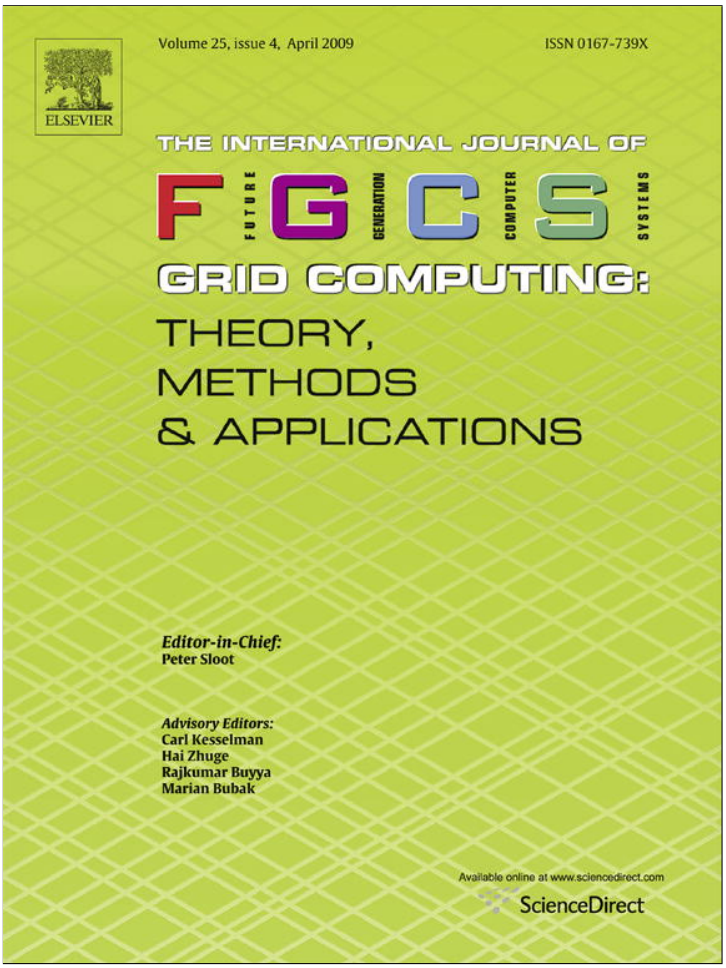

Future Generation Computer Systems 25 (2009) 426–435

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A grid middleware for data management exploiting peer-to-peer techniques☆

Athanasia Asiki*, Katerina Doka, Ioannis Konstantinou, Antonis Zissimos, Dimitrios Tsoumakos, Nectarios Koziris, Panayiotis Tsanakas

Computing Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, Zografou Campus, Zografou 15773, Greece

ARTICLE INFO

Article history:
Received 15 January 2008
Received in revised form
18 August 2008
Accepted 11 September 2008
Available online 30 September 2008

Keywords:
Grid architectures and systems
Peer-to-peer systems
Middleware
Multidimensional indexing
Information storage and retrieval
Data management

ABSTRACT

In this paper, we describe a service-oriented middleware architecture for Grid environments which enables efficient data management. Our design introduces concepts from Peer-to-Peer computing in order to provide a scalable and reliable infrastructure for storage, search and retrieval of annotated content. To ensure fast file lookups in the distributed repositories, our system incorporates a multidimensional indexing scheme which serves the need for supporting both exact match and range queries over a group of metadata attributes. Finally, file transfers are conducted using *GridTorrent*, a grid-enabled, Peer-to-Peer mechanism that performs efficient data transfers by enabling cooperation among participating nodes and balances the cost of file transfer among them. The proposed architecture is the middleware component used by the *GREDIA* project, in which both media and banking partners plan to share large loads of annotated content.

1. Introduction

Due to the explosion of network technologies and the impressive growth in the performance of computing systems, there is an increasing trend to apply high-performance and cluster-based techniques in completely distributed environments. One challenge is to achieve maximum utilization of idle computational cycles. Another challenge is to establish a global-scale data management scheme providing the required protocols and algorithms for sharing, searching and transferring data among geographically distributed resources.

Grid computing [1] focuses on studying distributed infrastructures, where users share geographically distributed resources integrated under a common middleware. It adopts the principles of the software model called *Service Oriented Architecture* (SOA) and provides seamless access to services deployed in a distributed manner. The basic characteristic of these systems is the existence of explicitly defined rules and policies to enable flexible, secure and coordinated resource sharing among dynamic virtual collections of users, named *Virtual Organizations* (hence VOs).

Another form of distributed computing is expressed by *Peer-to-Peer* (hence P2P) *computing*, which deals mainly with the sharing of large amounts of data over the network. Many P2P applications already exist for file sharing among an increasing number of users, gaining interest in the large Internet and scientific community. These applications are characterized by their decentralized nature and their unsupervised operation dynamically adjusting to node arrivals and departures. The above-mentioned features contribute to extensibility, resilience to faults and higher system availability. However, the distributed nature of P2P systems and the lack of centralized structures require an efficient object location mechanism. This mechanism should be capable of adapting to changing network topologies, while maintaining high lookup performance. Thus, much effort has been placed to the development of indexing methods for efficient search mechanisms.

As mentioned above, the peers in a P2P system are organized dynamically and without the existence of centralized structures. Anonymity is also another important issue in such systems as far as the identities of resource providers and information requesters are concerned. These features seem contradictory to the "strict" structure of Grid systems, which takes into account administrative and organizational boundaries. However, there is a growing trend of introducing P2P techniques and algorithms for efficient data management in Grid environments. Existing services, such as the Data Transfer service, responsible for file transfers among nodes (e.g., using the GridFTP protocol) or the Replica Location Service (RLS), keeping track of the physical locations of files, can be designed and developed according to the P2P philosophy.

☆ This work was partly supported by the European Commission in terms of the GREDIA FP6 IST Project (FP6-34363).

* Corresponding author. Tel.: +30 2107722867; fax: +30 210 7721292.
E-mail addresses: nasia@cslab.ece.ntua.gr (A. Asiki), katerina@cslab.ece.ntua.gr (K. Doka), ikons@cslab.ece.ntua.gr (I. Konstantinou), azisi@cslab.ece.ntua.gr (A. Zissimos), dtsouma@cslab.ece.ntua.gr (D. Tsoumakos), nkoziris@cslab.ece.ntua.gr (N. Koziris), panag@cs.ntua.gr (P. Tsanakas).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.09.005

These enhancements lead to more scalable, fault-tolerant and self-organized solutions.

Our target applications require the management of annotated multimedia content. Information from multimedia applications bombards our daily life and is produced by the majority of scientific and business applications. Large amounts of audiovisual data are becoming available on the World Wide Web, in broadcast data streams, in personal and business databases. However, multimedia files contain a lot of information that is difficult to organize. Thus, their utility depends on the existence of efficient mechanisms for discovery, filtering and managing this type of content.

In this paper, we present a service-oriented middleware architecture for data management in Grid environments. The provided services enable the efficient search, discovery and transfer of annotated multimedia content. The annotations are either provided by the user or generated automatically by the system.

Searches for the stored content are supported at two different levels. At the first level, a user can perform advanced searches over the annotations based on a predefined metadata schema. At the second level, a data item can be searched according to its unique identifier, which is stored in a distributed catalogue, containing mappings of unique identifiers of files to their physical locations. The search facilities are accommodated by P2P overlays using Distributed Hash Tables (hence DHTs). Lookup operations for specific data items benefit from the structure of DHT systems. However, their utilization is more complicated and costly as far as queries for range values and multiple attributes are concerned, in terms of messages and maintenance effort required by additional indexing mechanisms. To support the processing of complex queries, we have proceeded in modifications of the placement of data and the query routing in DHTs.

