Self-Organizing Schema Mappings in the GridVine Peer Data Management System

[Demonstration]

Philippe Cudré-Mauroux Suchit Agarwal Adriana Budura
Parisa Haghani Karl Aberer

School of Computer and Communication Sciences
EPFL – Switzerland
{firstname.lastname}@epfl.ch

ABSTRACT

GridVine is a Peer Data Management System based on a decentralized access structure. Built following the principle of data independence, it separates a logical layer – where data, schemas and mappings are managed – from a physical layer consisting of a structured Peer-to-Peer network supporting efficient routing of messages and index load-balancing. Our system is totally decentralized, yet it fosters semantic interoperability through pairwise schema mappings and query reformulation. In this demonstration, we present a set of algorithms to automatically organize the network of schema mappings. We concentrate on three key functionalities: (1) the sharing of data, schemas and schema mappings in the network, (2) the dynamic creation and deprecation of mappings to foster global interoperability, and (3) the propagation of queries using the mappings. We illustrate these functionalities using bioinformatic schemas and data in a network running on several hundreds of peers simultaneously.

1. INTRODUCTION

Peer-to-Peer (P2P) systems have become a compelling alternative to standard client-server infrastructures for large-scale settings. They rely on machine-to-machine ad-hoc communications to offer services to a community by eliminating all central components. Research on P2P networks initially focused on building structured overlays, such as Distributed Hash Tables (DHTs), to offer efficient hash table-like functionalities in large-scale settings. Recently, the P2P paradigm inspired several research efforts related to data management. PIER [5] is a scalable system based on a structured overlay network, which brings structured query processing to widely distributed environments. While PIER focuses on scalability, other efforts focus on interoperability by extending traditional integration systems to decentralized environments. Peer Data Management Systems (PDMSs) emerged as an attempt to decentralize the mediator architecture and allow the systems to scale gracefully with the number of heterogeneous sources. PDMSs do not require the definition of a global schema, but consider instead unstructured networks of mappings between pairs of schemas to iteratively disseminate a query from one database to all the other related databases. Piazza [9] and Hyperion [7] are two well-known systems following that paradigm.

Our three-tiered PDMS system, called GridVine [1], tackles both scalability – through the use of a structured DHT to support all distributed operations – and global interoperability – through automated mechanisms to organize the network of mappings in a dynamic way. Both scalability and global interoperability are critical issues that have to be tackled in PDMSs, which were from the start designed to integrate data in very large-scale and decentralized settings. We give in the following an overview of the general architecture of our system, before focusing on its self-organizing principles and giving an outline of our demonstration.

2. SYSTEM ARCHITECTURE

Our approach revisits the principle of data independence [4] by separating a logical from a physical layer. In the present case, we generalize Codd's notion of data independence to networked environments beyond storage systems by separating a logical mediation layer – responsible for structured data storage, data integration, and query resolution – from a decentralized P2P overlay layer – liable for index load-balancing and efficient routing of messages.

Figure 1 gives a conceptual overview of our architecture. The base layer, called *Internet Layer* in the figure, represents the various machines connected to the Internet and sharing structured information through our infrastructure. These machines self-organize into a structured P2P overlay layer. We use P-Grid (see the following section) to arrange the peers into a virtual binary search tree at the overlay layer. Finally, the semantic mediation layer sits on top of this architecture and takes advantage of the overlay layer to efficiently share heterogeneous and structured information

VLDB '07, September 23-28, 2007, Vienna, Austria.

