Hierarchical Storage Support and Management for Large-Scale Multidimensional Array Database Management Systems[1]

Bernd Reiner♦, Karl Hahn♦, Gabriele Höfling♦, Peter Baumann■

♦ FORWISS (Bavarian Research Center for Knowledge Based Systems)
Orleansstr. 34, 81667 Munich, Germany
{reiner, hahnk, hoefling}@forwiss.tu-muenchen.de
http://www.wibas.forwiss.tu-muenchen.de
■ Active Knowledge GmbH, Kirchenstrasse 88,
81675 Munich, Germany
baumann@active-knowledge.com
http://www.active-knowledge.de

Abstract. Large-scale scientific experiments or simulation programs often generate large amounts of multidimensional data. Data volume may reach hundreds of terabytes (up to petabytes). In the present and the near future, the only practicable way for storing such large volumes of multidimensional data are tertiary storage systems. But commercial (multidimensional) database systems are optimized for performance with primary and secondary memory access. So tertiary storage memory is only in an insufficient way supported for storing or retrieval of multidimensional array data. To combine the advantages of both techniques, storing large amounts of data on tertiary storage media and optimizing data access for retrieval with multidimensional database management systems is the intention of this paper. We introduce concepts for efficient hierarchical storage support and management for large-scale multidimensional array database management systems and their integration into the commercial array database management system RasDaMan.

1 Introduction

In many large-scale scientific domains, experimental and scanning devices or simulation programs generate large volumes of data. Examples are atmospheric data transmitted by satellites, climate-modeling simulations, flow modelling of chemical reactors, computational fluid dynamics and simulation of the dynamics of gene expressions. In principle, many natural phenomena can be modeled as spatio-temporal array data of some specific dimensionality. Their common characteristic is that a huge amount (hundreds of terabytes) of multidimensional discrete data (MDD) has to be

[1] This work was supported by the ESTEDI project (http://www.estedi.org). ESTEDI (European Spatio-Temporal Data Infrastructure for High-Performance Computing) is funded by the European Commission under FP5 grant no. IST-1999-11009.

stored. Actually the common state of the art of storing such large volumes of data is tertiary storage systems (mass storage systems), where data are stored as file. Typically, such tertiary storage systems have robot controlled tape libraries or jukeboxes. They provide automated access to hundreds or thousands of media (e.g. magnetic tapes, magneto optical tapes, CD-ROMs, DVDs, etc.). Concerning data access the main disadvantages are high access latency compared to hard disk devices and to have no direct access to specific subsets of data.

If only a subset of such a large data set is required, the whole file must be transferred from tertiary storage media. Taking into account the time required to load, search, read, rewind and unload several cartridges, it can take many hours to retrieve a subset of interest from a large data set. Entire files (data sets) must be loaded manually from the magnetic tape, even if only a subset of the file is needed for a further processing.

On the other hand, multidimensional database management systems (DBMS) offer efficient retrieval or manipulation of MDDs. They have extended query languages with special multidimensional operations like geometric, induced and aggregation operations [1, 2]. Concerning data storage the possibility of tertiary storage access is lacking, so mass data can't be managed directly by multidimensional array DBMS. As a consequence, in high performance computing applications DMBS are typically used for meta-data management only, where meta-data contains the information about the location of the data sets (on which medium).

The ESTEDI[1] project addresses the delivery bottleneck of large high-performance computing (HPC) results to the users with a flexible data management for spatio-temporal data. ESTEDI, an initiative of European database developers, software vendors and supercomputing centers, will establish an European standard for the storage and retrieval of multidimensional HPC array data. To this end, the multidimensional array DBMS RasDaMan will be enhanced with intelligent mass storage handling and optimized towards HPC [3].

