Grid Data Management: Open Problems and New Issues

Esther Pacitti, Patrick Valduriez, Marta Mattoso

Grid Data Management: open problems and new issues[1]

Esther Pacitti, Patrick Valduriez

INRIA and LINA, University of Nantes – France
Esther.Pacitti@lina.univ-nantes.fr
Patrick.Valduriez@inria.fr

Marta Mattoso

COPPE, Federal University of Rio de Janeiro – Brazil
marta@cos.ufrj.br

Abstract. Initially developed for the scientific community, Grid computing is now gaining much interest in important areas such as enterprise information systems. This makes data management critical since the techniques must scale up while addressing the autonomy, dynamicity and heterogeneity of the data sources. In this paper, we discuss the main open problems and new issues related to Grid data management. We first recall the main principles behind data management in distributed systems and the basic techniques. Then we make precise the requirements for Grid data management. Finally, we introduce the main techniques needed to address these requirements. This implies revisiting distributed database techniques in major ways, in particular, using P2P techniques.

1 Introduction

Initially developed for the scientific community as a generalization of cluster computing, Grid computing is now gaining much interest in other areas such as enterprise information systems [15,23]. For instance, IBM, Oracle and Microsoft are all promoting Grid computing with tools and services for both scientific and enterprise applications[2]. The Grid is a natural evolution of the Web and enables the virtualization of distributed, heterogeneous resources using Web services [9]. These resources can be data sources (files, databases, web sites, etc), computing resources (multiprocessors, supercomputers, clusters) and application resources (scientific applications, information management services, etc.). Unlike the Web, which is client-server oriented, the Grid is demand-oriented: users send requests to the Grid which allocates them to the most appropriate resources to handle them. A Grid is also an organized, secured environment managed and controlled by administrators.

[1] Work partially funded by ARA "Massive Data" of the French ministry of research (project Respire), the European Strep Grid4All project, the CAPES-COFECUB Daad project and the CNPq-INRIA Gridata project.

[2] One can notice the name of Oracle's last database version 10g where g stands for "Grid".

Compared with cluster computing which only deals with parallelism, the Grid is characterized with high heterogeneity, large-scale distribution and large-scale parallelism. Thus, it can offer advanced services on top of very large amounts of distributed data, e.g. the Southern California Earthquake Center digital library [16].

To realize the full potential of Grid computing, interoperability of tools and middlewares is of major importance. This has been addressed by the Open Grid Services Architecture (OGSA) and the Globus toolkit of the Globus alliance [15]. OGSA and Globus are contributing to the success of Grid computing, well beyond the original computational Grids. As the Grid is breaking into important areas such as enterprise information systems, data management becomes critical. In particular, the techniques must scale up while addressing the autonomy, dynamicity and heterogeneity of the data resources.

In this paper, we consider the general problem of Grid data management, with many applications and data sources distributed in a large-scale computer network. Applications and data sources can be fairly autonomous, i.e. locally owned and controlled, and highly heterogeneous in size and complexity. Managing and transparently accessing such resources in a highly distributed system with a good cost/performance trade-off is a hard problem. What we need are data management techniques that scale up while addressing the autonomy, dynamicity and heterogeneity of resources.

Data management in distributed systems has been traditionally achieved by distributed database systems [38] which enable users to transparently access and update several databases in a network using a high-level query language (e.g. SQL). Transparency is achieved through a global schema which hides the local databases' heterogeneity. In its simplest form, a *distributed database system* is a centralized server that supports a global schema and implements distributed database techniques (query processing, transaction management, consistency management, etc.). This approach has proved effective for applications that can benefit from centralized control and full-fledge database capabilities, e.g. information systems. However, it cannot scale up to more than tens of databases. Data integration systems [47] extend the distributed database approach to access data sources on the Internet with a simpler query language in read-only mode. Parallel database systems [48] also extend the distributed database approach to improve performance (transaction throughput or query response time) by exploiting database partitioning using a multiprocessor or cluster system. Although data integration systems and parallel database systems can scale up to hundreds of data sources or database partitions, they still rely on a centralized global schema and strong assumptions about the network.