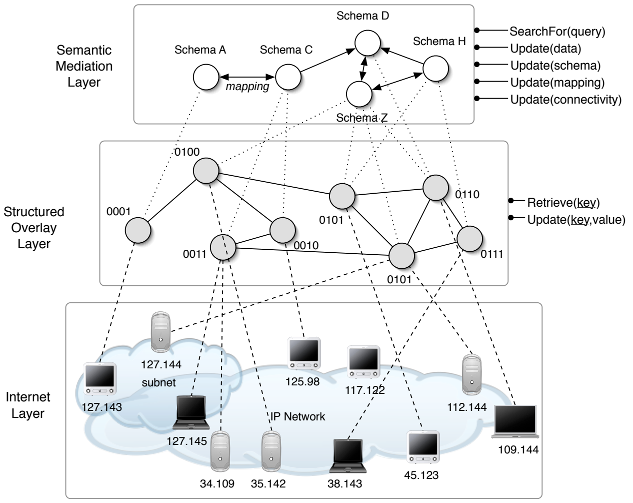

Figure 1: The GridVine PDMS: in our architecture, the semantic mediation layer shares structured data taking advantage of a structured overlay network, which is built on top of a physical network.

across the network. We describe both the overlay and the semantic mediation layer in more detail in the following.

2.1 Organizing Peers at the Overlay Layer

GridVine uses the P-Grid P2P access structure[1] at the intermediate overlay layer. P-Grid is a self-organizing and distributed access structure, which associates logical peers representing the machines in the network with data keys from a binary key space. Each peer is responsible for some part of the overall key space and maintains additional routing information to forward queries to neighboring peers.

Each peer $p \in \mathcal{P}$ is associated with a leaf of the binary tree. Each leaf corresponds to a binary string $\pi \in \Pi$. Thus, each peer p is associated with a path $\pi(p)$. For each level of the tree, each peer has references to some other peers that do not pertain to the peer's subtree at that level, which enables the implementation of prefix routing for efficient search. In addition, peers also maintain references $\sigma(p)$ to peers having the same path, i.e., their replicas that duplicate their content to ensure fault-tolerance and resilience to network churn.

P-Grid supports two basic operations: $Retrieve(\underline{key})$ for searching for a certain key and retrieving the associated value and $Update(\underline{key}, value)$ for inserting, updating or deleting values. Since P-Grid uses a binary tree, $Retrieve(\underline{key})$ is intuitively efficient, i.e., $\mathcal{O}(log(|\Pi|))$, measured in terms of the number of messages required for resolving a search request, for both balanced and unbalanced trees. The *Retrieve* and the *Update* operations provide probabilistic guarantees for data consistency and are efficient even in highly unreliable, dynamic environments.

2.2 Sharing Data at the Mediation Layer

GridVine takes advantage of the efficient, load-balanced and distributed index structure maintained at the overlay layer to share structured information at the semantic mediation layer. At the mediation layer, we support various operations to enable structured data storage and query processing. GridVine stores data as ternary relations called triples. Triples are a natural way to encode RDF information, but can also be used to encode arbitrary relational structures in distributed environments [6].

Triples t always take the following form:

$$t = \{t_{subject}, t_{predicate}, t_{object}\}$$

where $t_{subject}$(the *subject*) is the resource about which the statement is made, $t_{predicate}$ (the *predicate*) represents a specific property in the statement and t_{object} (the *object*) is the value (resource or literal) of the predicate in the statement. Each shared triple at the mediation layer is mapped to routable keys at the overlay layer in order to enable efficient sharing of information and query resolution. To support constraint searches on the triples' *subject*, *predicate* or *object*, we index each triple three times at the overlay layer, generating separate keys based on their *subject*, *predicate* and *object* values. The binary keys are generated using an order-preserving hash function $Hash()$ on the data. Thus, the insertion of a triple t is performed as follows:

$$\begin{aligned}&Update(t) \equiv\ Update(\underline{Hash(t_{subject})}, t),\\&\quad Update(\underline{Hash(t_{predicate})}, t), Update(\underline{Hash(t_{object})}, t).\end{aligned}$$

In that way, a triple insertion at the mediation layer triggers three *Update()* operations at the overlay layer. Update and deletion operations can be implemented using the same mechanism, which explains the generic name (*Update*) we gave to that primitive. Each peer p maintains a local database DB_p to store the triples it is responsible for, i.e., the triples t for which $Hash(t_{subject})$, $Hash(t_{predicate})$, or $Hash(t_{object}) \in \pi(p)$. Thus, the physical schemas of the local databases can all be identical and consist of three attributes $S_{DB} = (subject,\ predicate,\ object)$. The local databases support three standard relational algebra operators: projection π, selection σ and (self) join $\bowtie$.

