Serdica J. Computing **5** (2011), 207–236

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

FAST INFORMATION RETRIEVAL IN THE OPEN GRID SERVICE ARCHITECTURE*

Tobias Berka, Marian Vajteršic

Abstract. In research, grid computing is an established way of providing computer resources for information retrieval. However, e-science grids also contain, process and produce documents—thereby acting as digital libraries and requiring means for information discovery. In this paper, we discuss how distributed information retrieval can be integrated into the Open Grid Service Architecture (OGSA) to efficiently provide image retrieval for e-science grids. We identify two fundamental ways of performing information retrieval on the grid – as a batch job or as a distributed activity – and argue the case for the latter for reasons of efficiency. We give an analysis of the theoretic communication and computation complexity and demonstrate that bandwidth limitations provide a decisive argument to support our case. We describe further design decisions for our system architecture and give a brief comparison with other designs reported in literature. Lastly, we describe how the statelessness and isolation of web services impede data-intensive, distributed, cross-site activities in OGSA grids, and how to escape them.

ACM Computing Classification System (1998): C.2.4, H.3.3, D.2.11.

Key words: Grid computing, information retrieval, web services.

*This is an extended version of an article presented at the Second International Conference on Software, Services and Semantic Technologies, Sofia, Bulgaria, 11–12 September 2010.

1. Introduction. Computational grids provide computation like a utility. To give more scientists access to the computing power necessary to solve today's demandingproblems, scientific funding agencies have launched large programs for the adoption of grid technology. Grids are now going into full productive use in large scientific facilities. In recent research, grids have come as a natural fit to provide storage and computation for information retrieval systems. Information retrieval (IR) is concerned with search for unstructured information such as text documents, and its high computational requirements make it an ideal candidate for parallel processing. But the connection between e-science grids and information retrieval runs deeper:

— First, scientists collaborating in a virtual organization (VO) produce and consume documents as part of their daily work. This goes hand in hand with a need to discover documents of interest through search and retrieval.

— Second, documents may serve as input for jobs submitted to the grid, e.g. for applications in digital image processing, natural language processing and other fields. Consequently, grids may persistently store a large number of documents.

— And third, documents are also produced as an output of batch jobs, e.g. in seismic analysis. The automated production of documents depicted in Figure 1 is an important factor, because it quickly produces large numbers of documents.

This means that grids themselves resemble digital repositories, in that they provide virtual organizations with a means to collect, manage, store and even automatically produce digital documents. If multiple organizations join forces in a VO to pool their resources, it is clear that we can benefit from a mechanism for resource discovery, including images and text documents. It is easy to see the benefit of having readily available information retrieval machinery capable of indexing documents *across* the boundaries of individual groups and systems.

However, this raises several challenges for traditional grid computing:

1. The documents are inherently distributed across multiple grid sites and the search is necessarily distributed. As such, we must always search across grid sites, despite the fact that conventional grid wisdom forbids cross-site activities.
2. Queries must be answered sporadically and frequently. It is not possible to predict the arrival time of the next query and plan the query execution ahead of its arrival. At the same time, a search engine for a large virtual organization may have to deal with very many queries.

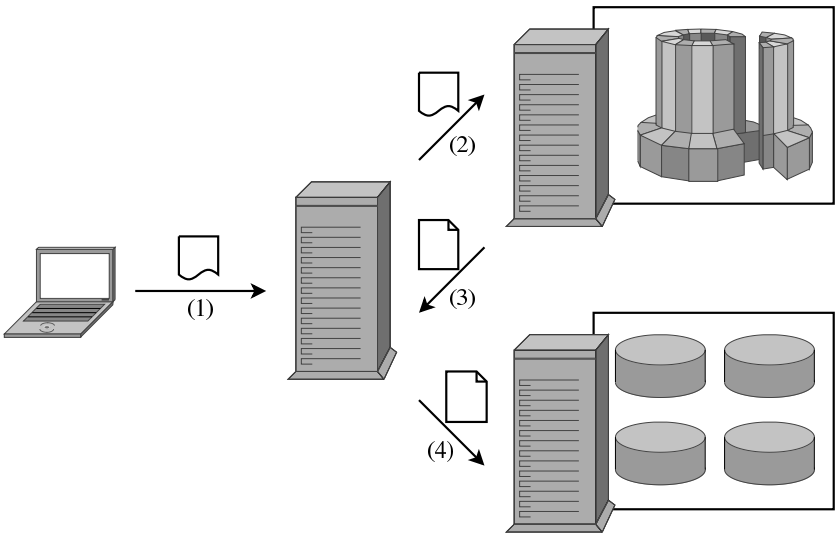

Fig. 1. Automated Document Creation on an e-Science Grid. A processing request is sent to a grid engine (1) where it is submitted to to a processing system (2). The job returns images or other documents (3) which are archived on the allocated storage system (4)

3. In addition, search engines must operate for extended durations—much longer than even expensive batch jobs.

These characteristics are very different from the usual, batch-job style of processing in high-performance computing grids. And indeed, as we will outline below, we had to find a new architecture for information retrieval on the grid in order to properly address these issues.

In this paper, we describe the most fundamental difficulties of the task at hand, and present a novel approach, which allowed us to circumvent some of these problems. We give a brief review of computational grids, traditional grid information retrieval and our alternative approach in Section 2, describe the design of our prototype for fast image retrieval in Section 3 and discuss some problematic issues that we encountered during the implementation in Section 4. Lastly, Section 5 summarizes our findings and presents our conclusions.

The research conducted on grid retrieval has been released in [8], and parts of this article have been published in [9] and [10].

2. Information Retrieval on Computational Grids. The Open Grid Forum (OGF) has opened the standardization process for IR on the grid

with a first official document [34]. Unfortunately, this standardization process has not seen much progress since. It can indeed be noted that grid information retrieval has been added to the agenda of the ACM's special interest group in information retrieval [4]. However, our own considerations regarding information retrieval on the grid start at a lower, more fundamental level.

For our goal of providing information retrieval on or for the grid it is an important question whether we consider information retrieval to be merely an application, which we split into individual computational tasks that we can dispatch to a computational grid for execution, or if we see information retrieval as part of the infrastructure. The latter perspective suggests that we should deploy information retrieval services alongside the other low-level services. But before we discuss the specific details of information retrieval on the grid, let us first review the definition and architecture of computational grids.

2.1. Computational Grids. Grid computing pursues the long-term goal of providing computing like a utility. The term "grid computing" itself is derived from the electric power grid. One day in the future, computing will be provided like electricity by simply plugging a simple terminal into an "outlet". Beyond this long-term vision there is also the mid-term perspective on grid computing. In this case, the broad concept of "computing" can be narrowed down to two cases: high-performance scientific computing and business-centric enterprise computing. The primary goal is to increase the utilization of geographically dispersed compute facilities. Consequently, the raw quantity of computing equipment is to be reduced, along with the cost of owning and operating it [24]. In both fields which are to be served, the primary task at hand is the batch processing of data intensive applications. There are several aspects that contribute to the complexity of this objective:

— Geographic dispersion: the facilities which constitute a grid are placed at different geographic locations. The scale of this physical distribution is nothing less than global. It is clear that both the large scale and the ubiquity of dispersion pose many technical challenges. Improved utilization requires performance optimization in an inherently distributed environment with an *extremely* broad cost spectrum.

— Social and legal dispersion: the facilities are owned and operated by different *legal entities* or organizations. Groups of organizations can form virtual organizations, which exist as units of administration on a grid. These units then share access to grid resources, such as computing facilities, and data sources, such as file systems, sensors or databases. Care must be taken that the various requirements of such organizations are met, including proper

authentication and authorization, auditing, billing and means of administration.

— Heterogeneous systems: at the individual level, the systems within a grid are not homogeneous. In fact, a grid is a large, collaborative system consisting of different hardware platforms, operating systems, software libraries and applications. While some differences may be obvious and relatively easy to deal with, such as different processor architectures or operating systems, other differences are more subtle and carry a much greater potential for implicit dependencies and thus failure. Minor differences between version, binary patches or simply the placement within the file system are known to wreck havoc on application behavior, compilation of software and distributed deployment. Needless to say, all these issues must be dealt with only to deploy and run a piece of software on a grid.