The intention of this paper is a concept of hierarchical storage support, by combining the advantages of both techniques, storing big amounts of data and realizing efficient data access for retrieval with the multidimensional array DBMS RasDaMan. Thus overcoming their shortcomings particularly for scientific applications. This paper is organized as follows: Section 2 gives an overview about the system architecture and describes the new-implemented tertiary storage support functionality. In section 3 we present a concept for efficient storage of multidimensional data. Section 4 will have a focus on the tertiary storage management and support for large MDDs. Performance aspects can be found in section 5. Section 6 summarizes the achievements and gives an outlook on future work.

---

[1] ESTEDI (European Spatio-Temporal Data Infrastructure for High-Performance Computing) project (http://www.estedi.org) is funded by the European Commission under FP5 grant no. IST-1999-11009. Project Partner are FORWISS (DE), Active Knowledge (DE), DLR (DE), MPIM (DE), University of Surrey (GB), CCLRC (GB), Numeca (BE), Cineca (IT), CSCS (CH) and IHPC&DB (RU).

2 System Architecture

We have implemented the hierarchical storage management concept and integrated it into the first commercial multidimensional array DBMS RasDaMan. RasDaMan is distributed by Active Knowledge GmbH (http://www.active-knowledge.com). The DBMS RasDaMan (Raster Data Management) is designed for multidimensional array data and provides an extended multidimensional query language RasQL [1, 2]. The original version of RasDaMan didn't have a connection to tertiary storage systems apart from conventional backup. Within the ESTEDI project we have extended the RasDaMan kernel with easy to use functionality to automatically store and retrieve data to/from tertiary storage systems. In Figure 1 the architecture of the extended RasDaMan system can be seen.

The left side of the figure depicts the original RasDaMan architecture with the RasDaMan client, RasDaMan server and conventional DBMS (e.g. Oracle, which is used by RasDaMan as storage and transaction manager). The additional components for the tertiary storage interface are the Tertiary Storage Manager (TS-Manager), File Storage Manager and Hierarchical Storage Management System (HSM-System). The TS-Manager and File Storage Manager are included in the RasDaMan server. The HSM-System is a conventional product like SAM (Storage Archiving System) from LSC Incorporation or UniTree. Such an HSM-System (to the bottom right in Figure 1) can be seen as a normal file system with unlimited storage capacity. In reality, the virtual file system of HSM-Systems is separated into a limited cache on which the user works (load or store his data) and a tertiary storage system with robot controlled tape libraries. The HSM-System automatically migrates or stages the data to or from the tertiary storage media, if necessary.