The proposed data transfer mechanism combines the cooperative sharing enforced in the BitTorrent protocol [2,3] with the GridFTP [4] protocol used in Grid environments. Despite improvements in the GridFTP protocol, it remains a centralized mechanism, unable to cope with flash crowd situations, when the number of potential clients and the volume of data increase. The allocated bandwidth to each data transfer in progress depends on the number of active connections in the GridFTP server, which becomes the bottleneck of the system. This transfer mechanism cannot take advantage of the already transferred data pieces among the participating nodes. In that case, nodes having completed the download of a data piece are unable to serve this piece to other nodes asking for it. The proposed scheme is based on the BitTorrent protocol, allowing the concurrent download of file pieces from several storage nodes while uploading the already downloaded ones as well. All participants cooperate in sharing data already transferred to them and aggregating their bandwidth optimizing the overall transfer rate.

Our design is completely decentralized, requiring no form of centralized communication among different types of services. Our middleware provides transparent services for efficient data search, discovery and transfer of annotated content. Our interest in the proposed architecture is based on the belief that it would provide users manipulating such content with the necessary services to build a promising Grid infrastructure, while it can be integrated with other systems and existing middlewares.

The rest of the paper is organized as follows: Section 3 addresses some general issues and requirements regarding the design of a large-scale system intending to manage annotated content. Section 4 presents an overview of the proposed architecture. Section 5 introduces a multidimensional scheme for metadata search. In Section 6, the implementation of a distributed DHT-based catalogue is analyzed. Section 7 describes a data transfer mechanism based on the BitTorrent protocol. Section 8 refers to the sequence of logical operations during insertion and search of data. Finally, Section 9 refers to some issues occurring during the implementation phase and Section 10 contains concluding remarks as well as outline directions for future research, involving possible extensions of our work.

2. Related work

Data Grids are wide-area distributed infrastructures of heterogeneous resources capable of managing immense amounts of data. The core services for accessing heterogeneous storage resources, storing, transferring and searching large datasets are described in [5].

A fundamental building block of the Data Grid architecture is the data transfer mechanism among storage nodes. The established protocol is *GridFTP* [4], a protocol defined by the Global Grid Forum and adopted by the majority of the existing middlewares. GridFTP extends the standard FTP protocol including features like *Grid Security Infrastructure* (GSI) [6] along with third-party control and data channel. A more distributed approach of the GridFTP service attracted the attention of the Grid community leading to the *Globus Stripped GridFTP* protocol [7], included in the current release of Globus Toolkit 4 [8]. The new features added to the GridFTP protocol support transfers of data striped or interleaved across multiple servers, partial file transfers and parallel data transfers using multiple TCP streams. However, the GridFTP protocol remains a centralized mechanism accommodating only server–client data transfers. This fact implies that a limit exists for concurrent transfers and GridFTP is unable to cope with flash crowd situations.

In Data Grids, the availability of data and the efficiency of data transfers depend on the number of existing file copies. These copies are called replicas and are distributed among resources. The physical instances of a file are located by the *Replica Location Service*, which interacts with a *Replica Catalogue* containing mappings between *Logical FileNames* (LFNs) and *Physical FileNames* (PFNs). The main drawback of the initially implemented RLS architecture was its centralized structure, which posed limitations to the scalability and the resilience of the system. Efforts on distributing the catalogue were the consequent step. They resulted in *Giggle* (Giga-scale global location engine) *Framework* [9,10], the most widespread solution currently deployed on the Grid concerning the management of replicas. It deploys an RLS architecture consisting of several global and local services in a multi-tier hierarchy. Giggle utilizes *Local Replica Catalogues* (LRCs), maintaining mainly mappings of LFNs to PFNs across a site, and global *Replica Location Indices* (RLIs) maintaining information about the catalogues and the associated LFNs. However, the updates in the LRCs induce a complex and bandwidth-consuming communication scheme between LRCs and RLIs.

The existence of efficient Metadata services plays an important role in data publishing and searching. In [11], Deelman et al. emphasize the need for services responsible for handling metadata descriptions of data objects. Their claim is being justified by real use cases from the scientific community. They introduce a *Metadata Catalogue Service* (MCS) implemented on top of a database. The scalability of the system relies on the extension of OGSA-DAI [12] to interact with the MCS. Nevertheless, the centralized metadata servers remain the bottleneck of the system, when the number of metadata and performed searches increase.

On the other hand, the P2P approach provides a scalable alternative solution to the conventional server–client approach. Depending on their structure, P2P systems are divided into three categories, namely structured, unstructured [13,14] and hybrid [15,16]. The model followed to build a structured overlay is the construction of a *Distributed Hash Table* (DHT), as in overlays

of Chord [17], Pastry [18], Kademlia [19], CAN [20], etc. Data items and nodes are assigned a unique identifier (ID) called *key*, which is usually generated by a hash function applied over their contents and addresses respectively. Keys are used during insertion and lookup of data items in the DHT overlay. During an insertion, a data item is forwarded among the nodes of the overlay, so as to be stored in the node with the closest ID to its key. The notion of distance is defined by a metric specified by the P2P protocol. The main advantage of DHT-based systems is that if a key exists in the overlay, their lookup algorithm guarantees to find it. The lookup cost is bounded to the logarithm of the search space, namely the number of nodes participating in the overlay.

An approach followed to predetermine data placement is the utilization of *Locality Preserving Hash Functions*. The generation of IDs based on *Space Filling Curves* (SFCs) falls into this category, since indexing methods based on SFCs map points from a multidimensional space into one dimension, while they tend to preserve locality. In Squid [21] and SCRAP [22], the set of attributes describing data items are considered to form a d-dimensional space. Each combination of their values represents a point in this space. The key to be used in the DHT is produced by the mapping of the point in a single dimension based on an SFC, such as the Hilbert curve or the Z-curve. Multidimensional indexing with SFCs is introduced in the CISS framework [23] as well. The Hilbert curve has been chosen by the authors due to its clustering properties. The critical point in using an SFC-based as a hash function is the consequences regarding the load balancing of the system.

In traditional DHTs, keys are produced randomly by a cryptographic hash function and thus distributed uniformly to all nodes. SFCs preserve locality by placing in close nodes data with similar attributes, against the uniform distribution of load among nodes. To avoid imbalance, Squid relies on the fact that the d-dimensional keyword space is sparse and so the data items are assigned to peers roughly in the same way. SCRAP faces this problem with Skip Graphs [24], a popular solution for range queries in DHTs. The authors of CISS propose that nodes which are heavily loaded can deal with the problem locally by giving part of their key ranges to their neighbors or globally by finding a lightly loaded node in the system.

Towards the efficient processing of complex and range queries, additional indexing mechanisms have been proposed. These structures are usually distributed trees formed with indices maintained among the nodes of the overlay. This is the case for the distributed *tries* introduced in [25–27]. Tries are prefix trees suitable for storing and processing strings. They are used for searching key ranges according to their common prefix. The *Prefix Hash Tree*, [25] is a distributed trie indexing binary strings based on their common prefix. The entire keys are stored in leaf nodes, which are mapped onto peers of an underlying DHT. The *P-Grid* [26] incorporates the trie structure in the routing mechanism of the proposed overlay. The *Distributed Lexical Placement Table* (DLPT) [27] is a dynamically constructed trie for indexing, enabling service discovery. Another similar approach for supporting complex queries in a large-scale distributed environment is the IMAGINE-P2P platform [28], which forwards the queries to be processed along semantic paths of an index. A tree index is constructed on top of a semantic overlay which contains semantic relationships among data items stored in a Chord DHT.

