

Community-Driven Data Grids*

Tobias Scholl
Supervised by Alfons Kemper (alfons.kemper@in.tum.de)

Department of Computer Science
Technische Universität München
Munich, Germany
tobias.scholl@in.tum.de

ABSTRACT

Beyond already existing huge data volumes, e-science communities face major challenges in managing the anticipated data deluge of forthcoming projects. *Community-driven data grids* target at domain-specific federations and provide a distributed, collaborative data management by employing dominant data characteristics (e. g., data skew) and query patterns to optimize the overall throughput. By combining well-established techniques for data partitioning and replication with Peer-to-Peer (P2P) technologies we can address several challenging problems: data load balancing, handling of query hot spots, and the adaption to short-term burst as well as long-term load redistributions.

1. INTRODUCTION

E-science projects of many research communities, e. g., biology, the geosciences, high-energy physics, or astrophysics, face huge data volumes from current experiments. Due to the expected data rates of upcoming astrophysical projects, e. g., the *Panoramic Survey Telescope and Rapid Response System (Pan-STARRS)*, producing several terabytes a day, current centralized data management approaches [23] offer only limited scalability.

Combining and correlating information from various experiments or observations are the key for finding new scientific insights. Mostly, data results are currently provided to the whole community by the institutions conducting the experiments, hosting the data on their own servers. This approach of autonomous data management is not well-suited for the application scenario just described as each data source needs to be queried individually and (probably large) intermediate results need to be shipped across the network.

1.1 Problem Statement

In order to deal with the sheer size of their resulting data, researchers within a community join forces in *Virtual Organizations* and build infrastructures for their scientific federations, so-called *data grids* [24]. These data grids interconnect dedicated resources

*This work is part of the AstroGrid-D project and of D-Grid and is funded by the German Federal Ministry of Education and Research (BMBF) under contract 01AK804F.

Permission to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than VLDB Endowment must be honored.

Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists requires prior specific permission and/or a fee. Request permission to republish from: Publications Dept., ACM, Inc. Fax +1 (212)869-0481 or permissions@acm.org.

using high-bandwidth networks and enable researchers to share and correlate their data sets within the community. In order to ensure reproducibility, published data sets are not changed. Instead, new additional versions are made available. Moreover, an *increasing popularity* within the user community puts high demands on the various architectural design choices, such as providing high query throughput. Further challenging aspects are *skewed data distributions* in the data sets as well as *query hot spots*.

E-Science communities need support for building an infrastructure for data sharing that 1) is able to directly deal with several terabytes or even petabytes of data, 2) integrates the existing high-bandwidth networks with several hundred nodes within the communities, and 3) offers high throughput to cope with a steadily growing user community. Given these requirements, *how can we provide a scalable infrastructure that is capable of using the shared resources and performs data as well as query load balancing?*

1.2 Our Approach

Community-driven data grids enable communities to individually address the two major issues, *high-throughput data management* and *correlation of distributed data sources*. With HiSbase, our prototypical implementation of a community-driven data grid, we explore design alternatives for data grids between the both (impractical) extremes of a centralized community warehouse and a fully-replicated data grid. We propose a decentralized and scalable approach to scientific data management using the existing available capacities—both CPU and main memory—of the community network resources. Based on distributed hash tables (DHTs), we partition data according to predominant query patterns and not according to the original data source. In case of multi-dimensional data, space filling curves preserve the spatial locality between closely related data.

1.3 Application Scenario

In astrophysics as well as in other scientific communities we expect exponential growth rates in addition to already existing enormous data volumes. Furthermore, the increasing access rates by researchers to these information systems support the need for a scalable and efficient data management. The correlation and combination of observational data or data gained from scientific simulations (e. g., covering different wave bands) is the key for gaining new scientific insights. The creation of likelihood maps for galaxy clusters [21] or the classification of spectral energy distributions [9] are examples for such applications. The Ph.D. project described in this paper is conducted in the context of AstroGrid-D [4], the astrophysics community project within the German e-science and Grid Computing initiative D-Grid. Collaborating with our partners from leading German astrophysical institutes, our goal is to provide a

