several techniques can be recognized for the parallel implementation of distributed ontology matching. Below, we give a short overview of the main techniques and show their applicability for the ontology matching tasks.

Based on code/data dependencies, several standard configurations of the workflow can be recognized for the ontology matching process, that allow for parallel reasoning (Figure 2):

- Single Code Multiple Data (SCMD workflow)

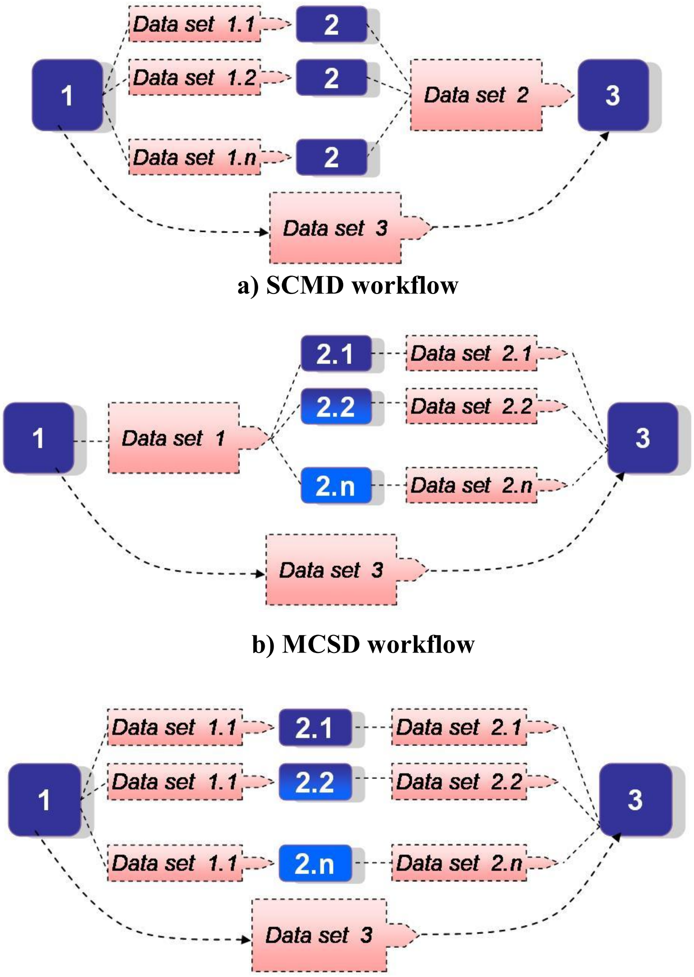
  In this case the data that is being processed in the code region can be constructed of subsets that have no dependencies between them (Figure 3a). The same operation is performed on each of the subsets.

- Multiple Code Single Data (MCSD workflow without conveyer dependencies)

  For this workflow, several different operations are performed on the same dataset. Herewith, no dependencies between processed data sets exist. This is typical for transformation of one data set to another one according to rules which are specific for each subset of the produced data (Figure 3b)

- Multiple Code Multiple Data (MCMD workflow)

  This type of workflow is the combination of both previous workflows (SCMD and MSCD), as Figure 3c shows.



a) SCMD workflow

b) MCSD workflow

c) MCMD workflow

Figure 2: Ontology matching standard workflows

With regard to the parallel execution, there are several strong arguments towards a use of the following approaches for the parallel implementation of ontology matching algorithms.

Distributed Data Parallelism

There are several software products handling distributed data sources. One of them - MapReduce - is a programming model and an associated implementation for processing and generating large data sets [14]. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. MapReduce allows to automatically parallelizing and executing the workflow and is applicable for large clusters of commodity machines (or sites of the grid/cloud environment). MapReduce is beneficial for all types of the workflow, operating big amounts of data.

Parallelism on Distributed Memory Architectures (MPI)

Message Passing Interface (MPI) is a standard for describing the message exchange between processes running in high performance environments including clusters of workstations or supercomputers [15]. MPI defines a set of operations for the processes which are executed in parallel. There are several implementations of the MPI standard for the main programming languages (C, Java), both commercial and open source. In our experiments with the Urban Computing ontology matching tasks we used the Open MPI library [16] that is open source and developed by a consortium of academic, research, and industry partners, HLRS is a partner of. MPI allows full-scale workflows to achieve high performance on large-scale architectures. Both MCSD and MCMD workflows, which require an information exchange between the processes running in parallel, will benefit from the implementation of the distributed memory parallelism by means of MPI.

Parallelism on Shared Memory Architectures (OpenMP)

Shared memory architectures allow in some cases for more efficient data exchange between processes by means of multithreading, a method of parallelization whereby the master thread (a series of instructions executed consecutively) forks a specified number of slave threads and a task is divided among them. The threads then run concurrently, with the runtime environment allocating threads to different processors. OpenMP [17] is an implementation of multithreading, a portable programming interface for shared memory parallel computers, offers significant advantages over both hand-threading and MPI for shared-memory computing architectures. However, the scalability of this approach is limited to the number of nodes connected by the shared memory.

