

The Architecture of a New Storage and Retrieval System for Scientific Studies

Ratko Orlandic¹ and Sachin Kulkarni²

¹ *University of Illinois at Springfield, Computer Science Department,
Springfield, IL 62703, USA
rorla2@uis.edu*

² *Illinois Institute of Technology, Department of Computer Science,
Chicago, IL 60616, USA
kulksac@iit.edu*

Abstract

The proliferation of large data repositories inspires continuing interest in storage systems and organization. This paper describes basic principles and architecture of a new prototype system for data storage and retrieval, called D, which is designed to support advanced scientific studies in Data Grid environments. Unlike typical storage systems, the D* system enables a tighter integration of data-storage and data-mining technologies. Through the application of innovative retrieval and clustering techniques for high-dimensional data, D* can support high-performance data access and provide scientific applications useful insights into the data that can facilitate subsequent processes of data preparation and data mining. The system scales well with the increasing dimensionality of scientific data and works well on incremental loads of data.*

Keywords: scientific databases, data grids, data clustering, access methods, data dimensionality.

1. Introduction

The development of the new information-technology frameworks for scientific research, called the *Data Grids*, is one of the most important undertakings in computational sciences today. The main objective of these endeavors is to enable “geographically dispersed extraction of complex scientific information from very large collections of measured data” [1].

The Data Grids face difficult problems of scale arising from the large volumes and high dimensionalities of scientific data. To support efficient and scalable retrieval of multi-dimensional scientific data on a designated execution site, the storage and retrieval system should employ a clustered storage organization

that maximizes the densities of clusters on storage. The clustering algorithm appropriate for this application should effectively isolate areas with points, reducing the amount of their internal empty space. Furthermore, the clustering method should scale in terms of the number of points and number of dimensions.

Considering these factors, we have developed a new prototype system for efficient storage and retrieval of multi-dimensional data, called D* (Data storage and retrieval). The system is based on certain principles of organizing and accessing multi-dimensional data on storage discussed later in the paper. D* is also designed to enable a tighter coupling between data-storage and data-mining technologies. Through the application of innovative retrieval and clustering techniques for high-dimensional data, D* can support high-performance data access and provide scientific applications useful insights into the data that can facilitate subsequent processes of data preparation and data mining. The basic processes of the system include: clustering, space partitioning, loading, and retrieval based on region queries and similarity searching.

2. Data Grid Vision

The general thinking behind Data Grids can be illustrated best with an example. Suppose that a scientist starts an analysis program at a certain local site shared by multiple clients. An important aspect of a grid is that the system must first determine the resources required for the program’s execution, including the data that must be accessed. The information required to perform this step, which is usually referred to as *request interpretation*, is maintained in a certain metadata catalog [2].

The second step determines where in the wide-area (possibly worldwide) network the required resources reside and, based on that, where the program should be

executed. Since data may be replicated at multiple locations, this determination must rely on some replica catalog, which maps logical data descriptions to one or more physical locations [2]. This critical step is called *request planning*.

Once the appropriate site has been selected, the system must develop an execution plan, which drives the final step of the process called *request execution*. The event-driven request execution is performed against the data of participating sites with the help of some elaborate data-management facilities, which include data access and data transfer, as well as storage management and allocation.

Various dialects of this vision differ with respect to the general architecture, the subdivision of tasks among the components, and the interaction between the components. For example, Foster et al. [4] propose a layered architecture consisting of the following levels: fabric (physical resources, catalogs, and code repositories), connectivity (communication and

authentication middleware services), resource (computing, storage, access, and network protocols), collective (replica management, request interpretation, and request planning), and application (application-specific services). The European Data Grid architecture assumes a somewhat different but also layered organization comprised of many work packages [7].

In order to draw closer attention to the components of our primary interest, while ignoring the low-level issues of the fabric and connectivity layers, we assume here our own dialect of the Data Grid vision illustrated in Figure 1. This simplified architecture is divided into three large components, which we call GridAPS (Grid Application Services), GridDIS (Grid Directory Services), and GridDMS (Grid Data-Management Services). However, in order to avoid cluttering the figure, some depicted sites include only selected components.

