GridVine: An Infrastructure for Peer Information Management

GridVine is a semantic overlay infrastructure based on a peer-to-peer (P2P) access structure. Built following the principle of data independence, it separates a logical layer — in which data, schemas, and schema mappings are managed — from a physical layer consisting of a structured P2P network supporting decentralized indexing, key load-balancing, and efficient routing. The system is decentralized, yet fosters semantic interoperability through pair-wise schema mappings and query reformulation. GridVine's heterogeneous but semantically related information sources can be queried transparently using iterative query reformulation. The authors discuss a reference implementation of the system and several mechanisms for resolving queries collaboratively.

Philippe Cudré-Mauroux, Suchit Agarwal, and Karl Aberer
Ecole Polytechnique Fédérale de Lausanne

Semantic Web technologies are rapidly gaining popularity as a way to organize large amounts of digital assets. Today, mainstream information-management applications ranging from database systems (such as Oracle's Semantic Technology Center; www.oracle.com/database/Enterprise_Edition.html) to creative asset managers (Extensis' Portfolio; www.extensis.com/en/products/asset_management/) let users create both data and schemas to customize how they organize their information. Contrary to the initial vision of the Semantic Web, however, these solutions typically operate in confined environments, such as intranets or predefined communities, and remain impractical for sharing data. Two issues prevent data sharing: lack of scalability resulting from the infrastructure's inability to evolve with the storage, network, or processing load; and semantic heterogeneity, which hinders the querying of data originating from different communities. Several recent research efforts, such as RDF Peers[1] or the Semantic Web and Peer-to-peer (SWAP) project[2] recently proposed efficient architectures

1089-7801/07/$25.00 © 2007 IEEE

Peer Data Management

Data integration has a long tradition of pursuing the design of systems and methods that allow transparent access to disparate and heterogeneous systems through a single interface. *Federated databases* were developed toward that goal to allow the retrieval of data from multiple noncontiguous databases with a single query, even when the constituent databases are heterogeneous. Thus, federated databases provide a solution for integrating data coming from heterogeneous databases interconnected via a computer network. They come in different flavors (see Amit P. Sheth and James Larson[1] for a taxonomy) but often revolve around a central *mediator*[2] component that stores a global schema and is responsible for reformulating queries in terms of all the other schemas the individual databases use.

With the explosion and decentralization of information production, however, it rapidly became clear that this centralized approach — requiring the definition of a central, global schema — could not be enforced on a large scale. The way GridVine disseminates the queries from one schema to another is typical of a new generation of data integration systems called *peer data management systems* (PDMSs). PDMSs emerged as an attempt to decentralize the mediator architecture and let the systems scale gracefully with the number of heterogeneous sources. They don't require the definition of a global schema because they consider loosely structured networks of mappings between pairs of schemas to iteratively disseminate a query from one database to all related databases.

Research on PDMSs is developing in several compelling directions. The complexity of iteratively reformulating queries to reach distant and heterogeneous databases in a PDMS is studied in the context of the Piazza[3] project. Hyperion[4] is a system inspired by the Local Relational Model[5] mapping data at both the instance and schema levels to enable global search capabilities in decentralized environments. SomeWhere[6] is a PDMS offering reasoning services through a distributed consequence finding algorithm. SQPeer[7] proposes a publish–subscribe mechanism and several compile and runtime optimization techniques to execute query plans in decentralized Semantic Web environments.

Our own efforts in GridVine focus on scalability and efficiency through the use of a structured overlay layer and on query diffusion based on probabilistic analyses of the mappings to determine the correctness of the semantic routes in the network.[8]