GridVine supports the sharing of user-defined schemas to structure the data shared at the mediation layer. For the sake of this demonstration, schemas are composed of sets of attributes that are used as *predicates* in the triples. Each schema is associated with a unique key at the overlay layer and inserted into the network at the corresponding location:

$$\begin{aligned}&Update(Schema) \equiv\\&\quad Update(\underline{Hash(Schema\_Name)}, Schema\_Definition).\end{aligned}$$

Whenever necessary, globally unique identifiers are created for local resources and schemas by concatenating the logical address $\pi(p)$ of the peer p posting the item with a hash of the local identifier or schema name.

2.3 Resolving Queries in GridVine

Mapping each triple at the semantic layer onto series of keys at the overlay layer enables us to resolve complex queries in GridVine. A *triple pattern* [8] is an expression of the form $(s,\ p,\ o)$ where s and p are URIs or variables, and o is a URI, a literal or a variable. The simplest queries supported by GridVine retrieve information based on a single triple pattern:

$$SearchFor(x?\ :\ (s,\ p,\ o)),$$

[1] *http://www.p-grid.org/*

where $x?$, the *distinguished* variable the query has to return, also appears in the triple pattern (s, p, o). For instance, the following triple pattern query

$$SearchFor(x? : (x?,\ EMBL\#Organism, \%Aspergillus\%))$$

is constrained on the value of the *Organism* predicate defined in the EMBL bioinformatic schema. It retrieves all nucleotide sequences whose organisms contain the string *Aspergillus*. In GridVine, triple pattern queries can be resolved by taking advantage of the $Retrieve(\underline{key})$ primitive at the overlay layer. A peer issuing a triple pattern query q first has to determine the address space $\underline{key}$ where it can find the answers. This can be determined by taking a hash of one of the constant terms *const* in the triple pattern:

$$\underline{key} = \underline{Hash(const)}.$$

When two constant terms appear in the triple pattern, the most specific one should be used. In our example, we choose the predicate: $\underline{key} = \underline{Hash(EMBL\#Organism)}$. Once the address space is discovered, the peer forwards the query to the peer(s) responsible for that space using a $Retrieve(\underline{key}, q)$ operation. As all triples are indexed on their *subject*, *predicate* and *object* in GridVine, the query can directly be answered by the peer(s) responsible for the corresponding address space. Thus, resolving a triple pattern query boils down to an overlay look-up generating $\mathcal{O}(log(|\Pi|))$ messages. At its final destination(s) $\underline{key}$, the query is resolved with a local relational query on the local database DB_{dest}. Defining $pos(term)$ as the position of a term (variable or constant) in a triple pattern, i.e., $pos(term)$ either takes *subject*, *predicate* or *object* as value, the set of results *Results* is obtained as follows:

$$Results = \pi_{pos(x)}\ \sigma_{pos(const)=const}\ (DB_{dest}).$$

In our example, the query is forwarded to the peer(s) responsible for $Hash(EMBL\#Organism)$, which then retrieves the results by issuing a query $\pi_{subject}\ \sigma_{predicate=EMBL\#Organism \wedge object=\%Aspergillus\%}$ on its local database. Once retrieved by the destination peer(s), the results are sent back to the original issuer of the query. Conjunctive queries can be resolved in a similar manner, by iteratively resolving each triple pattern contained in the query and aggregating the sets of results retrieved.

The access structure used at the overlay layer allows to resolve queries in an efficient manner even in large-scale environments. A recent deployment of GridVine on 340 machines scattered around the world sharing 17000 triples showed that 40% of the 23000 triple pattern queries we submitted were answered within one second only, and 75% within five seconds.

3. SELF-ORGANIZING MAPPINGS

GridVine's mediation layer allows the peers to share structured data in a scalable manner. Sharing information syntactically aligned as triples does not however ensure global interoperability. On the contrary, GridVine being a totally decentralized system, any peer in the network is free to come up with new schemas to structure its own data.

