Community Training: Partitioning Schemes in Good Shape for Federated Data Grids[*]

Tobias Scholl Richard Kuntschke Angelika Reiser Alfons Kemper
Technische Universität München
Munich, Germany
⟨*firstname.lastname*⟩@in.tum.de

Abstract

In federated Data Grids, individual institutions share their data sets within a community to enable collaborative data analysis. Data access needs to be provided in a scalable fashion since in most e-science communities, data sets do not only grow exponentially but also experience an increasing popularity. If data autonomy is retained, each individual institution has to ensure efficient access to its data. Analyzing application-specific data properties (such as data skew) or query characteristics (query patterns) and distributing data within Data Grids accordingly, allows for improved throughput for data-intensive applications and enables better load-balancing between shared resources. We propose a framework for investigating application-specific index structures for creating suitable partitioning schemes. We evaluate two variants of the well-known Quadtree data structure as well as the Zones approach, an index structure from the astrophysics domain, according to several criteria. Our framework improves data access within federated Data Grids and can be combined with well-established Grid methods as well as with more flexible P2P technologies.

1. Introduction

Many application domains such as high energy physics, astronomy, and geosciences have gathered and still collect enormous quantities of data. The sheer data volumes expected by forthcoming e-science data challenges make it impossible for a single institution to process all the data autonomously. In order to find new scientific results, researchers cooperate globally with colleagues and combine the outcome of different experiments, observations, and other data repositories.

For this purpose, many e-science communities join forces in order to build up *Virtual Organizations* and to create *Data Grids*, a network for sharing data and compute resources that is interconnected with high-bandwidth links to facilitate data-intensive applications. *Federated Data Grids* [16] are one of the various design options for Data Grids, where members of a community integrate their existing data repositories into a common infrastructure. However, these institutions mostly enforce *data autonomy*, i. e., they retain control over their data sets and keep the data within institutional boundaries. Members from other organizations are only allowed to access these data sets if they provide sufficient authentication.

In this paper, we describe a *training phase* which is the essential building block for generating partitioning schemes in our framework. Partitioning schemes are based on predominant data and query characteristics from a particular domain. We use an example from the astrophysics domain for illustration, where data is highly skewed and data-intensive applications exhibit a high degree of spatial locality. We compare the *Quadtree* [4, 13], a median-based Quadtree variant, and the *Zones* [7] index structure with regard to their fitness as partitioning schemes for federated astrophysical Data Grids.

For communities which do not need to enforce data autonomy, distributing the data partitions across several resources according to the partitioning scheme yields several benefits such as improved load-balancing and increased throughput. In combination with the partitioning scheme, the data partitions can then be accessed with familiar data access methods, such as OGSA-DAI, which additionally provide data integration capabilities and the necessary security features.

2. Motivation

Facing not only exponential data growth (terabytes per day, petabytes per year) but also a steadily increasing number of users, communities look into different technologies

[*]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 and by Microsoft Research Cambridge (MSRC) under contract 2005-041.

for providing scalable infrastructures. The number of users is rising as not only a few scientists have access to these data sets but also numerous amateur scientists, school classes, or students use the available data sets to perform their private research. Many data sets of publicly funded projects become (and remain) available to the public after a one-year grace period and can be accessed by the community. This multitude of data repositories, where institutions are no longer interested in keeping them "private", is in the main focus of our work.

Many e-science use cases exhibit a high degree of spatial locality. In astronomy, for example, data-intensive analysis tasks operate on data from the same area of the sky. We use the term *spatial locality* to denote that tasks use data from a particular area. These *region-based queries* define areas in the sky mainly by using the two-dimensional celestial coordinates *right ascension (ra)* and *declination (dec)*. Of course, queries can also contain predicates on other attributes. A partitioning scheme suitable for the astrophysics domain needs to preserve spatial locality as one of the predominant *query characteristics*. Another example from the business world are Enterprise Grids where large warehouses might aggregate data from RFID-sensors. Possible queries in such a setting contain not only the geographic location but also a third, temporal dimension in order to track the goods on their way from the warehouse to the customer.

A prevalent *data characteristic* in many application areas is data skew. Data skew originates from data spaces containing a combination of densely and sparsely populated areas. The differences in data density may arise from the original data distribution or from some regions having been investigated more extensively than others. In astrophysics, for example, 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).

Together with our partners from the AstroGrid-D [3] community project within the D-Grid initiative, we are building up a Grid environment that supports users at bringing their science to the Grid. Within that common effort, our main focus are applications that access scientific databases from the Grid or use Grid-based data stream management [9]. While our framework is also applicable to other domains, we concentrate on data-intensive tasks from astrophysics for illustration.

