T2: A Customizable Parallel Database For Multi-dimensional Data *

Chialin Chang†, Anurag Acharya‡, Alan Sussman†, Joel Saltz†+

† Dept. of Computer Science
University of Maryland
College Park, MD 20742

‡ Dept. of Computer Science
University of California
Santa Barbara, CA 93106

+ Dept. of Pathology
Johns Hopkins Medical Institutions
Baltimore, MD 21287

chialin@cs.umd.edu, acha@cs.ucsb.edu, als@cs.umd.edu, saltz@cs.umd.edu

1 Introduction

As computational power and storage capacity increase, processing and analyzing large volumes of data play an increasingly important part in many domains of scientific research. Typical examples of large scientific datasets include long running simulations of time-dependent phenomena that periodically generate snapshots of their state (e.g. hydrodynamics and chemical transport simulation for estimating pollution impact on water bodies [4, 6, 20], magnetohydrodynamics simulation of planetary magnetospheres [32], simulation of a flame sweeping through a volume [28], airplane wake simulations [21]), archives of raw and processed remote sensing data (e.g. AVHRR [25], Thematic Mapper [17], MODIS [22]), and archives of medical images (e.g. confocal light microscopy, CT imaging, MRI, sonography).

These datasets are usually multi-dimensional. The data dimensions can be spatial coordinates, time, or experimental conditions such as temperature, velocity or magnetic field. The importance of such datasets has been recognized by several database research groups and vendors, and several systems have been developed for managing and/or visualizing them [2, 7, 14, 19, 26, 27, 29, 31].

These systems, however, focus on lineage management, retrieval and visualization of multi-dimensional datasets. They provide little or no support for analyzing or processing these datasets – the assumption is that this is too application-specific to warrant common support. As a result, applications that process these datasets are usually decoupled from data storage and management, resulting in inefficiency due to copying and loss of locality. Furthermore, every application developer has to implement complex support for managing and scheduling the processing.

Over the past three years, we have been working with several scientific research groups to understand the processing requirements for such applications [1, 5, 6, 10, 18, 23, 24, 28]. Our study of a large set of applications indicates that the processing for such datasets is often highly stylized and shares several important characteristics. Usually, both the input dataset as well as the result being computed have underlying multi-dimensional grids, and queries into the dataset are in the form of ranges within each dimension of the grid. The basic processing step usually consists of transforming individual input items, mapping the transformed items to the output grid and computing output items by aggregating, in some way, all the transformed input items mapped to the corresponding grid point. For example, remote-sensing earth images are often generated by performing atmospheric correction on several days worth of raw telemetry data, mapping all the data to a latitude-longitude grid and selecting those measurements that provide the clearest view.

In this paper, we present *T2*, a customizable parallel database that integrates storage, retrieval and processing of multi-dimensional datasets. T2 provides support for many operations including index generation, data retrieval, memory management, scheduling of processing across a parallel machine and user interaction. It achieves its primary advantage from the ability to seamlessly integrate data retrieval and processing for a wide variety of applications and from the ability to maintain and process multiple datasets with different underlying grids. Most other systems for multi-dimensional data have focused on uniformly distributed datasets, such as images, maps, and dense multi-dimensional arrays. Many real datasets, how-

*This research was supported by the National Science Foundation under Grant #ASC 9318183, ARPA under Grant #DABT 63-94-C-0049 (Caltech Subcontract #9503), and the Office of Naval Research under Grant #N6600197C8534. Anurag Acharya is supported by a grant from the College of Engineering.

ever, are non-uniform or unstructured. For example, satellite data is a two dimensional strip that is embedded in a three dimensional space; water contamination studies use unstructured meshes to selectively simulate regions and so on. T2 can handle both uniform and non-uniform datasets.

T2 has been developed as a set of modular services. Since its structure mirrors that of a wide variety of applications, T2 is easy to customize for different types of processing. To build a version of T2 customized for a particular application, a user has to provide functions to pre-process the input data, map input data to elements in the output data, and aggregate multiple input data items that map to the same output element.

T2 presents a uniform interface to the end users (the clients of the database system). Users specify the dataset(s) of interest, a region of interest within the dataset(s), and the desired format and resolution of the output. In addition, they select the mapping and aggregation functions to be used. T2 analyzes the user request, builds a suitable plan to retrieve and process the datasets, executes the plan and presents the results in the desired format.

