HiSbase: Histogram-based P2P Main Memory Data Management

Tobias Scholl Bernhard Bauer Benjamin Gufler Richard Kuntschke Daniel Weber
Angelika Reiser Alfons Kemper

Technische Universität München
Munich, Germany

{scholl, bauerb, gufler, kuntschk, weberd, reiser, kemper}@in.tum.de

1. INTRODUCTION AND MOTIVATION

Many e-science communities, e. g., medicine, climatology, and astrophysics, are overwhelmed by the exponentially growing data volumes that need to be accessible by collaborating researchers. Nowadays, new scientific results are often obtained by exploring and cross-correlating data from different distributed sources [3]. However, neither centralized data processing by shipping the data to the processing site on demand nor a centralized data warehouse approach scale sufficiently to handle the huge data volumes and processing demands of future e-science communities and applications. The former suffers from high transmission costs while the latter cannot scale to the large amounts of data in combination with the growing number of queries.

Decentralized Peer-to-Peer (P2P) architectures lend themselves to provide scalable communication and data management to overcome the deficiencies of centralized approaches. Based on distributed hash tables (DHT), new peers and their resources are integrated seamlessly. Built on top of the P2P overlay network infrastructure, communities need a framework for data publication and efficient data access which adapts to the data and query characteristics of their specific domain. With HiSbase, collaborating researchers are able to share not only their CPU resources, but also otherwise unused main memory. HiSbase achieves higher throughput in query processing as data is distributed across numerous (e. g., thousands of) peers according to predominant query patterns. As a consequence, most processing tasks can be performed locally, achieving high cache locality as peers mainly process queries on logically related data. Figure 1 illustrates this approach on an abstract level. In the figure, logically related data originating from (possibly) different distributed sources are denoted by the same geometric shapes. HiSbase allocates data fed into the system by means of community-specific distribution functions. Thereby, related data objects of various sources—represented by identical geometric shapes in Figure 1—are mapped to the same peers.

The abstract scenario above is applicable to many e-science domains including climatology, geophysics, and medicine. We use data and use cases from the astrophysics domain for further illustrations, since we are currently developing a distributed information management platform for the German astrophysics community (AstroGrid-D) within *D-Grid*, the German e-science and Grid Computing initiative. This platform facilitates collaborations with national as well as international partners.

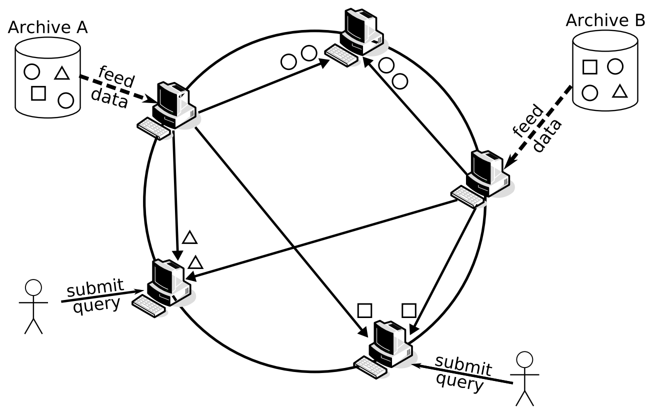


Figure 1: HiSbase architecture.

In e-science, results of different investigations (experiments, surveys, observations, etc.) are compared or combined to gain further insight or to see the complete picture of a particular phenomenon. Some astrophysics example use cases are the creation of probability maps for galaxy clusters and the combination of observational data from several archives covering, for example, various wavelength ranges in order to classify spectral energy distributions [5].

Researchers usually want to access logically related subsets of these data sets. The definitions of such subsets are mostly based on specific data characteristics. Typical access patterns over astrophysical data sets are point-near-point and point-in region queries. Such queries are usually *region-based*, i. e., they process data within certain regions of the sky. These regions are specified by the two-dimensional celestial coordinates *right ascension* and *declination*. Region-based queries can, of course, also contain predicates on attributes other than the celestial coordinates. In case of celestial objects, other attributes might comprise detection time, catalog-identifier, temperature, or energy level.

Preserving the locality of data is crucial to allow the efficient processing of queries on logically related data sets. Data locality is especially important for the performance of data analysis tasks in astrophysics as these are mostly accessing data in a region-based manner. Therefore, randomly mapping data objects across widely distributed information networks can severely impair the performance of typical astrophysical query patterns.

In many application domains, data sets are often highly skewed.