The main solutions for Grid data management, in the context of computational Grids, are file-based. A basic solution, used in Globus, is to combine global directory services to locate files and a secured file transfer protocol such as GridFTP [37]. Although simple, this solution does not provide distribution transparency as it requires the application to explicitly transfer files. However, high-level data management services can be provided on top of GridFTP, e.g. the Stork data placement scheduler in the Condor project [13]. Another solution is to use distributed file systems for the Grid, e.g. GridNFS [20], which provide location-independent file access. Recent solutions have also recognized the need for high-level data access and extended the distributed database architecture [8] whereby clients send database

requests to a Grid server (with a directory) that forwards them transparently to the appropriate database servers. These solutions rely on some form of global directory management, where directories can be distributed and replicated. However, current solutions focus on data sharing and collaboration for statically defined virtual organizations with powerful servers. They cannot be easily extended to satisfy the needs of dynamic virtual organizations such as professional communities where members contribute their own data sources, perhaps small but in high numbers, and may join and leave the Grid at will. In particular, current solutions require heavy organization, administration and tuning which are not appropriate for large numbers of small devices.

Peer-to-Peer (P2P) techniques which focus on scaling, dynamicity, autonomy and decentralized control can be very useful to Grid data management. The synergy between P2P computing and Grid computing has been advocated to help resolve their respective deficiencies [17]. For instance, Narada [34], P-Grid [1] and Organic Grid [11] develop self-organizing and scalable Grid services using P2P interactions. The Grid4All European project [19] which aims at democratizing the Grid is also using P2P techniques. As further evidence of this trend, the Global Grid Forum has recently created the OGSA-P2P group [10] to extend OGSA for the development of P2P applications.

Initial research on P2P systems has focused on improving the performance of query routing in the unstructured systems which rely on flooding. This work led to structured solutions based on distributed hash tables (DHT), e.g. CAN [40] and Chord [44], or hybrid solutions with super-peers that index subsets of peers [54]. Recent work on P2P data management has concentrated on supporting semantically rich data (e.g., XML documents, relational tables, etc.) using a high-level SQL-like query language and distributed database capabilities (schema management, query processing, replication, etc.), e.g. ActiveXML [2], Appa [4,5], Edutella [35], Piazza [46], Pier [21].

Although useful, these P2P data management capabilities are not sufficient for emerging Grid environments. Support for dynamic virtual organizations requires other advanced capabilities such as semantic-based resource discovery, workflow support, autonomic management and security. Resource discovery becomes important as resources can be added or removed frequently. Hence, it should be high-level and based on semantics (e.g. described by ontologies) rather than low-level (e.g. name-based). Workflow support is also critical to control the execution of complex requests which have to deal with several applications and data sources, possibly spanning different Grids. Autonomic management of the data, with self-organization, self-tuning and self-repairing, is critical to relieve the Grid users from administration tasks. Finally, data security and data privacy are major issues, in particular, to isolate between different virtual organizations and different grids.

In this paper, we discuss the main open problems and new issues related to Grid data management using P2P techniques. We first recall the main principles behind data management in distributed systems and the basic techniques needed for supporting advanced functionality (schema management, access control, query processing, transaction management, consistency management, reliability and replication). Then we make precise the requirements for Grid data management

within P2P. Finally, we introduce the main techniques needed to address these requirements.

The rest of the paper is organized as follows. Section 2 recalls the main capabilities of distributed database systems. Section 3 discusses requirements for Grid data management. Section 4 introduces the main techniques needed for Grid data management. Section 5 concludes.

2 Data Management in Distributed Systems

The fundamental principle behind data management is *data independence*, which enables applications and users to share data at a high conceptual level while ignoring implementation details. This principle has been achieved by *database systems* which provide advanced capabilities such as schema management, high-level query languages, access control, automatic query processing and optimization, transactions, data structures for supporting complex objects, etc.

A *distributed database* is a collection of multiple, logically interrelated databases distributed over a computer network. A *distributed database system* is defined as the software system that permits the management of the distributed database and makes the distribution *transparent* to the users [38]. Distribution transparency extends the principle of data independence so that distribution is not visible to users.

These definitions assume that each site logically consists of a single, independent computer. Therefore, each site has the capability to execute applications on its own. The sites are interconnected by a computer network with loose connection between sites which operate independently. Applications can then issue queries and transactions to the distributed database system which transforms them into local queries and local transactions (see Figure 1) and integrates the results. The distributed database system can run at any site s, not necessarily distinct from the data (i.e. it can be site 1 or 2 in Figure 1).