2.1. The Kademlia DHT overlay

The DHT-based overlays in our architecture are implemented with a modified version of the Kademlia protocol [19]. Both data items and nodes are assigned unique identifiers from a unified address space. Items to be inserted in the DHT are *(key, value)* pairs, where the value is the data item to be stored to the node with the closest ID to the key. The notion of distance between points in the identifier space is defined by the XOR metric. The Kademlia protocol locates data items based on their keys, only if these keys are known in advance. The query messages are routed in the overlay according to the information that each node maintains for other peers. This information is acquired by the messages that a peer receives. The symmetric feature of the XOR metric allows Kademlia participants to receive lookup queries from roughly the same distribution of nodes in their routing tables. Kademlia appears as an appropriate selection due to its simple routing table structure and its consistent algorithm throughout the lookup procedure.

The basic RPCs of the Kademlia protocol are:

FIND_NODE: This operation returns the *kappa* closest nodes to the target ID that the recipient node is aware of, where *kappa* is a system parameter.

FIND_VALUE: When a node of the Kademlia network is instructed to lookup a value in the network, it issues α parallel queries (FIND_VALUE) to the closest nodes it is aware of. A reply will be the value or routing information in the form of even closer nodes to the target key. The process continues until either the value or the *kappa* closest nodes to the target key are found. The system wide parameter *kappa* also specifies the number of copies maintained for each data item, and controls the size of routing tables in peers. Both α and *kappa* variables are set at each participant node and affect only local service performance.

STORE: At first, the node initializing the STORE operation calculates the key for this data item, usually a hash over its content. Then, it searches the network in several steps, until the closest nodes to that key are discovered. Afterwards, the data item is stored in these nodes.

PING: This RPC probes a node to see if it is online.

3. Challenges and requirements

The design of a generic middleware for efficient search and retrieval of annotated content in a distributed Grid environment has been largely motivated by the requirements posed by the GREDIA research project [29], supported by the European Committee. GREDIA's main objective is the development of a Grid application platform, providing high level support for the implementation of Grid business applications through a flexible graphical user interface. This generic platform will facilitate the provision of business services, which mainly demand access and sharing of large quantities of distributed annotated numerical and multimedia content. Furthermore, the GREDIA platform will exploit Grid technologies to enable access to the distributed content by mobile devices. A user may access the platform in order to upload content or perform searches.

The design and the implementation of such systems face several challenges. The most crucial one is the development of a distributed architecture for managing large amounts of data stored in geographically distributed resources. The efficiency of the system should not decline when the amount of stored data and the number of performed operations increase. The system should scale well, providing reliable and concrete services.

Regarding the publishing and discovery of content, the users of the system should be able to perform both exact match and range queries. The answer to a range query concerns a group of data having a common characteristic. For example, a user may ask for a data item specifying its author, subject and requiring its creation to be within a specific time interval and results satisfying this criterion should be returned. However, it is difficult to handle range

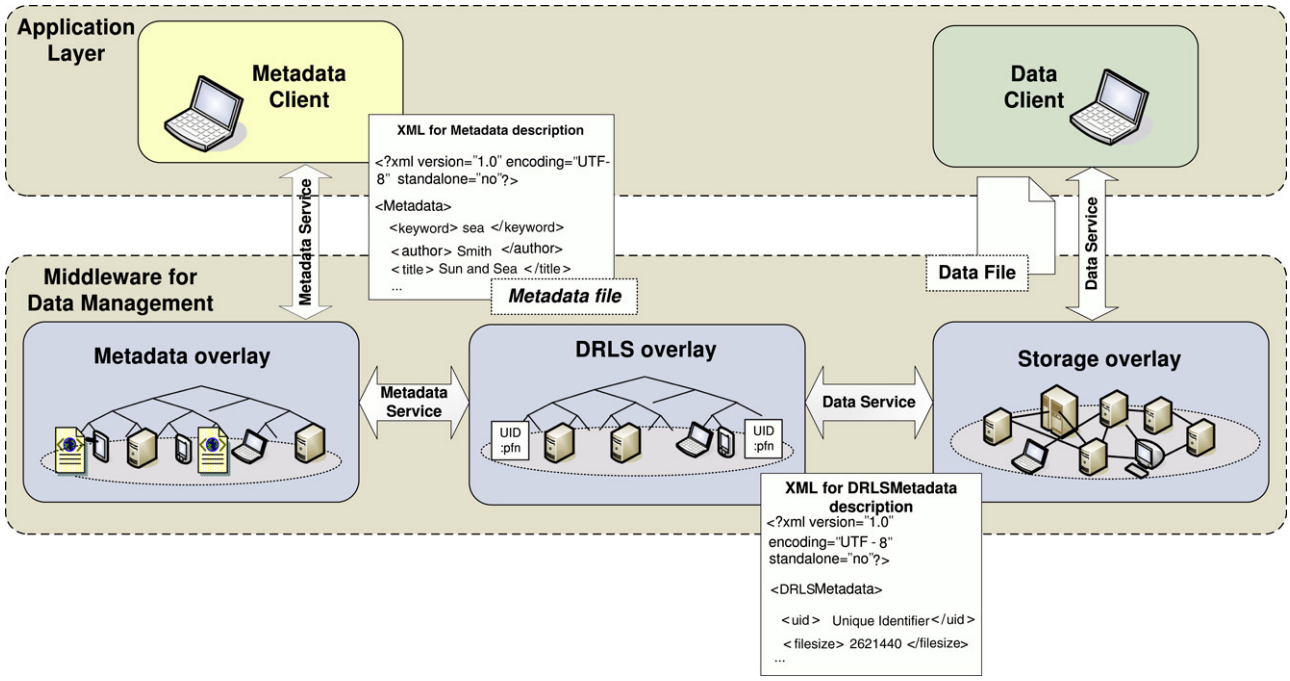

Fig. 1. Overview of the proposed architecture.

queries without centralized catalogues for indexing. Provisions should be made so that range queries are processed by a small number of nodes.

Finally, the deployment of a unified framework accessing heterogeneous resources and transferring large amounts of data is very challenging. Bandwidth limitations should be considered, especially when a node has to serve multiple concurrent requests for popular files.

4. Architecture

The proposed architecture deals with data management in a distributed environment consisting of resources belonging to different Virtual Organizations. A main characteristic of this environment is the heterogeneity of resources in terms of computational power, storage capacity and bandwidth. Resources with different features, for example laptops, desktop computers, dedicated servers or even mobile phones, may participate in this platform. Moreover, we assume that some resources may remain off the system for long periods of time. Therefore, the design should cope with node arrivals and departures to provide a robust operation.

The proposed architecture is layered and comprises of three different overlays, as shown in Fig. 1. These overlays are:

- The ***Storage overlay***, where the multimedia content is stored and the data transfer mechanism is implemented. The nodes in the *Storage* overlay act as file servers and should provide the corresponding services persistently.
- The ***Metadata overlay***, where the metadata description of each file is stored and retrieved. This overlay provides a powerful search mechanism supporting not only exact match queries but range and complex queries as well.
- The ***Distributed Replica Location Service overlay*** (hence *DRLS*), which maintains a list of useful information needed during data transfers in the *Storage* overlay. This overlay also acts as a link to correlate metadata descriptions to their corresponding data. It implements a distributed catalogue containing mappings of *Unique IDentifiers* (UIDs) to the physical locations of data files represented by *Physical FileNames* (PFNs).

Each node of the platform is allowed to participate in more than one overlay depending on its available storage space and the services it can host. The *Metadata* and the *DRLS* overlays are implemented according to the Kademlia protocol. Nodes with close IDs in the ID space are considered neighbors without being physically close as well. As shown in Fig. 1, each overlay interacts with the corresponding clients and the other overlays through *Grid services*. These services are the *Data service* and the *Metadata service*, deployed in the nodes of the *Storage* and *Metadata* overlays respectively.