— Heterogeneous grids: due to the decentralized and competitive nature of computing as a business and science, there is now a broad range of both commercial and scientific toolkits for grid computing. These can be cluster management tools [27] [28], cycle stealing tools [1] [2] or grid techniques integrated into enterprise computing solutions [14]. This has given rise to the need to standardize grid functionality. The Open Grid Forum[1] (OGF) is the standardization group which has taken on this effort. The Globus toolkit[2] serves as the reference implementation of this standardization effort [26].

Since we attempt to be independent from individual vendors or system designers in our work, we are closely following the doctrine and principles of the open grid forum and the design of the Globus toolkit. The design of the common grid architecture of the OGF is referred to as the Open Grid Services Architecture (OGSA) [25]. The approach taken by the OGF fundamentally relies on web-based service oriented architecture, as embodied by the World Wide Web Consortium's[3] web service standards, with extensions for security and stateful web services by the Organization for the Advancement of Structured Information Standards[4] (OASIS). The OGF provides standardized interfaces to the common functionality of various grid toolkits. In this manner, the OGSA and its reference implementation, the Globus toolkit, offer the usual range of facilities for a batch processing system.

[1] see `http://www.ogf.org`.
[2] see `http://www.globus.org`.
[3] see `http://www.w3.org`.
[4] see `http://www.oasis-open.org`.

2.2. Grid Information Retrieval. The deployment of information retrieval on the grid can be done in two basic forms: as a job or as a service. If IR is conducted as a job, then we utilize the grid as a computational utility. But if it is provided as a service, then we integrate it into the grid infrastructure. In this article, we are arguing the case for the latter approach, because it is more suitable for the task at hand.

For IR as a job, the computationally expensive tasks of information retrieval systems are submitted to a computational grid as jobs. In this case, we utilize the grid as an automated batch processing system, which allows us to perform computations by staging executable code and the data onto the system allocated for by the grid scheduler. However, since IR systems seek to maximize the user satisfaction by minimizing the response time [17], batch processing can only be applied to off-line functions without stringent requirements on the response time. Figure 2 sketches this traditional approach to grid IR.

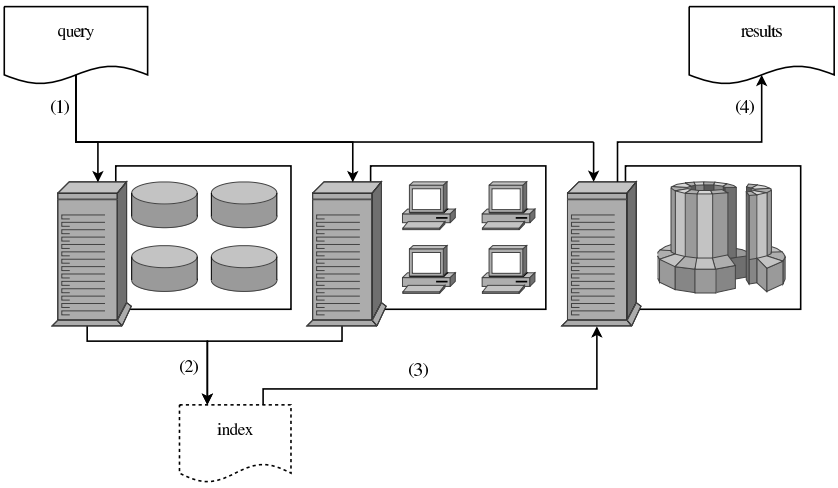

Fig. 2. Grid Information Retrieval as a Job. A client submits a query (1), the distributed indices are located (2) and staged to the dynamically allocated computing node (3). This node computes the query and returns the results (4)

As an example, consider a retrieval system using latent semantic indexing [21], which has to compute and update a singular value decomposition. These expensive updating and indexing steps can be performed as batch jobs on the grid. The query execution, on the other hand, should be performed in a different manner, because the time spent moving the index to the allocated processing system

is not proportional to the time spent computing the query results.

With IR as a service, we avoid this problem by conducting a distributed retrieval process amongst the nodes of a virtual organization's grid. The retrieval functionality is thus offered as a persistent, distributed service and the node-local indices can be kept ready to respond to incoming queries.

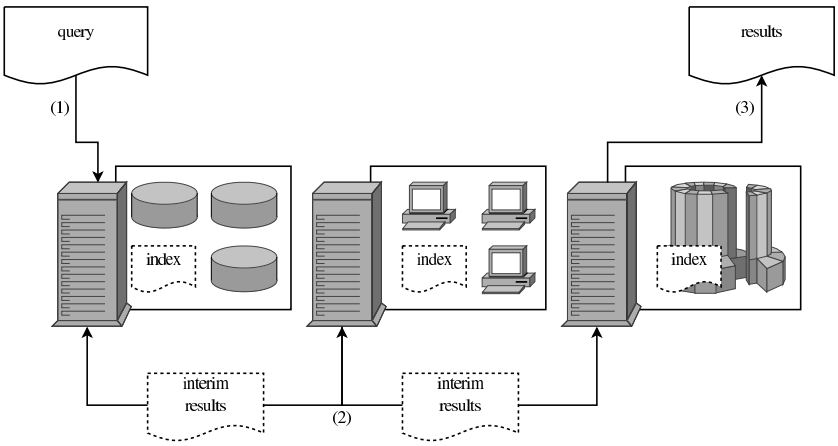

Fig. 3. Grid Information Retrieval as a Service. A query is submitted (1) and the grid nodes compute the results in parallel by exchanging intermediate data (2) *without* relocating the local indices. Once the parallel algorithm completes, the results are returned (3)

The requirements of IR support this approach. In all of its forms, an information retrieval system maintains some form of search index, which contains the associations (e.g. presence, relative frequency, relevance or some other quantification) between the features (e.g. words, tags, genes or image metrics) and the documents. This index is queried to provide search and retrieval functionality. For information retrieval as a job, it is necessary to move the entire index back and forth between the server, which provides the retrieval functionality, and the allocated computing nodes in the grid.

Clearly, this is not a very economical approach to the task at hand. The specific requirements of information retrieval are somewhat at odds with the job-submission paradigm inherent in computational grids. If we take the other route and integrate information retrieval into the grid infrastructure, we can eliminate this problem, making information retrieval as a service not only a valid, but indeed a valuable approach.

2.3. The Cost of Index Migration. To illustrate the advantage over the traditional approach, let us consider a system of N sites indexing a total of D documents using F features. We now analyze the theoretic gains we can make in terms of processing time and communication costs per query. For dense feature vectors used in image retrieval, the computational complexity of the query processing is dominated by computing the similarity between the query and all documents. Sorting the results with an optimal complexity of $O(D \log D)$ is barely noticeable compared to the similarity scoring computation with a typical serial time complexity of

$$T_S = O(FD).$$

If we split the document collection into N disjoint parts of equal size, the parallel time complexity is

$$T_P = O\left(\frac{FD}{N}\right).$$

Without considering communication cost, the processing time is reduced and we obtain a theoretic speed-up of

$$S = \frac{T_S}{T_P} = \frac{O(FD)}{O\left(\frac{FD}{N}\right)} = O(N).$$

For cross-site grid retrieval the communication overhead will not save us any time, but the burden placed on the individual site is reduced. Table 1 summarizes the mathematical symbols used here.

Table 1. Mathematical Symbols

Symbol	Description
N	Number of grid sites,
D	Number of documents,
F	Number of features,
T_S	Time complexity of the serial algorithm,
T_P	Time complexity of the parallel algorithm,
S	Speed-up of the parallel algorithm over the serial version.