Within AstroGrid-D, we access persistent data, such as scientific databases, using the components developed by the *Open Grid Services Architecture – Data Access and Integration (OGSA-DAI)* project [1]. The OGSA-DAI project integrates with the Globus Toolkit and participates in the standardization process of the Open Grid Forum (OGF) *Data Access and Integration Services (DAIS)* working group [2]. OGSA-DAI offers a unified way of accessing

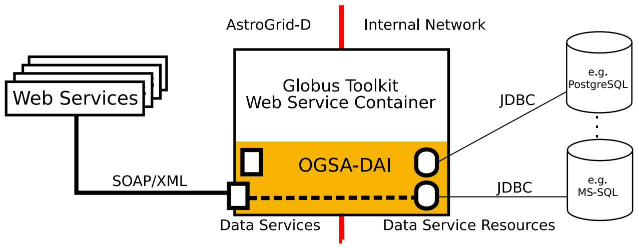


Figure 1. OGSA-DAI middleware

and integrating distributed, heterogeneous data resources using web services or Grid services. The interface to integrate resources is very flexible and supports files and RDF. The main focus of the OGSA-DAI support, however, are relational and XML databases.

Figure 1 gives a short overview of the main OGSA-DAI components and how they are used within AstroGrid-D. Integrated into a Globus Web Service Container, OGSA-DAI publishes a *data service* interface that allows web service or Grid service clients to interact with *data service resources*. These data service resources are databases exposed via the data service. OGSA-DAI can therefore hide the complexity of the individual database (driver, connection URL) from the Grid user, the database resources can be kept behind institutional firewalls, and database access is secured using the mechanisms based on Grid-certificates provided by the Globus Toolkit. If communities want to integrate a new type of resource, they can define an interface for that resource via the OGSA-DAI *activity*-framework.

3. Training Phase

We use a training phase to create partitioning schemes that describe how data objects are to be partitioned. The training phase comprises three steps:

1. Extract the training samples,
2. Create the partitioning scheme, and
3. Distribute data according to the partitioning scheme.

The training samples are representative subsets from all data archives that are to be distributed within the network. Based on these training samples, we build the partitioning scheme. Given the size of existing and anticipated data sources (several terabytes each), it is not feasible to perform the training on the complete data sets. We use functionality provided by relational database systems to extract random samples. The created partitioning scheme is then evaluated considering the identified application-specific data and query properties. After having decided for a partitioning scheme, we can create the individual *data partitions* (or regions) from the participating data archives. These data partitions are then distributed to shared resources within the Data Grid, e. g., using GridFTP. At the resources, the data is loaded into a database and is made accessible via OGSA-DAI. *Quadtrees*

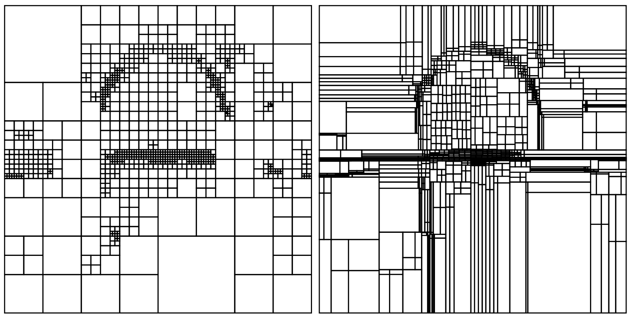


Figure 2. Partitioning scheme based on Quadtrees without (left) and with (right) median heuristics.

and the *Zones* are the two data structures we compare in the following with regard to their fitness for being used as a partitioning scheme for federated Data Grids in astrophysics.

Quadtrees [4, 13] are a well-known spatial data structure and use the principle of recursive decomposition to partition a d-dimensional data space. Quadtrees are recursively defined to be either a leaf with a d-dimensional hypercube data region or an inner node with 2^d subtrees. In a 2-dimensional data space, an inner node has four children (quadrants) that cover equal-sized rectangular data regions.

In Quadtree-based partitioning schemes, the partitions correspond to the leaves of the Quadtree. In particular, communities having skewed data sets can benefit from the capability of Quadtrees to adapt to the data resolution. Sparsely populated areas of the data space are represented by leaves covering a large data region and densely populated areas are partitioned into several leaves covering small areas. Thus, the amount of data within each leaf, and therefore within each partition, is approximately equal. In very pathological cases, however, where data is concentrated in a very small area, Quadtrees degenerate to a tree having many empty leaves. Partitioning schemes without empty leaves are preferable in our setting because they allow us to directly map data partitions to shared resources.