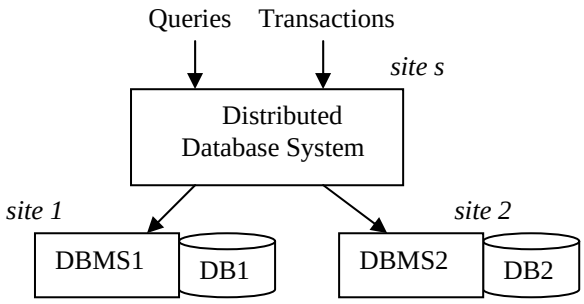

Figure 1. A distributed database system with two data sites

The database is physically distributed across the data sites by fragmenting and replicating the data. Given a relational database schema, for instance, fragmentation subdivides each relation into partitions based on some function applied to some

tuples' attributes. Based on the user access patterns, each of the fragments may also be replicated to improve locality of reference (and thus performance) and availability.

The functions provided by a distributed database system could be those of a database system (schema management, access control, query processing, transaction support, etc). But since they must deal with distribution, they are more complex to implement. Therefore, many systems support only a subset of these functions.

When the data and the databases already exist, one is faced with the problem of providing integrated access to heterogeneous data. This process is known as *data integration*: it consists in defining a *global schema* over the existing data and *mappings* between the global schema and the local database schemas. Data integration systems have received several names such as federated database systems, multidatabase systems and, more recently, mediators systems. Standard protocols such as Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) ease data integration using SQL. In the context of the Web, mediator systems [47] allow general access to autonomous data sources (such as files, databases, documents, etc.) in read only mode. Thus, they typically do not support all database functions such as transactions and replication.

When the architectural assumption of each site being a (logically) single, independent computer is relaxed, one gets a *parallel database system* [48], i.e. a database system implemented on a tightly-coupled multiprocessor or a cluster. The main difference with a distributed database system is that there is a single operating system which eases implementation and the network is typically faster and more reliable. The objective of parallel database systems is high-performance and high-availability. High-performance (i.e. improving transaction throughput or query response time) is obtained by exploiting data partitioning and query parallelism while high-availability is obtained by exploiting replication.

The distributed database approach has proved effective for applications that can benefit from static administration, centralized control and full-fledge database capabilities, e.g. information systems. For administrative reasons, the distributed database system typically runs on a separate server and this reduces scaling up to tens of databases. Data integration systems achieve better scale up to hundreds of data sources by restricting functionality (i.e. read-only querying). Parallel database systems can also scale up to large configurations with hundreds of processing nodes. However, both data integration systems and parallel database rely on a global schema that can be either centralized or replicated.

3 Requirements for Grid Data Management

A Grid is an organized collection of nodes in a network which contribute various resources, e.g. computation, storage, data, applications, etc. Depending on the contributed resources and the targeted applications, many different kinds of Grids and architectures are possible. Computational Grids, the earlier kind of Grid, typically aggregate very powerful nodes (supercomputers, clusters) to provide high-performance computing for scientific applications (e.g. physics, astronomy). On the contrary, scavenging Grids, such as the well-known Seti@home Grid [41], aggregate

very large numbers (up to millions) of desktop computers to provide relatively free CPU cycles. Data Grids aggregate heterogeneous data sources (like a distributed database) and provide additional services for data discovery, delivery and use to scientific applications. More recently, enterprise Grids [23] have been proposed to aggregate information system resources, such as Web servers, application servers and database servers, in the enterprise.

A common need to these different kinds of Grids is interoperability of heterogeneous resources. OGSA provides a framework to create solutions using Web service standards (HTTP, XML, SOAP, UDDI, WSDL, etc.) and a Service Oriented Architecture (SOA) which allows for loose-coupling [9]. In particular, distributed database management techniques can be implemented using Web services [8]. The adoption of Web services in enterprise information systems makes OGSA appealing and several offerings for enterprise Grids are already based on the Globus platform.