VLDB '07, September 23-28, 2007, Vienna, Austria.
Copyright 2007 VLDB Endowment, ACM 978-1-59593-649-3/07/09.

Data skew originates from data spaces with a mix of densely and sparsely populated regions. The differences in data density may arise from the original data distribution or from the fact that some regions have been investigated more extensively than others, i. e., more data has been collected and is available. In astrophysics, celestial objects are not distributed uniformly over the sky due to certain conditions or events that lead to data concentration (e. g., high data density in the galactic plane or a supernova).

With HiSbase, we demonstrate how to address this imbalance of the data distribution while preserving spatial proximity to efficiently process region-based queries (Section 2.1). By calculating a histogram that equips the DHT infrastructure with a community-specific data distribution, HiSbase achieves this goal. In this paper, we concentrate on the Z-quadtree histogram data structure (Section 2.2) that we designed to preserve spatial locality for astrophysics data sets. Z-quadtrees are *quadtrees* whose leaves correspond to histogram buckets and are linearized on the DHT key space using a space filling curve. These trees provide efficient access to histogram buckets (regions) while balancing the data load across peers.[1]

HiSbase incorporates multi-dimensional data and histograms as follows:

- We precompute the histogram of the actual data space in a preparatory training phase based on a *training set* and pass it to the initial HiSbase peer during startup.
- Additional peers subsequently joining the network receive their own local copy of the histogram from a neighboring HiSbase peer.
- HiSbase allocates data at peers according to the precomputed histogram (Section 2.3) and uses the histogram as a routing index. Data archives feed data into HiSbase by sending their data to any HiSbase peer (Section 2.4).
- Every HiSbase peer accepts queries and routes them to a coordinator peer which owns (some of) the data needed to process the query. If the coordinator does not cover all the data relevant to the query, it guides cooperative query processing among all peers contributing to the query result (Section 2.5).

HiSbase realizes a scalable information economy [1] by building on advances in proven DHT-based P2P systems such as Chord [10] and Pastry [7], as well as on achievements in P2P-based query processing [4]. HiSbase combines these techniques with histograms for preserving data locality, spatial data structures such as the quadtree [8] for efficient access to histogram buckets, and space filling curves [6] for mapping histogram buckets to the DHT key space.

Our main contributions within the context of HiSbase entail an investigation of the potential offered by P2P networks for increasing query throughput in data-intensive e-science applications. One of the main deficiencies of centralized data management is achieving sufficient query throughput.

HiSbase targets community-specific collaborations having vast data volumes with fairly stable data distributions and offers a framework for comparing various *à priori* calculated histogram data structures. This is a necessity for efficiently distributing and processing data sets at a large scale and distinguishes HiSbase from related work which use quadtrees to address frequent changes in data hot spots [9] or spatial objects [11], i. e., objects with a multi-dimensional extent. In [9], peers only cover quadtree leaves at the same level, while HiSbase does not enforce such a restriction. Tanin et al. [11] map quadtree leaves randomly on an underlying DHT and require additional routing steps to descend the distributed quadtree. HiSbase uses a space filling curve to map the regions and directly determines which regions contribute to a query result.

[1]In the following, we use the terms *regions* and *histogram buckets* interchangeably. The *leaves* of a Z-quadtree represent the histogram buckets for that particular histogram data structure.

2. HISBASE ARCHITECTURE

We use the distributed hash table (DHT) infrastructure *Pastry* [7] to manage peers and route messages in HiSbase. While the overall design of HiSbase abstracts from the underlying DHT implementation, we deliberately decided to base our implementation on *Pastry*, one of the well-established ring-based P2P overlay network structures with optimized routing performance. Like Chord [10], Pastry maps data and peers to a one-dimensional key ring. In contrast to Chord, Pastry optimizes the initial phase of routing by preferring physical neighbors to speed up communication within the overlay network.

2.1 Spatial Locality and Skew Handling

In the following, we use a simplified two-dimensional data sample (see Figure 2(i)) for illustration. Data skew can be addressed by determining an appropriate histogram function. HiSbase partitions the data space in a way such that histogram buckets contain approximately the same number of objects. The functionality of the histogram is comparable to that of the hash function in traditional DHT structures. In such DHT systems, data is hashed randomly onto the key ring to achieve an equal distribution of the data over the participating peers. However, such an equal distribution comes at the expense of losing the spatial locality of the data. HiSbase avoids this deficiency by using histograms to partition the data space into multi-dimensional contiguous regions.