But IR as a service also allows us to improve the network utilization. Merging and sorting the results of the individual sites can be performed with a distributed merge-sort algorithm, which requires $O(\log N)$ serial communication steps. The message size doubles with every step, but the total message size is less than twice the number of documents, or $O(D)$ asymptotically. In the traditional grid retrieval case, all contributing sites must send their entire indices to the indexing server. This effectively requires $O(N)$ message sends with a total message

size of $O(FD)$. Simultaneous connections are of course possible, but these incur additional overheads in the operating system and networking hardware, may cause network congestion and must share the available bandwidth. Consequently, they seldom perform better than a sequence of isolated send operations. We thus reduce the number of serial messaging steps from linear to logarithmic and the total message size by a factor of $O(F)$ *for every single query*[5].

Needless to say, a balanced distribution of documents is an important prerequisite for these savings. For asymmetric situations, where few sites hold many documents but most sites hold very few, we should either re-distribute the documents or use a monolithic retrieval engine.

To illustrate the difference in terms of concrete numbers, we will consider the transmission time based on the bandwidth. For simplicity, we will not consider the latency. If the network topology provides multiple links to the server, then the bandwidth is limited by the memory bandwidth of the processor. In the Intel Nehalem architecture the absolute limit is 30 GB/s. If we use 1,024 image features in single-precision, 32 bit floating point representation, we have a total size of 4 KiB per document. We can then transmit the vectors for 7,864,320 documents in one second. While merging the results from different sites, we transmit a 32 bit document identifier and a similarity score in 32 bit floating point representation for each document, with a total memory size of 8 bytes per result. The theoretic throughput is therefore 4,026,531,840 results per second. If we were to transmit the feature vectors for all of these documents, the total transmission time would be 512 seconds, or 8 minutes and 32 seconds. We can expect an improvement in transmission time from 8.5 minutes to about 2 seconds *for every query* if numerical identifiers are used. Use of string identifiers reduces this advantage. Assuming a document identifier length of 252 characters and single-precision similarity scores, we can still transmit 125,829,120 results per second, or 16 times more than document vectors.

2.4. The Choice of Middleware. An important question is the choice of communication middleware for such a persistent parallel information retrieval service. We can follow the OGF in their move towards a service-oriented architecture using web services as a means of communication between nodes [26]. This design rationale led to the open grid service architecture (OGSA) described in [25]. Web services are language independent and allow a great deal of flexibility regarding the design and implementation of the retrieval software running on individual nodes. The price we pay for this flexibility is high communication costs due to XML-based message formats. Furthermore, web services offer a different

[5]The savings are actually closer to $F/2$, but this fine point is lost in O-notation.

style of communication compared to traditional middleware for parallel high-performance applications—a gap that must be bridged by manual programming effort.

Another approach would be the use of grid-aware implementations of the message passing interface (MPI). Such systems use various techniques to bypass firewalls or other mechanisms obstructing cross-site communication and attempt to re-structure collective communication operations to minimize the use of high-latency links [18]. This approach obviously provides better communication performance and allows parallel services to use the popular MPI interfaces. But it limits the openness of the distributed retrieval system because all local implementations are forced to use a specialized MPI implementation. We lose the flexibility of mixing different implementations within the grid retrieval system of a virtual organization.

A third approach would be the use of a service-oriented architecture other than web services for information retrieval systems, e.g. the OSIRIS middleware framework [15]. These alternate forms of middleware may offer more flexibility than plain web services, but they are often available only in research implementations and do not enjoy widespread use.

We believe that we should choose the first option, comply with the OGSA and use web services as a means of communication despite the increase in cost of cross-site messaging. If we define interfaces for effective and efficient information retrieval, we not only avoid the index migration problem, but we can include information discovery mechanisms into the standard functionality of grid toolkits. And as we have noted above, this is an identified goal of the OGF.

Thus far, we have discussed our key architectural issue: *how* to conduct information retrieval on grids. The next design decision is to determine which distributed or parallel information retrieval architecture would be suitable for retrieval as a service. Distributed information retrieval offers an extremely broad range of retrieval models, system architectures and distribution methods. This field is so rich that a survey of the state-of-the-art is clearly out of scope. But before we describe our own design decisions, let us review some of the fundamental architectures that have been developed.

3. Fast Image Retrieval as a Service.

The design of any distributed retrieval system must strike a balance between flexibility and response time and/or retrieval quality. It is easy to see that we gain a maximum in flexibility if we design for a federated, heterogeneous retrieval system with distinct features and potentially overlapping document collections. But we can minimize the re-

sponse time if we build a homogeneous, centralized retrieval system with strictly partitioned features and documents. In order to make informed design decisions for our image search engine in e-science grids, we will sketch the background in parallel and distributed information retrieval.

3.1. Parallel Processing in Information Retrieval. Sparked by some prominent computer architectures of the past, it was natural to raise the issue of how to best utilize methods from the field of parallel computing for information retrieval purposes at a very early stage [48]. In more recent developments, MPI-based text search engine on high-performance middleware [38], a hybrid synchronous/asynchronous parallel search engine [40], a parallel algorithm for nearest neighbor search on a clustered index [29] or a parallel index for multimedia search [39].

Due to the past and present importance of text retrieval, no overview of distributed retrieval can be complete without mention of the research on parallel and distributed inverted files—the most common form of index representation. Instead of storing the plain text of a file sequentially, it can be transformed into an inverted file. In this representation, the original document content is stored as a list of words and their occurrences. In this format it is easy to construct word frequencies by counting the number of occurrences for every word. The inverted list thereby serves not only as an extended form of sparse vector representation for text indexing, but also allow search engines to locate the actual occurrences and reconstruct the surrounding text, which is a very useful feature for the presentation of search results for the end user.

Naturally, this representation has attracted much attention in the research of parallel and distributed retrieval architectures [52] [55] [33] [23] [50] [61]. It should also be pointed out that prior to inverted files, various other coding- or hash-based mechanisms have been used [51].

Unfortunately, these thoroughly understood techniques do not serve us well for our purpose of retrieving images. The reason for this is two-fold:

1. The feature vectors for image retrieval are dense rather than sparse. This means that the inverted list representation is inefficient compared to dense vector or matrix representations.
2. The features and measures provided by digital image processing typically do not have any meaningful location in the source image. While certain features are indeed spatially located, such as windowed histograms in the image domain, this kind of location information serves no use for the presentation of image search results. The added value introduced by inverted lists is thus not present in image retrieval systems.

3.2. The Background of Distributed Databases. Much of the more recent work in the field has taken place before the backdrop provided by distributed relational database systems. Several important distinctions for architectural and organizational principles have been carried over from the retrieval of data to that of information.

Using central processing units, random access memory and hard disk drives as building blocks, designers of distributed databases use a three-way classification of hardware architectures, e.g. in [31]:

1. Shared-everything systems allow arbitrary sharing of RAM and HDDs amongst a number of CPUs.
2. Shared-disk systems allow the sharing of HDDs amongst a number of CPUs with individual RAM.
3. Shared-nothing systems consist of individual hosts with isolated units of CPU, RAM and HDD communicating on one or more shared channels.

Since the grid is geographically dispersed over a very large area, our own system is placed firmly in the realm of shared-nothing systems. While the networking technology may be low-latency and high-bandwidth to support the demands of the grid infrastructure, from the perspective of our information retrieval system these links are nowhere near the performance of a local area network due to the geographic distances and cumbersome XML formats involved. This leaves us with relatively high costs for communication, which must be accounted for in the design of our algorithms.

While this model describes the sharing at the hardware level, [20] introduces an analogous classification for the sharing of information at application level:

1. Shared-index systems allow every cooperating computing element to access a common index in full.
2. Shared-vocabulary systems can utilize a global, shared set of features.
3. Shared-nothing systems consist of isolated hosts without any explicit sharing of information.

While this classification can describe some of today's distributed retrieval architectures, we believe that it fails to capture the difference between explicit sharing of information, by replication or storage on a dedicated server, and implicit sharing of information, where the global state is derived through distributed control but *never constructed explicitly.*

Another distinction describes the degree of local autonomy for the individual hosts. The literature refers to this with somewhat sharp notions, classifying

systems either as *centralized* (no local autonomy) or *federated*, which enjoy a varying degree of local independence.