Multiple overlays introduce extensibility and robustness to the system. The implemented services of each overlay are autonomous entities. The goal of our design is to provide portable and flexible services, which can be used either to accomplish a specific purpose or integrated in a more generic middleware. The interaction with other services occurs through well-defined interfaces. The execution of the services is transparent to the user, who only needs to know an endpoint to the provided service and not its various instances.

The data items handled in our system can be divided into two categories, namely the *data files* and the *metadata files*. A data file contains the actual content, which is stored and replicated in the *Storage* overlay. Each data file is described with attributes of a predefined metadata schema and their values are included in its metadata file. The metadata files are stored in the *Metadata* overlay, in different nodes from their data files. The link among these files is the UID attribute. The value of this attribute is the hash of the data file's content and is included in its metadata file. The *DRLS* overlay plays an important role, since it contains the mappings of UIDs to the physical locations of data files. Given a metadata file, a lookup operation for its UID in the *DRLS* overlay should be made in order for the corresponding data file to be found. The *DRLS* DHT stores (*key, value*) pairs, where the key is the UID of the data file. The value is an XML file containing mainly the physical locations of replicas.

These overlays store data belonging to users of different Virtual Organizations (VOs). We assume that the authorization process is carried out by another middleware component, which is beyond the scope of the described middleware services. This component rephrases the user's query according to his/her VO membership.

The result is that a data file can be stored or searched in a storage node, only if the user is granted the appropriate permissions in the node that his/her request arrives.

5. Exploiting a multi-dimensional indexing scheme in the metadata overlay

The Metadata service hosted by the *Metadata* overlay provides a search mechanism for the stored annotations of the multimedia content. Each data file is described by a set of metadata attributes. These attributes follow a predefined Metadata schema designed according to the needs of the users. The annotations are included in the metadata files. For example, an image can be tagged with attributes such as title, location, topic, keywords, creator name, duration, date, format, size. Some of these attributes (size, format, type, date) can be automatically completed by the Metadata client. The rest of them are filled in by the owner of the image or other authorized users through a Web form. We consider that only the most important attributes of the defined schema are indexed, in order to reduce the complexity of the indexing method and limit the search space for faster results.

Our approach to support multiattribute queries in the DHT is based on properties of SFCs. An SFC continuously maps a compact interval to a d-dimensional space. SFCs preserve locality, so that close points in the one-dimensional space are mapped to close points in the d-dimensional space.

The SFCs are utilized in the *Metadata* overlay in order to influence the data placement in the DHT without changing the routing algorithm of the Kademlia protocol. Our goal is metadata files with relative attributes to end up, with higher probability, in nodes with close IDs, so as to reduce flooding over the network for range queries.

In our multidimensional indexing scheme, we consider that the set of d attributes to be indexed forms a d-dimensional space. Each point in this space represents a combination of values for these indexed attributes. The points of the d-dimensional space are mapped down to a single dimension by the SFC component, which utilizes the Hilbert curve or the Z-curve. The result is the partitioning of the d-dimensional space into cells, which in turn are assigned an integer and mapped to points on a single dimension.

A basic property of SFCs is their recursive generation. The number of recursions is indicated by the factor k called the approximation order. This factor defines the number of space partitions and thus the precision of the algorithm. For example, in the Hilbert SFC case the d-dimensional space is subdivided into 2^d cells filled in by the First Order Curve initially, where k is considered to be equal to one. In the next recursion, each cell is subdivided 2^d more times and is filled in by the Second Order Curve. The same procedure is repeated until the k-th Order Curve is generated.

The derived values in the single dimension represent the keyset of the Kademlia-based DHT overlay. Each key is kd bits long and each node in the overlay manages data mainly in contiguous ranges of the SFC. We decided that the most appropriate SFC to consider for our case is the Hilbert SFC, since it appears to present the best clustering properties [30,31]. The Hilbert SFC can be approximated in a higher order by a combination of First Order Curves appropriately oriented.

The SFC component executes the following operations:

Find_SFCID: Maps the coordinates of a point in the d-dimensional space into its position in the SFC.

Find_Coordinates: Maps a point in the SFC to its coordinates in the d-dimensional space.

Traverse_SFC: Produces the SFC recursively enumerating the points of the d-dimensional space by transversing the SFC in order to find adjacent points.

The mapping of a point in the d-dimensional space to its position in the SFC and vice versa is not simple and the difficulty increases analogously to the number of dimensions. The generation of the SFC can be performed non-recursively by using circular shift and exclusive-or operations on bytes according to the Butz algorithm [32]. Based on this algorithm, we have implemented the mapping procedures from the d-dimensional space to one dimension and vice versa. The traversal of the SFC is done by generating the curve recursively.

The search of a metadata file requires knowing the exact key used during the insertion of the metadata file in the DHT. This knowledge is acquired by using the SFC component. Therefore, the values of the indexed attributes for the searched metadata file are given as input to the algorithm that maps them to the corresponding key. The output is the SFC ID which is the key used during the insertion phase of the specific metadata file. The insertion of metadata file is described in Algorithm 1. The node instructed to lookup the specific metadata file using its key in the network issues α parallel queries to the closest nodes it is aware of. A reply will be the metadata file or routing information in the form of even closer nodes to the respective key. The node continues the process until the metadata file is found or all *kappa* closest nodes to the key are contacted.

Algorithm 1 Insert Metadata File

AttrsToIndex $\Leftarrow$ *Parse*(*metadatafile*)
key $\Leftarrow$ *Find_SFCID*(*AttrsToIndex*)
Add UID to the metadata file
STORE(*key*, *metadatafile*)

The answer to a query might be a metadata file with a specific key or metadata files with keys inside one or more key ranges, depending on the type of the query. A single key is used to lookup a metadata file for an exact match query, if the user searches for a specific combination of the indexed attributes. The processing of a non exact match query consists of two consecutive phases. In the first phase, clusters of SFC points corresponding to keys of metadata files answering the query are determined. In the next phase, lookup operations for these cluster(s) start. Lookups for key ranges require the modification of the Kademlia protocol and mainly of the FIND_VALUE and the FIND_NODE operations. Assuming that churn will not be an issue for our target applications, each node is aware of the next node with the closest ID in the network and maintains an index towards it. Each time that a query for a key range arrives at a node, it scans its keys and replies with metadata files inserted with keys included in the key range. If the node is responsible only for a subrange, then it forwards the query to the next node. The forwarding of the query continues until objects within the key range are retrieved. The steps of the described search procedure are presented in Algorithm 2.

The number of bits used for value encoding in each dimension is equal to the approximation factor. We consider that there are three categories of attributes to be indexed: numerical, categorical and string attributes. For numerical types such as date, size, duration, we use their bit representations as input to the SFC component. The categorical attributes can be easily encoded due to the fact that the number of different values is limited. For string attributes, we use a hash function to map them into the available value ranges. A special case of string attributes is keywords. Our goal is to allow a user to describe a file with keywords without restricting their number. Moreover, keyword searches are the most popular in such platforms and the search mechanism should provide fast replies. For these reasons, we have decided not to include the keywords in the dimensions of the SFC scheme. Each keyword is hashed and inserted in the *Metadata* overlay separately, as the key of a

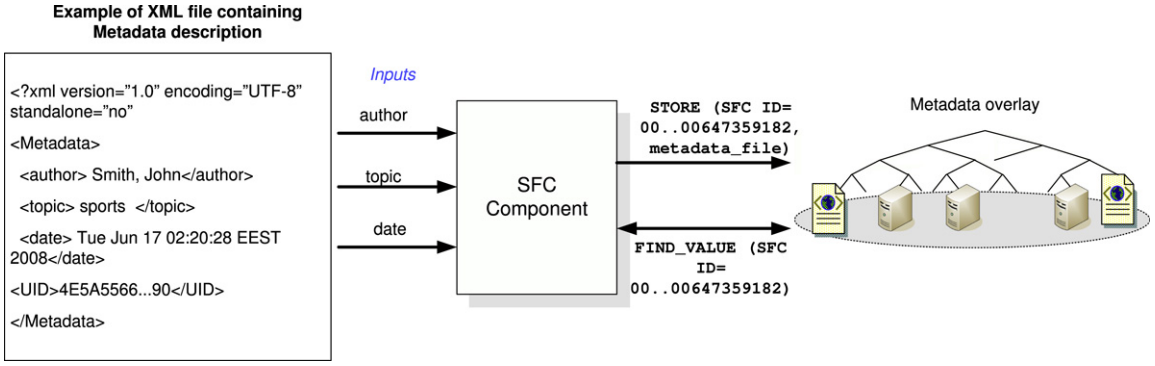

Fig. 2. Inserting and searching a metadata file in the *Metadata* overlay.