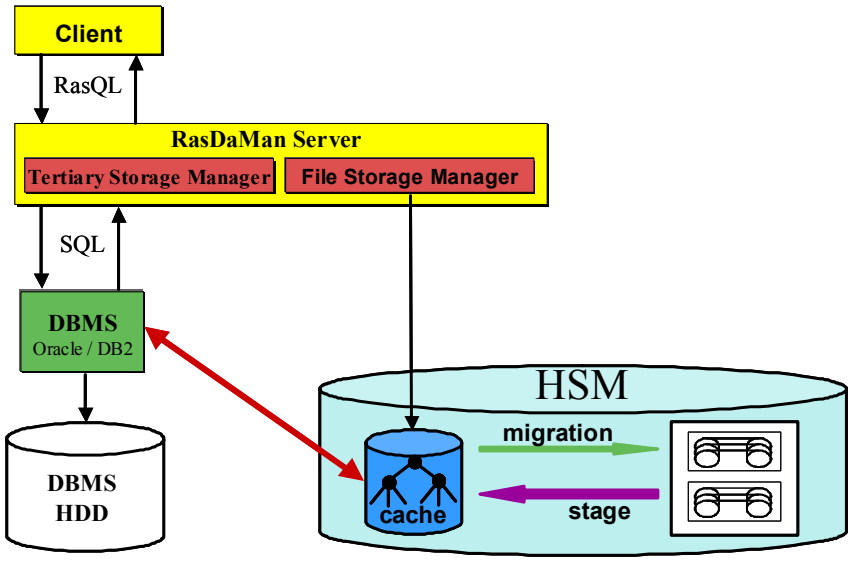


Figure 1: Extended RasDaMan architecture with tertiary storage interface

All in all two possibilities are available for connecting tertiary storage systems to the database system RasDaMan. First, an existing system like an HSM-System can be used. The other possibility is to develop a proprietary and new connection to a tertiary storage system. We decided to use conventional HSM-Systems for the connection of tertiary storage devices to RasDaMan. Such HSM-Systems have been developed to manage tertiary storage archive systems and to handle thousands of tertiary storage media (e.g. magnetic tape). Leading HSM-Systems are sophisticated and support robotic libraries of many manufactures. Another important reason for this decision was that such HSM-Systems with big robotic libraries (more than 100 TByte storage capacity) are already in use by the ESTEDI partners, which are also using RasDaMan.

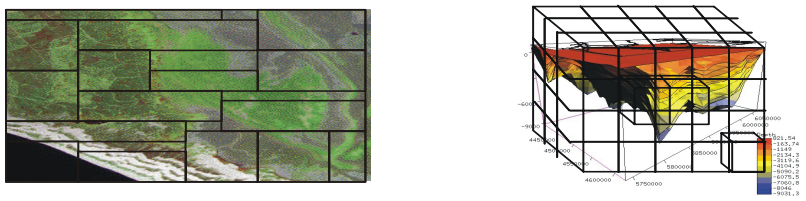
The new RasDaMan tertiary storage functionality is based on the TS-Manager module (shown in Figure 1). This TS-Manager is implemented and integrated into the RasDaMan kernel. If a query (RasQL) is executed, the TS-Manager knows whether the needed data sets are stored on hard disk or on a tertiary storage media. This meta-data used by the TS-Manger are stored in RasDaMan respectively the underlying DBMS (e.g. Oracle). The performance is much higher if the meta-data are stored permanently in the DBMS and not exported to tertiary storage media. If the data sets are on hard disk (in the DBMS), the query will be processed without specific tertiary storage management. This is the normal procedure of the RasDaMan system without tertiary storage connection. If the data sets are stored on one or more tertiary storage media, the data sets must be imported into the database system (cache area for tertiary storage data sets) first. The import of data sets stored on tertiary storage media is done by the TS-Manager automatically whenever a query is executed and those data sets are requested. After the import process of the data sets is done, RasDaMan can handle the data sets (cached in the DBMS) in the normal way. The TS-Manager of RasDaMan has information and meta-data about all data sets, for example where the data sets are stored, how the data sets are organized on media, etc.

After an insert of new data sets they will not be exported to tertiary storage media automatically. Sometimes it is necessary to store the data sets only in the DBMS if the data access time is critical, e.g. some users have frequent access to the data sets. The user can decide, whether the data sets are to be stored on hard disk (which is already done by the insert tool) or whether the data sets should be exported on tertiary storage media. These two possibilities are flexible in several cases. For example, data sets are very often requested by users at the beginning (insert time of data sets) and after several months the data sets are less important for these users. In this case, the data sets first inserted into the DBMS can be exported to tertiary storage media after several months. If the data sets should only be stored in the DBMS, the user does not need to do anything else. When the data sets should be exported to tertiary storage media, one has to issue an export command. For exporting data sets a new statement was integrated into the RasDaMan query language (EXPORT FROM <data-set> WHERE <condition>). We can export complete data sets or only specific parts to the HSM-System (i.e. to tertiary storage media). Before we will describe details about tertiary storage support for multidimensional DBMS (section 4) we have to discuss basics about efficient storage of large multidimensional data in the following section.