In Section 2 we first present several motivating applications and illustrate their common structure. Section 3 then presents an overview of T2, including its distinguishing features and a running example. Section 4 describes each database service in some detail. An example of how to customize several of the database services for a particular application is given in Section 5. T2 is a system in evolution. We conclude in Section 6 with a description of the current status of both the T2 design and the implementation of various applications with T2.

2 Motivating examples

Satellite data processing: Earth scientists study the earth by processing remotely-sensed data continuously acquired from satellite-based sensors, since a significant amount of earth science research is devoted to developing correlations between sensor radiometry and various properties of the surface of the earth. A typical analysis [1, 5, 18] processes satellite data for ten days to a year and generates one or more composite images of the area under study. Generating a composite image requires projection of the globe onto a two dimensional grid; each pixel in the composite image is computed by selecting the "best" sensor value that maps to the associated grid point. A variety of projections are used by earth scientists – the USGS cartographic transformation package supports 24 different projections [33] . An earth scientist specifies the projection that best suits her needs, maps the sensor data using the chosen projection, and generates an image by compositing the projected data. Sensor values are pre-processed to correct the effects of various distortions, such as instrument drift, atmospheric distortion and topographic effects, before they are used.

Virtual Microscope and Analysis of Microscopy Data : The Virtual Microscope [10] is an application we have developed to support the need to interactively view and process digitized data arising from tissue specimens. The Virtual Microscope provides a realistic digital emulation of a high power light microscope. The raw data for such a system can be captured by digitally scanning collections of full microscope slides under high power. At the basic level, it can emulate the usual behavior of a physical microscope including continuously moving the stage and changing magnification and focus. Used in this manner, the Virtual Microscope can support completely digital dynamic telepathology [34]. In addition, it enables new modes of behavior that cannot be achieved with a physical microscope, such as simultaneous viewing and manipulation of a single slide by multiple users, and three dimensional image reconstruction and registration from multiple microscope slides marked by various special stains.

The digitized image from a slide is effectively a three dimensional dataset, since each slide can contain multiple focal planes. In the operation of the virtual microscope, high resolution data is retrieved, decompressed and projected onto a grid of suitable resolution (governed by the desired magnification). A compositing algorithm is applied to all pixels mapping onto a single grid point to avoid introducing spurious artifacts into the displayed image. In the future, we plan to make use of multi-resolution data structures to make it possible to acquire and store data arising from different spatial regions at different levels of resolution.

Water contamination studies: Environmental scientists study the water quality of bays and estuaries using long running hydrodynamics and chemical transport simulations [4, 6, 20]. The hydrodynamics simulation imposes an unstructured grid on the area of interest and determines circulation patterns and fluid velocities over time. The chemical transport simulation models reactions and transport of contaminants, using the fluid velocity data generated by the hydrodynamics simulation. This simulation is performed on a different unstructured spatial grid, and often uses significantly coarser time steps. This is achieved by mapping the fluid velocity information from the circulation grid, averaged over multiple fine-grain time steps, to the chemical transport grid and computing smoothed fluid velocities for the points in the chemical transport grid. As the chemical reactions have little effect on the circulation patterns, the fluid velocity data can be generated once and used for many contamination studies.

3 Overview

In this section, we provide an overview of T2. We describe its distinguishing features and use a database that generates composite images out of raw satellite data as an example.

There are four distinguishing features of T2. First, it is targeted towards multi-dimensional datasets – the attributes of each dataset form some underlying multi-dimensional attribute spaces (e.g., spatial coordinates, time, temperature, velocity, etc.). T2 can simultaneously manage and process multiple datasets with different attribute spaces and different distributions of data within each attribute space. For example, T2 can manage satellite data at multiple stages in a processing chain, ranging from the initial raw data that consists of a two dimensional strip embedded in a three dimensional space to ten day composites that are two dimensional images in a suitable map projection to monthly composites that are 360x180 images with one pixel for each longitude-latitude element. T2 uses multi-dimensional indices (e.g., R^*-trees [3, 13], quad-trees [11]) to manage these datasets. For a given dataset, a separate index is created for every attribute space of interest. For example, the underlying attribute space for AVHRR satellite data has three axes - latitude (in 1/128th of a degree), longitude (1/128th of a degree) and time (in seconds). During processing, this attribute space is mapped to another attribute space, which is a grid in the *Interrupted Goodes Homolosine* map projection [30]. T2 allows users to index this dataset either on the underlying latitude-longitude-time attribute space or on the attribute space jointly defined by the Goodes map projection and time.

Second, T2 leverages commonality in processing requirements to seamlessly integrate data retrieval and processing for a wide variety of applications. It provides support for a variety of common operations such as index generation, data retrieval, memory management, scheduling of processing across the parallel machine and user interaction.