One approach to address the issue of empty leaves is a *median-based heuristics* that splits a leaf at the median instead of at the center. For our astronomical example, the heuristics uses the split point (m_{ra}, m_{dec}) by computing the median for *ra*-coordinates and *dec*-coordinates independently. Our heuristics is similar to the technique used by *optimized point Quadtrees* [4], which only compute the median in the first dimension and thus guarantee that no leaf contains more than half the data of the original leaf. Our heuristics, which computes the median in all dimensions independently, offers a better data distribution in the average case. Figure 2 shows a Quadtree with regular decomposition and a Quadtree using our median heuristics, respectively, both built during our evaluation.

Quadtrees can be created either top-down or bottom-up. Both approaches start with a single empty Quadtree leaf. During top-down creation, the training sample is sequentially inserted into the leaf until the predefined leaf capacity is reached. The leaf is split up and its data is distributed among its new subtrees. In the bottom-up approach, leaves have an unlimited capacity and all data is inserted into the initial leaf first. In the following, the biggest leaf is split in turn until the desired number of partitions has been reached.

We prefer the bottom-up approach over the top-down approach for several reasons. Building Quadtrees top-down requires to guess the leaf capacity in advance, which is hard for highly skewed data sets. Furthermore, building the partitioning scheme bottom-up is a requirement for other splitting strategies to work properly. For example, the median-based heuristics depends on all points within a leaf and would be inaccurate if calculated incrementally when the leaf threshold is reached.

The *Zones* index [7] is an index structure developed in order to improve the performance of typical query patterns in astrophysics such as points-in-region queries, self-match queries, and cross-match queries. The principle behind the Zones index is to divide the data space into zones of equal height h. The zone identifier of a point (*ra*, *dec*) is calculated by $floor((dec + 90.0)/h)$ because the domain of the *declination* coordinate is $[-90.0, +90.0]$. Due to their simplicity, Zones are efficiently implemented directly in SQL and thus exhibit very good performance in tracking down relevant zones for a particular task. Applying the Zones index to the algorithm for finding maximum-likelihood brightest cluster galaxies [11] is a very good example for increased performance by implementing an algorithm as close as possible to the data: directly inside the database. The Zones algorithm preserves spatial locality by grouping data elements from the same area in the sky into the same zone. When used as a clustered index (which also defines the physical layout of data besides accelerating data access), the Zones index can be used efficiently by database optimizers to determine the query result.

We limit our discussion to these three data structures—the two variants of the Quadtree and the Zones—and refer the interested reader to the survey by Gaede and Günther [6] or the book by Samet on multi-dimensional and metric data structures [14] for more information on multi-dimensional access methods.

4. Evaluation

We evaluated the fitness of the three different data structures described in the previous section by collecting several statistics. For the evaluation, we used a Java-based prototype which was developed for the HiSbase [15] system, which also employs a training phase. We used two different data sets for the evaluation. D_{skew} comprises about

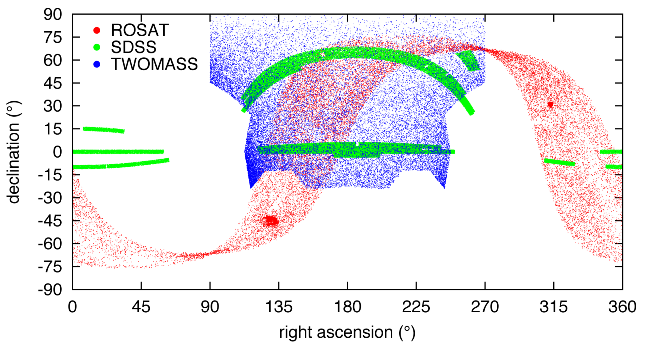


Figure 3. Data set D_{skew}

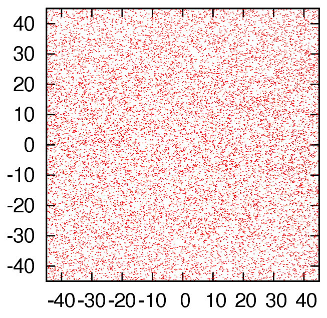


Figure 4. Data set $D_{uniform}$

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 (see Figure 3). We also drew samples from a uniformly distributed astrophysical simulation data set $D_{uniform}$ (160 million objects, Figure 4) of roughly equal size in order to specifically study the impact of data skew on the partitioning schemes. We varied the size s of the training samples to benchmark the quality of results obtained from small data sets. Finally, we generated partitioning schemes of different sizes n, with n varying from 16 (2^4) to 131 072 (2^{17}) partitions. If the partitioning scheme generates only non-empty partitions, we may generate as many partitions as we have distributed Grid servers available for distributing the data. If the construction method cannot guarantee the absence of empty partitions (which is generally the case), it is better to generate more partitions and assign multiple partitions to a server. We chose 2^{17} as upper limit, since the training phase is also useful in a P2P environment in which it is desirable to have many more regions than participating servers. Table 1 summarizes the evaluation parameters.