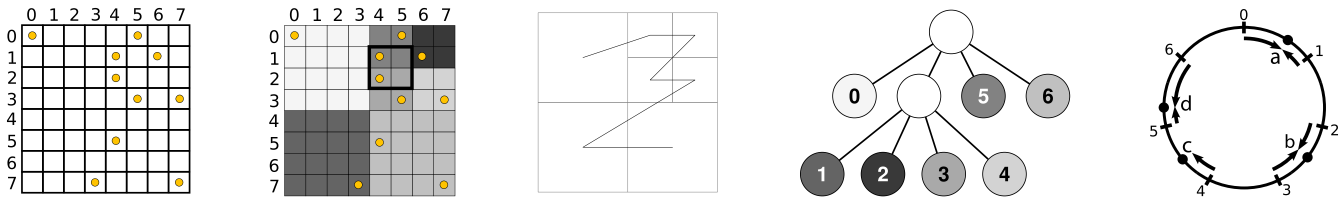
Based on a representative data set, the histogram is calculated during a training phase. This training set may either be the complete data set or a representative random sample. For our demo setup, we used random 10% of the input data as training set. Such an à priori analysis is applicable in many research areas, including astrophysics, because the distribution of published data remains fairly stable. In this paper, we use the Z-quadtree introduced in the next section as histogram data structure. After the training phase, the resulting Z-quadtree is passed to the initial peer when starting HiSbase. Newly arriving peers obtain the histogram from any existing peer when joining the network, i. e., each HiSbase peer keeps a local copy of the Z-quadtree.

We choose the number of histogram buckets to be 10 times higher than the anticipated number of peers. This allows additional peers to join the network and keeps the size of the histogram at a reasonable level compared to the amount of data transmitted during query processing. Since arriving peers most presumably join the P2P network via a physical neighbor peer, histogram transmission only insignificantly prolongs the standard joining process of the underlying P2P system.

2.2 Z-Quadtrees

Partitioning schemes resulting in regular shapes (squares or rectangles) are preferable in order to limit the complexity of query processing. Quadtrees [8], which are based on recursive decomposition, exhibit this characteristic and also adapt to skew in data distributions. In this paper, we employ a quadtree as histogram data structure. The leaves of the tree (i. e., histogram buckets) are mapped to the DHT key ring using the Z-order [6]. Consequently, we use the term Z-quadtree to denote this histogram data structure.

For a d-dimensional data space, a Z-quadtree is recursively defined to be either a leaf with a d-dimensional hypercube data region or an inner node with 2^d child trees. In our two-dimensional astrophysical data space, an inner node has four children (quadrants) with rectangular data regions. During Z-quadtree build-up, the training set is sequentially inserted into a single leaf until a predefined threshold is reached. After that, the leaf is split according to



(i) Data sample (ii) Data buckets (iii) Linearization (iv) Z-quadtree structure (v) Peer mapping

Figure 2: Processing an illustrative data sample with a Z-quadtree histogram.

the quadtree splitting strategy. Figure 2(ii) shows the final decomposition of the data sample with a bucket threshold of three objects. Finally, the Z-quadtree leaves are linearized using a space filling curve such as the Z-order, as illustrated in Figure 2(iii). This corresponds to numbering the leaves of the Z-quadtree (Figure 2(iv)) with *region ids* according to a depth-first tree traversal.

Space filling curves preserve spatial proximity which is essential if a peer manages several regions. Due to the use of a space filling curve, these regions constitute neighboring Z-quadtree leaves. Queries spanning the corresponding regions can thus be efficiently processed locally at the responsible peer.

In contrast to the original quadtree which is intended as a spatial data structure, the Z-quadtree is used for data dissemination, query processing, and as a routing index. The actual data is not stored in the Z-quadtree itself.

Lookups performed during the feeding stage and during query processing benefit from the regular structure of Z-quadtree leaf regions and are described in more detail below. Not all leaves will be completely filled because the object threshold for buckets merely is an upper bound. In pathological cases, e. g., if most data items are concentrated in a comparatively small area of the data space, the tree might degenerate having numerous empty leaves. While we define the Z-quadtree top-down, we actually build the histograms bottom-up during the training phase. This is advantageous, since building the quadtree top-down would additionally require to guess the splitting threshold in advance.

We currently investigate other splitting techniques with various trade-offs. The original quadtree bisects each dimension (which results in four quadrants in the two-dimensional case). One variant of the Z-quadtree uses a different splitting strategy by individually calculating the median for each dimension as the split coordinate.