Third, T2 can be customized for a wide variety of applications without compromising efficiency. To customize T2, a user has to provide (1) a transformation function to pre-process individual input items;

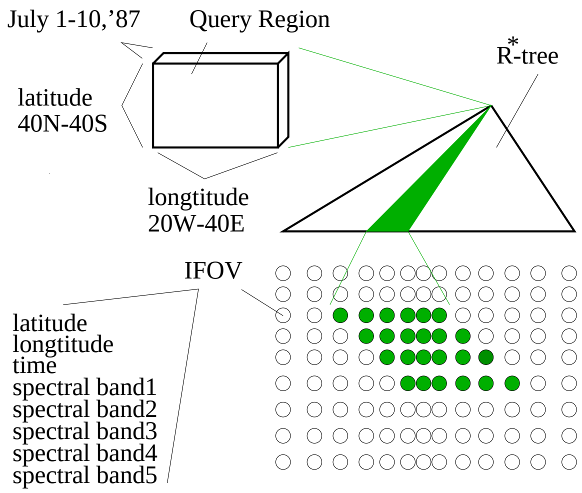

Figure 1: Example of a T2 query for an AVHRR dataset. The query region is specified in terms of the attribute space that underlies the AVHRR dataset.

(2) one or more mapping functions to map from the input attribute space to the output attribute space (multiple functions are automatically composed by T2); and (3) an aggregation function to compute an output data item given the set of input data items that map to it.

Fourth, T2 leverages the commonality in the structure of datasets and processing to present a uniform interface. A T2 query is specified by the dataset(s) of interest, a region of interest within the dataset(s), and the desired format, resolution and destination of the output. In addition, users select the transformation, mapping and aggregation functions. The output of a T2 query is also multi-dimensional, and the attribute space for the output is implicitly specified by the query. The region of interest can be specified in terms of any attribute space that the dataset has an index on. For example, a query to retrieve and process AVHRR data could specify its region of interest in terms of either the latitude-longitude-time attribute space that underlies the AVHRR dataset or the attribute space defined by the Goodes map projection and time.

Figures 1 and 2 show how T2 is used to generate an output image from processing raw AVHRR data. Each data item in the AVHRR dataset is referred to as an *instantaneous field of view* (IFOV), and consists of eight attributes – three key attributes that specify the spatio-temporal coordinates and five data attributes that contain observations in different parts of the electromagnetic spectrum. IFOVs from multiple orbits are stored in T2, although Figure 1 only shows a strip from one orbit.

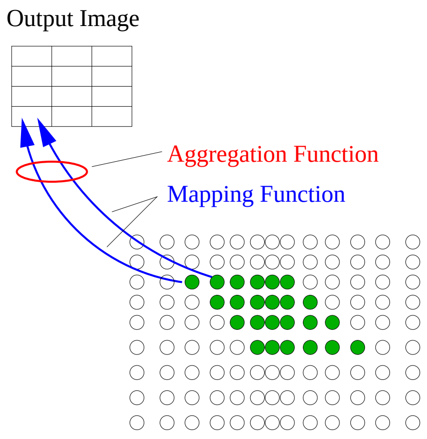

Figure 2: The output of a query into an AVHRR dataset is an image in the Goodes map projection. A transformation function is applied to each IFOV for correction, but is not shown.

A query consists of a query region that contains the IFOVs of interest, the parameters of the output image (i.e. grid resolution), references to a transformation function, a mapping function and an aggregation function, and what to do with the output image (e.g., store on disk, send to another program, etc.). The query region is specified in terms of the latitude-longitude-time attribute space, and an R^*-tree indexed over the IFOVs on that attribute space is used to identify the IFOVs of interest. Each IFOV selected for the query is pre-processed by the specified transformation function to correct the effects of various distortions – instrument drift, atmospheric distortion and topography. It is then mapped to a pixel in the output image by the specified mapping function. Since the query region extends over ten days and observations from different orbits overlap spatially, multiple IFOVs may map to an output pixel. The specified aggregation function for an output pixel selects the "best" corrected IFOV that maps to the output pixel, based on a measure of the clarity of the sensor readings. Figure 2 illustrates these operations.

4 System Architecture

T2 has been developed as a set of modular services, as shown in Figure 3. Some of the functions provided by these services, such as the indexing service, correspond directly to those provided by object-relational database systems; other functions are provided to support the stylized processing required by our target applications. While we expect that many applications will be able to use the services as is, we anticipate that some applications may need to replace or modify some of the services.