The difference between *homogeneous* and *heterogeneous* systems largely follows the same structure as distributed databases, but it is not explicitly noted and often intermingled with other distinctions. In information retrieval, heterogeneous search systems that cannot cooperate directly are typically referred to as *meta-search engines*, because they consist of search engines querying other search engines without knowing any details about their vocabulary, index or algorithmic, inner workings. Homogeneous systems, on the other hand, are often assumed to be cooperative, tightly coupled and willing to expose their data [46]. In both cases, the problems of selecting sub-collections, combining the results and assigning new, combined relevances, commonly referred to collection selection, result fusion and merging problems are also intense subjects of study, see e.g. [6] [13] [16] [45].

3.3. Information Retrieval on the World Wide Web. Although there is no strict, formal requirement for information retrieval on the World Wide Web to be parallel or distributed, the sheer volume of documents will require a combination of both. But it is related to retrieval on the grid because the documents we wish to index are inherently distributed. For the web, information retrieval services are provided by a number of well known search engines. The vast amount of literature on the matter lies beyond the scope of this article. For our purposes it is important to note that the grid differs from the web in ways which have a great impact on how to provide information retrieval.

First, the grid is *not* generally a *public information space* like the web. Most of its information is confidential to the general public and must be made available only to the members, or some of the members, of a virtual organization. The focus on e-science readily illustrates that internal, work-in-progress documents must not be leaked to individuals willing and able to subvert the principles of cooperation within the virtual organization. But the confidentiality required for grid information retrieval can go even further than access to the full documents. Even the existence and title of certain documents may have to be limited until the results have been published. Web search engines are typically owned and operated by a third party, which has no direct affiliation with the parties of the virtual organization. They operate on the documents exposed to the public on the World Wide Web by traversing the link structure of the hypertext and retrieve the documents' content using the HTTP protocol. To conduct information retrieval on the grid, the individual sites need more control over which documents are indexed and retrieved. This high degree of control suggests either

federated information retrieval systems with a high degree of local autonomy, or cooperative systems with distributed control operating within the boundary of a virtual organization but across the physical entities it contains.

Second, the grid is primarily a compute-enabled distributed system. It consists of a set of infrastructural services and supports additional execution by allocating resources for batch-jobs issued dynamically. This is an environment well suited to the deployment of distributed information retrieval systems. Considering the cost-ratio between local processing and storage versus transmission of data over networks, it can be much more economical to move the processing to the data. In terms of information retrieval this means indexing local document collections locally. Here we face a trade-off between cost and retrieval performance. A cost-effective system may consist of isolated indices with minimal interaction. More accurate systems may require more interaction between the individual hosts. But as we will discuss in this article, there are parameters other than cost and performance which further complicate the design of information retrieval systems on the grid.

Third, the web is a dedicated hypertext information system and relies on the HTML document format as its primary form of content. The linked structure of the web inherently provides a rich foundation for information retrieval because we can typically assume that there exists some textual document in which any other content is embedded. Not only is text the most thoroughly supported form of content, but it can also be used to retrieve other media types based on the text surrounding the hyperlink. Furthermore, the existence of links between documents can be exploited for associative models for information retrieval [47]. Grid retrieval cannot make such powerful assumptions about the nature of its content.

3.4. Peer-to-Peer Retrieval. Information retrieval in peer-to-peer systems bears several resemblances to grid retrieval:

— Structured peer-to-peer systems are almost identical to grid applications using structured overlay networks, which offer some form of communication topology as a virtual networking layer (similar to virtual private networks).
— The federated nature of peer-to-peer systems, where individual hosts are at liberty not only to join or leave at their discretion, but also to add or remove documents from the implicit, global corpus as they like, is almost identical to the degree of autonomy we wish to have in grid infrastructure.

The fundamental difference is that grids are primarily a compute infrastructure and not mere shared document bases. However, we can see many infrastructural concepts which are common among both grids and peer-to-peer

system (as discussed in [58]). As an example, consider the traditional publish-subscribe mechanism in distributed computing, which is present not only in the open grid forum's standards (through the common information model standard [22]), but also in general grid research [44], the research of structured overlay networks [57] and peer-to-peer systems [7].

Information retrieval for the grid is no different in this respect. Many of the techniques developed for peer-to-peer information retrieval lend themselves readily for federated grid retrieval with a very high degree of local autonomy.

The key research issues in peer-to-peer retrieval are much the same as in other fields of distributed information retrieval, but with a different emphasis. Of key importance is the question of query routing, i.e., how to channel a query through a peer-to-peer system to obtain optimal search results while keeping the overall strain on the communication links minimal. Primitive systems ignore the overhead and use flooding. More advanced systems use the established state-of-the-art in traditional peer-to-peer systems, which are primarily based on distributed hashing techniques [3]. This line of work tends to cater for the "retrieval" aspect in the sense of delivery via network. Of greater interest to us are other techniques, which are more advanced in terms of information retrieval. One class of solutions is semantic network overlays, whose structure is determined by some form of document similarity rather than communication topology and link latencies. A popular approach is to represent every host via the centroid of its local document vectors [19] [53] [54]. Other approaches use limited flooding or random walks through the overlay network until they find a suitable host or neighborhood of hosts [35].

3.5. Distributed Information Retrieval Architecture. The design of any distributed information retrieval system is governed by a number of basic properties. Two important, discriminating issues are the degree to which individual nodes of a system can make decisions affecting the global state and the extent to which individual nodes can become active without an external instruction to do so. As an example, consider a distributed information retrieval system organized as a peer-to-peer system, where each node holds the documents stored by the local user. All of these documents are represented as vectors in a common vocabulary of features. The users of this system may search the global collection by issuing queries at their local host. Such a system has a high degree of local autonomy regarding documents, because every node can add or remove documents as they see fit. The local autonomy regarding features is effectively zero, because all nodes must use the same vocabulary. We can also assert that nodes can exert a high degree of local activity, because the users can issue a query at any node,

which must then be processed in the entire distributed system.

Some systems expose their internal data to the outside world, while others keep it secret. But the wide deployment of service-oriented architectures allows information retrieval systems to *respond to technical inquiries* rather than to merely provide access to raw data. The important distinction is that it is possible to synthesize the information from a wide range of heterogeneous data representations. As an example, consider two text retrieval systems:

— System A keeps inverted lists for all of its documents and uses a Boolean query model.

— System B uses the vector space model with inverse document frequencies.

If the functionality to compute the base data for a global *idf* score is wrapped as a function, these two systems are free to query one another, in order to improve the retrieval performance of both systems. An important factor for real-world applications is the existence of billing techniques to make data available even to competitors—for a price.

Another question is whether or not the hosts use a common vocabulary of features. This issue can become relevant if we intend to integrate a number of open hosts with different vocabularies. In this case, we must take appropriate steps to reconcile the vocabularies used by the different hosts. While this poses few problems for text retrieval, the reconciliation of different image features may be extremely difficult or impossible without access to the source images and re-extraction of image features.

In systems with centralized control the distributed processing is orchestrated and controlled by a *coordinator*. This host sends requests to all other hosts which participate in the computation, initiates their processing and gathers their results. This means that all host interactions directly or indirectly involve the coordinator. In system with distributed control the activities and flow of information is determined and initiated *locally*. Host interactions occur without the involvement of a third party. While distributed control is certainly more difficult to implement, it can greatly reduce the communication overhead.

One obvious means of improving the performance of a distributed information retrieval system is to conduct the queries on a restricted subset of hosts. So instead of operating on the complete, distributed state of the system we can choose to conduct queries on a constrained, incomplete part of the total index. Needless to say, there is a trade-off between retrieval accuracy and query response time.