Web service standards are useful for Web data management: XML for data exchange, XSchema for schema description, SOAP for remote procedure calls, UDDI for directory access, WSDL for data source description, WS-Transaction for distributed transactions, BPEL for workflow control, etc. However, Web services focus on loosely-coupled interaction among services and thus are not sufficient to address the specific data management requirements of emerging Grids, in particular, those with dynamic virtual organizations. When considering data management, the main requirements of a Grid are:

- **Data transparency and consistency**: applications should be able to access consistent data without knowing their location. Consistency is of major importance for replicated data.
- **Autonomy support**: an autonomous node should be able to join or leave the Grid at any time. It should also be able to control the data it stores and which other nodes can store its data, e.g. some other trusted nodes.
- **Resource discovery**: data resources which can be added or removed frequently should be able to be discovered easily, with a high-level of semantics (e.g. using ontologies) rather than using names in an LDAP directory.
- **Query expressiveness**: the query language should allow the user to describe the desired data at the appropriate level of detail. The simplest form of query is key look-up which is only appropriate for finding files. Keyword search with ranking of results is appropriate for searching documents. But for more structured data, an SQL-like query language which relies on some form of global schema is necessary.
- **Efficiency**: the efficient use of the Grid resources (bandwidth, computing power, storage) should result in lower cost and thus higher throughput of queries, i.e. a higher number of queries can be processed by the Grid in a given time.
- **Quality of service**: refers to the user-perceived efficiency of the system, e.g. completeness of query results, data consistency, data availability, query response time, etc.
- **Fault-tolerance**: efficiency and quality of services should be provided despite the occurrence of nodes' failures. Given the dynamic nature of nodes which may leave or fail at any time, the only solution is to rely on data replication.

- **Workflow support**: Several applications are being specified through workflows. A workflow should be defined through a high level language, supporting most workflow constructs [50], e.g. BPEL. The Grid must support a workflow execution engine for such languages taking full advantage of its resources accounting for the dynamic nature of nodes.
- **Autonomic management**: Autonomic management of the data, with self-organization, self-tuning, self-repairing, self-healing is critical to provide efficiency and quality of service. It should also relieve the Grid users from administrating their own resources.
- **Security**: the semi-open nature of a Grid makes security a major challenge since one cannot rely on trusted servers. For data management, the main security issue is access control and data privacy which includes enforcing intellectual property rights on data contents.

These requirements cannot be achieved by simply combining distributed database techniques and Web services into the Grid environment. New data management techniques are necessary.

4 Grid Data Management Techniques

P2P data management techniques which focus on scaling, dynamicity, autonomy and decentralized control can be very useful to address some of the above requirements. However, there are many different P2P architectures and topologies that are possible for Grid systems. Depending on the architecture, the above requirements are more or less difficult to achieve. For simplicity, let us consider the three main classes of P2P systems [49]: unstructured, structured and super-peer.

In *unstructured systems*, the simplest ones, each peer can directly communicate with its neighbors. Autonomy is high since a peer only needs to know its neighbors to log in. Searching for information is simple and proceeds by flooding the network with queries, each peer processing and redirecting the incoming queries to its neighbors. There is no restriction on the expressiveness of the query language which could be high. Such query routing based on flooding is general but does not scale up to large numbers of peers. Also, the incompleteness of the results can be high since some peers containing relevant data or their neighbors may not be reached because they are off-line. However, since all peers are equal and able to replicate data, fault-tolerance is very high.

Structured systems improve the performance of unstructured systems based on distributed hash tables (DHT). A DHT system provides a hash table interface with primitives put(key, value) and get(key), where key is typically a file name and each peer is responsible for storing the values (file contents) corresponding to a certain range of keys. There is an overlay routing scheme that delivers requests for a given key to the peer responsible for that key. This allows one to find a peer responsible for a key in $O(\log n)$, where n is the number of peers in the network. Because a peer is responsible for storing the values corresponding to its range of keys, autonomy is limited. Furthermore, DHT queries are typically limited to exact match keyword

search. Active research is on-going to extend the capabilities of DHT systems to deal with more general queries such as range queries and join queries [21].

Super-peer P2P systems are hybrid between pure systems and client-server systems. Some peers, the super-peers, act as dedicated servers for some other peers and can perform complex functions such as indexing, query processing, access control and meta-data management. Using only one super-peer reduces to client-server with all the problems associated with a single server. Super-peers can also be organized in a P2P fashion and communicate with one another in sophisticated ways. Thus, unlike client-server systems, global information is not necessarily centralized and can be partitioned or replicated across all super-peers. The main advantage of super-peer is efficiency and quality of service. A requesting peer simply sends the request, which can be expressed in a high-level language, to its responsible super-peer which can then find the relevant peers either directly through its index or indirectly using its neighbor super-peers. Access control can also be better enforced since directory and security information can be maintained at the super-peers. However, autonomy is restricted since peers cannot log in freely to any super-peer. Fault-tolerance is typically low since super-peers are single points of failure for their sub-peers.