Algorithm 2 Search

```
if query is a exact match query then
  key ⇐ Find_SFCID(Indexed_Attr)
  lookup key
else
  find key_ranges
  for each key_range do
    node ⇐ lookup(key_range)
    search(node, key_range)
    if node is NOT responsible for the whole range then
      forward the query to the next node and repeat if it is
      needed until the range is covered
    end if
  end for
end if
```

(*key, value*) pair, where the value is the SFC ID produced by the SFC component. Therefore, queries for keywords can be answered directly with simple lookup operations in the DHT.

An example of insertion and search of a metadata file is shown in Fig. 2. The XML file contains a metadata description with three attributes to be indexed. The author is a string attribute, while the topic is considered to be categorical. The date is parsed, so that its numerical representation is to be used. The XML file is enhanced with the UID of the data file as well. The values of the three attributes to be indexed are given as input to the SFC component. The produced SFC ID is used as key to store the XML file in the proper node of the *Metadata* overlay. In order to search the specific metadata file, the values of the author, topic and date attributes should be given to the SFC component. The output is the SFC ID of the metadata file. The lookup operation of the Kademlia protocol is used for this SFC ID, so that the metadata file is to be found.

In DHT-based systems, the even distribution of load among nodes is critical for the performance of the system. Traditional DHT systems deal with this problem by using a hash function in order to achieve random generation of keys. The random hashing for node IDs means that each node is responsible for just a small interval of the address space (between itself and the next node), while the random mapping of items means that roughly the same number of items get IDs belonging to the interval of the address space owned by each node. On the other hand, the multidimensional indexing based on SFCs aims to preserve locality. This feature and the heterogeneity of nodes in terms of storage capacity and bandwidth have negative impact on the load distribution. To deal with this problem, we intend to implement a solution based on virtual servers, proposed in [33]. A virtual server acts like a physical node: it is responsible for a contiguous portion of the DHT's identifier space, thus for all data items whose IDs fall into that portion. Moreover, it is a single entity, meaning it has its own routing table. A physical node can host multiple virtual servers, thus it *owns* multiple, non-contiguous intervals of the identifier space. Load balancing will be achieved by physical nodes exchanging virtual servers in order to lighten their burden. When a physical node is overloaded by virtue of available storage space or bandwidth, it may move one or more of its virtual servers to another, underloaded physical node.

6. A distributed replica location service

The DRLS overlay implements a *Distributed Replica Location Service* [34] using a DHT by correlating its inherent (*key, value*) pairs to (UID, XML with PFNs) mappings. Every data file is assigned with a UID, which is considered to uniquely identify the file in the system. The UID is included in the metadata file as well and links the metadata description to the actual data.

The main problem associated with the usage of a DHT to store replica locations lies in the difficulty of the P2P network to handle mutable data. Each (*key, value*) pair is stored to nodes with IDs close to the keys and cached around the network. Therefore, it is difficult to ensure in a given moment that all replicas have been informed of the existence of a new version during an update procedure. While this may seem sufficient for storing read-only files, it is not adequate to serve the needs of a Replica Location Service in a grid environment. Update operations are necessary for storing replica locations, since the PFNs of files could change frequently. For this reason, there should be a way for propagating the modifications throughout the network as soon as possible.

In order to establish a scheme that enables the handling of mutable data, we take into account the fact that peers of a DHT may distribute data in numerous peers of the system. There is a significant probability that upon subsequent queries for the same key, at least one of the updated ones will be contacted, if a value is changed in one of these nodes. Nevertheless, node arrivals and departures may result in unpredictable differentiations in storage relationships between data items. As a consequence, lookups should always query all nodes responsible for a specific (*key, value*) pair, compare the results based on some predefined version vector (indicating the latest update of the value) and propagate the changes to the found nodes, which have not been up-to-date with the latest value yet. This requires that the algorithm for locating data items does not halt when the first value is returned, but continues until all available versions of the pair are returned. The querying node will then decide which version to keep and send corresponding store messages back to the peers that seem to hold older or invalid values. Updates can therefore be implemented through the predefined set operation and version checking can also be done by nodes receiving store commands. The latter should check their local storage repositories for an already present (*key, value*) pair and keep the latest version of the two values in case of a conflict. The version of a (*key, value*) pair is determined by a timestamp indicator.

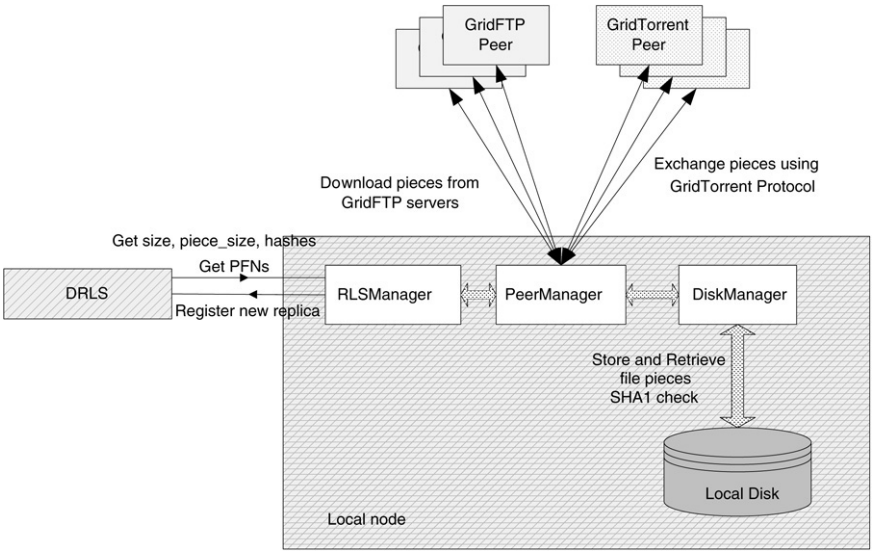

Fig. 3. GridTorrent architecture.

Our modified lookup algorithm works in a way similar to the FIND_NODE loop of Kademlia, originally used for storing values in the network. We first find all closest nodes to the requested (*key, value*) pair, through FIND_NODE RPCs, and then send them FIND_VALUE messages. The querying node checks all values returned, finds the most recent version and notifies the nodes having stale copies of the change. Of course, if a peer replies to the FIND_VALUE RPC with a list of nodes it is marked as not up to date. When the top *kappa* nodes have returned a result (either a value or a list of nodes), we send the appropriate STORE RPCs. Nodes receiving a STORE command should replace their local copy of the (*key, value*) pair with its updated version. Storing a new key in the system is done exactly in the same way, with the only difference that the latest version of the data item is provided by the user. Moreover, deleting a value equals updating it to zero length. Deleted data will eventually be removed from the system when it expires. Updating the mappings is an atomic operation, i.e. it happens immediately after the responsible peer receives the original update. Thus, there exists only a single version each time.