One common approach is to cluster documents into groups of geometrically similar ones and to pre-select suitable clusters based on the similarity

between the query and the centroids of the clusters [30]. In another approach, clusters are represented as vectors containing the inverse document frequencies *within* the individual collections [60]. If the individual hosts of a system are uncooperative, a meta-searcher can access individual documents and create a representation of the underlying collection via query-based sampling [42] [43].

One important functional perspective on different types of distributed information retrieval systems lies in the way queries are routed amongst hosts. There are several approaches to the query routing problem. These differ primarily in the required volume of global data. Some approaches use large quantities of highly dynamic global information but may provide very efficient ways of directing queries to the target hosts. Other approaches minimize the amount of globally stored data and sacrifice the efficiency of routing the queries to achieve this.

The first type of query routing, routing by index, uses a complete index describing which features or documents are available at every host of the system. Using such an index we can determine the *set of holding hosts* for every entity in the system and contact these directly. We use the notion of a *partial* routing index to refer to a form of index which uses an incomplete representation to eliminate the need for an *explicitly stored* complete index. Such partial indices can be based on distributed hash tables as used in peer-to-peer networks. Another, more expensive approach is to introduce integer surrogate identifiers for every entity and a (global) mapping to translate between the external, original identifiers and their internal representation. This requires additional attention when allocating surrogate identifiers and when deleting or relocating documents to prevent a degenerate distribution of the entities amongst the hosts. While this approach has a higher communication overhead, it introduces some additional flexibility, because the surrogate management can be implemented as an independent, reusable sub-system.

Another approach avoids the *explicit* representation of the global index. Instead, queries are dynamically routed through the system. We refer to this as *interactive query routing*. The straightforward approach is to use message flooding to spread the query through the entire system. This approach can be improved by using local host information or proximity information about a host's neighborhood to eliminate some of the message overhead and processing costs caused by unselective flooding. Interactive routing strategies often involve a content-based selection of the routing target. This decision is typically made locally, meaning that the target for the next hop in the transmission is selected at every host. In autonomous and dynamic systems, this process is often inexact and requires some amount of message branching to search in an area rather than an individual host.

But all of these approaches suffer degradation in retrieval performance. The interactive routing approach has its origin in federated, heterogeneous, distributed relational database systems. It can eliminate the complication of maintaining a global state in a highly dynamic or heterogeneous system. But this flexibility comes at a cost, because the global state is partly derived during every single query.

3.6. Organization in our Approach. Before we can make our choices regarding the organization of our information retrieval system we must briefly review the goals of our system. The target "audience" of our system consists of the members of virtual organizations on computational grids used for e-science applications. Such applications will be designed for domain experts who are able to conduct complex searches and require accurate and complete results for their work (quite unlike the retrieval of holiday images on the Web). The individual applications will need to choose suitable image features for the specific tasks at hand. Based on these requirements we have chosen the following organizational principles for our system:

— Image retrieval as a service: The most fundamental decision we have made for our system is that, based on our previous discussion about information retrieval as an infrastructural task, we seek to provide image retrieval functionality on the grid as part of the web-service based infrastructure. While the kind of machinery we develop is optimized for the retrieval of images, it is important to note that the interface definitions certainly also suffice for text retrieval.

— Shared-nothing architecture: Since our system will be part of a computational grid's infrastructure we are dealing with a shared-nothing architecture. Note that the individual hosts on which our system is deployed can themselves be more complex than single-CPU systems, but our focus is on the overall architecture, rather than hardware-specific optimizations of the host-local processing.

— Centralized system: The system as we envision it does not grant autonomy to the individual hosts. Members of the VO can all submit documents to the distributed system, but they do so through a single designated server. Our motivation for a centralized approach is threefold. First, a well-designed front-end server can easily handle large numbers of requests, since the actual workload is carried primarily by the working hosts. Second, we do not consider the virtual organizations to be so extremely short-lived to warrant the additional design complexity and the run-time overhead incurred by allowing more local autonomy than is needed. Third, current grid toolkits

have very limited support for peer-to-peer functionality. Even the abstraction of the network layer using structured overlay networks is not widely available (even though it is a well-researched topic). To make matters worse, those software libraries which provide functionality for federated, distributed information systems are not compatible with existing grid toolkits. As a result, every piece of federated functionality must be designed and implemented specifically for our system as a custom solution. Clearly, this task lies beyond the scope of the project.

— Master-slave organization: As an immediate consequence, we will base the design of our distributed mechanisms on a master-slave organization. The system will consist of a single master, who is responsible for receiving and answering client requests, and a group of compute hosts, who perform the distributed query processing.

— Open hosts: The individual hosts in our system should readily answer questions about their internal state, such as statistics for features or documents. One particular goal is distributed feature weighting, which we do not intend to limit to specific approaches due to the differences between image and text retrieval systems outlined above.

— Query-routing by index: In order to reduce the number of messages used by the system, yet retain the full flexibility regarding the placement of documents, we have decided to opt for query routing using a complete index. The master server retains a complete list of which features and documents belong to which hosts. The storage and look-up overhead caused by this index is easily compensated by the fact that the operations on a single host are much faster than web-service based, interactive query routing.

— Complete queries: We use complete knowledge of the entire index to answer queries. Since we are building a system for e-science applications, we favor retrieval accuracy and completeness over retrieval performance. We therefore gather the entire, dispersed information required to answer queries exactly as specified by the general vector space model.

Since we are constructing an image search engine, which uses image features, we must work with feature vectors that are *dense*—a substantial difference to text retrieval, which uses very sparse vectors since most documents contain only very few words. But for a wide range of modern text retrieval models the feature vectors are dense rather than sparse. Today, we have a wide range of dimensionality-reducing techniques, such as latent semantic indexing [21], spectral decomposition of the feature covariance matrix [36] or rare term vector replacement [11].

We decided to choose a specific retrieval model to obtain simplicity in the design and efficiency in the implementation and selected the vector space model. It was originally introduced in [49], but has been extended in ways too numerous to describe in this paper. Suffice it to say that it is a tried-and-true retrieval technique and lends itself well to various modern retrieval tasks.

Our system uses two types of distribution: both the features and the documents can be partitioned amongst hosts of the system. In addition, both partitionings can be combined into a hybrid partitioning strategy. The document partitioning can be used to index and search the documents of multiple sites of a VO. Within a site, a feature partitioning may be used to further accelerate the search process. We adapted the data parallel retrieval algorithm described in [12].

For the future, we have the option to add additional layers for improved efficiency through caching or replication, or for fault tolerance through redundancy. Furthermore, the local similarity computation can be exchanged for a sparse implementation of the vector space model, without having to modify the distributed query processing.

Unfortunately, it is not possible to give a full survey of the related literature and we focus on prior work with immediate relevance to our approach. In research, conventional information retrieval systems have been successfully deployed as jobs on the grid [32] or in more intricate architectural forms using workflow engines [37]. Other research directions include the development of new similarity index structures and algorithms specifically for the requirements of grid infrastructure [5]. Research on distributed information retrieval with a service-oriented architecture is not limited to the OGSA, see e.g. [41]. For text retrieval in the vector space model using inverted files, the feature and document partitioning [33] and the hybrid partitioning [59] have been applied successfully.

4. In-Memory Retrieval with Web Services. Due to the disparate scales of performance, executing search operations on an index in main memory is clearly preferable to disk-based searching. In-memory retrieval is therefore a field of study in its own right [56]. For infrastructural information retrieval in the OGSA, it becomes a major cause for concern.

We have previously argued that information retrieval is best offered in a server-like manner in order to prevent costly migration of indices. Initially, it would seem that web services are a natural fit for the task at hand. However, two fundamental issues cause an enormous amount of manual labor in the programming of such systems in practice. In all major implementations of web

services, the delivery of messages or requests is processed by a user-supplied function, which is isolated inside a web service container. We had to add the message queues and message-passing facilities in our own software, and faced further challenges to overcome the isolating barrier of the web service container.