References

1. A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, vol. 22, no. 3, 1990, pp. 183–236.
2. G. Wiederhold, "Mediators in the Architecture of Future Information Systems," *Computer*, vol. 25, no. 3, 1992, pp. 38–49.
3. A.Y. Halevy et al., "Piazza: Data Management Infrastructure for Semantic Web Applications," *Proc. Int'l World Wide Web Conf.* (WWW 03), 2003, ACM Press, pp. 556–567.
4. M. Arenas et al., "The Hyperion Project: From Data Integration to Data Coordination," *SIGMOD Record*, special issue on peer-to-peer data management, vol. 32, no. 3, 2003 pp. 53–58.
5. P. Bernstein et al., "Data Management for Peer-to-Peer Computing: A Vision," *Proc. Int'l Workshop on the Web and Databases* (WebDB 02), 2002, pp. 89–94.
6. P. Adjiman et al., "Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web," *J. Artificial Intelligence Research*, vol. 25, 2006, pp. 269–314.
7. G. Kokkinidis and V. Christophides, "Semantic Query Routing and Processing in P2P Database Systems: The ICS-FORTH SQPeer Middleware," *Proc. Int'l Conf. Current Trends in Database Technology*, 2004, Springer-Verlag, pp. 486–495.
8. P. Cudré-Mauroux, K. Aberer, and A. Feher, "Probabilistic Message Passing in Peer Data Management Systems," *Proc. Int'l Conf. Data Engineering* (ICDE 06), 2006, IEEE CS Press, pp. 41–50.

for handling structured information in a scalable manner. Other projects have focused on semantic heterogeneity and proposed decentralized information-integration architectures (see the "Peer Data Management" sidebar).

In this article, we describe GridVine, the first system that simultaneously addresses both scalability and semantic heterogeneity. Built following the peer-to-peer (P2P) paradigm, it eliminates central components and concentrates instead on bottom-up and decentralized processes. Rather than requesting services from centralized servers, participating peers collaboratively contribute resources to support higher-level semantic applications. This ensures graceful scalability, as new clients entering the system can in turn provide resources and act as servers. The self-organizing and decentralized P2P access structure supports several higher-level semantic services, such as distributed search, persistent storage, and semantic integration. Such services recursively use the underlying P2P access structure to locate relevant information and dynamically distribute the machines' load.

GridVine: A Three-Tier Semantic Overlay Network

A key aspect of our approach is to apply the principle of data independence by separating a logical layer from a physical one. This principle is well-known from its use in databases and has largely contributed to the success of modern database systems by opening the door to optimizing logical higher-level primitives at the physical layer. In this

case, we generalize the notion of data independence to networked environments beyond storage systems[3] by separating a logical semantic layer — responsible for structured data storage, integration, and query resolution — from a self-organizing P2P infrastructure that's liable for indexing, load-balancing, and efficient routing.

Figure 1 gives a conceptual overview of our architecture. The base layer, called the *Internet layer* in the figure, represents the various machines connected to the Internet that share structured information throughout our infrastructure. These machines self-organize into a structured P2P overlay layer for efficient message routing and index load-balancing. We use P-Grid (which we describe in more detail in the next 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 exploits the overlay layer to efficiently share and integrate structured information across the network.

Structured information (data, schemas, and mappings) is managed and stored through a database at the semantic mediation layer, but indexed and load-balanced by the overlay layer. Distinguishing the semantic layer from the overlay layer lets us offer higher-level primitives at the uppermost layer, while benefiting from an efficient, decentralized, and load-balanced access structure that the P2P overlay maintains. Note that the three layers are uncorrelated: the organization of the machines at the Internet layer is independent of the organization of the peers at the overlay layer, which is itself dissociated from the information's structure at the semantic layer. We describe both the overlay and the semantic mediation layer in more detail in the following sections.

The Overlay Layer

GridVine takes advantage of the P-Grid P2P access structure (www.p-grid.org) at the intermediate overlay layer. P-Grid is a self-organizing, distributed access structure, which associates logical peers representing the machines in the network with keys from a key space representing the underlying data structure. Each peer is responsible for some part of the overall key space and maintains additional routing information to forward queries to neighboring peers. As the number of machines taking part in the network and the amount of shared information evolve, peers opportunistically organize their routing tables according to a dynamic and distributed binary search tree.

Each peer $p \in P$ is associated with a leaf of the binary tree, and each leaf corresponds to a binary string $\pi \in \Pi$. Thus, each peer p is associated with a path $\pi(p)$. For each tree level, each peer stores references to other peers that don't pertain to the peer's subtree at that level, thus enabling the implementation of prefix routing for efficient search. Keys index all shared data items in the system. The key $key(d)$ of a data item d is generated using an order-preserving hash function $Hash()$. Each peer is responsible for storing the keys that fall under its current key space $key \in \pi(p)$. The partition of the key space is load-balanced[4] in such a way that all peers are responsible for the same amount of data, irrespective of the keys' actual distribution. In addition, peers also maintain references $\pi(p)$ to peers with the same path; that is, they're replicas that duplicate their content to ensure persistent storage and resilience to network churn.