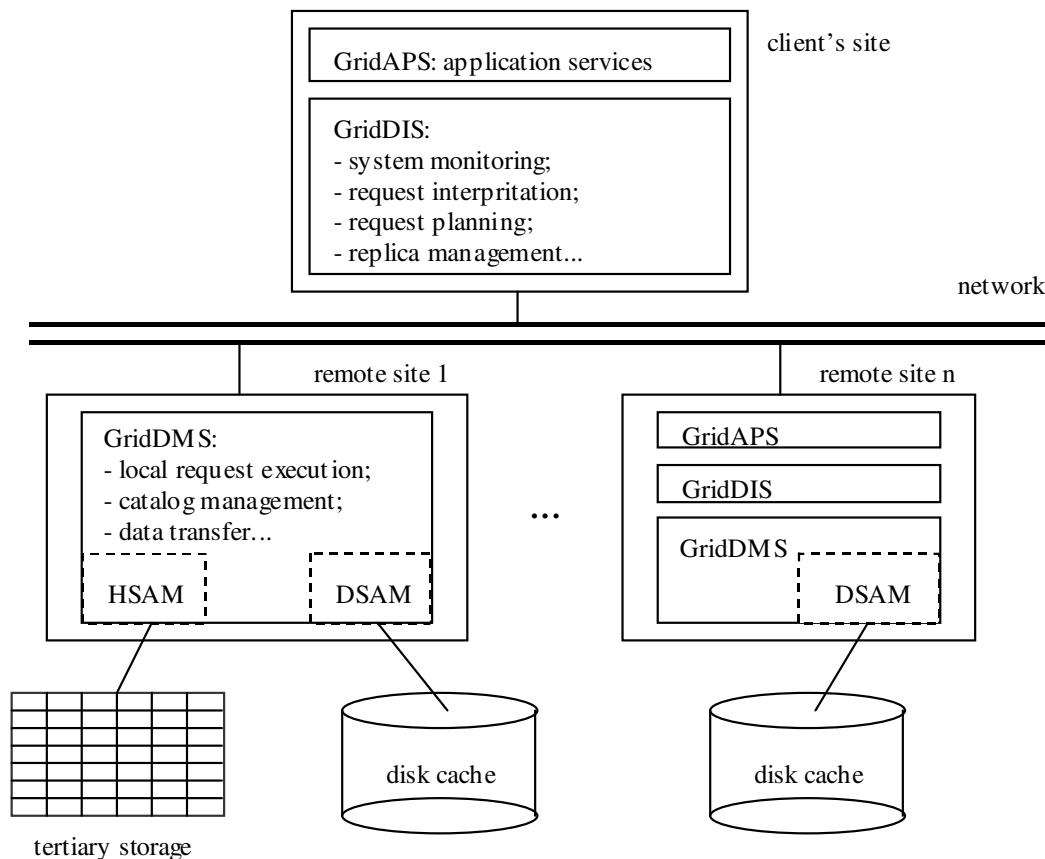


Figure 1: A schematic representation of a simplified Data Grid architecture.

In the figure, DSAM and HSAM represent two components for *storage and access management*. Both components are concerned with storage organization and data access, but they manage data on different types of storage. The *Disk Storage and Access Manager (DSAM)* dynamically handles data on a shared disk. The *Hierarchical Storage and Access Manager (HSAM)* interfaces with the underlying mass storage system to manage data on tertiary storage. In addition, it manages a disk cache for temporary storage of files. The indices for efficient access to the data are also maintained on the high-speed disk.

In a typical scientific application, each data item (*raw data*) is usually depicted by a large number d of descriptive attributes, generally represented as numeric or enumerated types. This *summary information* is interpreted as a feature vector defining a point in the d -dimensional space. While the raw data may also be of interest, the analytical studies are frequently concerned with summary information [14, 11]. If the analytical programs use raw data, each raw sample is stored next to its summary information. Otherwise, the summary items are organized separately in a way that is convenient for subsequent querying and analysis.

3. Massive Data and the Data Grid

The performance of the Data Grid operation has a significant bearing on the success of the Data Grid initiative. Since much of this performance depends on how fast one can find the desired items in the large data repositories, an appropriate storage organization is critical for the success of the Data Grids. Several research efforts, directly related to the Data Grid initiative, deal with the problem of managing massive data repositories [8, 13, 14]. Yet despite these efforts, the problem of storage organization is not sufficiently explored.

One can argue that the problem of massive data can never be fully resolved outside a distributed and replicated environment such as the Data Grid. In conjunction with request planning, data replication increases the availability of data and, by reducing network latencies, the performance of analytical processing. However, the adoption of the appropriate storage organization is also critically important. This is because, *when the size of a data collection exceeds certain limits, the capabilities of the given site in a distributed and replicated environment such as a Data Grid are determined not only by its computational and*

data resources, but also by the way the data is organized on storage.