During the training phase, we gathered the following information: 1) which method is the fastest, 2) what method results in the best distribution across partitions (measured in comparison to the partition storing the largest amount of data), 3) the variance in data population, 4) the number of empty partitions, 5) the differences among the results of training sets of varying size, and 6) the accuracy obtained during the training phase in comparison to calculating the partitioning schemes on the complete data set. The rationale behind investigating these characteristics is as follows.

1) Duration The duration[1] of the training phase provides a notion of how long it takes to get a good partitioning scheme. We observe that in many e-science communities, data sets are updated only every few months or on a yearly basis. Due to the invariance of the data, the training phase merely is a one-time cost.

2) Average data population The average data population in comparison to the biggest leaves gives a notion about how similar the data distribution is among the individual leaves.

3) Variance in partition population The variance is similar to the average data population. A low variance implies that all partitions contain approximately the same number of objects.

4) Empty partitions Without empty partitions, we can directly map data partitions to the shared servers. In cases where we cannot avoid empty leaves, we can create more partitions than the anticipated number of servers and assign multiple partitions to a server.

5) Size of the training set If outcomes of a small training set are comparable to the results of a large training set, we can use the smaller one and thus reduce the cost of the training phase.

6) Baseline comparison Computing the histogram on the complete data set would generate the exact partitioning scheme. However, in many circumstances this is not feasible due to the enormous data volumes.

4.1. Duration

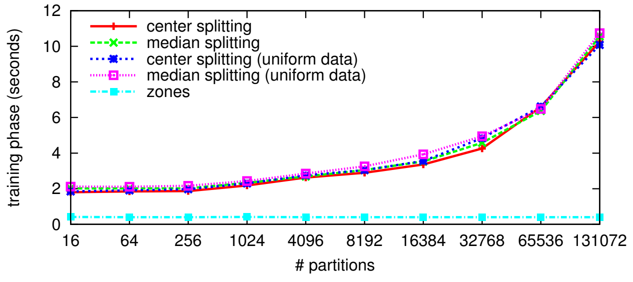
The graphs in Figure 5 show the duration of the training phase (in seconds) for both Quadtree variants and the Zone-partitioning on the different training sets for the 0.1% training sample and 10% training sample, respectively. The Zone-partitioning is invariant to the data distribution and can be computed very fast by providing the partition cardinality. For partitioning schemes which consider data distribution, the duration increases with the number of partitions. The larger the training sample, the more the duration is dominated by the processing of the training sample. Therefore, the influence of a higher number of partitions is less prominent in Figure 5b compared to Figure 5a.

For median-based Quadtrees, the training phase lasts longer than for standard Quadtrees. In addition to finding the biggest leaf, which is necessary in both cases, the median needs to be calculated. This is more expensive than only splitting a leaf. As we use an $O(n)$ order statistic algorithm to compute the median instead of an $O(n \log n)$ sorting-based algorithm, the duration increases in a similar way for both variants.

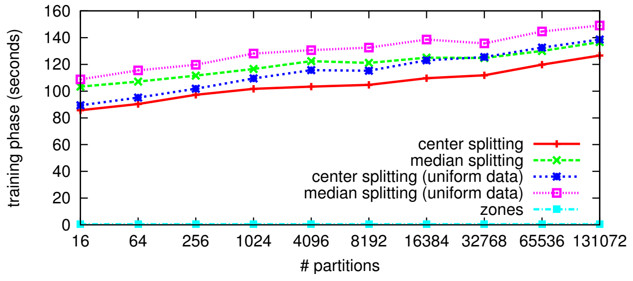
[1]We performed the evaluation of the training phase on a Linux server equipped with an Intel Xeon processor at 2.8 GHz and 4 GB of RAM.

Parameter	Value	Description
D	D_{skew}, $D_{uniform}$	Datasets used for training sample extraction.
s	0.01%, 0.1%, 1%, 10%	Size of the training set (of D).
n	$2^4, 2^6, 2^8, 2^{10}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}$	Size of the partitioning scheme.

Table 1. Parameters for the training phase evaluation.



(a) 0.1% training sample



(b) 10% training sample

Figure 5. Duration of the training phase.