P-Grid supports two basic operations: *Retrieve*(*key*) for searching for a certain key and retrieving the associated data value, and *Update*(*key*, *value*) for inserting, updating, or deleting keys. Depending on the application, different values can be associated with the keys. In the context of GridVine, key values either point to unstructured content (such as images) that the overlay layer manages, or to structured data managed by a database at the semantic layer. Because P-Grid uses a binary tree, *Retrieve*(*key*) is intuitively efficient — for example, $O(\log(|\Pi|))$ — measured in terms of how many messages are required to resolve a search request in a balanced tree. For skewed distributions, other work has shown[5] that due to P-Grid's probabilistic nature, the expected search cost remains logarithmic, independently of how the P-Grid is structured. As a result, we can view P-Grid as a persistent and load-balanced index layer that supports efficient key look-ups in a totally decentralized manner.

Semantic Mediation Layer

GridVine takes advantage of the efficient and distributed index structure maintained at the overlay layer to globally manage semantic information at the semantic mediation layer. We support various operations to maintain the semantic layer, including instance, schema, and schema-mapping insertion. Capitalizing on the popularity of Semantic Web standards, we provide those mechanisms

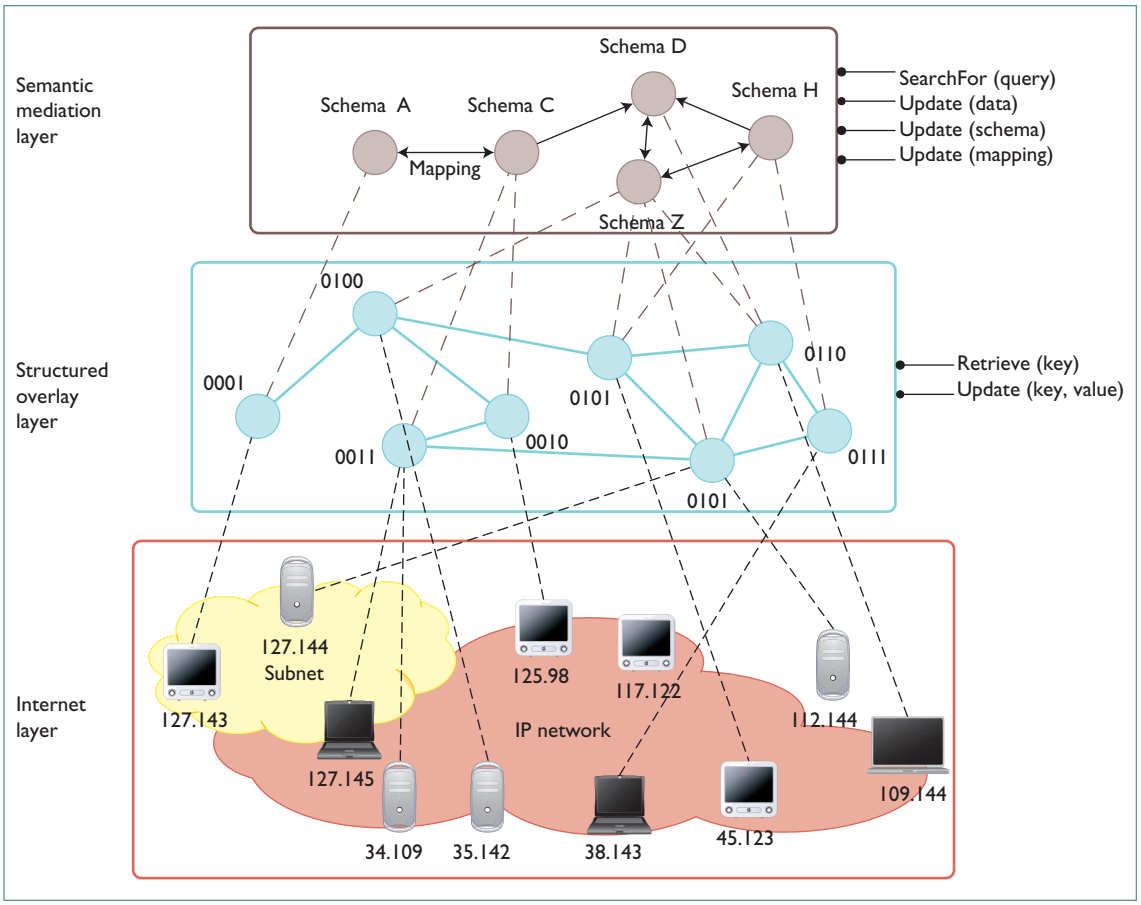

Figure 1. The GridVine semantic overlay network. In our architecture, the machines self-organize into a structured overlay network of peers (blue dots). This P2P network is then used to index semantic information such as schemas (brown dots) or mappings in a totally decentralized way.

within the standard syntactic framework of RDF and the Web Ontology Language (OWL). This requires mapping semantic data and operations to the two operations provided at the overlay layer, and hence, mapping semantic information to routable keys.

RDF stores information as *triples t* representing various statements; triples always take the following form:

$$t_i = t_{subject}, t_{predicate}, t_{object}$$

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. All resources in our system are identifiable through URIs, which the system creates on-the-fly upon insertion when missing. A structured overlay network lets developers implement application-specific addressing spaces. In our case, we introduce the specific URI schemes *pgrid*, for elements that the overlay layer manages (index keys and unstructured content), and *pgrids*, for structured elements that the semantic layer manages (instances, schemas, and schema mappings). This doesn't exclude the use of other URI schemes in conjunction with P-Grid's specific ones.

We map each triple at the semantic layer to routable keys at the overlay layer to enable efficient and load-balanced information indexing. The index's granularity, as well as the exact mechanism used for mapping information at the semantic layer to keys at the overlay layer, are of utmost importance because they directly influence the system's query-processing capabilities. We want to support searches on individual statements and thus must index each triple separately. Most RDF query languages are based on constraint searches on the triples' *subject*, *predicate*, or *object*; as such, we have to reference each individual triple three times,