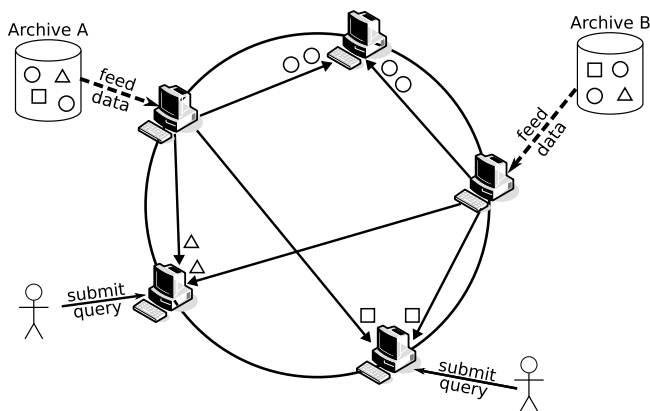


Figure 1: HiSbase architecture

scalable and distributed data management infrastructure to research communities, especially for the astrophysics community as one of the most active communities with regards to Grid-based e-science solutions for collaborative scientific research.

The rest of the paper is organized as follows. After describing some of the relevant literature in Section 2, we describe in Section 3 the main characteristics of HiSbase, our prototype implementation of community-driven data grids. In Section 4, we describe some of the load balancing issues we are currently working on and plan to address during the course of our work. Finally, we conclude in Section 5.

2. RELATED WORK

In recent years, research on peer-to-peer (P2P) systems has contributed significantly in the context of decentralized data processing and scalable management of fast growing data volumes. Especially, DHT-based systems, like Chord [22] and Pastry [17], offer highly scalable, failure-resilient approaches that allow efficient point-based queries even in the presence of data skew. They achieve this by uniformly distributing the data to the participating peers, however, at the cost of destroying the locality relationship of individual objects. This technique prohibits an efficient support for range queries, but these play a fundamental role in many scientific disciplines, especially astrophysics.

In contrast, P-Ring [1] supports range queries by additionally guaranteeing logarithmic search costs especially for skewed data distributions. To achieve their data load balancing guarantees, P-Ring uses so-called *helper peers*, peers that support other nodes which are overloaded due to many data insertions. An overloaded node redistributes its data between itself and its helper peers. Nodes whose load drops beyond a certain threshold either redistribute their data or hand off their data completely and offer themselves as helper peers. Compared to HiSbase, P-Ring assumes a uniform workload distribution and does not address query hot spots.

HotRoD [13] proposes an approach for dealing with query hot spots on one-dimensional data. Data is distributed on a key ring according to an order-preserving hash function in order to support range queries. Each node collects statistics about query frequencies and query ranges for achieving a uniform query load distribution. Popular (*hot*) data of overloaded peers are replicated and stored on additional virtual, rotated key rings. Data skew, however, is not addressed by HotRoD.

Multiple virtual rings are also used by BORG [8] in order to manage replicated data objects and their updates in a decentralized

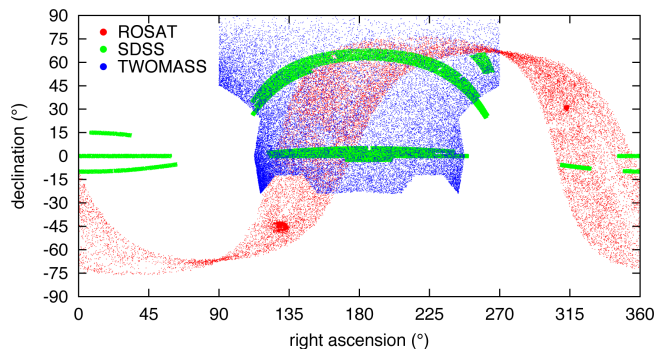


Figure 2: Data sample of the three astrophysical catalogs ROSAT, SDSS, and TWOMASS

fashion. Compared to HotRoD all replicas of an object are stored in their individual ring and nodes participate in rings of data objects they maintain. Updates within the BORG system can be initiated by any node which stores a replica. A node from the particular replication ring is selected as temporary master in order to coordinate the update procedure. However, the authors do not discuss data skew and query hot spots.