To summarize, unstructured systems have better autonomy and fault-tolerance but can be quite inefficient because they rely on flooding for query routing. Hybrid systems have better potential to satisfy high-level data management requirements. However, DHT systems are best for key-based search and could be combined with super-peer systems for more complex searching. Thus, there is no single P2P architecture that dominates and we could envision different combinations for different Grids.

To address the requirements of Grid data management using P2P techniques, distributed database techniques must be revised and extended in major ways. Grid and P2P data management must deal with semantically rich data (e.g., XML documents, relational tables, etc.) and needs functions similar to those of distributed database systems. In particular, users should be able to use a high-level query language to describe the desired data as with OGSA-DAI, an OGSA standard for accessing and integrating distributed data [8]. In OGSA-DAI, distributed query execution is being addressed by OGSA-DQP. But the characteristics of Grid and P2P create new issues. First, the dynamic and autonomous nature of nodes makes it hard to give guarantees about result completeness and makes traditional (static) query optimization impossible. Second, data management techniques need to scale up to high numbers of nodes. Third, the lack of centralized control makes global schema management and access control difficult. Finally, even when using replication, it is hard to achieve fault-tolerance and availability in the presence of unstable nodes. The main techniques which we think require further research are the following.

- **Schema management.** Users should be able to express high-level queries over their own schema without relying on a global schema. Thus the problem is to support decentralized schema mapping so that a query on one node's schema can be reformulated efficiently in a query on another node's schema. Solutions have been proposed for specific P2P systems [35,46] but need to be generalized to Grids.
- **Data source discovery.** Data source discovery in Grids typically relies on LDAP directories using name-based or key-word based queries. When data

resources can be added frequently, resource publication and discovery becomes difficult. Decentralized P2P techniques can be used to improve the performance of resource discovery [22]. Another promising solution is to exploit more semantics, e.g. using ontologies, together with information retrieval techniques [45]. This requires being able to reason across semantic domains.

- **Query processing.** Recent work on query processing in P2P and large-scale networks has focused on supporting some complex queries such as Top-k queries [7]. More work is needed to generalize to different classes of queries like OLAP queries which may access very large amounts of data. Efficiency in query processing can be obtained by exploiting the parallelism and load balancing of the distributed architecture. Adaptive algorithms designed for database clusters [26, 27] are useful but need be significantly revised to deal with the unpredictable nature of a Grid environment.
- **Load balancing.** In a Grid, load balancing is critical to achieve overall good performance while satisfying the interests of users and resource providers. Grids typically use economic models for resource trading, e.g. a model based on auctions [12]. Load balancing of data access requests can take advantage of an economic model but must be extended to capture providers and users' satisfaction based on a notion of quality. Data access requests following spatial relationships should be clustered accordingly by the Grid [32]. In particular, it should be able to achieve equitable mediation between asymmetric interests [25]. In a Grid, load balancing should also deal with multiple mediators and the dynamic behaviour and autonomy of nodes, as well as the large scale of the system.
- **Replication and caching.** Replicating or caching data are important to improve performance (by avoiding network accesses). Furthermore, replication is important to increase fault-tolerance. The most general form of replication is multi-master, i.e. whereby the same replica may be updated by several (master) nodes. However, conflicting updates of the same data at different nodes can introduce replica divergence. A practical solution is optimistic replication [39, 42] which allows the independent updating of replicas and divergence until reconciliation. Reconciliation solutions have been proposed for specific P2P systems [30, 31] and need be generalized to Grids. Another useful solution is to live with divergence and provide the ability to access the most current replicas [6]. Replication can also be useful to cache the frequently accessed data at some specific nodes. Data caching has been successfully adopted to improve query response time in large-scale distributed environments [52]. However, a problem is to address the dynamic nature of the system, where nodes can join and leave the system at any time. In particular, caches may get frequently changed or unavailable.
- **Workflow management.** Workflow management has been supported since the first versions of Grid infrastructures such as Condor-G [18] and Pegasus [43]. Recently, several workflow management systems for Grid (WfMSG) have been proposed, such as: Askalon [51], Kepler [28], P-Grade [24], Swift [55], Taverna [36], Triana [13]. However, processing workflows in a Grid is an open issue and imposes many challenges [53]. The scheduling of a