4.2. Average Data Population

The covered area of the data partitions can differ in size either due to the data resolution (Quadtrees increase the resolution for highly populated data regions) or position (Zones close to the poles cover a smaller area than regions close to the equator). When inserting the training samples into the data structures, each data object is assigned to a containing partition. We call the data objects that are assigned to a partition p, the *population* of p. Having defined the population of a partition, we now can compare the *average data population (ADP)* to the maximum population among the partitions. A more homogeneous data distribution results in a higher ADP. If the ADP is 100% then all partitions are equally populated.

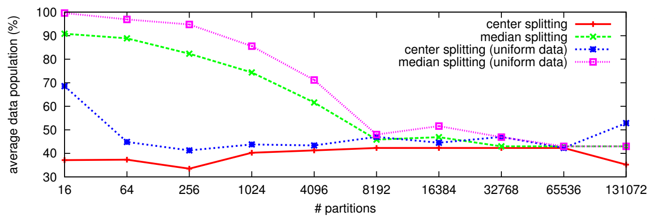
The ADP graphs for the Quadtree variants for different training samples and the graph for the Zones on the complete data set are shown in Figure 6. Constructing Quadtrees bottom-up only considers the data distribution for selecting the most populated leaf. Therefore, results for all training sets are very similar. In conjunction with the median heuristics, Quadtrees achieve a very good average population in both data sets, especially in $D_{uniform}$. Figure 7 explains why the values for median-based Quadtrees drop significantly in Figures 6b and 6c for 8 192 partitions. In order to divide all 4 096 partitions equally, 4-times more, that is 16 384, leaves are necessary. Therefore, the algorithm is "still underway" during the configurations that exhibit this sudden change in average population. Zones achieve a good ADP in homogeneous environments, whereas in skewed data sets, the ADP is lower in comparison to the Quadtree variants.

4.3. Variance in Data Distribution

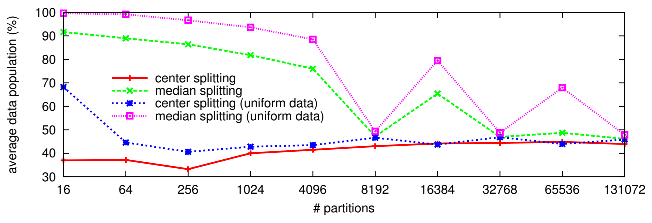
Next, we measured the variance for each partition. The larger the variance, the more diverse is the distribution of data points among the partitions. The graphs in Figure 8 depict the variance on a logarithmic scale. The median-based heuristics decreases the variance for both data sets, especially for the uniform data. Generating 4 096 partitions from the 10% training sample results in a variance of 8 479 and 2 052 645 for the median-based variant and of 199 275 and 3 336 748 for the standard Quadtree for $D_{uniform}$ and D_{skew}, respectively. In each graph, the variance decreases because the size of the training set is kept constant and therefore the amount of data objects for each partition is reduced. The variance of the Zones approach develops similar to that of the Quadtree.

4.4. Empty Partitions

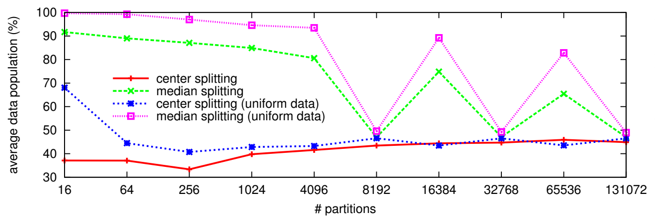
The next comparison is with regard to the number of empty partitions. The results are presented in Figure 9 as percentage of all partitions. While D_{skew} has large areas with no data (Figure 3), $D_{uniform}$ does not. For both small samples (0.01% and 0.1%), however, the number of empty leaves increases significantly for both datasets from 16 384 partitions onwards (see Figure 9a). We discuss this deficiency of small data samples in the following section. For D_{skew}, all Zones configurations have about 7% empty partitions. Roughly 13° of 180° at the bottom of Figure 3 contain no data, i. e., 7% of the declination domain are empty. As the Zones divide the data space regularly, this ratio remains constant. For $D_{uniform}$, no empty Zone partitions exist. For the 1% and 10% training samples of D_{skew} (the latter is shown in Figure 9b), the percentage of empty partitions steadily decreases for the Quadtree, while the median heuristics eliminates empty regions completely. At 256 Quadtree partitions, all "white space" has been roughly approximated with 6 empty partitions and additional empty regions only develop at the "edges" of dense data areas, which explains our measurements for D_{skew}. We do not show the graphs for the samples of the uniform data set, as all three data structures created no empty partitions if an sufficiently large training sample is used.