3 Efficient Storage of Large Multidimensional Data

In this section we want introduce techniques of storing large multidimensional data efficiently. Later on this techniques will be extended for tertiary storage access. MDD, resulting from sampling and quantizing of phenomena like temperature or velocity in multidimensional space or represented statistical data, is a commonly used data type, in particular the 2D special case of raster images. A MDD object consists of an array of cells of some base type (e.g. integer, float or arbitrary complex types), which are located on a regular multidimensional grid. For example an MDD can be a 4 dimensional spatio-temporal object, resulting from scientific experiments or simulations and can become very large. Since linear storage of those MDDs as binary large objects (BLOB) makes it impossible to access only specific areas of interest from one MDD. Special multidimensional array DBMS (e.g. RasDaMan) are required for efficient MDD support.

The insufficient support for multidimensional arrays in commercial DBMS has inspired research on providing DBMS services (query language, transactions) for MDD, aiming at application areas different from traditional DBMS, for instance, scientific data management, geographic information systems, environmental data management, storage structures techniques. An often discussed approach is chunking or tiling of large data sets and is commonly used for multidimensional arrays in different application areas [5, 7, 13]. Chunking means subdividing of multidimensional arrays into disjoint sub-arrays with the same dimensionality as the original array. All chunks have the same shape and size and are therefore aligned. Tiling is more general than chunking, because sub-arrays don't have to be aligned or have the same size. MDDs can be subdivided in regular or arbitrary tiles. Regular or aligned tiling is identical with chunking and is the most common tiling concept in array systems. Further information about tiling strategies can be found in [8]. Figure 2 depicts examples of arbitrary and regular tiling.



Arbitrary Tiling (2D) Regular Tiling (3D)

Figure 2: Arbitrary and regular tiling strategy

If tiling is supported by DBMS (e.g. RasDaMan), it is possible to transfer only a subset of large MDDs from the database (or tertiary storage media) to client applications, because every tile is stored as one single BLOB in the relational database system. This will mainly reduce access time and network traffic. The query response time scales with size of query box, not with size of MDD. Now we can handle large data sets efficiently.

In the commercial DBMS RasDaMan, BLOBs (tiles) are the smallest units of data access. In order to manage these units in main memory, a limit on tile size is usually

imposed. This limit should be set to multiples of one database page. Typical sizes of tiles stored in RasDaMan range from 32 KByte to 640 KByte and are optimized for hard disk access [8]. As we will see those tile sizes are much to small for data sets held on tertiary storage media. It is necessary to choose different granularities for hard disk and tape access, because they differ significantly in their access characteristics. Hard disks have fast random access, whereas tape systems have sequential access with much higher access latency. More details about performance of tertiary storage devices can be found in [10]. It is important to use data management techniques for efficiently retrieving arbitrary areas of interest from large data sets stored on tertiary storage devices. We have to find some partitioning techniques that partition data sets into clusters based on optimized data access patterns and storage device characteristics.

4 Tertiary Storage Support for Multidimensional DBMS

The average access time (e.g. load, switch, positioning time) for tape systems (20 – 180s) is by order of magnitude slower than for hard disk drives (5 - 12ms), whereas the difference between transfer rate of hard disk and tertiary storage systems isn't so important (factor 2 or 3). The main goal is to minimize the number of media load and search operations and to reduce the access time of clusters read from tertiary storage system when subsets are needed [6]. Generally there are several possibilities for optimization of the access costs. An important point is the granularity of the data stored on media and read from media. Particularly the size of the data blocks, which are moved from tape to the hard disk cache, is critical. On the one hand, preferably small data blocks should be transferred over the network to minimize the network traffic. On the other hand, the transfer rate of tertiary storage systems is not bad and the quantity of tape access should be minimized. On this reason the size of data blocks stored on tertiary storage media should be more than 100 MByte. It is unreasonable to increase the RasDaMan MDD tile size (32 – 640 KByte), because then we would loose the advantage of transferring only small subsets of the MDDs to the client application. A promising idea is to introduce an additional data granularity as provided by the so-called Super-Tile concept. In the following section we will present the newly developed Super-Tile concept.