workflow focuses on mapping and managing the execution of tasks on Grid shared resources that are not directly under the control of these workflow systems [29]. To address such mismatch, the P-Grade Grid portal provides automatic mappings for several Grid environments while managing executions with many different parameter sets. However, in current WfMSG, scheduling is left to the user and addressed through static allocation strategies. When dynamic allocation strategies are available, resources are randomly chosen. Recently, dynamic adaptive workflow planning has been proposed to take advantage of most active Grid nodes [33]. Workflow based on Web services can discover equivalent services at run time thus providing many opportunities for dynamic scheduling and re-planning. Thus, choosing the best strategy for workflow execution in a Grid is a challenging research area.

- **Autonomic data management.** Self-management of the data by the Grid is critical for emerging Grid applications, in particular, to support dynamic virtual organizations and large numbers of small devices. Modern database systems already provide good self-administration, self-tuning and self-repairing capabilities which ease application deployment and evolution. However, extending these capabilities to the level of a Grid is hard with different data resources, each with different autonomic capabilities. In particular, an open problem is the automatic definition and allocation of data replicas to deal with load increases.
- **Data security and privacy**. Data security and access control in a Grid typically relies on a LDAP directory for authenticating users, possibly with single sign-on, and secured communication protocols such as SSL to exchange data. However, the semi-open nature of a Grid makes security and privacy a major challenge since one cannot rely on trusted servers. In some applications, it is important that data privacy be preserved. This has become a main challenge for database systems [3] since, by providing high-level query capabilities, it remains possible to infer private data. Furthermore, enforcing intellectual property rights on data contents is an open issue.

5 Conclusion

As Grid computing is gaining much interest in important areas where access to information is essential, data management becomes critical. In a Grid environment, data management techniques must scale up while addressing the autonomy, dynamicity and heterogeneity of the resources. In particular, as addressed in the Grid4All European project [19], they should be able to satisfy the needs of dynamic virtual organizations such as professional communities where members contribute their own resources, perhaps small but in high numbers, and may join and leave the Grid at will.

When considering data management, the main requirements of a Grid are: autonomy support, resource discovery, query expressiveness, efficiency, quality of service, fault-tolerance, workflow support, autonomic management, and security. Addressing these requirements implies revisiting distributed database techniques in

major ways. P2P techniques which focus on scaling, dynamicity, autonomy and decentralized control can be very useful to Grid data management. However, they should deal with semantically rich data (e.g., XML documents, relational tables, etc.) with a high-level query language as with OGSA-DAI. We analyzed the potential of three main P2P architectures (unstructured, structured and super-peer) for the Grid. An important conclusion is that there is no single P2P architecture that dominates and different combinations can be envisioned for different Grids.

To address the requirements of Grid data management using P2P techniques, the main techniques which require further research are: schema management, data source discovery, query processing, load balancing, replication and caching, workflow management, autonomic data management, data security and privacy. Research in some of these areas, in particular, autonomic data management, data security and privacy, is hard and will require much work before concrete solutions emerge.

Extensive experimentation with a large variety of applications exhibiting very different properties (large-scale, dynamic behavior, autonomy, etc) is also important to characterize which P2P techniques are best for which applications. For instance, the integration of light-weight devices (personal digital assistants, smart phones, secured chips, etc.) and mobile networks in a Grid is likely to require specific data management techniques.

Acknowledgements

The authors wish to thank the anonymous reviewers, as well as Peter Kacsuk, for their helpful comments and suggestions.