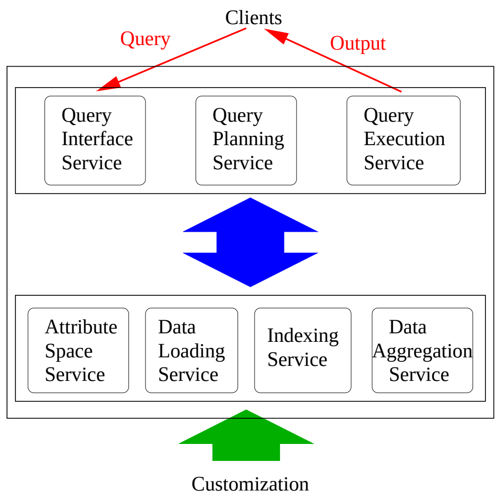

Figure 3: The architecture of T2.

4.1 The attribute space service

The attribute space service manages the registration and use of attribute spaces and mapping functions. Mapping functions are used to either map individual points between previously registered attribute spaces or to map points from a registered attribute space to define a new attribute space. In this section, we describe how attribute spaces and mapping functions are specified and maintained.

Multi-dimensional attribute spaces are the central structures in T2. All other structures and operations are specified in terms of these attribute spaces. An attribute space is specified by the number of dimensions and the range of values in each dimension. For user convenience, additional information can be stored with an attribute space. For example, a name and the resolution of the values in a dimension can be specified (e.g. for the latitude-longitude-time attribute space from Section 3, the resolution of the latitude dimension is 1/128-th of a degree).

T2 supports two kinds of attribute spaces: *base* and *derived*. *Base* attribute spaces are explicitly specified by the user and are persistent, so can be identified by names that are visible to users. A *derived* attribute space is specified as a (*base* attribute space, mapping function) pair. Logically, a *derived* space is defined as the space generated by mapping every point in the *base* space using the mapping function.

Mapping functions are specified by the domain and range attribute spaces and an algorithm for the mapping between them. Currently, mapping functions are statically linked; we plan to provide dynamic linking in the near future.

4.2 The data loading service

The data loading service manages the process of loading new datasets into T2. To load a new dataset into T2, a user has to specify the format, location and metadata for the dataset and the data loading service takes care of loading the dataset and integrating it into the database.

T2 expects incoming datasets to be partitioned into *chunks*, each chunk consisting of one or more data items. T2 allows users to pick any chunk size (all chunks do not have to be the same size); users should pick chunk sizes that allow for efficient retrieval from disk, because chunks are the unit of disk retrieval in T2 (for the IBM Starfire disks that our T2 prototypes run on, we pick chunk sizes greater than 128 KB). The format of the dataset is specified by: (1) the name of a *base* attribute space that underlies the dataset (we call this the *native* attribute space of the dataset); (2) the size of each chunk in the dataset; (3) the number of chunks in the dataset; (4) an iterator function that iterates over the set of data items in a single chunk; and (5) and an access function that given a data item, returns its coordinates in the underlying attribute space.

The metadata for the dataset consists of placement information. T2 assumes that a disk farm is attached to the processors and placement information is needed to determine the data layout. There are two components of the placement information, both of which are optional. The first is a list of *minimum bounding rectangles* (*mbr*) for each chunk being loaded. An *mbr* for a chunk is a specification of the extent of the data items in the chunk in the attribute space. If the *mbr* information is not specified, it is automatically computed using the iterator and the access functions. The second is a pair of algorithms – one to decluster the chunks to individual disks and the other to cluster them on individual disks. Each algorithm is specified by name. As for mapping functions, T2 currently supports only static linking. By default, T2 uses the *minimax* algorithm [23, 24] for declustering and the Short Spanning Path (SSP) algorithm [9] for clustering. In addition, T2 allows the data layout to be separately computed and provided in a file. This would be useful if the algorithms used to compute the placements were embedded in some other application that could not be structured to fit T2's interface requirements.

Once the data layout is specified, the service computes an efficient schedule for moving the chunks to their destinations and executes the schedule.

4.3 The indexing service

The indexing service creates an index for a given (dataset, attribute space) pair. An attribute space can be used for indexing a dataset if and only if it is either the *native* attribute space of the dataset or the target of a chain of mapping functions that maps the *native* attribute space to the new attribute space. T2 allows users to optionally specify an indexing algorithm; by default it uses a variant of R^*-trees.