For example, a simple temporal ordering of data on storage is more than adequate for certain tasks that access data in their temporal order, e.g. data replication or some complex analysis over the entire repository. However, when the repository is queried on the intrinsic properties of data, the items that satisfy the query appear to be randomly distributed across the storage. Therefore, a typical query must access many storage units, and usually only a small fraction of the retrieved data is relevant. In the Data Grid environments, this not only adversely affects the performance of analytical processing, but also has the effect of “blinding” the process of request planning. As noted earlier, the latter process selects the execution site for the given program based in part on an estimate as to how much data will be accessed.

An alternative idea is to cluster the data on storage so that a single access to the storage would retrieve an entire set of similar items [6]. For queries with relatively good selectivity, a fully clustered organization could easily generate 2 orders of magnitude fewer accesses to the storage than the temporal layout of data. Unfortunately, to cluster large data collections, one would need considerable computational and storage resources that may exceed the capabilities of even the richest super-computing centers. Moreover, since typical scientific data repositories are not static (data is usually streaming in over a prolonged period of time), this would require periodic re-clustering of the entire repository.

Another approach, exploiting the knowledge of the future access patterns, is a recurring theme in storage management. Some data repositories are clustered on a single feature [15] or a small subset of dimensions that tend to be restricted by typical queries more often than others are. However, since the values of the remaining dimensions become disseminated across the files in a virtually random fashion, the benefits of such clustering tend to be limited, and they could easily become lost in the associated costs.

Further complicating the matter is the fact that access patterns within or across applications change over time, sometimes violently. Therefore, any storage organization chosen for the given repository is likely to become inappropriate at one time or another. To address this problem, the data repository should be replicated. However, since different types of processing may require radically different storage organizations, the

layout of data on individual replica sites should not be the same throughout the enterprise.

In this context, the main challenge is to find a dynamically clustered storage organization that can take the advantage of any number of dimensions, possibly all dimensions, of data. In order to avoid a global restructuring of the data repository on any given site, while clustering data on any number of dimensions, we have proposed the idea of *two levels of data clustering* [9]. The idea involves a mix of static and dynamic decisions designed to achieve full benefits of data clustering at reasonable costs. The organization takes into account the typical representation of data in a scientific repository discussed earlier.

In the proposed organization [9], a static partition of the multi-dimensional space is used to organize data into clusters corresponding to different regions in the space (“*a priori* clustering”). For best results, the space partition should be derived after clustering an early sample of summary data. Even though the space partition is statically defined, the process of *a priori* clustering is performed dynamically while the data is arriving. Depending on the type storage, each *a priori* cluster can be maintained in a multi-dimensional index on disk (e.g., the KDB-tree [12] or R-tree [5]) or a group of files on tertiary storage.

Subsequently, the data of individual *a priori* clusters are organized into sub-clusters, each of which is assigned to a single unit of storage. This is called “*a posteriori* clustering”. If the data is maintained on the disk, the sub-clusters are formed implicitly by the indexing technique while the data is inserted. In this case, each sub-cluster represents a set of items in a

certain index page. In the case of tertiary storage, *a posteriori* clustering necessitates explicit clustering of data in an *a priori* cluster. Since at any given time *a posteriori* clustering operates on a subset of the data, it can be performed even in environments with limited resources.

The idea of the two-level clustering of data raises several questions. Which strategy should one use to partition the given space? How would one derive the appropriate space partition? What mechanism should be used for clustering of data? The answers to these questions must take into account environmental concerns, such as the volume and dimensionality of data as well as the nature of storage. They must also make sure that data is clustered on all relevant (externally selected) dimensions.

4. The D* System

Considering these factors, we have developed a prototype system for efficient storage and retrieval multi-dimensional data, called D*. The current implementation of the system assumes disk as the storage media. The basic processes of the system are illustrated in Figure 2. They include: data clustering, two variants of space partitioning, initial and incremental data loading, and data retrieval based on multi-dimensional region queries and similarity searching.

The *data clustering* module is based on a new clustering algorithm, called GARDEN_{HD}, which scales both in terms of the number and dimensionality of data [10].