7. GridTorrent: A data transfer mechanism for the storage overlay

The purpose of this layer is to provide a data transfer mechanism that effectively deals with large and concurrent file uploads and downloads, even when numerous requests rely on a single data source, maximizing bandwidth utilization. The proposed solution, *GridTorrent* [35], constitutes a decentralized approach, that, unlike GridFTP, takes advantage of multiple replicas to boost aggregate transfer throughput.

GridTorrent is an implementation of the popular BitTorrent protocol designed to interface and integrate with well-defined and deployed Data Grid components and protocols (e.g. GridFTP, RLS). Just like BitTorrent, GridTorrent is based on P2P techniques, that allow clients to download files from multiple sources while uploading them to other users at the same time, rather than obtaining them from a central server. By dividing files into fragments, GridTorrent can combine the best out of the two protocols:

- It exploits BitTorrent's peer and fragment selection, thus providing an optimized data transfer service.
- It takes advantage of the striped version of GridFTP protocol, as it has the ability to directly communicate with GridFTP servers, thus being backwards compatible.

In short, GridTorrent works as follows: A request to GridTorrent for a file triggers a query to the *DRLS* overlay. This procedure is repeated periodically, in order to detect any changes in the locations of file replicas or of any joins or departures of nodes. The file can be located in GridFTP servers or GridTorrent peers. GridTorrent nodes can be classified into leechers still downloading pieces and seeds, meaning peers having the whole file. Upon receiving the list of peers, GridTorrent acts according to the protocol prefix of the PFN. If it concerns a GridTorrent client, the two involved peers initiate communication by exchanging the BitTorrent *bit field* message, informing each other of the pieces they possess. Furthermore, each time a peer downloads a piece, it sends a *have* message notifying all peers connected to it of its new acquisition. In order to download data from another GridTorrent client, the peer issues a *request* message for blocks. Blocks are parts of a piece, referenced by the piece index, a zero-based byte offset within the piece and their length. Having information about the available pieces, GridTorrent starts downloading pieces following a rarest-first policy. In case of a GridFTP server, the peer does not need to exchange bit field messages. As for the downloading technique, the client issues a GridFTP partial get message for the data within the specific block it intends to download.

As shown in Fig. 3, GridTorrent consists of the following components:

- The RLSManager, a component that communicates with the *DRLS* overlay to acquire the DRLS Metadata description containing the size of each file fragment, the hash for every piece, used for integrity reasons imposed by this extended fragmentation and the physical locations of the various replicas. The physical locations are identified by a GridTorrent URL, a unique identifier of the following form, so as to preserve the backwards compatibility with existing Grid transfer mechanisms: gtp://site.fqdn/path/to/file.
- The PeerManager, which handles all the communication with other GridTorrent or GridFTP enabled peers.
- The DiskManager, which manages all disk I/O for storing and retrieving files. The DiskManager is also responsible for verifying the correctness of the file by comparing the SHA1 of each downloaded piece against the SHA1 provided by the DRLS.

Although GridTorrent is evolving work, preliminary experimental results indicate notable improvement in the average transfer rate for large data files in WAN network conditions. Fig. 4 summarizes these results, presenting the minimum, maximum and average completion time in seconds when distributing a file to a constant set of nodes. More precisely, having a GridFTP server running on a node, we invoked concurrent file transfers to 16 client nodes of our testbed, first using GridFTP and then using GridTorrent, and measured the total completion time. This experiment was conducted for file sizes varying from 16 to 512 MB.

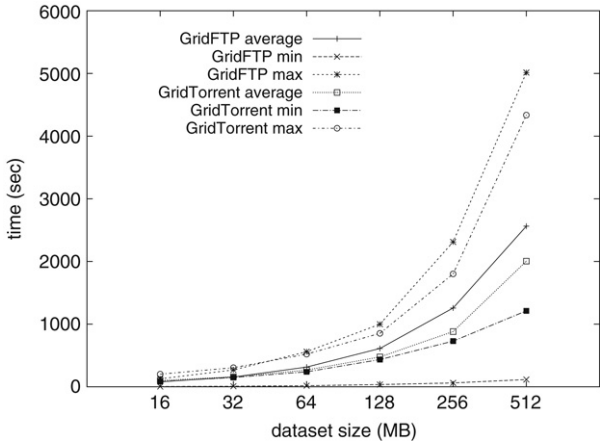

Fig. 4. Minimum, maximum and average time required for 16 independent downloaders to transfer a file using either GridFTP or GridTorrent.

8. Description of insertion and search

So far, we have described a grid middleware for storage, search and retrieval of data. In Section 4, the overall architecture along with the implemented services have been presented. The exact procedure for *insert* and *search* operations as well as the interaction among the various components are shown in Fig. 5(a) and (b).

Insertion: The Data Client receives a data file and generates its UID. Then, the UID is added in the metadata file among the rest of the description provided by the user. The Data Client sends the data file to the Data Service, which uploads it in the *Storage* overlay using the GridTorrent mechanism. If the user's node participates in the *Storage* overlay, the file is stored locally as well. The PFNs of the nodes, where the file has been successfully uploaded, are returned to the Data Service. Afterwards, the Data Service creates the XML file for the *DRLS* Metadata description including these PFNs and inserts it in the *DRLS*. The Metadata Client receives the metadata file along with the UID of the data file and passes the metadata file through the SFC component to generate its SFC ID. Finally, the metadata file is stored and replicated in the *Metadata* overlay. The described procedure along with the succession of events for inserting a new file is shown in Fig. 5(a).

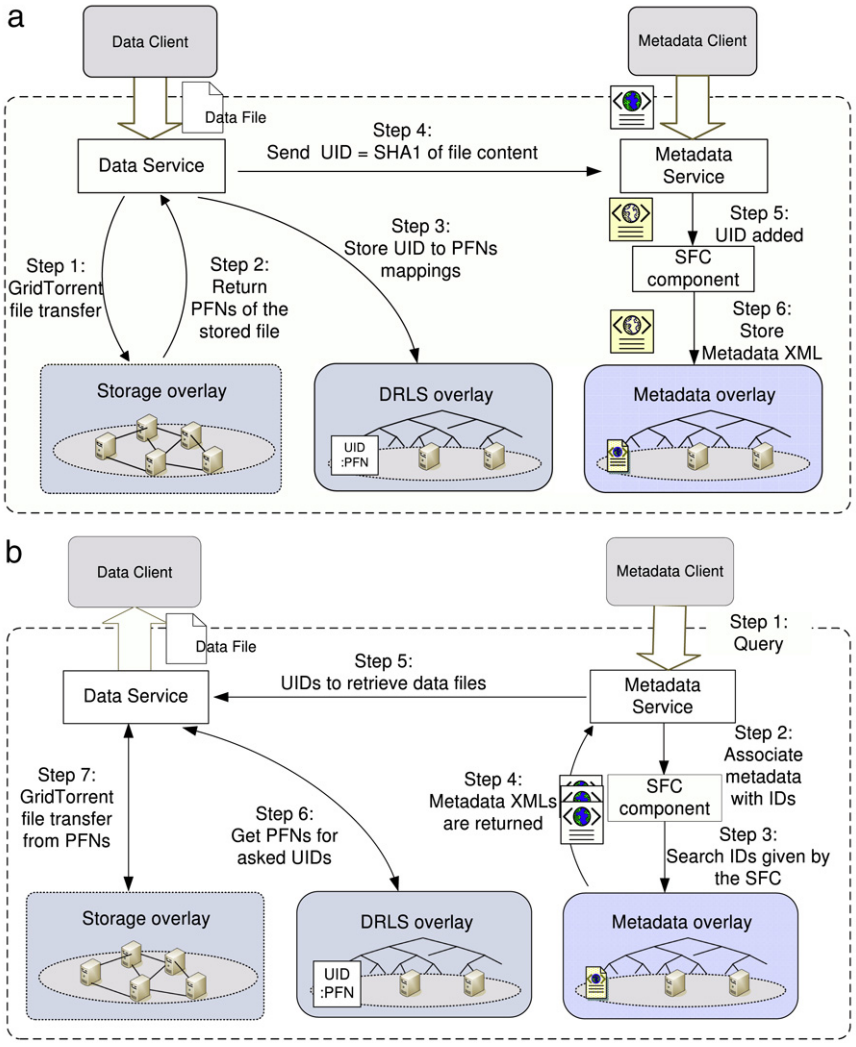

Fig. 5. (a) Logical operations occurring when inserting a new file. (b) Logical operations occurring when retrieving a file using user-defined metadata attributes.