To integrate all semantically related but syntactically heterogeneous information shared by the peers, GridVine supports the definition of pairwise schema mappings. Mappings allow the reformulation of a query posed against a given schema into a new query posed against a semantically similar schema. By iterating this process over several mappings, a query can traverse a sequence of schemas at the mediation layer and retrieve all relevant results, irrespective of their schemas. Given the provision of a sufficient number of mappings, GridVine fosters in that way global semantic interoperability in a totally decentralized fashion.

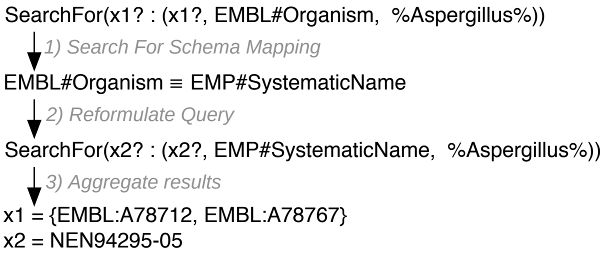
SearchFor(x1? : (x1?, EMBL#Organism, %Aspergillus%))
1) Search For Schema Mapping
EMBL#Organism ≡ EMP#SystematicName
2) Reformulate Query
SearchFor(x2? : (x2?, EMP#SystematicName, %Aspergillus%))
3) Aggregate results
x1 = {EMBL:A78712, EMBL:A78767}
x2 = NEN94295-05

Figure 2: A simple example of query reformulation using a schema mapping.

GridVine allows for the definition of both equivalence and inclusion (subsumption) GAV mappings. For the sake of this demonstration, mappings relate semantically similar *predicates* defined in different schemas. Queries are then reformulated by replacing the predicates with the definition of their equivalent or subsumed predicates (view unfolding). Schema mappings are inserted at the key space corresponding to the source schema at the overlay layer – or at the key spaces corresponding to both schemas if the mapping is bidirectional:

$$\begin{aligned} &Update(Schema\_Mapping) \equiv \\ &\qquad Update(\underline{Source\_Schema\_Key}, Schema\_Mapping). \end{aligned}$$

Figure 2 shows a simple example of query reformulation.

GridVine offers several functionalities to organize the network of mappings at the mediation layer in an automated way. The system ensures that the network of schemas and mappings at the mediation layer is connected in order to enforce global interoperability. It creates additional mappings whenever necessary, tries to assess the quality of the mappings and discards the mappings that are detected as being erroneous.

3.1 Connectivity at the Mediation Layer

GridVine maintains information about the graph of schemas and mappings. Upon inserting a new schema, GridVine asks for the manual definition of schema mappings between the new schema and some already inserted schemas. The system then periodically ensures that the network of schemas and mappings forms a *strongly* connected graph, such that a query posed locally using any schema can be disseminated globally to all related schemas using the network of mappings.

Repeatedly crawling a decentralized and potentially large graph of schemas connected by mappings would be costly in our setting. In GridVine, the peers determine the degree of connectivity of the mediation layer from the degree distribution of its schemas. Each peer storing a schema definition is responsible for updating the number of incoming and outgoing mappings attached to its schema:

$$Update(Domain\_Connectivity) \equiv$$
$$Update(\underline{Hash(Domain)}, \{Schema, InDegree, OutDegree\})$$

where $Domain$ is the name of the application domain related to the mediation layer (e.g., *protein sequences*, see below Section 4). The peer p responsible for $Hash(Domain)$ at the overlay layer can then locally derive the degree distribution of the graph of schemas by aggregating these numbers. It evaluates the connectivity of the mediation layer by computing a connectivity indicator ci_{domain}:

$$ci_{domain} = \sum_{j,k}(jk - k)\; p_{jk}$$

where p_{jk} stands for the probability of a schema to have in-degree j and out-degree k. $ci_{domain} \geq 0$ indicates the emergence of a giant connected component in the graph of schemas and mappings [2]. Thus, the mediation layer is not strongly connected as long as $ci_{domain} < 0$.