generating separate keys based on their *subject*, *predicate*, and *object* values. Thus, the system inserts each triple t as follows:

$$Update(t) \equiv Update(Hash(t_{subject}),\ t),\\ Update(Hash(t_{predicate}),\ t),\ Update(Hash(t_{object}),\ t).$$

In this way, each triple is associated with three keys at the overlay layer. The system processes update and delete operations using the same mechanism, which explains the generic name (*Update*) we give to this primitive. Each peer p maintains a local database DB_p at the semantic layer to store the triples whose keys fall under its key space. Because RDFS statements can be written as ternary relations, local databases' physical schemas 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.

GridVine also supports the sharing of *schemas* that define classes of resources and their related properties. Each schema is associated with a unique key and indexed in an atomic operation:

$$Update(RDF\_Schema) \equiv Update(\pi(p):\\ Hash(Schema\_Name),\ Schema\_Definition),$$

where the logical address $\pi(p)$ of the peer p posting the schema is concatenated to a hash of the schema name to create a unique key for the schema whenever necessary. As class and property definitions can also be written as triples in RDFS, we use the same database to store both triples and schemas.

Integrating Data at the Semantic Mediation Layer

GridVine's semantic layer lets peers efficiently share knowledge in a global manner. They can query for any information at the semantic layer by issuing series of *Retrieve*(*key*) operations on the corresponding keys at the overlay layer (we'll explore this more in the next section). However, sharing information that's syntactically aligned as RDF triples in the network doesn't ensure global interoperability. On the contrary, because GridVine is totally decentralized, any peer in the network is free to come up with new schemas to structure its own information.

To integrate all semantically related but syntactically heterogeneous information that peers share, GridVine supports the definition of pairwise schema mappings. A mapping allows 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 global semantic interoperability in a totally decentralized fashion.

We encode schema mappings using simple OWL statements relating semantically similar classes and properties from two different schemas using *owl:equivalentClass* and *owl:equivalentProperty* properties. Schema mappings are indexed 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:

$$Update(Schema\_Mapping) \equiv\\ Update(Source\_Schema\_key,\ Schema\_Mapping).$$

Figure 2 shows a simplified example of query reformulation in GridVine; a peer issues a query to retrieve a book annotated with a given schema (*EPFL* schema), reformulates the query thanks to a schema mapping, finds a relevant resource annotated with the second schema, and, finally, retrieves the relevant resource by querying the structured overlay layer.