Where possible, we will try to combine all the approaches mentioned above in order to reach the maximal productivity of the ontology matching parallel execution.

4 Conclusions

The presented approach for matching ontologies by the use of distribution methods and parallelization techniques is an effective method to solve the challenge of matching in a scalable, robust and timesaving way. Through this, it is of high interest to topics of the semantic web such as ontology matching. However, the presented work is an overview about ideas and methods which are analyzed to solve the challenge of effective ontology matching in a scalable way. Furthermore, within the LarKC project parallelization and distribution techniques for executing semantic data structures are analyzed and developed. These techniques are considered for this work as well.

Parallelization and Distribution techniques are effective methods for ontology matching when thinking about large-scale ontologies and data sets respectively. Hence, these are valuable techniques for Urban Computing scenarios where ontology matching strategies are required in order to manage the available data structures in a certain period of time. With regard to time dependency when thinking of the use of data structures for Urban Computing scenarios it is of high interest to ensure an adequate performance considering the execution time of ontology matching techniques. For this, the presented approaches are beneficial to manage the challenge of ontology matching in a time saving manner by the use of job distribution on several nodes of a cluster and parallelization of ontology matching workflows.

Acknowledgments. This work has been supported by the LarKC project (http://www.larkc.eu) and has been partly funded by the European Commission's IST activity of the 7th Framework Program under contract number 215535. This work expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work.

References

1. Alasoud**,** A.**,** Haarslev, V., Shiri N.: An empirical comparison of ontology matching techniques. Journal of Information Science 35, 379--397 (2009)
2 Kindberg, T., Chalmers, M., Paulos E.: Introduction: Urban Computing. IEEE Pervasive Computing 6, 18--20 (2007)
3. Reades, J., Calabrese, F., Sevtsuk, A., Ratti, C.: Cellular Census: Explorations in Urban Data Collection. IEEE Pervasive Computing 6, 30--38 (2007)
4- Kane, L., Verma, B., Jain, S.: Vehicle tracking in public transport domain and associated spatio-temporal query processing. Computer Communications 31, 2862--2869 (2008)
5. Sense Networks, http://www.citysense.com/home.php
6. Transport for London, http://journeyplanner.tfl.gov.uk
7. Helsinki City Transport, http://transport.wspgroup.fi/hklkartta/defaultEn.aspx
8. Kim, W., Seo, J.: Classifying Schematic and Data Heterogeneity in Multidatabase Systems. Computer 24, 12--18 (1991).
9. Kostakos, V., Nicolai, T., Yoneki, E., O'Neill, E., Kenn, H., Crowcroft, J.: Understanding and measuring the urban pervasive infrastructure. Personal Ubiquitous Comput. 13, 355--364 (2009)
10.Kerner, B.S., Demir, C., Herrtwich, R.G., Klenov, S.L., Rehborn, H., Aleksic, M., Haug, A.: Traffic state detection with floating car data in road networks. In: 8th International IEEE Conference on Intelligent Transportation Systems, pp. 44--49, IEEE Press (2005)
11.Euzenat, J., Shvaiko, P.: Ontology Matching. Springer. Berlin; Heidelberg (2007)
12.Alasoud, A., Haarslev, V., Shiri, N.: An empirical comparison of ontology matching techniques. Journal of Information Science, 1--19 (2009)

13. Shvaiko, P., Euzenat, J.: Ten Challenges for Ontology Matching. In: Proceedings of ODBASE, LNCS 5332, pp. 1164--1182, Springer (2008)
14. Dean, J., Ghemawat, S.: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco (2004)
15. Dongarra, J.J., Otto, S.W., Snir, M., Walker, D.: A message passing standard for MPP and workstations. Communications of the ACM 39, 84--90 (1996)
16. Open MPI website, http://www.open-mpi.org/
17. van der Pas, R.: An Overview of OpenMP 3.0. Tutorial IWOMP 2009 – TU Dresden, June 3, 2009. Available at https://iwomp.zih.tu-dresden.de/downloads/2.Overview_OpenMP.pdf
18. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to Map between Ontologies on the Semantic Web. In: Proc. of the 11th Int. World Wide Web Conference (WWW 2002), pages 662–673, Honolulu, Hawaii, USA, May (2002)
19. The project EDAMOK, http://edamok.itc.it/
20. KAON website, http://kaon.semanticweb.org/
21. Tangmunarunkit, H., Decker, S., Kesselman, C.: Ontology-based Resource Matching in the Grid - The Grid Meets the SemanticWeb. In: Proc. of the 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID) at WWW 2003, pages 706–721, Budapest, Hungary, May (2003)