4.1 The Super-Tile Concept

The main goal of the new Super-Tile concept is a smart combination of several small MDD tiles to one Super-Tile for minimizing tertiary storage access costs. Smart means to exploit the good transfer rate of tertiary storage devices and to take advantage of other concepts like clustering of data. In ESTEDI, where RasDaMan is used as multidimensional DBMS the multidimensional index R+ tree for realizing fast random access of arbitrary tiles stored on disk is used [9, 12]. The conventional R+ tree index structure of the multidimensional DBMS was extended to handle such Super-Tiles stored on tertiary storage media. This means that information regarding which tiles are stored on hard disk and which are stored on tertiary storage media must be integrated

into the index. Tiles of the same subindex of the R+ tree are combined into a Super-Tile and stored on tertiary storage medium. The specific tertiary storage manager knows on the basis of the structure of the multidimensional R+ tree that all tiles below this subindex (subtree of a R+ tree node) are combined to a Super-Tile and stored on the same tertiary storage media. A node of such a subindex is called a Super-Tile node.

Figure 3 depicts an example of the R+ tree index of one MDD with the corresponding Super-Tile nodes. Only complete nodes of the R+ tree can become Super-Tile nodes (e.g. the ST1 node in Figure 3). This means that all tiles of the included leaf nodes (in an R+ tree, data is only stored in leaf nodes) of one Super-Tile node are combined to one Super-Tile (light gray circle/oval). As a consequence, Super-Tiles can only be multiples of tiles.

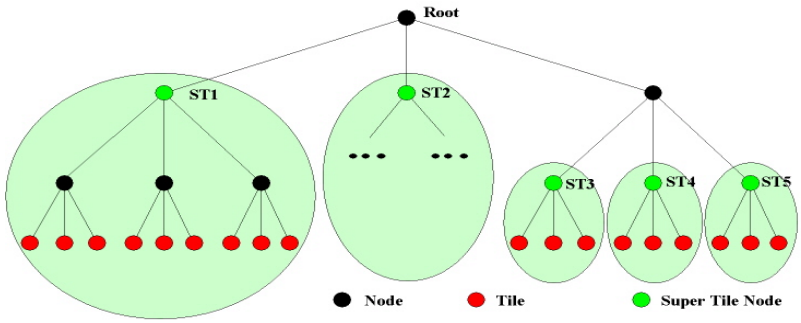


Figure 3: Example R+ tree index of one MDD with Super-Tile nodes

Super-Tile nodes can be on arbitrary levels of the R+ tree. In the example of Figure 3 we have 5 Super-Tile nodes (ST1 - ST5). The Super-Tile nodes ST1 and ST2 are on the second level of the tree and the corresponding Super-Tiles include 9 Tiles. The Super-Tile nodes ST3, ST4 and ST5 are on the third level of the tree and the corresponding Super-Tiles include only 3 Tiles.

We developed an algorithm for computing Super-Tile nodes inside the R+ tree. A general restriction of the Super-Tile algorithm is that only one Super-Tile node is contained in one path (from node to root) of the R+ tree. The algorithm traverse the R+ tree bottom-up and compute the size of the data (tiles), which is referenced in the subtree. For the detection of Super-Tile nodes the predefined size of the Super-Tiles is used. If the summarized size of the subtree of an index node is greater than the predefined Super-Tile size all child nodes get Super-Tile nodes. If the summarized size of tiles contained in the subtree of a node is smaller than the predefined Super-Tile size this node remains a candidate. The user can define suitable Super-Tile sizes, optimized for the data and tertiary storage access characteristics. If the user defines no Super-Tile size, a default maximum size of 200 MByte will be used. Extensive tests have shown that this Super-Tile size shows good performance characteristics in most cases and the Super-Tile algorithm produces Super-Tiles, which have about 60% to 80% of the size of the predefined Super-Tile size. More details about determining optimal file sizes on tertiary storage media can be found in [4].