2.3 Mapping Peers

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 peers onto the identifier space. Region (bucket) assignment is done implicitly by the routing mechanism of the underlying DHT structure, i. e., region ids are mapped to the closest peer on the ring. Peer communication is performed independently from the current number of peers by routing messages to the DHT identifiers of regions. The histogram structure created during the training phase therefore also serves as a routing index for HiSbase. Figure 2(v) shows how the evenly distributed region ids (0–6) are mapped to the randomly distributed peers (a, b, c, d) on the DHT key space.

2.4 Feeding the Data

Archives feed data directly into the P2P architecture as suggested by Figure 1. Data integration is not in the focus of this demo—we assume that the data being fed into HiSbase adheres to a common schema or is already properly transformed. In HiSbase, the histogram, a community-specific data allocation function, determines which peers should store which data objects. Thus, every peer maintains all data from any archive located in its assigned regions.

Data feeding proceeds by data archives sending their data to any HiSbase peer. For each data item d (e. g., a tuple from a relational database system), the peer performs a lookup in the histogram H, retrieving the region id r the data item belongs to. Subsequently, d is sent to r. Finally, HiSbase uses the histogram to route d to the peer responsible for r, which is the peer closest to r on the DHT key ring (cf. Figure 2(v)). Of course, this approach can be optimized by using *bulk feeding* where data items for the same region are transmitted in a single message.

Peers store data objects contained in their assigned regions in a database system. HiSbase does not require any specific database system, thus enabling the use and comparison of traditional as well as main memory based database systems.

2.5 Region-based Queries

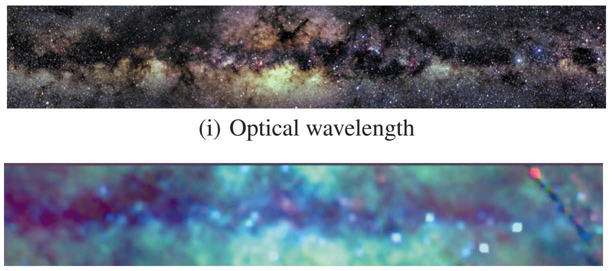
Region-based queries can be submitted to any peer in the HiSbase network. The peer receiving the query extracts the referenced multi-dimensional area A from the query predicate. The set S of region identifiers intersecting with A is determined by a lookup in the histogram H. We select an arbitrary region id r_c from S and choose the peer c which is responsible for r_c as the *coordinator*. The coordinator distributes the query to peers covering region ids from S and collects intermediate results.

Using a Z-quadtree histogram, S is obtained by a tree traversal. If the region of an inner node intersects with the query area A, its children are processed recursively. Identifiers of intersecting leaf regions are added to S.

For example, given the HiSbase instance of Figure 2(v), peer a may receive the query with the query area depicted by the thick-lined rectangle in Figure 2(ii). By coincidence, this peer covers one of the two relevant region ids 1 and 3. As the coordinator, it forwards the query to region id 3, which is covered by peer b. After receiving both, the results from its local database (it covers region id 1) and those from peer b, the coordinator sends the results back to the client for further processing.

Since HiSbase routes queries to responsible peers via region identifiers, peers covering several regions will receive the same query multiple times, once for each region. To avoid multiple query evaluation, a hash value is calculated for each query to recognize whether the query has been processed previously.

We compared query throughput of a small local area HiSbase configuration and of a wide-are deployment on PlanetLab with a central database server. First results indicate that HiSbase achieves super-linear query throughput and provides a stable and scalable data management infrastructure. Currently, we are combining our data load balancing approach with query load balancing techniques to efficiently handle query hot spots. We are considering several approaches such as replication and introducing a peer hierarchy where idle peers can support overloaded nodes by trading their own histogram buckets for data from the overloaded peer. We also plan to incorporate query statistics in the histogram creation during the training phase.



(i) Optical wavelength

(ii) X-ray wavelength

Figure 3: A multi-wavelength view on the milky way (Source: `http://adc.gsfc.nasa.gov/mw/`).