An index can be created at any time, although it is expected that most indices will be created as a part of the data loading operation. To create an index, the indexing service uses information about the *mbr* for each chunk in the dataset and about the physical location of each chunk on disk. It obtains this information from the data loading service. For *derived* attribute spaces, the indexing service uses the associated mapping function to first map the *mbr* for each chunk into the *derived* attribute space.[1]

4.4 The data aggregation service

The data aggregation service manages the user-provided functions to be used in aggregation operations. It manages the namespace of these functions and performs type checking both when the functions are registered (as a part of customization) and when they are used in response to a query. This service manages two kinds of functions: *transformation functions* and *aggregation functions*, as described in Section 3. Transformation functions are used to preprocess data items before aggregation. Aggregation functions are assumed to be commutative and associative and can be applied to individual data items in parallel and in any order. T2 is able to deal with both *distributive* and *algebraic* aggregation functions as defined by Gray et. al [12].

Functions are specified by a (function name, object file name) pair. The query interface service uses namespace information from the data aggregation service to allow the user to find the set of transformation functions and aggregation functions that can be applied to a given dataset.

4.5 The query interface service

The query interface service has two functions. First, it allows clients to find out what datasets are available and what functions and indices are associated with each dataset. Second, it allows clients to formulate and present valid queries.

As a part of the first function, the query interface service allows clients to browse all the namespaces

[1]Recall that a *derived* attribute space is specified as a (*base* attribute space, mapping function) pair.

in T2: (1) attribute spaces, (2) datasets, (3) indices, (4) placement algorithms, (5) mapping functions, (6) transformation functions, and (7) aggregation functions. As a part of the second function, it ensures that for each query: (1) the domain of the transformation function selected is the same as that of the input dataset (i.e. the types are the same); (2) the range of the transformation function has the same type as the domain of the aggregation function; and (3) the chain of mapping functions is consistent (that is, all the types and shapes match) and the input attribute space of the first mapping function matches the native attribute space of the dataset selected.

4.6 The query planning service

To be able to efficiently integrate data retrieval and processing on a parallel machine, T2 manages the allocation and scheduling of all resources, including processor, memory, disk bandwidth and network bandwidth. The task of the query planning service is to determine a schedule for the use of these resources to satisfy a query. Given the stylized nature of the computations supported by T2, use of several of these resources is not independent (e.g., it is not possible to use disk bandwidth without having memory to store the data being transferred from disk). In addition, the associative and commutative nature of the aggregation operations must be leveraged to form loosely synchronized schedules – the schedules for individual processors need not proceed in lock-step and only need to synchronize infrequently.

The input to the T2 query planning service consists of: (1) the list of *chunks* that need to be processed, their location on disk and the region of the output attribute space that each of them maps to; (2) the dependencies between *chunks* – dependencies occur when multiple datasets are being processed simultaneously; (3) a description of the output dataset, including the underlying attribute space and the size of each output data item; and (4) the amount of memory available on each processor. The output of the planning service consists of a set of ordered lists of *chunks*, one list per disk in the machine configuration. Each list consists of a sequence of sublists separated by synchronization markers. The operations in each sublist can be performed in any order; all operations in a sublist must be completed before any operation in the subsequent sublist can be initiated. This restriction is enforced to ensure schedulability.

We now briefly describe how these resources are taken into consideration during the planning, assuming a shared-nothing database architecture.

Load balancing: the query planning service considers two classes of load balancing. The first class, referred to as *input partitioning*, requires each processor to generate an independent intermediate result based on the chunks that are stored on its disks. These intermediate results are merged in a second phase to obtain the final output. This yields correct results due to the order-independent nature of the processing. The second class, referred to as *output partitioning*, partitions the final output; the data needed to compute the portion of the output assigned to a processor is forwarded to it by all the other processors in the machine configuration. The choice between these approaches is based on several factors, including the distribution of the data in the output attribute space, the placement of the input data chunks needed to answer the query on disk, and the machine characteristics (i.e. the relative costs of computation, interprocessor communication and disk accesses).

Memory: T2 uses memory for three purposes – to hold the data read from disk or received from the network, to hold the intermediate results for the aggregation operation and to hold the final output. If enough memory is available for all three purposes, operations for all *chunks* in a sublist are scheduled together. Otherwise, memory is first allocated to hold the incoming input data and the remaining memory is partitioned between the other two uses. Each sublist, then, is processed in a sequence of iterations – each iteration being scheduled such that all data for the iteration fits into memory.