References

1. K. Aberer et al. P-Grid: a Self-organizing Structured P2P System. *SIGMOD Record, 32(3)*, 2003.
2. S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, T. Milo. Dynamic XML Documents with Distribution and Replication. *ACM SIGMOD Int. Conf. on Management of Data*, 2003.
3. R. Agrawal, J. Kiernan, R. Srikant, Y. Xu. Hippocratic Databases. *VLDB Conf.*, 2002.
4. R. Akbarinia, V. Martins, E. Pacitti, P. Valduriez. Design and Implementation of APPA. *Global Data Management* (Eds. R. Baldoni, G. Cortese, F. Davide), IOS Press, 2006.
5. R. Akbarinia, V. Martins. Data Management in the APPA System. *Journal of Grid Computing*, this issue, 2007.
6. R. Akbarinia, E. Pacitti, P. Valduriez. Data Currency in DHTs. *ACM SIGMOD Int. Conf. on Management of Data*, 2007.
7. R. Akbarinia, E. Pacitti, P. Valduriez. Reducing Network Traffic in Unstructured P2P Systems Using Top-K Queries. *Distributed and Parallel Databases* 19(2), 67-86, 2006.
8. M. Antonioletti et al. The Design and Implementation of Grid Database Services in OGSA-DAI. *Concurrency - Practice and Experience* 17(2-4): 357-376, 2005.
9. M. P. Atkinson et al. Web Service Grids: an Evolutionary Approach. *Concurrency - Practice and Experience* 17(2-4): 377-389, 2005.

10. K. Bhatia. OGSA-P2P Research Group: Peer-To-Peer Requirements on the Open Grid Services Architecture Framework. *Global Grid Forum Document GFD-I.049*, 2005.
11. A. J. Chakravarti, G. Baumgartner, M. Lauria. The Organic Grid: Self-organizing Computation on a Peer-to-peer Network. *IEEE Transactions on Systems, Man, and Cybernetics*, Part A 35(3): 373-384, 2005.
12. M. Chen, G. Yang, X. Liu. Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P Computing. *Int. Workshop on Grid and Cooperative Computing*, 2003.
13. D. Churches, G. Gombás, A. Harrison, et al., Programming scientific and distributed workflow with Triana services, *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 18(10),* pp. 1021-1037, 2006.
14. Condor Project. http://www.cs.wisc.edu/condor.
15. Enterprise Grid Alliance. http://www.gridalliance.org.
16. M. Faerman, R.W. Moore, B. Minster, P. Maechling, Y. Cui, Y. Hu, J. Zhu. Managing Large Scale Data for Earthquake Simulations. *Journal of Grid Computing*, this issue, 2007.
17. I. T. Foster, A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. *Int. Workshop on P2P Systems (IPTPS)*, 2003.
18. J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke. Condor-G: a Computation Management Agent for Multi-Institutional Grids. *IEEE Symp. on High Performance Distributed Computing (HPDC)*, 2001.
19. Grid4all: Democratizing the Grid. http://www.grid4all.eu.
20. P. Honeyman,W.A. Adamson, Shawn McKee. GridNFS: Global Storage for Global Collaborations. *IEEE Int. Symp. on Global Data Interoperability - Challenges and Technologies*, 111-115, 2005.
21. R. Huebsch et al. Querying the Internet with PIER. *VLDB Conf.*, 2003.
22. A. Iamnitchi, I. Foster. On Fully Decentralized Resource Discovery in Grid Environments. *Int.Workshop on Grid Computing*, 2001.
23. R. Jiménez-Peris, M. Patiño-Martínez, B. Kemme. Enterprise Grids: Challenges Ahead. *Journal of Grid Computing*, this issue, 2007.
24. P. Kacsuk, Z. Farkas, G. Sipos, et al., Workflow-level Parameter Study Management in multi-Grid environments by the P-GRADE Grid portal, GCE06 - Grid Computing Environments Workshop in Conjunction with Supercomputing, 2006.
25. P. Lamarre, S. Cazalens, S. Lemp, P. Valduriez. A Flexible Mediation Process for Large Distributed Information Systems. *CoopIS Conf.*, 2004.
26. A. Lima, M. Mattoso, P. Valduriez. Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster. *Brazilian Symposium on Databases (SBBD)*, 2004.
27. A. Lima, M. Mattoso, P. Valduriez. OLAP Query Processing in a Database Cluster. *Int. Conf. on Parallel and Distributed Computing (Euro-Par*), *LNCS 3149*, pp. 355-362, 2004.
28. B. Ludäscher, I. Altintas, C. Berkley, et al., Scientific workflow management and the Kepler system, *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 18(10),* pp. 1039-1065, 2006.
29. A. Mandal, K. Kennedy, C. Koelbel, et al., Scheduling Strategies for Mapping Application Workflows onto the Grid, *14th IEEE International Symposium on High-Performance Distributed Computing (HPDC-14)*, 2005.
30. V. Martins, R. Akbarinia, E. Pacitti, P. Valduriez. Reconciliation in the APPA P2P System. *IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS)*, 2006.
31. V. Martins, E. Pacitti. Dynamic and Distributed Reconciliation in P2P-DHT Networks. *European Conf. on Parallel Computing (Euro-Par)*, 2006.
32. L. Meyer, M. Mattoso, I. Foster, et al., Planning Spatial Workflow to Optimize Grid Performance. *ACM Symposium of Applied Computing*, pp. 786-790, 2006.