Super-Tiles are the access (import/export) granularity of MDD on tertiary storage media. The retrieval of data stored on hard disk or on tertiary storage media is transparent for the user. Only the access time is higher if data stored on tertiary storage media. In order to improve performance, the whole index of all data (held on hard disk and tertiary storage media) is stored on hard disk. We will now discuss two further strategies for reducing tertiary storage access time, clustering and caching [11].

4.2 Strategies for Reducing Tertiary Storage Access Time

Clustering

Clustering is particularly important for tertiary storage systems where positioning time of the device is very high. The clustering of data sets reduces the positioning and exchange time of tertiary storage media. Clustering uses the spatial neighborhood of tiles within the data sets. Clustering of tiles according to spatial neighborhood on one disk or tertiary storage system proceed one step further in the preservation of spatial proximity, which is important for the typical access patterns of array data, because users often request data using range queries, which implies spatial neighbourhood. The used R+ tree index to address MDD's tiles already defines the clustering of the stored MDDs. With the developed Super-Tile concept we can distinguish intra Super-Tile clustering and inter Super-Tile clustering.

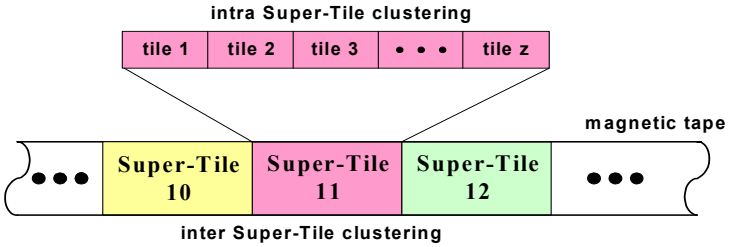


Figure 4: Inter and intra Super-Tile clustering of tiles stored on magnetic tape

The implemented algorithm for computing the Super-Tiles (see chapter 4.1) maintains the predefined clustering of subtrees (of Super-Tile nodes) of the R+ tree index and realizes intra Super-Tile clustering. Inside one Super-Tile we have clustering, i.e. neighborhood of the spatial location of the included tiles. The export algorithm (export of Super-Tiles to tertiary storage) realizes the inter Super-Tile clustering within one MDD. The various Super-Tiles of one MDD are written to tertiary storage media in the clustered order (predefined R+ tree clustering). Inter and intra Super-Tile clustering of tiles stored on magnetic tape is shown in Figure 4.

Caching

In order to reduce expensive tertiary storage media access the underlying DBMS of RasDaMan can be used as hard disk cache for data sets held on tertiary storage media. The general goal of caching tertiary storage data (Super-Tile granularity) is to mini-

mize expensive loading, rewinding and reading operations from slower storage levels (e.g. magnetic tape). In the tertiary storage version, requested data sets held on tertiary storage media are migrated to the underlying DBMS of RasDaMan (see Figure 1). The migrated Super-Tiles are now cached in the DBMS. After the migration the RasDaMan server transfers the requested tiles from the DBMS to the client application. For a better differentiation of data stored persistent in the DBMS (i.e. how data sets were held in the original version of the multidimensional DBMS) and data, which are only be, cached in the DBMS we call the storage location of the cached objects DBMS cache area.

The advantage of caching is that the data held on tertiary storage media doesn't have to be imported to the DBMS cache area for every request. The import of the requested data from the tertiary storage media is extremely expensive because the transfer from tertiary storage system to hard disk is slow (media loading time, media exchange time, media rewind time, seek time, read time, transfer rate, etc.). The second request to this data is very fast because the data is already held in the DBMS cache area. The tertiary storage Cache-Manager only evicts data (Super-Tiles granularity) from the DBMS cache area if necessary (the upper limit of the cache size is reached). At the moment the LRU (Least Recently Used) replacement strategy is supported and shows good performance. Other replacement strategies will be tested with tertiary storage devices. The general goal of these policies is to substitute only the data (Super-Tiles) that is least likely to be reused. In this research area, optimization algorithms especially for tertiary storage systems must be found.