Search: The Metadata service provides a search mechanism for complex queries in the indexed attributes of the predefined metadata schema. When a user aims to retrieve a data file, the succession of events is depicted in Fig. 5(b). The SFC component produces the SFC IDs for the metadata files, which reflect the user's requirements. In case of a query based on keywords, a lookup operation for the hashed value of each keyword is performed. This operation returns the SFC IDs of the metadata files with this keyword. The returned SFC IDs are compared and only the common ones for all keywords are searched in the *Metadata* overlay. The retrieved (if any) XML files are returned to the Metadata Service and presented to the Metadata Client according to their relevance with the query. The user chooses the files to download and their UIDs are exported from the corresponding metadata files. *DRLS* is queried using these UIDs, thus producing to the Data Service a list of PFNs in order to get the file(s) from. Finally, the GridTorrent file transfer module downloads the selected file(s) which can be locally saved.

9. Implementation issues

As noted above, our platform intends to host large-scale data management applications, which will be developed on top of the middleware layer. We are currently working on the implementation of the prototype middleware according to the described architecture. The proposed services are developed as *Grid services* according to the Open Grid Service Architecture (OGSA) [36], using the libraries of Globus Toolkit 4 (GT4) [8]. One of the essential requirements of OGSA is the implementation of the underlying software with stateful services. For this reason, all services in the different overlays of our architecture are implemented according to the Web Service Resource Framework (WSRF) and communicate with each other through well-defined interfaces. What is more, the services are compatible with the components of the *Grid Security Infrastructure* (GSI) for reasons related to authentication, secure communication and the application of uniform policies across the infrastructure. All services are implemented in Java and we aim to ensure their portability and their deployment to be independent from the underlying platform.

The metadata description that accompanies each data item is included in an XML file and is validated by an XSD schema, defined by the application's requirements. The attributes describing the data are chosen according to the needs of the specific community and common acceptable standards. A use case designed for this platform is a geographically distributed repository for journalists. In this case, the data items are multimedia files, thus the metadata schema is based on the MPEG-7 standard [37].

10. Conclusion and future work

This paper addresses issues on efficient sharing and management of annotated content among distributed heterogeneous resources. In this context, we proposed a service-oriented middleware architecture, that provides store, search and retrieve primitives for manipulation of data. We introduced the idea of DHT overlays for metadata and data management, thus avoiding the use of centralized entities. Moreover, a multidimensional indexing scheme, that speeds up the searching procedure and supports multiattributes and range queries has been described. Finally, we presented GridTorrent, a transfer protocol resilient to flash crowd conditions and completely compatible with existing grid middleware. We believe that the proposed architecture, where various P2P techniques have been applied, provides a robust and scalable infrastructure for storing annotated data and searching over a large set of metadata descriptions.

The implementation of the described platform is currently at a prototype status. The functionality of basic components has been predetermined and the corresponding interfaces have been deployed. However, we explore optimization issues in various aspects of our design. We are dealing with the challenge of load imbalance, induced by the heterogeneity of nodes in terms of storage capacity and bandwidth and the property of SFCs to preserve locality. Towards this direction, we plan to elaborate on the idea of virtual servers proposed in [33]. Finally, the clustering of SFC IDs produced by the SFC component and their routing in the *Metadata* overlay for more efficient query processing are also under consideration.

References

[1] I. Foster, C. Kesselman (Eds.), The grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
[2] BitTorrent.org, URL: http://www.bittorrent.org/index.html.
[3] B. Cohen, Incentives build robustness in BitTorrent, in: Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, 2003.
[4] B. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, Data management and transfer in high-performance computational grid environments, Parallel Computing 28 (5) (2002) 749–771.
[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, The data grid: Towards an architecture for the distributed management and analysis of large scale scientific datasets, Journal of Network and Computer Applications 23 (3) (2000) 187–200.
[6] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, A security architecture for computational grids, in: Proc. of the 5th ACM Conference on Computer and Communications Security, 1998, pp. 83–92.
[7] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, The globus striped GridFTP framework and server, in: Proc. of the 2005 ACM/IEEE Conference on Supercomputing, 2005.
[8] I. Foster, Globus Toolkit Version 4: Software for service-oriented systems, Journal of Computer Science and Technology 21 (4) (2006) 513–520.
[9] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, Giggle: A framework for constructing scalable replica location services, in: Proc. of the 2002 ACM/IEEE conference on Supercomputing, 2002, pp. 58–58.
[10] A. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf, H. Stockinger, B. Tierney, Performance and Scalability of a replica location service, in: Proc. of the 13th IEEE International Symposium on High Performance Distributed Computing Conference (HPDC-13), Honolulu, pp. 182–191, 1998.
[11] E. Deelman, G. Singh, M. Atkinson, A. Chervenak, N.C. Hong, C. Kesselman, S. Patil, L. Pearlman, M.-H. Su, Grid-based metadata services, in: Proc. of 16th Conference on Scientific and Statistical Database Management, Los Angeles, CA, USA, pp. 393–402, 2004.
[12] K. Karasavvas, M. Antonioletti, M. Atkinson, N.C. Hong, T. Sugden, A. Hume, M. Jackson, A. Krause, C. Palansuriya, Introduction to OGSA-DAI Services, in: Lecture Notes in Computer Science, vol. 3458, 2005, pp. 1–12.
[13] Gnutella website: http://rfc-gnutella.sourceforge.net/.
[14] I. Clarke, O. Sandberg, B. Wiley, T. Hong, Freenet: A Distributed Anonymous Information Storage and Retrieval System, in: Lecture Notes in Computer Science, 2009/2001.
[15] M. Stokes, Gnutella2: http://g2.trillinux.org/.
[16] S. Daswani, A. Fisk, Gnutella UDP Extension for Scalable Searches (GUESS) v0.1.
[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proc. of the 2001 ACM SIGCOMM Conference, 2001, pp. 149–160.
[18] A.I.T. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, Springer-Verlag, 2001, pp. 329–350.
[19] P. Maymounkov, D. Mazieres, Kademlia: A peer-to-peer information system based on the XOR metric, in: Proc. of the 1st International Workshop on Peer-to-Peer Systems, IPTPS02, Cambridge, USA, 2002.
[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Schenker, A scalable content-addressable network, in: Proc. of the 2001 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, SIGCOMM01, ACM, San Diego, CA, United States, 2001, pp. 161–172.
[21] C. Schmidt, M. Parashar, Enabling flexible queries with guarantees in P2P systems, IEEE Internet Computing 8 (3) (2004) 19–26.
[22] P. Ganesan, B. Yang, H. Garcia-Molina, One torus to rule them all: multi-dimensional queries in P2P systems, in: Proc. of the 7th International Workshop on the Web and Databases, WebDB'04, ACM, Paris, France, 2004, pp. 19–24.
[23] J. Lee, H. Lee, S. Kang, S.M. Kim, J. Song, CISS: An efficient object clustering framework for DHT-based peer-to-peer applications, Computer Networks 51 (4) (2007) 1072–1094.