In theory, web services are designed to be closed operations without any protocol-specific state, which are executed within the isolation of a web-service container, i.e. a separate sub-process contains the execution of the web service to limit the harm it can do to the overall system. If web services are used in an actual, non-trivial application to provide operations to external clients, we are confronted with an entirely different story. Here, web services must operate on the application's state.

As a consequence, the principles of statelessness and isolation are therefore subverted. The most common way to do this is to store the application state in a relational database, as depicted in Figure 4. The implementation of the web service uses a database connectivity driver to read and modify the state.

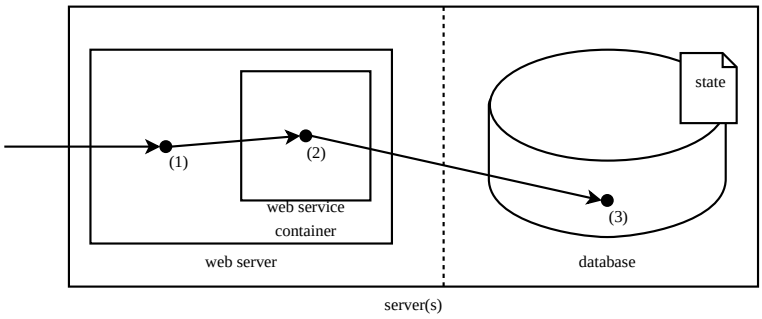

Fig. 4. Overcoming statelessness and isolation of web services using a database. The incoming web service call (1) triggers the execution of the associated function (2), which escapes from the web service container using a database driver (3)

Recognizing the need for better tools for the job, the organization for the advancement of structured information standards (OASIS) has developed the web service resource framework (WSRF). It is a collection of XML-based micro-standards for the creation, use and management of state information for web services. As shown in Figure 5, a stateful web service has means for declaring and manipulating a state, which is persistently stored in the file system in an XML file. This state is presented to the web service whenever a call is made, and the changes are written back to the file system once the call has completed. Similarly, a web service implementation could simply use the file system to store its state in a custom file format.

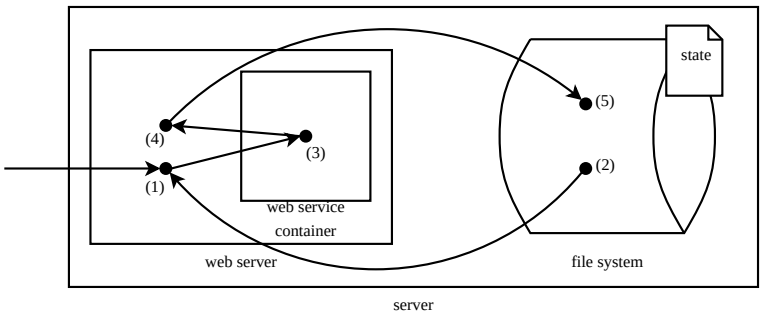

Fig. 5. Overcoming statelessness and isolation with the web service resource framework. For every incoming message (1) the old state is fetched from the file system (2) and the handler function is called (3). Upon completion, the new state is handed back to the web server (4) and persistently stored in the file system (5)

For our purposes of providing an application programming interface (API) for distributed, parallel information retrieval, statelessness and isolation are highly problematic. The key reason for realizing information retrieval as a service was to prevent index migration for efficiency. Now, web services create a similar problem: we must avoid moving the index to and from expensive persistent storage. Statelessness prevents us from holding the local portion of the search index in memory across multiple queries and service invocations. Isolation makes it impossible to share the search index with another process which executes the retrieval functionality. At the same time, a relational database and the OASIS's web service resource framework are both unsuitable for our needs. Neither a relational database nor an XML file is well-suited for storing dense matrix data, which constitutes the bulk of our application state. Direct usage of the file system was also out of question to avoid touching the slow, persistent storage mechanism of the hard disk drive.

Therefore, we had to develop a new way of escaping from the web service container. We decided to use remote procedure calls (RPCs), as provided by the open network computing ONC-RPC framework, which is readily available on every major Linux distribution. These remote procedure calls can be made from the computer hosting the web server to an RPC daemon on the same host, or to another machine used exclusively for indexing. The web service simply reformats the data to data structures suitable for RPC transmission, dispatches a call to the RPC handler, which executes the implementing function for the call. The implementation of each remote procedure places the message content in one of two message queues: one for processing requests and another for delivery of

intermediate results. These two queues are shared between a *messaging thread* for the execution of the remote procedure calls and an *application thread*, which performs the actual retrieval functionality, as depicted in Figure 6.

Web services provide either remote procedure call or message passing semantics. In practice, however, every web service invocation is mapped directly to a function call, which is executed in isolation within a web service container. Currently, there is no realization providing actual message passing facilities. This is a severe impediment, since we require messages to communicate intermediate results during the distributed search process. But because the message trans-

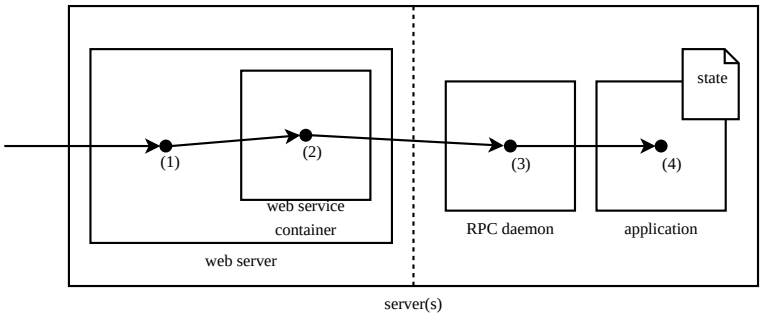

Fig. 6. Overcoming statelessness and isolation with remote procedure calls. An incoming message (1) is passed to the handler (2). An associated remote procedure call is dispatched to target host, where a daemon receives the call (3) and invokes the associated function (4)

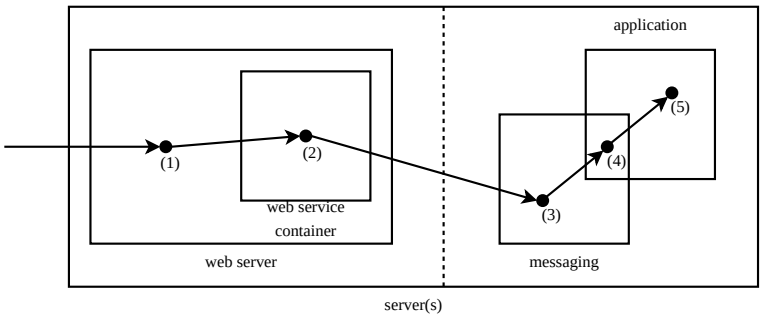

Fig. 7. Our strategy for overcoming statelessness and isolation. The web server (1) passes the message to the executing web service container (2), which uses remote procedure calls to transmit the message to the RPC daemon (3). The implementing function places the data in a shared message queue (4). This queue can be accessed by the application thread (5), which performs the actual retrieval functionality