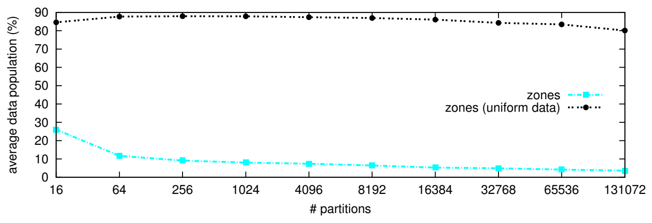
(a) 0.1% training sample



(b) 1% training sample



(c) 10% training sample



(d) Complete data

Figure 6. Average population of a partition in comparison to the leaf with the highest population.

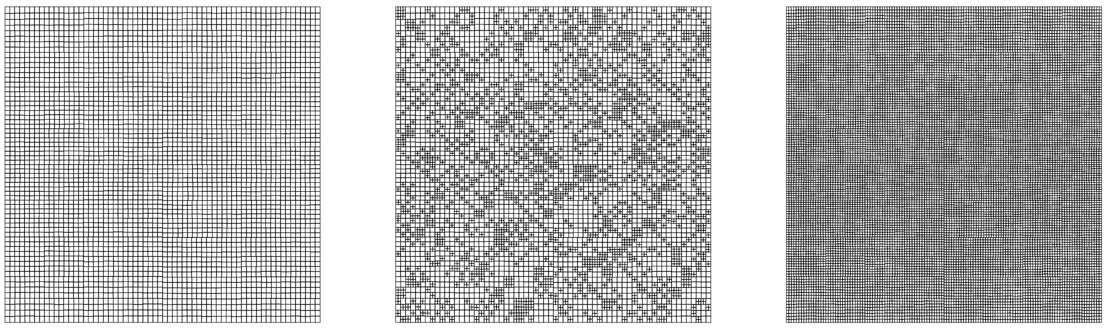
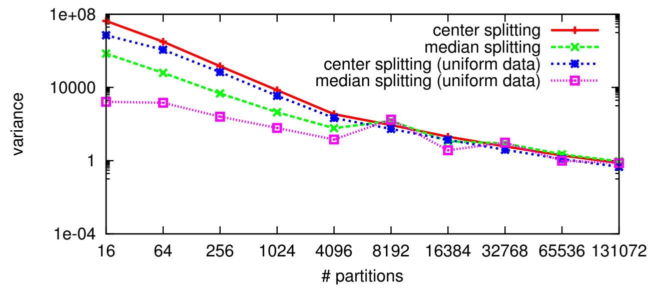
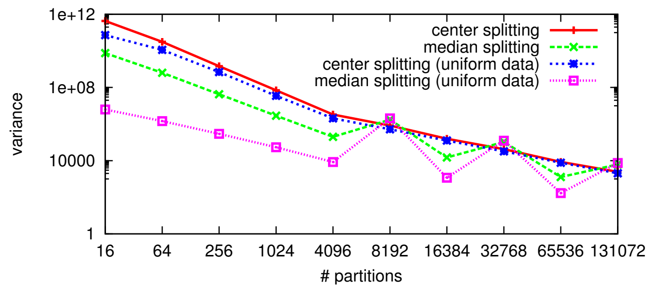


Figure 7. Median-based Quadtree for $D_{uniform}$ with 4 096, 8 192, and 16 384 partitions.

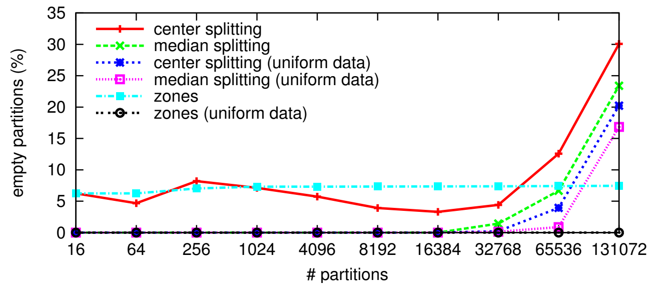


(a) 0.1% training sample

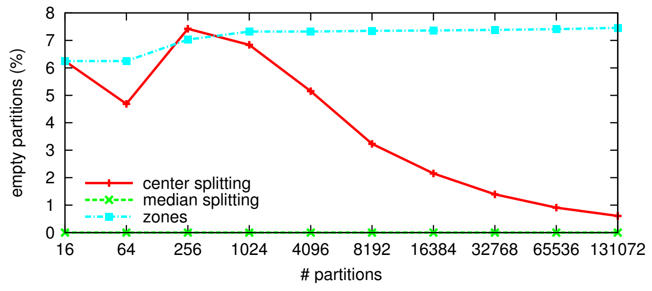


(b) 10% training sample

Figure 8. Variance in data distribution.