4.7 The query execution service

The query execution service manages all the resources in the system using the schedule created by the planning service. The primary feature of the T2 query execution service is its ability to integrate data retrieval and processing. It achieves this in three ways. First, it creates a *query environment* containing the set of functions that capture application-specific aspects of the processing. The query environment includes: (1) the access functions for individual data items; (2) the iterator to iterate over the data items in a chunk; (3) the transformation function; (4) the sequence of mapping functions that are applied to map each data item to the corresponding result data item; and (5) the aggregation functions needed to compute the output. In effect, explicitly maintaining this environment allows the query execution service to push processing operations into the storage manager and allows them to be performed directly on the buffer used to hold data arriving from disk. This avoids one or more levels of copying that would be needed in a layered architecture, where the storage manager and the processing belonged to different layers.

Second, this service overlaps the disk operations, network operations and the actual processing as much as possible. It does this by maintaining explicit queues for each kind of operation (data retrieval, message sends and receives, processing) and switches between them as required.

Third, it maximizes the utility of each disk retrieval by performing all processing for a data chunk while the chunk is in memory. As a result, a data chunk has to be retrieved only once. This is similar to the strip-mining and/or blocking operations performed for optimizing cache usage for matrix operations [8, 16].

The query execution service performs two kinds of synchronization. First, it enforces the synchronization indicated by the markers in the list of chunks retrieved from every disk (computed by the planning service). That is, the operations between a pair of markers can be performed in any order; all operations before a marker must be completed before any operation after the marker can be initiated. This restriction is used to avoid deadlocks.

The second type of synchronization attempts to preserve load balance by reordering operations. If a particular processor is unable to keep up with its peers, the other processors reorder their operations to reduce the amount of data that is sent to that processor. This mechanism can be used only between synchronization markers.

Assuming a shared-nothing architecture, for each iteration specified by the query plan, the query execution service goes through three phases: (1) memory allocation and initialization for intermediate and final results; (2) retrieval and processing of data; and (3) dispatching of the intermediate results – either to disk for use in a later iteration, or to another processor for further processing. The second phase consists of two sub-phases – a *local reduction phase* and a *global combine phase*. During the local reduction phase, chunks are retrieved and forwarded to wherever they should be processed, as specified by the query plan. Appropriate functions are invoked whenever a chunk arrives, either from the local disks or from the network interface. These functions iterate through the data items in a chunk, apply the transformation function to each data item, map the transformed data item to an intermediate result item using the mapping function, and finally aggregate the data items that map to each result item. After all chunks for an iteration have been retrieved and processed, the global combine phase is performed to aggregate the intermediate results.

Once all the chunks for the entire query plan have been processed, the final output dataset is computed from the intermediate results and sent to the destination specified by the query.

5 Customization example: AVHRR database

In this section, we illustrate customization in more detail using the AVHRR satellite database described in Section 3 as an example. This example is loosely based on Titan [5], a prototype data server capable of producing composite images out of raw remotely-sensed data.

The AVHRR dataset is partitioned into IFOV chunks based on the geometry of the IFOVs and the performance characteristics of the disks used to store the data. On the machine used for Titan, one reasonable partitioning creates chunks of 204x204 IFOVs – the size of each chunk is 187 KB. The format of the chunk is specified using an iterator that understands the multi-spectral nature of the values.

The three dimensional latitude-longitude-time attribute space that underlies the IFOVs is registered as a *base* attribute space with the attribute space service. An access function is used to extract the coordinate attributes from an IFOV, and the coordinates of the four corner IFOVs are used to compute for each chunk a minimum bounding rectangle in the latitude-longitude-time attribute space. The default T2 declustering and clustering algorithms described in Section 4.2 can be used to assign disk locations for the IFOV chunks. The data loading service then records all the relevant information about the AVHRR dataset, and moves the IFOV chunks to their assigned disk locations. A simplified R^*-tree suffices for indexing this dataset, and uses the spatio-temporal bounds of the IFOV chunks as access keys. The spatio-temporal bounds are specified as a region in the latitude-longitude-time attribute space. The R^*-tree shown in Figure 1 actually indexes over the IFOV chunks, not the individual IFOVs.

Since the standard AVHRR data product is presented in the Goodes map projection, a three dimensional attribute space jointly defined by the Goodes map projection and time is registered with the attribute space service as another *base* attribute space, and a mapping function is defined accordingly to map points from the latitude-longitude-time attribute space to this attribute space. This allows the indexing service to map the *mbr* of each IFOV chunk from the latitude-longitude-time attribute space to the Goodes-time attribute space, and build an index for the AVHRR dataset on the Goodes-time attribute space. With this additional index, a query region then can be specified in terms of the Goodes map projection. A two dimensional spatial attribute space can be derived from either of the three dimensional spatio-temporal attribute spaces, with a mapping function that discards the temporal coordinate. This derived spatial attribute space is used for the standard AVHRR data product.