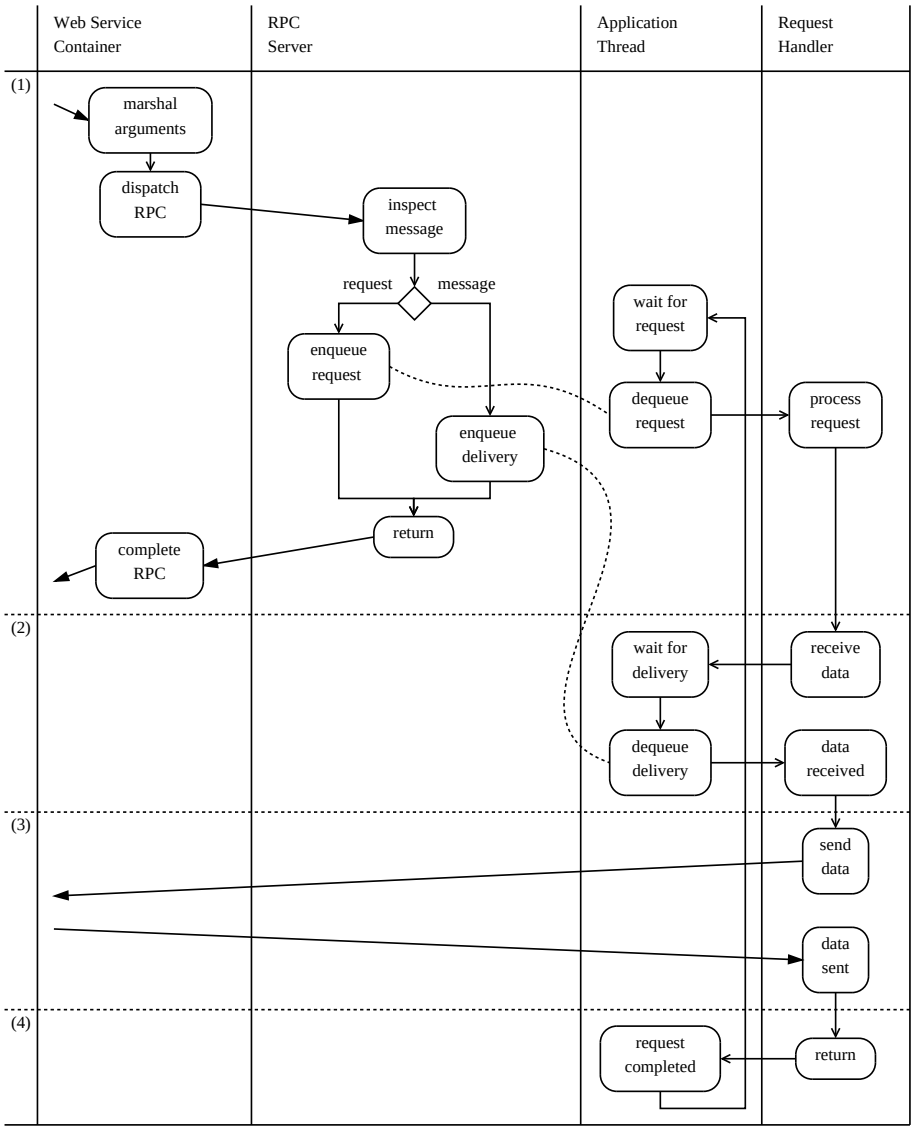

Fig. 8. Swimlane flowchart for the queue operation. An incoming web service invocation is received by the handler and placed in the request or message queue (1). If intermediate data from another node is received while a request is being processed (2), the message queue acts as an asynchronous buffer. Intermediate send operations (3) are performed directly by the request thread by dispatching a web service invocation. Once the execution of a request terminates (4), the processing thread returns to the request queue and awaits the next job

mission can take place concurrently with the local result processing, we further need an asynchronous, concurrent message passing facility. As shown in Figure 7, we are using two queues for our asynchronous message passing system: one for processing requests, i.e. queries or modifications to the documents, and another for data messages, i.e. intermediate results from other nodes. These two queues are shared between a *messaging thread* for the execution of the remote procedure calls and an *application thread*, which performs the actual retrieval functionality, as depicted in Figure 7.

In this approach, the implementation of the web service's associated function degenerates to simple data handling. The swimlane flowchart in Figure 8 illustrates the details of the queue-based, threaded communication. The actual handlers merely unmarshal the data from the message and either resend or enqueue them.

5. Conclusions. In this paper, we have argued that information retrieval for virtual organizations is an inherently distributed activity, which is complicated by the fact that queries must be answered sporadically and frequently. We have briefly reviewed the traditional notion of grid information retrieval as a job and argued that it suffers from the problem of prohibitively high index migration costs. As a remedy, we have introduced the concept of information retrieval as a service to prevent the negative impact of index migration on response time—and consequently user experience. We have sketched various characteristics of different approaches to distributed information retrieval and suggested that it is necessary to make specific choices. As an example, we have described the design decisions for our image retrieval system for e-science grids. Lastly, we have described the problems we encountered while implementing our approach as part of the open grid service architecture and sketched our resolution. Thus we have shown that it is indeed possible, albeit difficult, to build such a system. In the future, we hope to conduct a thorough performance analysis to empirically support the case for grid retrieval as part of the grid infrastructure.

Acknowledgement. The authors wish to acknowledge their funding by the Austrian ministry for science and research under grant no. P-142201-06.

REFERENCES

[1] Anderson D. P. BOINC: A System for Public-Resource Computing and Storage. In: Proc. GRID, IEEE, USA, 2004, 4–10.

[2] Anderson D. P., J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer. SETI@home: An Experiment in Public-Resource Computing. *Commun. ACM*, **45** (2002), No 11, 56–61.

[3] Androutsellis-Theotokis S., D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Comput. Surv.*, **36** (2004), No 4, 335–371.

[4] Baraglia R., D. Laforenza, F. Silvestri. SIGIR Workshop Report: The SIGIR Heterogeneous and Distributed Information Retrieval Workshop. *SIGIR Forum*, **39** 2005, No 2, 19–24.

[5] Batko M., V. Dohnal, P. Zezula. M-Grid: Similarity Searching in Grid. In: Proc. P2PIR, ACM, New York, 2006, 17–24.

[6] Baumgarten C. A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. In: Proc. SIGIR, ACM, USA, 1999, 246–253.

[7] Berberich K., M. Koubarakis, C. Tryfonopoulos, G. Weikum, C. Zimmer. MAPS: Approximate Publish/Subscribe Functionality in Peer-to-Peer Networks. In: Proc. ADPUC, ACM, USA, 2006, 1.

[8] Berka T. Distributed Image Retrieval on the Grid using the Vector Space Model. Master's thesis, University of Salzburg, Austria, 2009.

[9] Berka T., R. Kutil, M. Vajteršic. Fast Distributed Image Retrieval for e-Science Grids: Motivations and Challenges. In: Proc. ICCP, IEEE, 2010, 163–170.

[10] Berka T., M. Vajteršic. Fast Information Retrieval in the Open Grid Service Architecture. In: Proc. S3T, 2010, 202–206.

[11] T. Berka, M. Vajteršic. Dimensionality Reduction for Information Retrieval using Vector Replacement of Rare Terms. In: Proc. TM, 2011.

[12] Berka T., M. Vajteršic. Parallel Retrieval of Dense Vectors in the Vector Space Model. *Computing and Informatics*, **2** (2011), 247–265.

[13] Berretti S., A. del Bimbo, P. Pala. Merging Results for Distributed Content Based Image Retrieval. *Multimedia Tools Appl.*, **24**(2004), No 3, 215–232.

[14] Boden T. The Grid Enterprise — Structuring the Agile Business of the Future. *BT Technology Journal*, **22**(2004), No 1, 107–117.

[15] G. Brettlecker, D. Milano, P. Ranaldi, H. Schuldt. DelosDLMS – A Next-Generation Digital Library Management System. In: Proc. ICIAP,IEEE, USA, 2007, 83–88.

[16] Callan J., F. Crestani, H. Nottelmann, P. Pala, X. M. Shou. Resource Selection and Data Fusion in Multimedia Distributed Digital Libraries. In: Proc. SIGIR, ACM, USA, 2003, 363–364.

[17] Chowdhury A., G. Pass. Operational Requirements for Scalable Search Systems. In: Proc. CIKM, ACM, New York, 2003. 435–442.

[18] Coti C., T. Herault, S. Peyronnet, A. Rezmerita, F. Cappello. Grid Services for MPI. In: Proc. CCGRID, IEEE, USA, 2008, 417–424.

[19] Cuenca-Acuna F. M., T. D. Nguyen. Text-Based Content Search and Retrieval in Ad-hoc P2P Communities. In: Proc. NETWORKING, Springer, UK, 2002, 220–234.

[20] De Kretser O., A. Moffat, T. Shimmin, J. Zobel. Methodologies for Distributed Information Retrieval. In: Proc. Proceedings of the The 18th International Conference on Distributed Computing Systems, IEEE, Computer Society Washington, DC, USA, 1998.