In [2], the authors show that caching is not sufficient as single means for addressing hot spots and query load imbalance. Redundant routes which provide fault-tolerance network connections can also serve during query load balancing. A weight function, which considers both message routing load and processing load of the individual nodes, selects the least loaded route for communication. It is open, whether these techniques can be applied to the envisioned data-intensive application scenarios of e-science communities.

SD-Rtrees [3] are a scalable distributed data structure, which is built especially for point- and range-queries for large-scale spatial data sets. Based on the R-tree, new servers are added to network in case of an overload and affected data objects are distributed evenly. Users interact with an SD-Rtree using an image of the distributed data structure. This image can possibly be outdated due to data redistributions but it is incrementally updated via special messages. Replication is currently not considered in this data structure.

3. COMMUNITY-DRIVEN DATA GRIDS

HiSbase, our prototype for community-driven data grids, partitions the available data sets using a partitioning scheme tailored to the individual community and distributes or, if applicable, replicates the data to the community data nodes.

3.1 Architectural Overview

Figure 1 shows the high-level functionality of HiSbase from a users point of view. Distributed data providers (e. g., archives) send their data to a known, possibly adjacent, HiSbase node into the community network and all stations can submit queries. In the following, we assume that schema information of the used data sets is known and data sets either have been preprocessed or at least have a common sub-schema. Orthogonal techniques for data fusion [11] or schema-matching [15] can be integrated in this phase to prepare the scientific data sets. Data objects are distributed in such a fashion that logically related objects (denoted by their geometric shape) are mapped to the same node, even if they origin from different data archives. Thus, such data can be correlated locally.

We use the DHT-structure *Pastry* [17] for managing the HiSbase

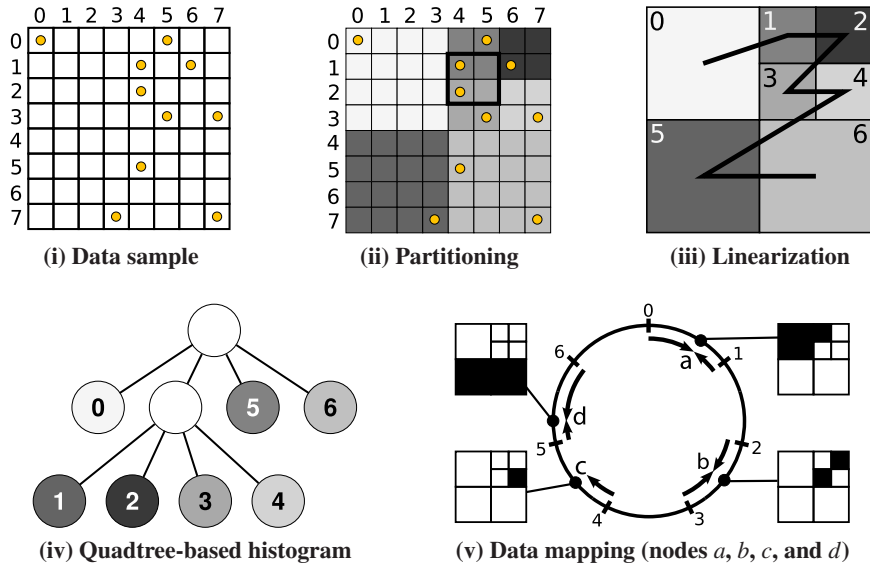


Figure 3: Processing of an illustrative example with a quadtree-based histogram

nodes and the message routing using the overlay-network. Similar to Chord [22], Pastry uses a one-dimensional key ring. In comparison to other DHT-systems [16, 22], Pastry optimizes the message routing. During the first routing phase, it prefers physical neighbors and thus accelerates the overlay communication.

Further details on our prototype implementation based on the *FreePastry*¹ library are presented in our demonstration paper [19].