Resolving Queries in GridVine

Indexing each triple based on three keys at the overlay layer lets us resolve complex higher-level queries at the semantic layer. A *triple pattern*[6] 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 semantic queries that GridVine supports retrieve information based on a single triple pattern:

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

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:

$$SearchFor(x_2? : (x_1?,\ pgrids : \underline{//0100:EPFL\#Title}),\\ x_2?))$$

retrieves the title of all the images annotated with

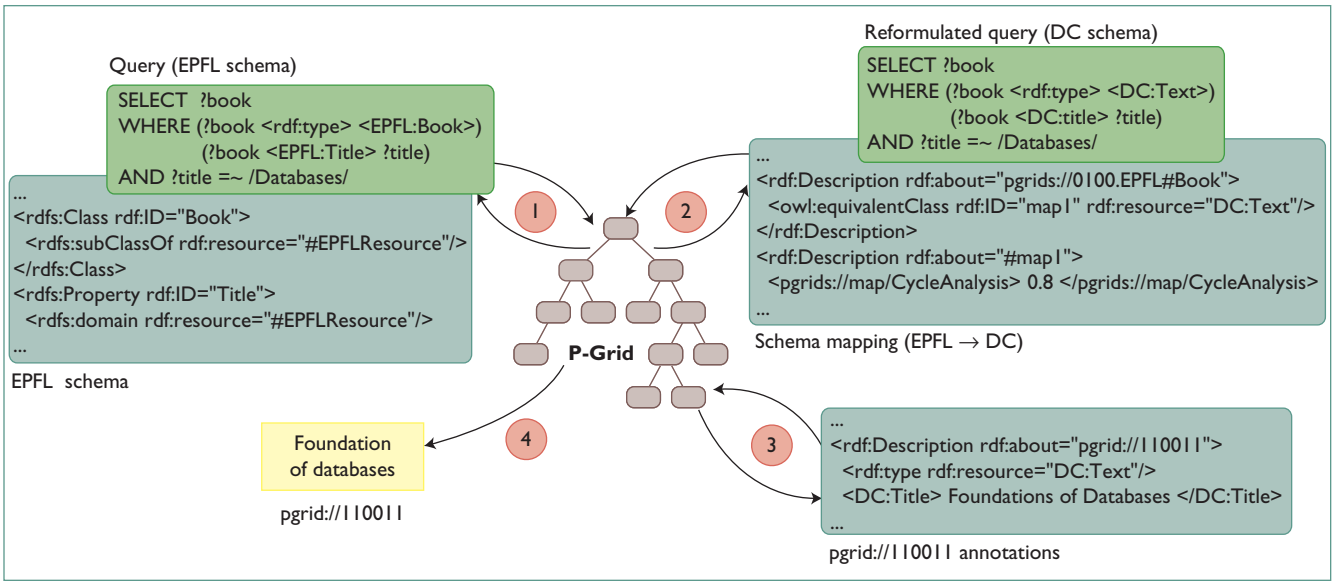

Figure 2. A simple example of query reformulation using a schema mapping. In the figure, a peer issues a query to retrieve a book annotated with a given schema (EPFL schema) (1), reformulates the query using a schema mapping (2), finds a relevant resource annotated with the second schema (3), and finally retrieves the relevant resource by querying the structured overlay layer (4).

the *pgrids*://0100:EPFL schema. We call such a query *atomic* because it contains only one triple pattern.

