UniStore: Querying a DHT-based Universal Storage*

Marcel Karnstedt, Kai-Uwe Sattler, Martin Richtarsky, Jessica Müller
TU Ilmenau
Germany

Manfred Hauswirth
National University of Ireland
Galway

Roman Schmidt, Renault John
EPFL
Switzerland

Abstract

In recent time, the idea of collecting and combining large public data sets and services became more and more popular. The special characteristics of such systems and the requirements of the participants demand for strictly decentralized solutions. However, this comes along with several ambitious challenges a corresponding system has to overcome. In this demonstration paper, we present a light-weight distributed universal storage capable of dealing with those challenges, and providing a powerful and flexible way of building Internet-scale public data management systems. We introduce our approach based on a triple storage on top of a distributed hash table (DHT) overlay system, based on the ideas of a universal relation model and the Resource Description Framework (RDF), and outline solved challenges as well as open issues.

1. Introduction

An increasing number of applications on the Web are based on the idea of collecting and combining large public data sets and services. In such *public data management* scenarios, the information, its structure, and its semantics are controlled by a large number of participants. Despite being distributed or decentralized in respect to data from a conceptual point of view, the supporting infrastructures of these systems still are based on inherently centralized concepts. The downsides at the physical layer of such centralized systems, such as bottlenecks, single-point-of-failures and enormous costs for providing the needed resources, are extended by problems on a more logical level, e.g., the problem of integrating data/services and the need of database processing functionality. Examples of such applications include (specialized) Web search engines, scientific database applications, naming or directory services and "social" applications such as file/picture sharing, encyclopedias, friend-of-a-friend networks or recommender systems.

In this paper, we argue for a decentralization of data management by creating a universal distributed storage for such public data/metadata, which exploits the gigantic storage and processing capacity of the worldwide available Internet nodes in the same way as the network layer exploits the worldwide communication devices for routing messages between nodes. Information sources are highly distributed, data is described according to heterogeneous schemas, no participant has a global view of all information, and data and service quality can only be guaranteed in a best effort way. In this context, the global challenge is to develop a light-weight, generic data management component playing the same role as the TCP/IP stack and a highly scalable infrastructure enforcing a fair distribution of storage and processing load in a highly dynamic world without any central control.

We present a platform based on a triple storage on top of the P-Grid [1] overlay, a universal relation model treating data and schema information uniformly and a light-weight query language inspired by RDF query languages. The software is not intended to run simulations, rather we introduce a platform intended for usage.

2. A Universal Storage based on DHTs

Structured P2P overlays are a good basis for a distributed universal storage as we propose, because problems like scalability, robustness and fair balance of load and work are covered. DHT-based overlays offer logarithmic search complexity in the number of nodes and are based on hashing for data placement, which allows for realizing efficient query processing strategies. Additionally, they offer guarantees and limits needed for defining an appropriate cost model.

Figure 1 shows the architecture of the implemented system. Based on the P-Grid [1] DHT layer, triple storage functionality is provided by a second layer, which is used by P-Grid's *StorageService* to store triple data and to process structured queries.

P-Grid organizes nodes at the leaf level of a virtual binary trie inducing no hierarchy of nodes and supports efficient substring search and range queries through its basic infrastructure, where other DHTs require additional structures.

*The work presented in this paper was supported (in part) by the Líon project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and was (partly) carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project NEPOMUK No FP6-027705.

Additionally, P-Grid includes a mature load-balancing technique able to deal with nearly arbitrary data skews, comes with an update functionality with lose consistency guarantees and enables the merging of two, formerly independent, overlays in a parallel fashion. P-Grid is implemented in Java and available from http://www.p-grid.org/.

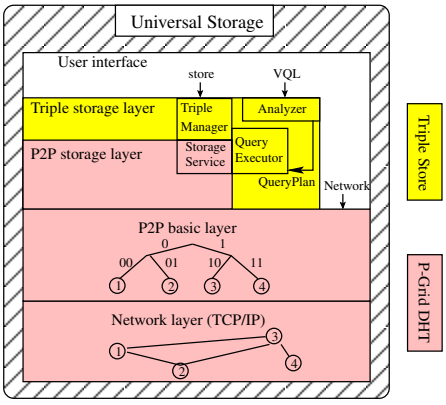

Figure 1: Architecture

In order to face the challenges of data organization, we follow the idea of the universal relation model allowing schema-independent query formulation. However, because exploiting the features of a DHT for fast lookups requires to index all attributes, we store data vertically, similar to the idea of RDF. If we assume relational data, each tuple $(OID, v_1, \ldots, v_n)$ of a given relation schema $R(A_1, \ldots, A_n)$ is stored in the form of n triples $(OID, A_1, v_1), \ldots, (OID, A_n, v_n)$ where OID is a unique key, e.g., a URI, and the attribute names A_i may contain a namespace prefix ns which allows the user to distinguish different relations and avoid conflicts.