3.2 Multi-dimensional Data and Data Skew

Many scientific domains use multi-dimensional data. These domains include climatology, medicine, and especially astrophysics. Data objects from observational data sources are considered to be logically related when they are in close vicinity to each other according to a astronomical coordinate system. It is common to use the celestial coordinate system, which specifies the coordinates *right ascension* and *declination* on a unit sphere around the geocenter.

Figure 2 shows data samples of three astrophysical catalogs, which are provided by our cooperation partners. It clearly exhibits data skew, as there are areas of the sky that contain data from all three catalogs as well as areas in our sample that contain no data.

Areas being investigated more intensively or showing a naturally high density are a common source for skewed data distributions in many application domains. We address and compensate such data imbalance by choosing a qualified data distribution function and a corresponding partitioning scheme.

In the following, we illustrate our HiSbase approach using Figure 3 on a simplified example. Data objects are denoted in Figure 3(i) with a point-like shape.

3.3 Training Phase

The partitioning scheme is determined during a training phase. As training data set we can either use the complete data or representative data samples. Due to the similarity with equi-depth histograms [14], we denote the used distribution function and the according data structure in HiSbase as *histogram*. This histogram is known at all participating nodes.

Quadtrees [5, 18] lend themselves as histogram data structure for achieving a balanced data distribution because they adapt to the data distribution and frequently partition densely populated areas. The regular shape of the data partitions (rectangles in the two-dimensional case) additionally supports the efficient calculation of relevant data regions during query processing. In previous work [20], we conducted an extensive evaluation of the training phase, comparing various data structures and defining several statistics to support the communities in determining which data structure fits their requirements best.

We create the histogram data structure by recursively decomposing the data space. At the beginning of the training phase, we start with a single-leaf quadtree that covers the whole data space and contains all training samples. We then continuously split the leaf which contains the most data objects until the desired number of partitions is created. Figure 3(ii) shows a quadtree-based data partitioning scheme with seven partitions or *regions*.

In order to preserve data locality during mapping the regions to the one-dimensional key space of the P2P infrastructure, we use *space filling curves* such as the Hilbert curve [6] or Z-order [12]. The capabilities of space filling curves as index structure for multi-dimensional databases have been intensively investigated in related work [10, 12]. Figure 3(iii) shows the linearization of the partitions using the Z-curve which is an intuitive choice as it corresponds to an in-order numbering of all quadtree leaves (see Figure 3(iv)).

3.4 Data Mapping

Region identifiers—determined by the space filling curve—are mapped to the one-dimensional DHT address space in ascending order using an equidistant distribution to cover the whole key ring. HiSbase randomly hashes nodes onto the identifier space and regions are implicitly assigned by the underlying DHT structure. Thus, the probability that a node handles a specific region is equal for all regions, even if they cover areas of different sizes. Figure 3(v) illustrates the mapping of seven regions (0–6) to four nodes (*a–d*). The nodes communicate using the region identifiers in order to reach the responsible node. Therefore, HiSbase uses the histogram data structure not only as distribution function but also as routing index.