3.2 Creation & Deprecation of Mappings

Peers responsible for a schema periodically inquire about the connectivity of the mediation layer by issuing a query to the corresponding key space. $ci < 0$ indicates that some of the schemas shared at the mediation layer cannot always be accessed by following series of mappings. In that case, more mappings are needed to ensure global interoperability. This triggers the automatic creation of additional schema mappings to reinforce the existing network. The exact method used to choose the pair of schemas and to create the mapping depends on the application domain (see below).

The quality of the mappings created in this way is periodically assessed as the networks of peers, schemas, and mappings evolve. GridVine uses a Bayesian analysis comparing transitive closures of mappings to assess the quality of the mappings [3]. The mappings manually created by the users are always considered as correct in this analysis, while probabilistic correctness values are inferred for mappings that were created automatically. A mapping detected as incorrect is marked as deprecated in the system, and is from then on ignored, both for the reformulation of the queries and for the connectivity analysis. The deprecation of mappings fosters the creation of a new topology of mappings, which will ensure the global interoperability of the system eventually.

4. DEMONSTRATION

We demonstrate the applicability of our ideas using real bioinformatic data shared in a network of several hundreds of peers. We export structured data from a public repository of the European Bioinformatics Institute[2]. We consider 50 distinct schemas, all related to protein and nucleotide sequences. We insert data, schemas and a set of manually created mappings in a network of several hundreds of peers.

As more and more schemas and mappings get inserted, we monitor the connectivity at the mediation layer and the automatic creation of mappings to integrate the heterogeneous data sources. We take advantage of shared references to the same protein sequence to select pairs of candidate schemas, and create the automatic mappings using a combination of lexicographical measures and set distance measures between the predicates defined in both schemas. Removing some of the existing mappings fosters the creation of additional mappings, some of which get deprecated by the Bayesian analysis and are gradually replaced by other mapping paths.

At any point of time, we can issue a query locally and observe its reformulation. In reformulating queries, we support two approaches: iterative, where a peer iteratively looks for paths of mappings and reformulates the query by itself, and recursive, where the successive reformulations are delegated to intermediate peers. Finally, we monitor the list of results received for each query. In a sparse network of mappings, few results get returned initially (low recall), while more and more results are retrieved as mappings get created automatically to ensure the global interoperability of the system.

5. ACKNOWLEDGMENTS

The work presented in this paper was supported by the Swiss National Competence Center in Research on Mobile Information and Communication Systems (NCCR MICS, grant number 5005-67322) and by the EPFL Center for Global Computing as part of the European project NEPOMUK No FP6-027705.

6. REFERENCES

[1] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *International Semantic Web Conference (ISWC)*, 2004.

[2] P. Cudré-Mauroux and K. Aberer. A Necessary Condition For Semantic Interoperability in the Large. In *Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE)*, 2004.

[3] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic Message Passing in Peer Data Management Systems. In *International Conference on Data Engineering (ICDE)*, 2006.

[4] J. M. Hellerstein. Toward Network Data Independence. *ACM SIGMOD Record*, 32(3), 2003.

[5] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The Architecture of PIER: an Internet-Scale Query Processor. In *In Conference on Innovative Data Systems Research (CIDR)*, 2005.

[6] M. Karnstedt, K.-U. Sattler, M. Richtarsky, J. Mueller, M. Hauswirth, R. Schmidt, and R. John. UniStore: Querying a DHT-based Universal Storage. In *International Conference on Data Engineering (ICDE)*, 2007.

[7] P. Rodríguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R.J. Miller, and J. Mylopoulos. Data Sharing in the Hyperion Peer Database System. In *International Conference on Very Large Databases (VLDB)*, 2005.

[8] A. Seaborne. RDQL - A Query Language for RDF. W3C Member Submission, 2004. `http://www.w3.org/Submission/RDQL/`.

[9] I. Tatarinov, Z. Ives, J. Madhavan amd A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyaska, G. Miklau, and P. Mork. The Piazza Peer Data Management Project. *ACM SIGMOD Record*, 32(3), 2003.

[2] the data is publicly available at *http://srs.ebi.ac.uk/*