In GridVine, atomic queries are resolved by first locating relevant peers, thanks to the index provided at the overlay layer, and then by processing structured data those peers handle at the semantic layer. A peer issuing an atomic query q first has to determine the key space *key* where it can find the answers. The peer can determine this key space by taking a hash of one of the constant terms *const* in the triple pattern:

$$key = Hash(const).$$

In our example, $key = Hash(pgrids://0100:EPFL\#Title)$. When two constant terms appear in the triple pattern, both can retrieve the results. Once the key space is discovered, the peer simply forwards the query to the peers responsible for that space using $Retrieve(key, q)$. As all triples are indexed on their *subject*, *predicate*, and *object* in GridVine, the query can be directly answered by the peers responsible for this key space, which stores the corresponding triples in its database. Thus, atomic query resolution boils down to a standard P-Grid look-up generating $O(\log(|\Pi|))$ messages. Once arrived at its final destination *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, *pos(term)* either takes *subject*, *predicate*, or *object* as value, and produces a set of results *Results* as follows:

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

In our example, the query is forwarded to the peers responsible for $Hash(pgrids://0100:EPFL\#Title)$, which can retrieve the results by issuing a query $=\pi_{object}\sigma_{predicate=pgrids://0100:EPFL\#Title}$ on its local database. Once the destination peer retrieves them, the results are sent to the query's original issuer. Conjunctive and disjunctive queries can be resolved in a similar manner,[7] by iteratively resolving each triple pattern contained in the query and aggregating the sets of results through local join operations.[8] GridVine also supports queries on value ranges, as our overlay network can natively process them.

Performance Evaluation

We carried out several large-scale experiments in various settings to validate our infrastructure's design. We details the experiments in the next section. Figure 3a shows the distribution of query processing time in several large-scale networks of 50 to 340 peers scattered around the world. Each peer ran on a distinct machine on the PlanetLab network (www.planet-lab.org). We

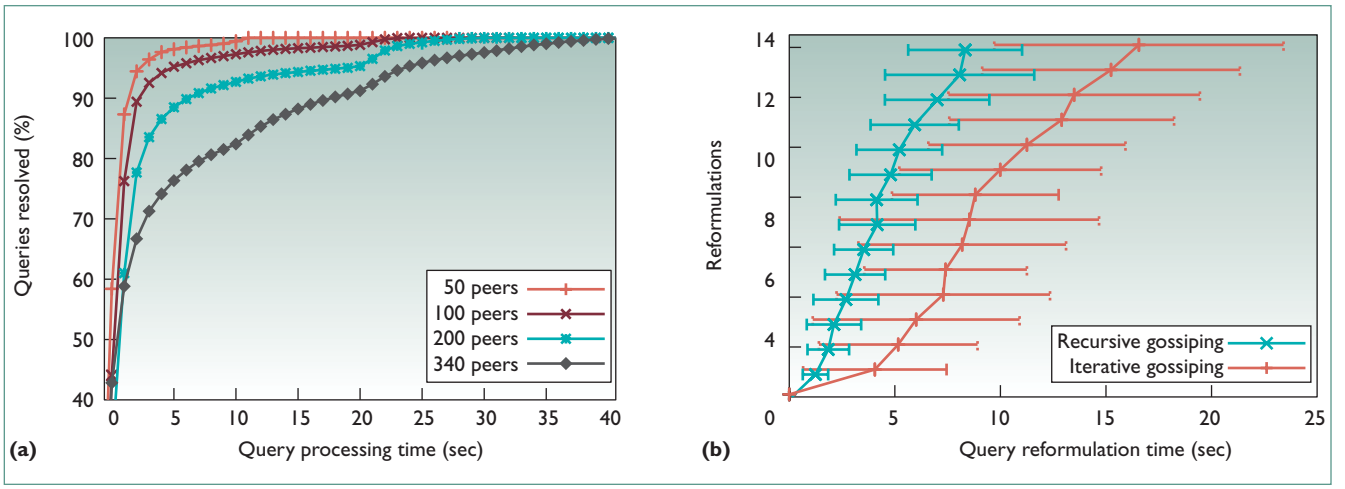

Figure 3. Two GridVine deployments. (a) The cumulative distribution of query resolution time for networks ranging from 50 to 340 peers distributed globally. (b) The reformulation steps for up to 14 query reformulations in a network with 40 schemas stored on 40 different peers.

inserted between 4000 and 80,000 triples (depending on the network size) in the system and monitored the resolution of thousands of atomic queries that each peer issued over several hours. For the largest network consisting of 340 peers, 43 percent of the queries were answered within one second, and 75 percent within five seconds. Note that the machines used for the experiment were heavily loaded due to the processes inherent to the PlanetLab infrastructure and to several other experiments running concurrently. Despite the heavily skewed distribution of the keys generated when indexing the triples, the partition of the key space at the overlay layer and thus the storage load at the semantic layer remained balanced thanks to P-Grid's proactive load-balancing mechanisms. Although several peers went offline during the test, all queries were answered properly due to the dynamic index replication triggered at the overlay layer.