¹<http://freepastry.org>

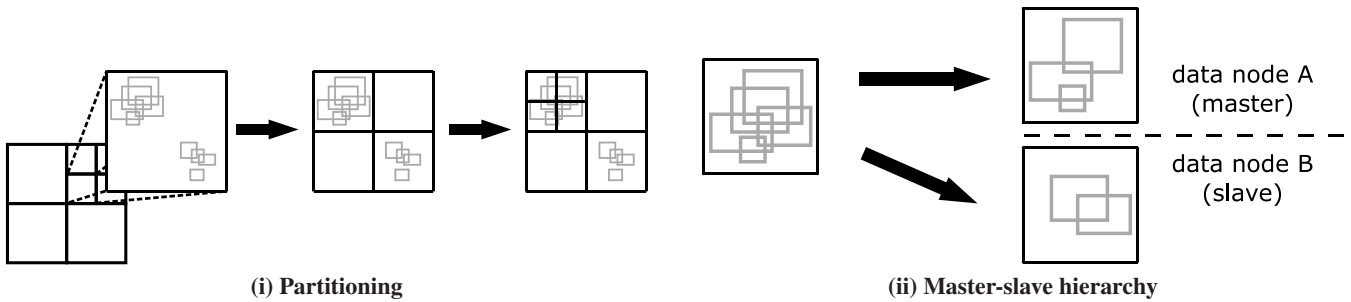


Figure 4: Handling of query hot spots in HiSbase

3.5 Query Processing

Each HiSbase node can either submit or receive region-based queries and calculates the relevant regions during query analysis. The node, which was used to inject the query, coordinates the query processing if it covers itself one of the relevant regions. Otherwise, one of the responsible nodes becomes the coordinator for this query. The coordinator forwards the query to all nodes covering relevant regions and finally combines the intermediate results. Assume that the example query (the thick-framed rectangle) from Figure 3(ii) is issued by a client at node a . Node a coordinates the query processing as it covers one of the two relevant regions (1 and 3) and forwards the query to node b . After combining the result from its local database with the result from node b , it returns the final result to the client.

3.6 Data Feeding

Integrated data sources (archives) directly feed their content into the HiSbase network. The community-specific distribution function determines (independently of the source archive) which peer is responsible for a part of the data space. Therefore, each node maintains all data objects that fall within its managed regions.

4. NEXT CHALLENGES

Having described the core ideas for our proposed community-driven data grids in the previous section, we discuss our ongoing work on load balancing and further challenges we want to address within this Ph.D. project.

4.1 Load Balancing in HiSbase

Achieving load balancing via data partitioning, replication and parallelism is a central aspect of the HiSbase architecture. *Data load balancing* aspects have been discussed in the previous section.

Once published, data of astrophysical projects does not change. This is important to guarantee reproducibility for other scientific results which are based on the public data sets. Therefore individual catalogs provide (mostly on a yearly basis) new versions in parallel to the already published ones. Communities can integrate additional data (new catalogs or new versions) in HiSbase as soon as the schema information has been distributed to all nodes.

In this case, however, data load balancing is possibly no longer optimal. Having added new data, it is therefore advisable to assess if histogram and data distribution can be further optimized.

Empirical studies [7] show that scientific data sets not only exhibit data skew but also are subject to query hot spots, i.e., the query workload is not uniformly distributed. For example, the discovery of a new phenomenon can lead to an increased interest of the research community in that particular area(s) of the sky. As a con-

sequence, one or multiple regions would experience an increased query density and the responsible data nodes would be overloaded. Besides such short-term hot spots, new and even stronger query hot spots can develop and persist for a long-lasting period.

Most dynamic, tree-based distributed P2P data management systems (e.g., [3]) have the deficiency of only being able to address query hot spots by repartitioning data instead of replicating it. Figure 4(i) illustrates such a scenario with two query hot spots within a histogram region. A dynamic P2P system could separate both query hot spots by splitting the original region; with the second iteration, however, this approach would have to accept that it also fragments region-based queries and thus introduces additional communication costs due to queries that span multiple regions. In HiSbase, we can avoid such a fragmentation by using our master-slave approach (see Section 4.1.2) in order to replicate the relevant data for such query hot spots and continue to preserve the data locality.

We first cover the aspects of long-term load balancing before presenting our approach to address short-term query hot spots below.

4.1.1 Long-term Load Balancing

In Section 3.3, our training was only based on the representative data samples and the region which contained the highest amount of data was always split next. If the various data sources already provide query traces it is advisable to integrate those during the training phase. During the integration we need to trade off the conflicting goals of data load balancing and preserving query locality.

A Workload-aware training phase. An easy and efficient option to integrate queries during the training phase is to preferably partition those regions which contain many data objects as well as many queries. If there are several smaller query hot spots in a data area, the hot spots will be distributed across multiple regions and the individual load on a data node is decreased. This is basically the strategy shown in Figure 4(i) and works pretty well at the beginning of the creation of histograms.

Repartitioning regions with a high-density query hot spot in a single area becomes counterproductive. Further repartitioning may result in too small regions and thus data for individual queries may be distributed across multiple partitions. As a consequence, we would violate query locality. Motivated by this observation, we use an extended approach that stops splitting a region and rather replicates it as soon as many queries cover more than a predefined percentage of the region's area (Figure 4(ii)).