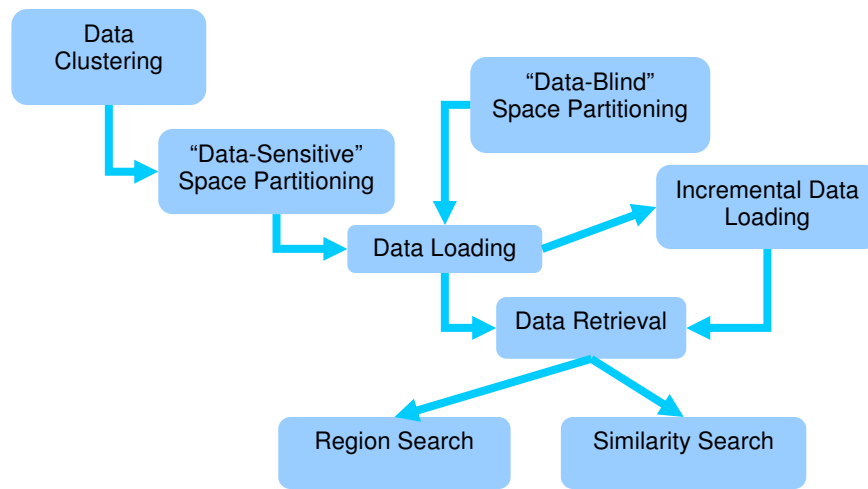


Figure 2: The architecture of the D* system.

This density-based clustering technique is organized around the idea of data space reduction, i.e. the process of detecting dense areas (dense cells) in the space. The method performs an effective and efficient elimination of empty areas that characterize large multi-dimensional spaces and an efficient agglomeration of the detected dense cells into larger clusters. In order to eliminate the need for dimensionality reduction, the clustering techniques exploits the properties of the Gamma partitioning strategy [11], which are discussed below. The result of GARDEN_{HD} is a compact representation of data that effectively captures the essential properties of the data distribution.

Operating on the compact representation of data produced by GARDEN_{HD}, the next process of the D* system performs a *data-sensitive (data-aware) space partitioning* using a novel partitioning technique, called DSGP (Data Sensitive Gamma Partitioning). The resulting space partition is based on the Gamma partitioning strategy. It is maintained by a memory-resident structure called the Gamma Filter. For each Gamma region, the Gamma Filter dynamically maintains its live region, i.e. the minimum bounding hyper-rectangle enclosing all points in the Gamma region.

For the environments where explicit clustering of data is not possible or viable, the D* system provides an option of deriving the Gamma space partition “blindly”, with no insight into the data distribution. This *data-blind*

space partitioning produces a fixed number of Gamma regions with equal volume.

In this organization, the Gamma Filter is used to impose a desirable behavior on traditional multi-dimensional access methods, such as B⁺-trees, KDB-trees, and R-trees, which normally do not exhibit this behavior in high-dimensional spaces. The space partition enables all dimensions (in the case of data-blind space partitioning) or all dimensions that differentiate the data clusters (in the case of data-sensitive space partitioning) to contribute to the search process. Due to the effects of live regions, the Gamma Filter also facilitates a significant reduction of the search space before accessing the storage. Moreover, due to the properties of Gamma partitions [10], the in-memory processing of the Gamma Filter is very efficient.

Each region in the space partition is assigned a separate multi-dimensional index. The indices, which are populated during *initial* and *incremental loading* of data, perform dynamic assignment of multi-dimensional data to index pages. Even though arbitrary multi-dimensional access methods can be used in this organization, the D* system employs the KDB-tree as the underlying access method in order to eliminate any overlap between the index regions. The KDB-tree indices perform an implicit “slicing” of Gamma regions into “tight” index regions that tend to be restricted along all data dimensions.

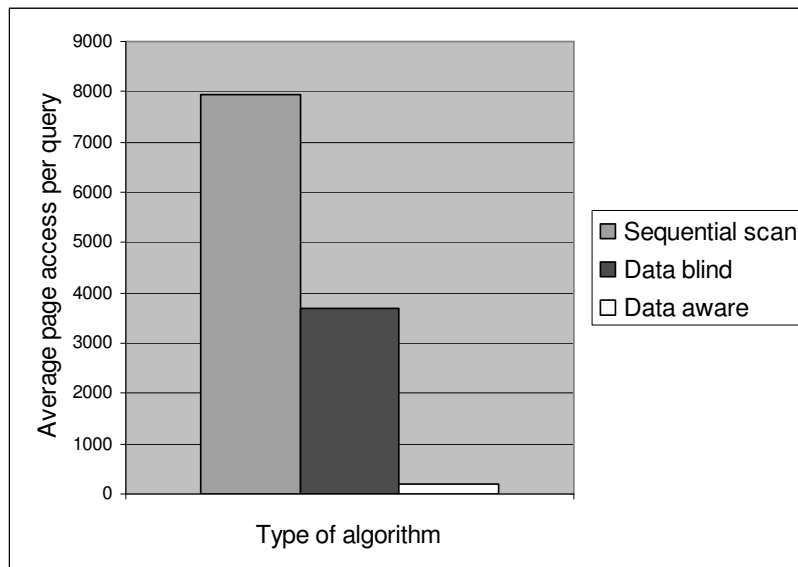


Figure 3: Experimentally observed performance of D* with or without explicit clustering.