[24] J. Aspnes, G. Shah, Skip graphs, in: Proc. of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2003, pp. 384–393.
[25] S. Ramabhadran, S. Ratnasamy, J.M. Hellerstein, S. Shenker, Brief announcement: Prefix Hash Tree, in: Proc. of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing, PODC'04, ACM, New York, NY, USA, 2004, pp. 368–368.
[26] A. Datta, M. Hauswirth, R. John, R. Schmidt, K. Aberer, Range queries in trie-structured overlays, in: Fifth IEEE International Conference on Peer-to-Peer Computing, 2005, pp. 57–66.
[27] E. Caron, F. Desprez, C. Tedeschi, Enhancing computational grids with peer-to-peer technology for large scale service discovery, Journal of Grid Computing 5 (3) (2007) 337–360.
[28] H. Zhuge, X. Sun, J. Liu, E. Yao, X. Chen, A scalable P2P platform for the knowledge grid, IEEE Transactions on Knowledge and Data Engineering 17 (12) (2005) 1721–1736.
[29] Grid Enabled access to rich media content (GREDIA) IST project, URL: http://www.gredia.eu.
[30] M.F. Mokbel, W.G. Aref, I. Kamel, Analysis of multi-dimensional space-filling curves, Geoinformatica 7 (3) (2003) 179–209.
[31] B. Moon, H. Jagadish, C. Faloutsos, J. Saltz, Analysis of the clustering properties of the Hilbert space-filling curve, IEEE Transactions on Knowledge and Data Engineering 13 (1) (2001) 124–141.
[32] A.R. Butz, Alternative algorithm for Hilbert's Space Filling Curve, IEEE Transactions on Computers C-20 (4) (1971) 424–426.
[33] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, Wide-area cooperative storage with CFS, in: Proc. of the Eighteenth ACM Symposium on Operating Systems Principles, SOSP'01, ACM, New York, NY, USA, 2001, pp. 202–215.
[34] A. Chazapis, A. Zissimos, N. Koziris, A peer-to-peer replica management service for high-throughput Grids, in: Proc. of the 2005 International Conference on Parallel Processing, ICPP05, Oslo, Norway, 2005, pp. 443–451.
[35] A. Zissimos, K. Doka, A. Chazapis, N. Koziris, GridTorrent: Optimizing data transfers in the Grid with collaborative sharing, in: Proc. of the 11th Panhellenic Conference on Informatics, PCI2007, Patras, Greece, 2007.
[36] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J.V. Reich, The open grid services architecture, Version 1.0, Informational document, Global Grid Forum (GGF), 2005.
[37] MPEG-7 Overview, 2004. URL: http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

Athanasia Asiki received her Diploma in Electrical and Computer Engineering (2005) from the National Technical University of Athens, Greece. She is currently a Ph.D. student in the School of Electrical and Computer Engineering, National Technical University of Athens. Her research interests include large-scale distributed systems, grid middleware, development of grid applications, and search techniques in Peer-to-Peer systems. She is a student member of the IEEE and a member of the Technical Chamber of Greece.

Katerina Doka received her undergraduate degree in Electrical and Computer Engineering from the National Technical University of Athens in July 2005. She is currently a Ph.D. student at the Computing Systems Laboratory, Dept. of Electrical and Computer Engineering of the National Technical University of Athens, doing research in the field of Large Scale Distributed Systems, Peer-to-Peer Technologies and Grid Computing. Since 2006 she has participated in a number of European and National R&D Projects.

Ioannis Konstantinou received his undergraduate degree in Electrical and Computer Engineering from the National Technical University of Athens in November 2004. He is currently working at the Computing Systems Laboratory at the National Technical University of Athens, where he is pursuing his Ph.D. in the field of Grid Computing, Large scale distributed systems and Peer to Peer technologies.

Antonis Zissimos graduated from the School of Electrical and Computer Engineering at the National Technical University of Athens in 2003. Today, he is a Ph.D. student at the Computing Systems Laboratory and his advisor is Professor Nectarios Koziris. He has teaching assistant experience in undergraduate courses of ECE/NTUA. He is also a reviewer at parallel and distributed conferences and journals of IEEE/ACM. Currently, he is working as a Research Associate at the Institute of Computer and Communication Systems/Computing Systems Laboratory at NTUA and he is involved in European and National R&D programs in the field of Grid/High Performance Computing, Networking and Storage for IT systems. He is a member of the IEEE Computer Society and the Technical Chamber of Greece.

Dimitrios Tsoumakos is a senior researcher in the Computing Systems Laboratory of the Department of Electrical and Computer Engineering of the National Technical University of Athens (NTUA). He received his Diploma in Electrical and Computer Engineering from NTUA in 1999. He joined the graduate program in Computer Sciences at the University of Maryland in 2000, where he received his M.Sc. (2002) and Ph.D. (2006). His research interests lie in the area of distributed systems/algorithms, particularly in designing and implementing adaptive, bandwidth-efficient schemes for data retrieval and dissemination in both structured and unstructured Peer-to-Peer systems. Furthermore, he is interested in Grid Computing applications, considered to be the natural evolution of Peer-to-Peer networking. He has also been involved in Database research, especially in applying distributed schemes to autonomous interconnected databases (Peer Databases) with limited or no schema information.

Nectarios Koziris, Associate Professor, received his Diploma in Electrical Engineering from the National Technical University of Athens (NTUA) and his Ph.D. in Computer Engineering from NTUA (1997). He joined the Computer Science Department, School of Electrical and Computer Engineering at the National Technical University of Athens in 1998, where he currently serves as an Associate Professor. His research interests include Computer Architecture, Parallel Processing, Parallel Architectures (OS and Compiler Support, Loop Compilation Techniques, Automatic Algorithm Mapping and Partitioning) and Communication Architectures for Clusters. He has published more than 60 research papers in international refereed journals and in the proceedings of international conferences and workshops. Nectarios Koziris is a recipient of the IPDPS 2001 best paper award for the paper "Minimising Completion Time for Loop Tiling with Computation and Communication Overlapping". He serves as a reviewer in International Journals (TPDS, JPDC, JSC, etc) and as a Program Committee member in various parallel computing conferences (IPDPS, HiPC, ICPP, IPDPS, CAC, PDSEC, SAC, etc). He is a member of IEEE Computer Society, member of IEEE TCPP and TCCA, member of ACM and chairs the Greek IEEE Computer Society Chapter. He also serves as Vice-Chairman for the Greek Research and Education Network (GRNET-Greek NREN, www.grnet.gr).

Panayiotis Tsanakas (Professor) received his Diploma in Electrical Engineering from the University of Thessaloniki (1982), his M.Sc. in Computer Engineering from Ohio University (1985), and his Ph.D. in Computer Engineering from the National Technical University of Athens (1988). He is now serving as Professor at the School of Electrical and Computing Engineering of the National Technical University of Athens. His research interests include high performance architectures, and distributed applications in medicine and education. He co-authored six textbooks (in Greek): Operating System Principles, Introduction to Computer Architecture, The Operating System EMPIX, Parallel Computing Systems, Mapping Algorithms into Parallel Processing Architectures, Computer Architecture and Operating Systems. He has published more than 15 papers in refereed scientific journals, and more than 40 papers in the proceedings of international conferences and workshops. He has served as reviewer for several International Journals and Conferences, and as evaluator for research project proposals. Prof. Tsanakas is the Chair for the Greek Research and Education Network (GRNET-Greek NREN, www.grnet.gr).