(a) 0.1% training sample



(b) 10% training sample, D_{skew}

Figure 9. Empty partitions.

Sample	D_{skew}	$D_{uniform}$
0.01%	14 103	16 851
0.1%	138 590	169 058
1%	1 382 150	1 692 883
10%	13 837 271	16 938 822

Table 2. Size of the used training samples.

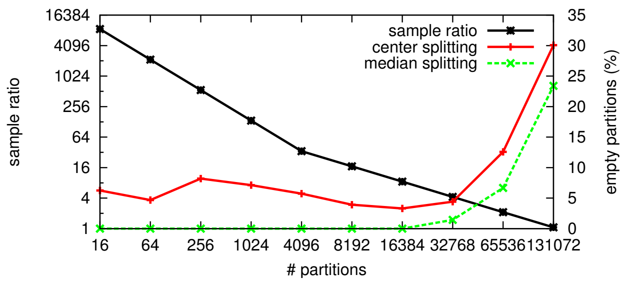


Figure 10. Effect of decreasing sample ratio.

4.5. Size of the Training Set

In the following, let the *sample ratio* $r^s_D(n)$ be the size s of a training sample of a data set D divided by the number n of partitions. The sample ratio explains the high percentage of empty partitions for small (0.01% and 0.1%) samples. Table 2 shows the cardinality of the training samples for D_{skew} and $D_{uniform}$, respectively. In Figure 10, the first curve is $r^{0.1}_{skew}$, the sample ratio for the 0.1% training sample of D_{skew}. Increasing the number of partitions, we have about as many objects as partitions, i. e., the ratio converges towards 1 (on the left-hand axis). Thus, a leaf of a Quadtree is empty with high probability. For easier comparison, we added the curves of both Quadtrees from Figure 9a, showing how the percentage of empty partitions raises. The observations for the 0.01% training sample are similar for both data sets. To summarize, a training set is only suitable for the training phase, if its sample ratio is sufficiently high.

4.6. Baseline Comparison

To evaluate the quality of partitioning schemes generated during the training phase, we compared the partitioning schemes obtained from our training samples with the partitioning scheme computed using the complete data as training sample. Due to space constraints, we only discuss the results for the Quadtree-based partitioning schemes of D_{skew} with 16 partitions. The partitioning schemes determined on the basis of the different training samples (0.01%, 0.1%, 1%, and 10%) were identical to the scheme computed on the basis of the complete data. Thus, in this particular case (Quadtrees, D_{skew}, 16 regions), all our training sets yield perfect partitioning schemes. Figure 11 shows the population of each individual partition relative to the size of the corresponding training sample, sorted in ascending order. We can see that the training samples are indeed representative because the distribution of the population is similar for all samples as well as for the original data. Note that computing partitioning schemes based on complete data sets is prohibitively expensive when dealing with very large current or future data sets. Thus, a baseline comparison as the one above is infeasible under such circumstances.

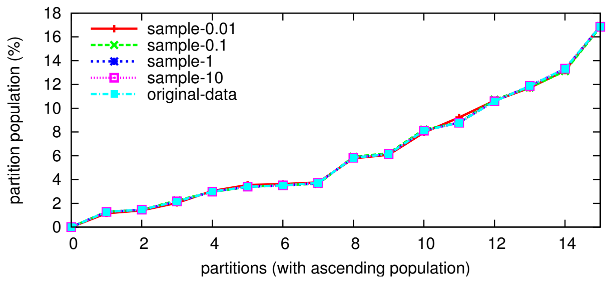


Figure 11. Baseline comparison for the standard Quadtree, D_{skew}, and 16 partitions.

4.7. Discussion

We shortly summarize the observations made during our evaluation. The training phase itself does not take very long, so it could be worthwhile comparing several training sets. This, for example, allows the detection of non-representative samples, if the curve in Figure 11 diverges too much from the others. Generating all 170 partitioning schemes (80 for each Quadtree variant, and 10 Zone-based) lasted about 90 minutes in our evaluation setup. Computing partitioning schemes such as median-based Quadtrees on the complete data set is a far more complex task. If training samples have a sufficiently high sample ratio, comparatively small data samples already provide good partitioning schemes. For our skewed data sample, median-based Quadtrees achieved the best load-balancing and had quite homogeneous average data populations. Depending on the data structure, the number of partitions influences the data distribution characteristics. For example, 2-dimensional Quadtrees should be generated with multiples of 4 as partition cardinalities. Besides the parameters used in our evaluation, there are several other means to compare the data structures and further properties of the data or queries that can be relevant for choosing the best partitioning scheme. If the partitioning scheme changes frequently and is transmitted regularly, the size of the data structure is also relevant for the decision process. Furthermore, evaluating the data partitioning against a typical workload can give further valuable insights.