As described in Section 3, the transformation function registered with the data aggregation service performs a sequence of corrections to each IFOV. In addition, it also computes the Normalized Difference Vegetation Index (NDVI) [15] for each IFOV, using corrected values from the first two bands of each IFOV. A registered aggregation function selects the NDVI value with the "best" IFOV among all IFOVs that map to a single output pixel, based on the clarity of the IFOV and the angular position of the satellite when the observation was made.

A typical query specifies an area of interest, usually corresponding to a geo-political area of world, and a temporal bound, which gets translated into a query region in either of the two *base* attribute spaces. The query would choose the AVHRR-correction/NDVI-generation algorithm as the transformation function, and the previously described NDVI aggregation algorithm as the aggregation function. The query also specifies the desired resolution of and where to send the output image (e.g., to disk or to another processing program). The query interface service validates the received query, and the query planning service generates an efficient schedule by taking into account the available machine resources. The query execution service carries out data retrieval and processing according to the generated schedule, and sends the output image to the destination.

6 Current Status and Future Work

We have presented T2, a customizable parallel database that integrates storage, retrieval and processing of multi-dimensional datasets. We have described the various services provided by T2, and further shown how several of those services can be customized for a particular application. In particular, we have shown how an AVHRR database, based on an existing system for handling raw remotely-sensed AVHRR satellite data, can be implemented using the services provided by T2.

We are currently in the process of implementing the various T2 services, and are designing the planning algorithm and cost models for the query planning service. We are also working on generalizing the design of the various services to handle multiple simultaneous queries. In addition, we are beginning to implement Titan, the Virtual Microscope, and a system for storing hydrodynamics simulation results for environmental studies using T2.

References

[1] A. Acharya, M. Uysal, R. Bennett, A. Mendelson, M. Beynon, J. Hollingsworth, J. Saltz, and A. Sussman. Tuning the performance of I/O-intensive parallel applications. In *Proceedings of the Fourth ACM Workshop on I/O in Parallel and Distributed Systems*, May 1996.

[2] P. Baumann, P. Furtado, R. Ritsch, and N. Widmann. Geo/environmental and medical data management in the RasDaMan system. In *Proceedings of the 23th VLDB Conference*, pages 548–552, Aug. 1997.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM-SIGMOD Conference*, pages 322–331, May 1990.

[4] C. F. Cerco and T. Cole. User's guide to the CE-QUAL-ICM three-dimensional eutrophication model, release version 1.0. Technical Report EL-95-15, US Army Corps of Engineers Water Experiment Station, Vicksburg, MS, 1995.

[5] C. Chang, B. Moon, A. Acharya, C. Shock, A. Sussman, and J. Saltz. Titan: A high performance remote-sensing database. In *Proceedings of the 1997 International Conference on Data Engineering*, pages 375–384. IEEE Computer Society Press, Apr. 1997.

[6] S. Chippada, C. N. Dawson, M. L. Martínez, and M. F. Wheeler. A Godunov-type finite volume method for the system of shallow water equations. *Computer Methods in Applied Mechanics and Engineering (to appear)*, 1997. Also a TICAM Report 96-57, University of Texas, Austin, TX 78712.

[7] D. J. DeWitt, N. Kabra, J. Luo, J. M. Patel, and J.-B. Yu. Client–server Paradise. In *Proceedings of the 20th VLDB Conference*, 1994.

[8] J. Dongarra, J. D. Croz, S. Hammarling, and I. S. Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, Mar. 1990.

[9] M. T. Fang, R. C. T. Lee, and C. C. Chang. The idea of de-clustering and its applications. In *Proceedings of the 12th VLDB Conference*, pages 181–188, 1986.

[10] R. Ferreira, B. Moon, J. Humphries, A. Sussman, J. Saltz, R. Miller, and A. Demarzo. The Virtual Microscope. In *Proceedings of the 1997 AMIA Annual Fall Symposium*, pages 449–453. American Medical Informatics Association, Hanley and Belfus, Inc.,

Oct. 1997. Also available as University of Maryland Technical Report CS-TR-3777 and UMIACS-TR-97-35.

[11] R. A. Finkel and J. L. Bentley. Quad-Trees - a data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[12] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Proceedings of the 1996 International Conference on Data Engineering*, pages 152–159, Feb. 1996.