The *retrieval* queries undergo several stages of refinement, both within the memory-resident Gamma Filter and the multi-dimensional indices on the disk. The supported retrieval processes include similarity (k nearest neighbor) searching and region queries. Fast computations on the Gamma Filter eliminate from inspection a potentially large number of data points without distance calculations. Since the bulk of filtering is performed in memory, a very good retrieval performance can be achieved.

In order to observe the effectiveness of the clustered organization, consider Figure 3, which shows the results of an experiment on a set of real 54-dimensional data with 580,900 items, called “covtype”, which was obtained from the UCI Machine Learning Repository. The experiment was conducted to compare the performance of the system with or without explicit clustering of data, i.e. with the “data-blind” and “data-aware” space partitioning, respectively. The performance was measured in terms of the average page accesses over 100 region queries with a randomly chosen center. Each query was restricted to 1% of the total space. The results are compared to simple sequential scan of the data.

High retrieval performance is only one of many benefits of the D* system. The system also enables a tighter coupling between data storage and data mining technologies. For example, since the GARDEN_{HD} clustering algorithm provides a compact but accurate insight into the data distribution, it is an effective data-reduction mechanism. On the reduced representation of data produced by GARDEN_{HD}, other processes of data preparation (e.g., imputation of missing values and dimensionality reduction) can be performed both efficiently and effectively. The D* system can also facilitate instance-based (k nearest neighbor) data classification [3] on dynamically growing sets of training data. As a result, D* can evolve from a storage and retrieval system into a full-fledged analytical engine for advanced scientific studies.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant no. IIS-0312266.

References

[1] P. Avery and I. Foster, “The GriPhyN Project: Towards Petascale Virtual Data Grids,” GriPhyN-14, 2001. <http://www.griphyn.org>

[2] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, “The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets,” *Journal of Network and Computer Applications*, 23:187-200, 2001.

[3] R. Fagin, R. Kumar, D. Shivakumar: “Efficient Similarity Search and Classification via Rank Aggregation,” *Proc. ACM SIGMOD Conf.*, 301-312, 2003.

[4] I. Foster, C. Kesselman and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organization,” *Int. Journal of High Performance Computing Applications*, 15(3):200-222, 2001.

[5] A. Guttman, “R-trees: A Dynamic Index Structure for Spatial Searching,” *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 47--54, 1984.

[6] K. Holtman, P. van der Stok and I. Willers, “Automatic Reclustering of Objects in Very Large Databases for High Energy Physics,” *Proc. Int. Database Engineering and Applications Symposium*, 132-140, 1998.

[7] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger and K. Stockinger, “Data Management in an International Data Grid Project,” *Proc. First IEEE/ACM Int. Workshop on Grid Computing*, 2000.

[8] T. Kurc, U. Catalyurek, C. Chang, A. Sussman and J. Saltz, “Visualization of Large Datasets with the Active Data Repository,” *IEEE Computer Graphics and Applications*, 21(4):24-33, 2001.

[9] R. Orlandic, “Effective Management of Hierarchical Storage Using Two Levels of Data Clustering,” *Proc. 20th IEEE / 11th NASA Goddard Conf. on Mass Storage Systems and Technologies*, 270-279, 2003.

[10] R. Orlandic, Y. Lai and W.G. Yee, “Clustering High-Dimensional Data Using an Efficient and Effective Data Space Reduction,” *Proc. ACM Conf. on Information and Knowledge Management CIKM’05*, 201-208, 2005.

[11] R. Orlandic, J. Lukaszuk and C. Swietlik, “The Design of a Retrieval Technique for High-Dimensional

Data on Tertiary Storage,” SIGMOD Record, 31(2):15-21, 2002.

[12] J.T. Robinson, “The K-D-B Tree: A Search Structure for Large Multidimensional Dynamic Indexes,” Proc. ACM SIGMOD Int. Conf. on Management of Data, 10-18, 1981.

[13] X. Shen and A. Choudhary, “A Distributed Multi-Storage I/O System for High Performance Data Intensive Computing,” Proc. IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid), 2002.

[14] A. Shoshani, L.M. Bernardo, H. Nordberg, D. Rotem and A. Sim, “Multidimensional Indexing and Query Coordination for Tertiary Storage Management,” Proc. 11th Int. Conf. on Scientific and Statistical Database Management (SSDBM), 214-225, 1999.

[15] C. Yu, S. Bressan, B.C. Ooi and K.-L. Tan, “Querying High-Dimensional Data in Single-Dimensional Space,” VLDB Journal 13(2):105-119 2004.