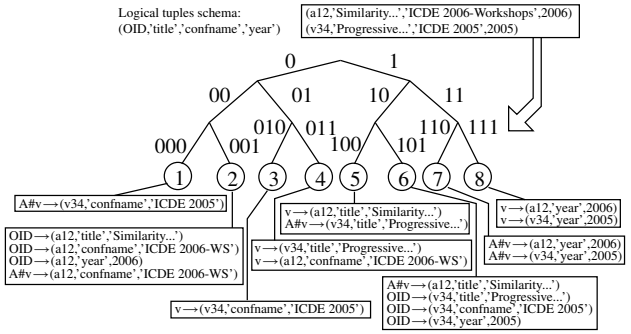

Figure 2: Example tuples in the triple store

Each triple can be indexed multiple times, using different (combinations of) triple entries. Different indexes are used for leveraging the performance of processing different query types. Figure 2 illustrates this for two example tuples, each containing three attributes: 18 resulting triples are distributed in the network of 8 peers (corresponding hash keys are sketched: e.g., `OID→t` means triple `t` was inserted according to `hash(OID)`).

The set of triples can be extended by metadata (in triple form), e.g., schema mappings,which can be queried explicitly by the user – or even automatically by the system to retrieve relevant data without needing the user to interact.

In order to support the formulation and processing of DB-like queries, we propose a structured query language VQL (Vertical Query Language), which is derived from SPARQL [4], and introduce an according logical algebra.

The algebra supports traditional "relational" operators $(\pi, \sigma, \bowtie, \ldots)$ as well as special operators needed to query the distributed triple storage. Operators of both classes can be freely combined and are applicable to schema, instance and metadata level. Furthermore, in order to support large-scaled and heterogeneous data collections, we extend the set of operators by special operators like similarity operators (e.g., similarity join) and ranking operators (e.g., top-N, skyline). Similarity operations are an extremely important and essential part of a universal storage as we propose.

For each logical operator there are several physical implementations available and in development. Key lookups, range queries on key level and prefix search are supported by P-Grid. All of the implemented operators only rely on functionality provided by the overlay system. They differ in the kind of used indexes, applied routing strategy, parallelism, etc. For example, in [3] we introduced a q-gram index (q-gram: a substring of fixed length q) in order to be able to process string similarity efficiently.

For each physical operator, and thus, for each query plan, we can determine worst-case guarantees (almost all are logarithmic) and predict exact costs [2]. We base these calculations on the characteristics of the used overlay system and the actual data distribution. By this, we derive a cost model for choosing concrete query plans, which is repeatedly applied at each peer involved in a query, resulting in an adaptive query processing approach.

For more details of the system, the underlying approach and aspects of physical query processing we refer to [2, 3].

The key benefits of our system are the generic, self-descriptive and flexible schema, a light-weight query language supporting fuzzy and ranking predicates on both, schema and instance level, several physical operators captured by an appropriate cost model and the ability to create a robust, scalable and reliable massively distributed (up to 1000 peers and more) storage in arbitrary environments.

References

[1] K. Aberer. P-Grid: A self-organizing access structure for P2P information systems. In *CoopIS*, 2001.

[2] M. Karnstedt, K.-U. Sattler, M. Hauswirth, and R. Schmidt. Cost-Aware Processing of Similarity Queries in Structured Overlays. In *IEEE P2P2006*, 2006.

[3] M. Karnstedt, K.-U. Sattler, M. Hauswirth, and R. Schmidt. Similarity Queries on Structured Data in Structured Overlays. In *NetDB'06*, 2006.

[4] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Cand. Recommendation 6 Apr 2006. http://www.w3.org/TR/rdf-sparql-query/.