[13] A. Guttman. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM-SIGMOD Conference*, pages 47–57, June 1984.

[14] Y. Ioannidis, M. Livny, S. Gupta, and N. Ponnekanti. ZOO: A desktop experiment management environment. In *Proc. 22nd International VLDB Conference*, pages 274–85, 1996.

[15] C. O. Justice, J. R. G. Townshend, B. N. Holben, and C. J. Tucker. Analysis of the phenology of global vegetation using meteorological satellite data. *International Journal of Remote Sensing*, pages 1271–1318, 1985.

[16] I. Kodukula, N. Ahmed, and K. Pingali. Data-centric multi-level blocking. In *Proceedings of the SIGPLAN '97 Conference on Programming Language Design and Implementation*, pages 346–357. ACM Press, June 1997. ACM SIGPLAN Notices, Vol. 32, No. 5.

[17] Land Satellite Thematic Mapper (TM). *http://edcwww.cr.usgs.gov/nsdi/html/landsat_tm/ landsat_tm*.

[18] S. Liang, L. Davis, J. Townshend, R. Chellappa, R. Dubayah, S. Goward, J. JaJa, S. Krishnamachari, N. Roussopoulos, J. Saltz, H. Samet, T. Shock, and M. Srinivasan. Land cover dynamics investigation using parallel computers. In *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium, Quantitative Remote Sensing for Science and Applications.*, pages 332–4, July 1995.

[19] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. DEVise: integrated querying and visual exploration of large datasets. In *Proceedings of ACM SIGMOD*, pages 301–12, 1997.

[20] R. A. Luettich, J. J. Westerink, and N. W. Scheffner. *ADCIRC*: An advanced three-dimensional circulation model for shelves, coasts, and estuaries. Technical Report 1, Department of the Army, U.S. Army Corps of Engineers, Washington, D.C. 20314-1000, December 1991.

[21] K.-L. Ma and Z. Zheng. 3D visualization of unsteady 2D airplane wake vortices. In *Proceedings of Visualization'94*, pages 124–31, Oct 1994.

[22] The Moderate Resolution Imaging Spectrometer. *http://ltpwww.gsfc.nasa.gov/MODIS/MODIS.html.*

[23] B. Moon, A. Acharya, and J. Saltz. Study of scalable declustering algorithms for parallel grid files. In *Proceedings of the Tenth International Parallel Processing Symposium*, pages 434–440. IEEE Computer Society Press, Apr. 1996.

[24] B. Moon and J. H. Saltz. Scalability analysis of declustering methods for multidimensional range queries. *IEEE Transactions on Knowledge and Data Engineering*, 1997. To appear.

[25] NASA Goddard Distributed Active Archive Center (DAAC). Advanced Very High Resolution Radiometer Global Area Coverage (AVHRR GAC) data. *http://daac.gsfc.nasa.gov/CAMPAIGN_DOCS/ LAND_BIO/origins.html.*

[26] The Oracle 8 spatial data cartridge, 1997. *http://www.oracle.com/st/cartridges/spatial/.*

[27] J. Patel et al. Building a scaleable geo-spatial DBMS: technology, implementation, and evaluation. In *Proceedings of ACM SIGMOD*, pages 336–47, 1997.

[28] G. Patnaik, K. Kailasnath, and E. Oran. Effect of gravity on flame instabilities in premixed gases. *AIAA Journal*, 29(12):2141–8, Dec 1991.

[29] SpatialWare DataBlade Module (from MapInfo) Corp, 1997. *http://www.informix.com/informix/ bussol/iusdb/databld/dbtech/sheets/spware.htm.*

[30] D. R. Steinwand. Mapping raster imagery to the Interrupted Goodes Homolosine projection. *http://edcwww.cr.usgs.gov/landdaac/1KM/goodesarticle.html.*

[31] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. The Sequoia 2000 storage benchmark. In *Proceedings of the 1993 ACM SIGMOD Conference*, May 1993.

[32] T. Tanaka. Configurations of the solar wind flow and magnetic field around the planets with no magnetic field: calculation by a new MHD. *Jounal of Geophysical Research*, 98(A10):17251–62, Oct 1993.

[33] The USGS General Cartographic Transformation Package, version 2.0.2. *ftp://mapping.usgs.gov/ pub/software/current_software/gctp/*, 1997.

[34] R. S. Weinstein, A. Bhattacharyya, A. R. Graham, and J. R. Davis. Telepathology: A ten-year progress report. *Human Pathology*, 28(1):1–7, Jan. 1997.