Figure 3b illustrates a network-intensive deployment focusing on data integration, where GridVine was deployed over 40 peers running on 20 cores in a local-area cluster. The peers were interconnected through a realistic network setting (ModelNet; http://modelnet.ucsd.edu) based on a client-stub topology, with 5 millisecond delays and 200 Kbyte-per-second links. Each peer was set up to be responsible for a distinct schema. The schemas were related through a random graph of schema mappings to create a fully-connected semantic mediation layer. Each peer issued several thousands of atomic queries in various points of this network to disseminate them throughout the semantic layer. The figure shows the successive reformulation steps, with confidence intervals set to 95 percent and a maximum of 14 reformulations for each query. We tested two approaches: in *iterative gossiping*, the peer issuing the original query is responsible for retrieving all mappings and reformulating all queries by itself iteratively. In *recursive gossiping*, the reformulation process is iteratively delegated to those peers receiving reformulated queries. Recursive gossiping performs systematically better because it distributes the reformulation load more evenly among peers.

Building Large-Scale Collaborative Applications with GridVine

Originally developed as a simple file-sharing application supporting RDF annotations,[7] GridVine has evolved into a general-purpose semantic infrastructure over the past two years. Today, it supports efficient search in very large-scale networks and offers persistent storage, load-balancing, identity management, RDF entailment, and semantic query diffusion to higher-level applications. We describe our experiences using GridVine for two applications: a large-scale photo sharing platform and a distributed semantic desktop.

PicShark: Sharing Partially Annotated Content

PicShark (http://lsirwww.epfl.ch/PicShark) is a distributed application for sharing annotated photos. Our goal with PicShark is to let a potentially very large population of users upload both photos and

arbitrary metadata pertaining to the photos. Because annotating photos is a time-consuming task that's difficult to automate, we assume that much of the metadata are missing, but take advantage of similar metadata and photos in the network to complete missing metadata with plausible values whenever possible.

As the application called for both scalable storage and efficient query resolution mechanisms, we decided to use GridVine as the underlying infrastructure. After having developed wrappers to export various annotation formats into RDF schemas and statements, we could let GridVine take care of all the indexing, load-balancing, storage, and query-resolution operations and concentrate on algorithms to elicit information for the missing metadata. Being able to efficiently query the whole network for some specific information — such as a name appearing in the annotations — lets us find related photos or schemas on a global level. To improve the process, we additionally implemented fuzzy search in GridVine based on lexicographical distances, which allows simultaneous retrieval of information related to both *John Doe* and *Jonn Doe*, for example. Finally, we took advantage of GridVine's data integration capabilities to not only search for similar annotations in a given community of interest, but to expand our searches to related communities by reformulating the queries through mappings interconnecting the schemas defined by the various communities.

Nepomuk: The Distributed Semantic Desktop

Nepomuk (http://nepomuk.semanticdesktop.org/) is a large-scale effort aimed at enriching normal desktop environments with semantics and information sharing capabilities to foster knowledge reuse, social interactions, and global collaboration. In this context also, the project consortium chose GridVine to support higher-layer information processes. In Nepomuk, GridVine is used to share semantic information globally and to integrate heterogeneous annotations that the users provided.

One of the project's key objectives is to automatically discover social communities and dynamically organize the users into meaningful groups. Logging some of the user interactions in GridVine lets us globally detect similar behaviors and discover related sets of users using fuzzy searches. Once related users are clustered in groups, we analyze the structured information (such as address books and annotated files) they share. To foster interoperability among all the users belonging to a given social group, we automatically create mappings between the schemas they use to encode their data. Hence, GridVine can help to automatically discover social trends emerging from the users, to cluster socially similar users, and finally, to promote knowledge exchange by ensuring data interoperability between users belonging to the same group. The whole process scales gracefully with the number of users, as all queries still only take $O(\log(|\Pi|))$ messages — where $||\Pi||$ is the number of users in the network — to reach their final destination.