33. L. Meyer, M. Wilde, M. Mattoso, I. Foster. An Opportunistic Algorithm for Scheduling Workflows on Grids. *Int. Conf. on High Performance Computing for Computational Science (VecPar), LNCS 4395*, pp. 212 – 224, Springer 2007.
34. Narada project. http://aspen.ucs.indiana.edu/users/shrideep/narada.
35. W. Nejdl, W. Siberski, M. Sintek. Design Issues and Challenges for RDF- and Schema-based Peer-to-Peer Systems. *SIGMOD Record, 32(3),* 2003.
36. T. Oinn, R. Greenwood, C. Goble, Chris Wroe, et al., Taverna: lessons in creating a workflow environment for the life sciences, *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 18(10),* pp. 1067-1100, 2006.
37. Open Grid Services Architecture. http://www.globus.org/ogsa/.
38. T. Özsu, P. Valduriez. *Principles of Distributed Database Systems*. 3rd Edition, Prentice Hall, Forthcoming.
39. E. Pacitti, O. Dedieu. Algorithms for Optimistic Replication on the Web. *Journal of the Brazilian Computing Society*, *8(2)*, 2002.
40. S. Ratnasamy et al. A Scalable Content-Addressable Network. *Proc. of SIGCOMM*, 2001.
41. SETI@home. http://www.setiathome.ssl.berkeley.edu/.
42. Y. Saito, M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1):42-81, 2005.
43. G. Singh, C. Kesselman, E. Deelman. Optimizing Grid-Based Workflow Execution. *Journal of Grid Computing*, 3(3-4), 201-219, 2005.
44. I. Stoica et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. of SIGCOMM*, 2001.
45. H. Tangmunarunkit, S. Decker, C. Kesselman. Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web. *Int. Semantic Web Conference (ISWC)*, 2003.
46. I. Tatarinov et al. The Piazza Peer Data Management Project. *SIGMOD Record, 32(3),* 2003.
47. A. Tomasic, L. Raschid, P. Valduriez. Scaling access to heterogeneous data sources with DISCO. *IEEE Trans. on Knowledge and Data Engineering*, 10(5), 808-823, 1998.
48. P. Valduriez: Parallel Database Systems: open problems and new issues. *Int. Journal on Distributed and Parallel Databases*, 1(2), 137-165, 1993.
49. P. Valduriez, E. Pacitti. Data Management in Large-scale P2P Systems. *Int. Conf. on High Performance Computing for Computational Science (VecPar)*, LNCS 3402, 109-122, Springer 2005.
50. W. Van der Aalst, et al., Life After BPEL? LNCS 3670, Springer, 35-50, 2005.
51. M. Wieczorek, R. Prodan, T. Fahringer, Scheduling of Scientific Workflows in the ASKALON Grid Environment, *SIGMOD Record, 34(3)*, pp. 56-62, 2005.
52. K. Yagoub, D. Florescu, V. Issarny, P. Valduriez. Caching Strategies for Data-intensive Web Sites. *Int. Conf. on VLDB*, 2000.
53. J. Yu, R. Buyya, A Taxonomy of Scientific Workflow Systems for Grid Computing, *SIGMOD Record, Special Section on Scientific Workflows, 34(3)*, pp. 44-49, 2005.
54. B. Yang, H. Garcia-Molina. Designing a Super-peer Network. *Int. Conf. on Data Engineering*, 2003.
55. Y. Zhao, J. Dobson, I. Foster, L. Moreau and M. Wilde, A Notation and System for Expressing and Executing Cleanly Typed Workflows on Messy Scientific Data, *SIGMOD Record, 34(3)*, pp. 37-43, 2005.