Collecting monitoring data about the queries and the load on the individual nodes is not only useful for detecting short-term query hot spots but also helpful in developing the partitioning scheme further in order to adapt to new application conditions.

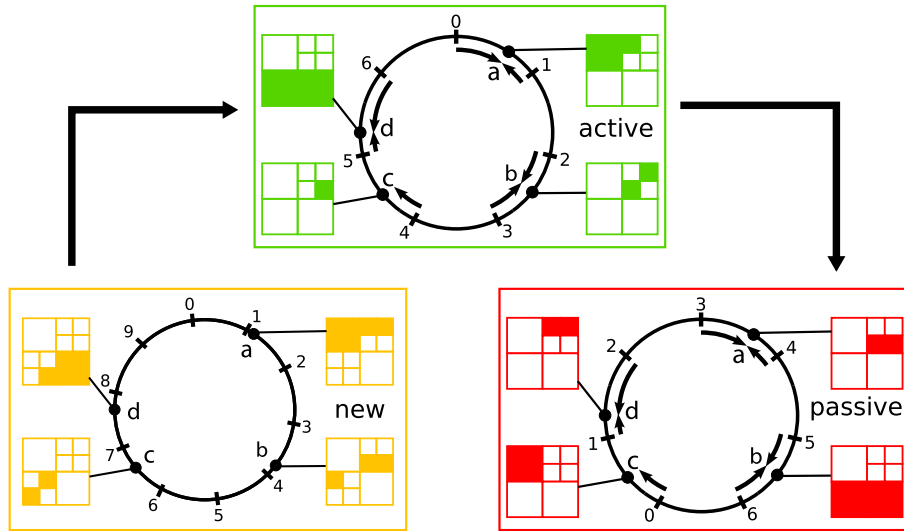


Figure 5: Evolution of histograms within HiSbase

Histogram evolution. Each HiSbase histogram passes through the three development phases that are depicted in Figure 5. According to the *new* histogram, new data partitions are created and disseminated in the network (see Section 3.6). Once the data distribution is completed, data and histogram are *activated* and HiSbase uses both for query processing. In addition, each HiSbase-node keeps a *passive* histogram and data copy. The passive histogram is used for providing data availability and as fallback support during query processing.

In order to determine the passive histogram and its data set, we see two general design options: we use 1) the *outdated* active histogram and its according data sets or 2) we create another slightly modified histogram from the active one. Depending on the cause which triggered the histogram evolution—be it a new data source or additional query hot spots—choosing one or the other alternative impacts the system differently.

Using the outdated histogram. Given the case that a new data source has been integrated into the HiSbase network together with the new histogram, the outdated (formerly active) histogram does not contain the new data set. Thus for updated data regions, the passive histogram can only provide approximate query results. When the new data set is already integrated into the outdated histogram, the data load balancing according to this histogram might not be optimal—as it triggered the creation of the new histogram—but at least the complete data is accessible via the outdated histogram. Whenever new query hot spots triggered the next histogram evolution, again, the outdated histogram does not optimally balance the query processing load but contains all data.

Using the new histogram. For building the passive copy also based on the new histogram we use the same histogram data structure twice and distribute the data twofold. By rotating the origin of the linearization function (as in Figure 5) or by using a different space filling curve we achieve a twofold data availability. This alternative is especially applicable for the initial setup of a HiSbase instance as no new data and statistics are available. Given that active and passive histograms (and their data sets) are identical, we can use both versions directly for load balancing purposes by using both histograms during query processing. Which copy to use can be based on the monitored load information or performed in a round-robin fashion.

Data distribution. For distributing the data during the histogram evolution process, we see two alternatives, one aims at achieving a high data availability early while the other targets an early query processing.

The first design choice distributes the data twice, according to both the new histogram and its *rotated* variant. Thus, HiSbase prepares and conducts the replacement of both the active and passive histogram at once. This may cause high traffic as the data has to be sent twice across the same network link at the feeding archive.