To the best of our knowledge, GridVine is the first infrastructure supporting both scalable storage and structured query processing while promoting global semantic interoperability through purely decentralized and self-organizing processes. Mapping structured information at the semantic layer to routable keys at the overlay layer lets us exploit a scalable, efficient, and totally decentralized index structure to resolve operations at the higher layers. Following the principle of data independence, our approach separates the logical and physical aspects such that it can be generalized to any physical infrastructure that provides functionalities that are similar to our P-Grid P2P system. GridVine is both efficient and stable, as demonstrated in our tests on several hundreds of machines. Also, it provides a convenient abstraction to build higher-layer semantic applications, as we described for two concrete case-studies related to large-scale knowledge sharing applications.

The past decade saw the rise of various mechanisms for organizing minimally-structured, human-processable data in the large. Today, we believe that a new revolution targeting declarative, semistructured and machine-processable information is on its way. End users, who used to be restricted to passively consuming manually curated digital information, are today evolving into industrious supervisors of semiautomatic processes, creating digital artifacts on a continuous basis. As more and more structured information is today generated in automated and decentralized ways, it's increasingly important to support incremental interoperability mechanisms to meaningfully process data on a large scale. In that context, we see the self-organizing and decentralized architecture we propose as not only complementing the

traditional approaches that organize information through top-down consensus creation but also as the only scalable resort for organizing data in the distributed, autonomous, and complex data spaces currently emerging.

Acknowledgments

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

References

1. M. Cai and M. Frank, "RDFPeers: A Scalable Distributed RDF Repository based on a Structured Peer-to-Peer Network," *Proc. Int'l World Wide Web Conf.* (WWW 04), ACM Press, 2004, pp. 650–657.
2. P. Haase and R. Siebes, "Peer Selection in Peer-to-Peer Networks with Semantic Topologies," *Proc. Int'l Conf. Semantics in a Networked World* (ICSNW 04), Springer-Verlag, 2004, pp. 108–125.
3. J.M. Hellerstein, "Toward Network Data Independence," *SIGMOD Record*, vol. 32, no. 3, 2003, pp. 34–40.
4. K. Aberer et al., "Indexing Data-Oriented Overlay Networks," *Proc. Int'l Conf. Very Large Databases* (VLDB 05), 2005, ACM Press, pp. 685–696.
5. K. Aberer, *Efficient Search in Unbalanced, Randomized Peer-To-Peer Search Trees*, tech. report IC/2002/79, Swiss Federal Inst. Technology, Lausanne (EPFL), 2002; www.p-grid.org/Papers/TR-IC-2002-79.pdf.
6. A. Seaborne, "RDQL — A Query Language for RDF," W3C Member Submission, 2004; www.w3.org/Submission/RDQL/.
7. K. Aberer et al., "GridVine: Building Internet-Scale Semantic Overlay Networks," *Proc. Int'l Semantic Web Conf.* (ISWC 04), Springer-Verlag, 2004, pp. 107–121.
8. E. Liarou, S. Idreos, and M. Koubarakis, "Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks," *Proc. Int'l Semantic Web Conf.* (ISWC 06), Springer-Verlag, 2006, pp. 399–413.

Philippe Cudré-Mauroux is a senior researcher and lecturer at Ecole Polythechnique Fédérale de Lausanne (EPFL) in Switzerland. His current research interests are in information management for large-scale settings, with an emphasis on semantic overlay networks and peer data management systems. Cudré-Mauroux has a PhD in distributed information systems from EPFL. He is a member of the IEEE and the ACM. Contact him at philippe.cudre-mauroux@epfl.ch.

Suchit Agarwal is a research assistant at EPFL, currently working on the GridVine project. He has a B.Tech. degree (with honors) in information technology from IIIT-Allahabad, India. His current interests involve large-scale distributed information systems and structural analysis of large graphs. Contact him at suchit.agarwal@epfl.ch.

Karl Aberer is a professor at EPFL. His particular interests are in decentralized system architectures, self-organization mechanisms, and emergent structures in information systems. He also serves as the director of the Swiss National Research Center for Mobile Information and Communication Systems (NCCR-MICS). Aberer has a PhD in mathematics from Eidgenössische Technische Hochschule in Zürich. He is a member of the IEEE and the ACM. Contact him at karl.aberer@epfl.ch.