5 Performance Aspects

First we show the main advantage of the developed hierarchical storage support for large-scale DBMS. The ESTEDI partners typically have to load whole MDDs (stored as single files) from tertiary storage devices, even if only a subset is required for further analysis. This means that the request response time scales with the size of the stored MDD. Using RasDaMan and the integrated tertiary storage access subsets of MDDs can be loaded and for this reason the query response time scales with the size of the query box and not with the size of the MDD (see Figure 5).

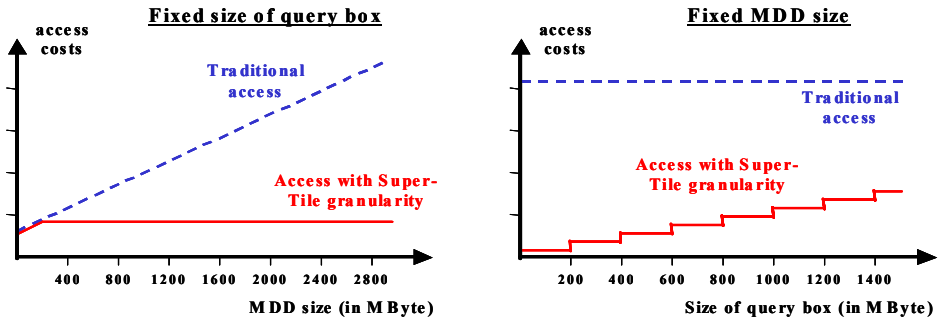


Figure 5: Access comparison with fixed query box size and with fixed MDD size

In this example we assume a Super-Tile size of 200 MByte. On the left side of Figure 5 we have a fixed query box size of 400 MByte and the MDD size is increasing, which means two Super-Tiles must be loaded at least and the access costs are stagnating. In the traditional case the access costs are scaling with the MDD size. If the MDD size is smaller than the predefined Super-Tile size the complete MDD has to be loaded and therefore the same access time is expected. Typically such small MDDs are not stored on tertiary storage medium. On the right side of Figure 5 the MDD size is fixed (3 GByte) and the query box is increasing. If we have access with Super-Tile granularity the access costs are increasing step by step. Otherwise with traditional access we have constant access costs, as the complete MDD must be loaded.

Now we briefly discuss the performance of the export and retrieval functionality of the new tertiary storage version of RasDaMan. The export of data sets to tertiary storage media is very fast because the data sets just have to be written to the virtual file system of the HSM-System. This means storing data sets on the hard disk cache of the HSM-System. The migration of the data sets from the HSM cache to the tertiary storage media does not concern the RasDaMan system. For the retrieval functionality three cases must be distinguished. In the first case we assume that the data sets needed are already held in the DMBS cache area of RasDaMan. This request operation is very fast, because no import of data from the HSM-System has to be done. In the second case the data sets required are held in the hard disk cache of the HSM-System. This is quite likely because the size of the HSM cache is normally hundreds of GByte. In this case the import of the data sets is as fast as the export of the data sets because the data sets don't have to be staged from the tertiary storage media. This access is about factor 1.8 to 3 slower compared to normal DMBS access (dependent on network traffic, transfer rate, etc.). We assume for the third case that the data sets requested are not held in the HSM cache. This means the HSM-System must first stage the data sets needed from the tertiary storage media to the HSM cache and then the data sets are transferred to the RasDaMan system. Compared with DBMS access we measured a slowdown of factor 3 to 10 (dependent on tertiary storage device, transfer rate, etc.).

6 Conclusion and Future Work

The initial point of our development was that members of the ESTEDI project have large volumes of multidimensional array data, generated by scientific simulations, which are stored as files on tertiary storage media.