3. DEMONSTRATION OUTLINE

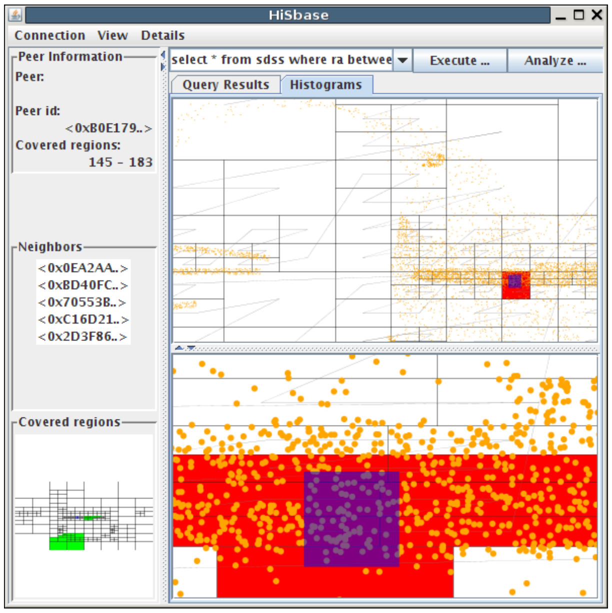
We use *cross-match* queries as an example application from the astrophysics domain to describe the functionality of HiSbase. Astrophysicists use cross-matching to determine whether data points from different archives are likely to stem from the same celestial object. Researchers take several point sources from an area (e. g., the milky way in Figure 3) in one data set and look for matching sources in other data sets.

Using catalog data from our astrophysical cooperation partners, we will demonstrate the following aspects of HiSbase:

- We present a *Java-based implementation* of HiSbase using the *FreePastry*[2] implementation of Pastry. People can interact with the HiSbase network during the demonstration, using a *graphical user interface* (see Figure 4).
- The demonstrated HiSbase instance is *deployed in a wide-area setting*, on about one hundred nodes distributed world-wide on the PlanetLab[3] testing platform and on the resources of the AstroGrid-D project using different database systems (DB2, PostgreSQL, Derby) to reflect the heterogeneity of the database landscape within e-science communities.
- The application interface offers additional access to the *current state* of the demonstration node. The state information includes currently covered regions and the neighbor connectivity. Additionally, the data samples used and the histograms generated during the training phase are shown.
- The demonstration laptop is one of the HiSbase peers, covers part of the data, and takes part in the *processing of region-based queries*. Users can either write their own SQL queries or choose cross-matching queries from a predefined set. We adopted existing code for SQL cross-matching queries [2] and added a special *xmatch* pattern to simplify queries.
- We show the *efficient coordination* of queries spanning multiple peers.
- We *evaluate* the Z-quadtree and other *histogram techniques* with regard to query throughput and the preservation of spatial locality. Queries submitted to the HiSbase application can be displayed in both the histogram currently used by the infrastructure and in an alternative histogram to allow the comparison of histogram variants.

The data sample for the demonstration comprises about 137 million objects from subsets of the ROSAT (25 million objects), SDSS (84 million objects), and TWOMASS (28 million objects) catalogs and has a size of about 50 GB.

[2]`http://freepastry.org`
[3]`http://www.planet-lab.org`



Peer Information
Peer:
Peer id:
<0xB0E179..>
Covered regions:
145 - 183

select * from sdss where ra betwee

<0x0EA2AA..>
<0xBD40FC..>
<0x70553B..>
<0xC16D21..>
<0x2D3F86..>

Figure 4: HiSbase GUI.

Acknowledgments. This work is part of the AstroGrid-D project within the D-Grid initiative and is funded by the German Federal Ministry of Education and Research (BMBF) under contract 01AK804F and by Microsoft Research Cambridge (MSRC) under contract 2005-041.

4. REFERENCES

[1] R. Braumandl, A. Kemper, and D. Kossmann. Quality of Service in an Information Economy. *ACM Trans. on Internet Technology*, 3(4):291–333, Nov. 2003.

[2] J. Gray, M. A. N. Santisteban, and A. S. Szalay. The Zones Algorithm for Finding Points-Near-Point or Cross-Matching Spatial Datasets. Technical Report MSR-TR-2006-52, Microsoft Research, Microsoft Cooperation, Redmond, WA, USA, Apr. 2006.

[3] J. Gray and A. Szalay. The World-Wide Telescope. *Communications of the ACM*, 45(11):50–55, Nov. 2002.

[4] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 321–332, Berlin, Germany, Sept. 2003.

[5] 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.

[6] 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.

[7] 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*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[8] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley, 1990.

[9] Y. Shu, B. C. Ooi, K.-L. Tan, and A. Zhou. Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems. In *Proc. of the IEEE Intl. Conf. on Peer-to-Peer Computing*, pages 173–180, Aug. 2005.

[10] 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.

[11] E. Tanin, A. Harwood, and H. Samet. Using a distributed quadtree index in peer-to-peer networks. *VLDB Journal*, 16:165–178, Feb. 2007.