5. Related Work

A recent survey [16] summarizes many of the various design options for Data Grids with regards to organizational structure, data transport, data replication, and scheduling by defining a comprehensive taxonomy for Data Grids and maps existing Data Grids to that topology. Throughout this paper, we focused on *federated Data Grids*.

GIME [17] takes a different approach to geotechnical information management in federated Data Grids. The system adheres to the data autonomy of the participating institutions and uses a replicated index (based on Quadtrees or R-trees) for managing the bounding boxes of participating archives. Thus, it reduces the number of messages by submitting the query only to such archives whose minimum bounding box actually intersects the query area. Load imbalance can arise for data archives covering a large area. These archives have to process more queries than small archives and several data sets cannot be combined directly on-site. AstroPortal [12] combines digital images from several astronomical archives by offering a Grid-based stacking service in order to create a "complete picture".

HiSbase [15] investigates the potential of P2P networks for data-intensive applications within the convergence of P2P technologies and Grid Computing [5]. HiSbase enables histogram-based P2P data management by using the training phase described in this paper. It achieves high flexibility by introducing an additional indirection by mapping data partitions to an overlay network instead of mapping partitions directly to servers. In first evaluations, HiSbase improved query throughput significantly because of increased cache locality, parallelism, and better load-balancing.

Decentralized Grid-based data stream management [9] is a different approach to increase the scalability of e-science research efforts. In this approach, complex data-intensive workflows can benefit from the data stream sharing [8, 10] optimization technique which comprises in-network query processing and multi-query optimization.

6. Summary

In this paper, we describe a flexible framework for investigating community-specific index structures used as partitioning scheme for federated Data Grids. Distributing the data across several Grid resources to level skewed data distributions while preserving spatial locality yields improved throughput and better load balancing for data-intensive applications. We evaluated Quadtree variants and the Zones algorithm and their capabilities to partition data from several repositories for federated Data Grids. Our criteria can be applied or extended in many other e-science communities. For future work, we plan to combine the flexibility of HiSbase with OGSA-DAI as well-established Grid-middleware for data access in order to shape community Grids for forthcoming data challenges.

References

[1] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. C. Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead. The design and implementation of Grid database services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17(2-4):357–376, 2005.

[2] M. Antonioletti, M. Atkinson, A. Krause, S. Laws, S. Malaika, N. W. Paton, D. Pearson, and G. Riccardi. Web Services Data Access and Integration - The Core (WS-DAI) Specification, Version 1.0. http://www.ogf.org/documents/GFD.74.pdf, July 2006.

[3] H. Enke, M. Steinmetz, T. Radke, A. Reiser, T. Röblitz, and M. Högqvist. AstroGrid-D: Enhancing Astronomic Science with Grid Technology. In *Proc. of the German e-Science Conference*, Baden-Baden, Germany, May 2007.

[4] R. A. Finkel and J. L. Bentley. Quad Trees A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4:1–9, Mar. 1974.

[5] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, volume 2, 2003.

[6] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231, June 1998.

[7] J. Gray, M. A. Nieto-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.

[8] R. Kuntschke and A. Kemper. Data Stream Sharing. In *Proc. of the Intl. Workshop on Pervasive Information Management*, Munich, Germany, Mar. 2006.

[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] R. Kuntschke, B. Stegmaier, A. Kemper, and A. Reiser. StreamGlobe: Processing and Sharing Data Streams in Grid-Based P2P Infrastructures. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 1259–1262, Trondheim, Norway, Aug. 2005.

[11] M. A. Nieto-Santisteban, J. Gray, A. S. Szalay, J. Annis, A. R. Thakar, and W. J. O'Mullane. When Database Systems Meet the Grid. In *Proc. of the Conference on Innovative Data Systems Research*, pages 154–161, Asilomar, CA, USA, Jan. 2005.

[12] I. Raicu, I. Foster, A. Szalay, and G. Turcu. AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis. In *Proc. of the TeraGrid Conf.*, June 2006.

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

[14] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[15] 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*, Vienna, Austria, Sept. 2007. Accepted for publication.

[16] 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.

[17] R. Zimmermann, W.-S. Ku, H. Wang, A. Zand, and J.-P. Bardet. A Distributed Geotechnical Information Management and Exchange Architecture. *IEEE Internet Computing*, (5):26–33, 2006.