In comparison, the data can be first transmitted according to the new histogram. Once completed, this new data copy can be used for query processing. The new data can be processed early on, yet without additional redundancy. For further replicating the data we can use the first, already distributed copy of the data. Thus the transmitting load is equally balanced across the data nodes as each node distributes its regions according to the *rotated* histogram.

4.1.2 Short-term Load Balancing

Short-term query hot spots are compensated by a master-slave approach, where overloaded nodes replicate parts of their data on slave-nodes. Matching masters to “willing” slaves is achieved using a multicast spanning tree. As there also exist P2P-based multicast implementations, e. g., Scribe [17], HiSbase does not require a centralized component to implement an efficient communication. In this context, the statistics collected during the histogram evolution process come handy to identify overloaded nodes.

Once we identified an overloaded node, we need to decide which data to replicate, how many copies to create, and which HiSbase nodes are suited best for managing the replicas.

For these three criteria we need load statistics as well as information about the stability of the individual data nodes. In the context of community-driven data grids such as HiSbase, load information might be of primary interest as we can assume dedicated nodes with stable, high-bandwidth network interconnections. Such an assumption seems reasonable when dealing with the anticipated amounts of data. If two machines have the same load profile, i. e., based on the load statistics both nodes are candidates for replication, we prefer to replicate the data of the machine being less stable.

The number of copies is proportional to the ratio between the load on the overloaded node and the load of neighboring nodes.

Should a node have three-times the load of its neighbors, we should replicate the data three times in order to level the load between the individual nodes.

Selecting a peer to be responsible for a replicated copy can be based on several criteria. In general, we can choose between *logical* neighbors (peers, which are neighbors on the underlying P2P key space) and *physical* neighbors (peers, which are close within the physical network).

For the decision, we need to trade off a contemporary replication that requires high bandwidth (which physical neighbors can provide) and preserving the data locality (for completely transferring queries that span multiple regions to another node) that is achieved by replicating data to neighbors on the identifier space.

Within our master-slave approach, we use physical neighbor nodes as data transfers are presumably faster than transmitting data over wide-area-network links. If queries span multiple regions which are managed by multiple peers, the use of physical neighbors further reduces the communication overhead.

4.2 Data Feeding Strategies

Conceptually, data is distributed into the HiSbase network as described in Section 3.6. However transmitting each data object individually will not scale to the anticipated data volumes. Defining the right “portion”, i. e., a data buffer which is transmitted in one message, based on the regions of the histogram and/or on a time-slot are anticipated optimizations. Furthermore, database vendors put tremendous efforts in optimizing their *bulk loading*-tools, so these might be a valid option compared to millions of `insert` statements.

5. CONCLUSION

Scientific data management is among the inspiring challenges for the database research community. With community-driven data grids, we propose an architecture that performs throughput optimization by adapting to the community-specific data and query characteristics and provides means for data and load balancing.

Initial experiments in our local LAN and on the AstroGrid-D test bed show a promising super-linear throughput increase due to high parallelism, load balancing, and higher cache locality.

We aim at enabling the researchers to actively design their data management. Such an effort not only fosters great inter-disciplinary collaborations. Furthermore, solving these challenges allows us to shape the data management for future e-science communities.