[21] Deerwester S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, R. A. Harshman. Indexing by Latent Semantic Analysis. *JSIS*, **41**(1990), No 6, 391–407.

[22] Distributed Management Task Force. Specification for CIM Operations over HTTP, Version 1.2. Online publication of the DMTF. `http://www.dmtf.org/standards/published_documents/DSP200.html`, January 2007.

[23] Efraimidis P., C. Glymidakis, B. Mamalis, P. Spirakis, B. Tampakas. Parallel Text Retrieval on a High Performance Supercomputer using the Vector Space Model. In: Proc. SIGIR, ACM, USA, 1995, 58–66.

[24] Foster I., C. Kesselman. Computational Grids. The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufman, USA, 1999, 15–51.

[25] Foster I., H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. von Reich. The Open Grid Services Architecture, Version 1.5. `http://www.ogf.org/documents/GFD.80.pdf`, July 2006.

[26] Foster I. T. Globus Toolkit Version 4: Software for Service-Oriented Systems. In: NPC, (Eds H. Jin, D. A. Reed, W. Jiang), LNCS, Vol **3779**, Springer, 2005, 2–13.

[27] Frey V., T. Tannenbaum, M. Livny, I. Foster, S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, **5** (2002), No 3, 237–246.

[28] Gentzsch W. Sun Grid Engine: Towards Creating a Compute Power Grid. In: Proc. CCGRID, IEEE, USA, 2001, 35-36.

[29] Gil-Costa V., M. Marin, N. Reyes. Parallel Query Processing on Distributed Clustering Indexes. *JDA*, **7** (2009), No 1, 3 – 17.

[30] Gravano L., H. García-Molina, A. Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM Trans. Database Syst.*, **24** (1999), No 2, 229–264.

[31] Hua K. A., C. Lee. Handling Data Skew in Multiprocessor Database Computers Using Partition Tuning. In: Proc. VLDB, Morgan Kaufmann Publishers Inc, USA, 1991, 525–535.

[32] Hughes B., S. Venugopal, R. Buyya. Grid-based Indexing of a Newswire Corpus. In: Proc. GRID, IEEE, USA, 2004, 320–327.

[33] Jeong B.-S., E. Omiecinski. Inverted File Partitioning Schemes in Multiple Disk Systems. *IEEE Trans. Parallel Distrib. Syst.*, **6** (1995), No 2, 142–153.

[34] Kim Y. Grid Information Retrieval System for Dynamically Reconfigurable Virtual Organization. `http://www.ogf.org/documents/GFD.82.pdf`, February 2006.

[35] King I., C. H. Ng, K. C. Sia. Distributed Content-based Visual Information Retrieval System on Peer-to-Peer Networks. *ACM Trans. Inf. Syst.*, **22** (2004), No 3, 477–501.

[36] Kobayashi M., M. Aono, H. Takeuchi, H. Samukawa. Matrix Computations for Information Retrieval, Major and Outlier Cluster Detection. *JCAM*, **149** (2002), No 1, 119–129.

[37] Larson R. R., R. Sanderson. Grid-based Digital Libraries: Cheshire3 and Distributed Retrieval. In: Proc. JCDL, ACM, New York, 2005, 112–113.

[38] MacFarlane A., J. McCann, S. Robertson. PLIERS: A Parallel Information Retrieval System Using MPI.In: Euro PVM/MPI, LNCS, Vol **1697**, Springer, 1999, 674–674.

[39] Marin M., V. Gil-Costa, C. Bonacic. A Search Engine Index for Multimedia Content. In: Proc. Euro-Par, LNCS, Vol **5168**, Springer, 2008, 866–875.

[40] Marin M., V. Gil-Costa, C. Bonacic, R. Baeza-Yates, I. D. Scherson. Sync/Async Parallel Search for the Efficient Design and Construction of Web Search Engines. *Parallel Computing*, **36** (2010), No 4, 153–168.

[41] Hyperdatabase Infrastructure for Management and Search of Multimedia Collections. In: Proc. DELOS Workshop, 2004, 25–36.

[42] Ogilvie P., J. Callan. The Effectiveness of Query Expansion for Distributed Information Retrieval. In: Proc. CIKM, ACM, USA, 2001, 183–190.

[43] Powell A. L., J. C. French. Comparing the Performance of Collection Selection Algorithms. ACM Trans. Inf. Syst., **21**(2003), No 4, 412–456.

[44] Ranjan R., L. Chan, A. Harwood, S. Karunasekera, R. Buyya. Decentralised Resource Discovery Service for Large Scale Federated Grids. In: Proc. E-SCIENCE, IEEE, USA, 2007, 379–387.

[45] Rasolofo Y., F. Abbaci, J. Savoy. Approaches to Collection Selection and Results Merging for Distributed Information Retrieval. In: Proc. CIKM, ACM, USA, 2001, 191–198.

[46] Ribeiro-Neto B. A., R. A. Barbosa. Query Performance for Tightly Coupled Distributed Digital Libraries. In: Proc. DL, ACM, USA, 1998, 182–190.

[47] Salton G. Associative Document Retrieval Techniques Using Bibliographic Information. *J. ACM*, **10**(1963), No 4, 440–457.

[48] Salton G., D. Bergmark. Parallel Computations in Information Retrieval. In: Proc. CONPAR, Springer, UK, 1981, 328–342.

[49] Salton G., A. Wong, C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, **18** (1975), No 11, 613–620.

[50] Sornil O. Parallel Inverted Index for Large-Scale, Dynamic Digital Libraries. PhD thesis, Virginia Polytechnic Institute and State University, 2001.

[51] Stanfill C., B. Kahle. Parallel Free-Text Search on the Connection Machine System. Commun. ACM, **29** (1986), No 2, 1229–1239.

[52] Stanfill C., R. Thau, D. Waltz. A Parallel Indexed Algorithm for Information Retrieval. *SIGIR Forum*, **23** (1989), 88–97.

[53] Tang C., Z. Xu, S. Dwarkadas. Peer-to-Peer Information Retrieval using Self-Organizing Semantic Overlay Networks. In: Proc. SIGCOMM, ACM, USA, 2003, 175–186.

[54] Tang C., Z. Xu, M. Mahalingam. pSearch: Information Retrieval in Structured Overlays. *SIGCOMM Comput. Commun. Rev.*, **33** (2003), No 1, 89–94.

[55] Tomasic A., H. Garcia-Molina. Performance of Inverted Indices in Shared-Nothing Distributed Text Document Information Retrieval Systems. In: Proc. PDIS, San Diego, CA, USA, 1993, 8–17.

[56] Transier F., P. Sanders. Engineering Basic Algorithms of an In-Memory Text Search Engine. *ACM Trans. Inf. Syst.*, **29** (2010), No 1, 2–37.

[57] Tryfonopoulos C., S. Idreos, M. Koubarakis. Publish/Subscribe Functionality in IR Environments using Structured Overlay Networks. In: Proc. SIGIR, ACM, USA, 2005, 322–329.

[58] Venugopal S., R. Buyya, K. Ramamohanarao. A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. *ACM Comput. Surv.*, **38** (2006), Issue 1, Article No 3.

[59] Xi W., O. Sornil, M. Luo, E. A. Fox. Hybrid Partition Inverted Files: Experimental Validation. In: Proc. ECDL, Springer, London, 2002, 422–431.

[60] Xu J., J. Callan. Effective Retrieval with Distributed Collections. In: Proc. SIGIR, ACM, USA, 1998, 112–120.

[61] Zobel J., A. Moffat. Inverted Files for Text Search Engines. *ACM Comput. Surv.*, **38**(2006), Issue 2, Article No 6.

Tobias Berka
Department of Computer Sciences
University of Salzburg
Austria
e-mail: tberka@cosy.sbg.ac.at

Marian Vajteršic
Department of Informatics
Mathematical Institute
Slovak Academy of Sciences
Slovakia

Received April 30, 2011
Final Accepted August 25, 2011