The main goal of our development was to realize a fast and efficient access to tertiary storage media and provide access functionality like retrieval of subsets as common for DBMS since a long time. This means the request response time scale now with the size of the query box, not with the size of MDD like the traditional case of the ESTEDI partner. In our approach we use a multidimensional array DBMS for optimal storage, retrieval and manipulation of large MDD. A major bottleneck of the commercial multidimensional array DBMS RasDaMan is that it was originally not designed to use tertiary storage media for storing hundreds of terabytes (up to petabytes). To handle the data amounts stored on tertiary storage an interface was presented to connect

tertiary storage systems to the multidimensional array DBMS RasDaMan. Consequently, we created a hierarchical storage support and management system for large-scale multidimensional array DBMS, which is specifically designed and optimized (using clustering and caching of Super-Tiles) for storing multidimensional array data on tertiary storage media.

Future work will be the development of scheduling techniques for multidimensional array data streamlined for Super-Tiles. Scheduling for tertiary storage media means the optimization of the media read order. This optimization reduces expensive media seek and exchange operations. The focus is on scheduling policies that process all requests on a loaded medium before exchanging it in the loading station of the robotic library. We can differ intra and inter query scheduling. Intra query scheduling will optimize the request order within one query. Inter query scheduling can be done in RasDaMan by examining the query queue. If the actual query needs Super-Tiles from one MDD and a further query of the query queue also needs several Super-Tiles from the same MDD, all needed Super-Tiles will be imported at the same time into the RasDaMan cache area.

References

1. Active Knowledge GmbH: RasDaMan Query Language Guide version 5.0, Active Knowledge GmbH, Munich 2001
2. Baumann P.: A Database Array Algebra for Spatio-Temporal Data and Beyond, Proc. of the 4th Int. Workshop on Next Generation Information Technologies and Systems (NGITS), p. 76-93, 1999
3. Baumann P.: Array Databases Meet Supercomputing Data – the ESTEDI Project, Internal ESTEDI Report, 2000
4. Bernardo L. M., Nordberg H., Rotem D., Shoshani A.: Determining the Optimal File Size on Tertiary Storage Systems Based on the Distribution of Query Sizes, Proc. of the 10th Int. Conf. on Scientific and Statistical Database Management, p. 22-31, 1998
5. Chen L. T., Drach R., Keating M., Louis S., Rotem D., Shoshani A.: Efficient organization and access of multi-dimensional datasets on tertiary storage, Information Systems, vol. 20, no. 2, p. 155-183, 1995
6. Chen L. T., Rotem D., Shoshani A., Drach R.: Optimizing Tertiary Storage Organization and Access for Spatio-Temporal Datasets, NASA Goddard Conf. on Mass Storage Systems, 1995
7. Furtado P. A., Baumann P.: Storage of Multidimensional Arrays Based on Arbitrary Tiling, Proc. Of the ICDE'99, p. 480-489, 1999
8. Furtado P. A.: Storage Management of Multidimensional Arrays in Database Management Systems, PhD Thesis of Technical University Munich, 1999
9. Gaede V., Günther O.: Multidimensional Access Methods, ACM Computing Surveys, vol. 30, no. 2, 1998
10. Johnson T., Miller E. L.: Performance Measurements of Tertiary Storage Devices, Proc. of the 24th VLDB Conf., New York, USA, 1998
11. Reiner B.: Tertiary Storage Support for Multidimensional Data, VLDB Supercomputing Databases Workshop, Rome, September 2001
12. Rigaux P., Scholl M., Voisard A.: Spatial Databases – with application to GIS, Academic Press, 2002
13. Sarawagi S., Stonebraker M.: Efficient Organization of Large Multidimensional Arrays, Proc. of Int. Conf. On Data Engineering, volume 10, p. 328-336, 1994