6. REFERENCES

- [1] A. Crăiniceanu, P. Linga, A. Machanavajjhala, J. Gehrke, and J. Shanmugasundaram. P-Ring: An Efficient and Robust P2P Range Index Structure. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 223–234, Beijing, China, June 2007.
- [2] A. Datta, R. Schmidt, and K. Aberer. Query-load balancing in structured overlays. In *Proc. of the IEEE/ACM Int. Symposium on Cluster Computing and the Grid*, pages 453–460, Rio de Janeiro, Brazil, May 2007.
- [3] C. du Mouza, W. Litwin, and P. Rigaux. SD-Rtree: A Scalable Distributed Rtree. In *Proc. of the Intl. Conf. on Data Engineering*, pages 296–305, Istanbul, Turkey, Apr. 2005.
- [4] H. Enke, M. Steinmetz, T. Radke, A. Reiser, T. Röblitz, and M. Högvist. AstroGrid-D: Enhancing Astronomic Science with Grid Technology. In *Proc. of the German e-Science Conference*, Baden-Baden, Germany, May 2007.
- [5] R. A. Finkel and J. L. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4:1–9, Mar. 1974.
- [6] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Math. Ann.*, 38:459–460, 1891.
- [7] M. Ivanova, N. Nes, R. Goncalves, and M. Kersten. MonetDB/SQL Meets SkyServer: the Challenges of a Scientific Database. In *Proc. of the Intl. Conf. on Scientific and Statistical Database Management*, page 13, Banff, Canada, July 2007.
- [8] D. Klan, K.-U. Sattler, K. Hose, and M. Karnstedt. Decentralized Managing of Replication Objects in Massively Distributed Systems. In *Proc. of the Intl. Workshop on Data Management in P2P Systems*, Nantes, France, Mar. 2008.
- [9] R. Kuntschke, T. Scholl, S. Huber, A. Kemper, A. Reiser, H.-M. Adorf, G. Lemson, and W. Voges. Grid-based Data Stream Processing in e-Science. In *Proc. of the IEEE Intl. Conf. on e-Science and Grid Computing*, page 30, Amsterdam, The Netherlands, Dec. 2006.
- [10] V. Markl and R. Bayer. Processing Relational OLAP Queries with UB-Trees and Multidimensional Hierarchical Clustering. In *Proc. of the Intl. Workshop on Design and Management of Data Warehouses*, page 1, Stockholm, Sweden, June 2000.
- [11] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data Fusion in Three Steps: Resolving Schema, Tuple, and Value Inconsistencies. *IEEE Data Engineering Bulletin*, 29(2):21–31, 2006.
- [12] J. Orenstein and T. Merrett. A class of data structures for associative searching. In *Proc. of the ACM SIGACT-SIGMOD Symp. on Principles of Database Sys.*, pages 181–190, Waterloo, Ontario, Canada, Apr. 1984.
- [13] T. Pitoura, N. Ntarmos, and P. Triantafillou. Replication, Load Balancing, and Efficient Range Query Processing in DHT Data Networks. In *Proc. of the Intl. Conf. on Extending Database Technology*, pages 131–148, Munich, Germany, Mar. 2006.
- [14] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 294–305, Montreal, Quebec, Canada, June 1996.
- [15] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
- [16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM Intl. Conf. on Data Communication*, pages 161–172, 2001.
- [17] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, Nov. 2001.
- [18] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, 1990.
- [19] T. Scholl, B. Bauer, B. Gufler, R. Kuntschke, D. Weber, A. Reiser, and A. Kemper. HiSbase: Histogram-based P2P Main Memory Data Management. In *Proc. of the Intl. Conf. on Very Large Data Bases (demo)*, pages 1394–1397, Vienna, Austria, Sept. 2007.
- [20] T. Scholl, R. Kuntschke, A. Reiser, and A. Kemper. Community Training: Partitioning Schemes in Good Shape for Federated Data Grids. In *Proc. of the IEEE Intl. Conf. on e-Science and Grid Computing*, pages 195–203, Bangalore, India, Dec. 2007.
- [21] P. Schücker, H. Böhringer, and W. Voges. Detection of X-ray Clusters of Galaxies by Matching RASS Photons and SDSS Galaxies within GAVO. *Astronomy & Astrophysics*, 420:61–74, 2004.
- [22] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of the ACM SIGCOMM Intl. Conf. on Data Communication*, pages 149–160, San Diego, CA, USA, Aug. 2001.
- [23] A. S. Szalay, J. Gray, A. R. Thakar, P. Z. Kunszt, T. Malik, J. Raddick, C. Stoughton, and J. vanderBerg. The SDSS skyserver: public access to the sloan digital sky server data. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 570–581, Madison, WI, USA, 2002.
- [24] S. Venugopal, R. Buyya, and K. Ramamohanarao. A Taxonomy of Data Grids for Distributed Data Sharing, Management, and Processing. *ACM Computing Surveys*, 38(1):3, Mar. 2006.