On Heterogeneous Distributed Geoscientific Query Processing *

Eddie C. Shek[†‡], Edmond Mesrobian[†], and Richard R. Muntz[†]
Computer Science Department[†]
University of California
Los Angeles, CA 90024
Information Sciences Laboratory[‡]
Hughes Research Laboratories
Malibu, CA 90265

Abstract

Geoscience studies produce data from various observations, experiments, and simulations at an enormous rate. With proliferation of geographic applications, scientific data formats, and storage systems, interoperability remains an important geoscientific data management issue that is often overlooked in current geoscientific query processing research. In this paper, we present how some issues concerning interoperability in geoscientific query processing are addressed in the Conquest parallel geoscientific query processing system under development at UCLA. The design of Conquest is based on the Volcano extensive query processing system which encapsulates parallel computation through the exchange operator. Conquest extends the exchange operator to support multicasting of data streams and hide the heterogeneity in hardware and operating system platforms from users developing parallel geoscientific applications. The Conquest data model captures some important structural and semantic properties of common geoscientific datasets and is used as a canonical model for a wide variety of scientific and non-scientific datasets. In addition, Conquest supports a uniform interface to a variety of scientific data sources. Access to data managed by a remote data repository is optimized by "pushing" Conquest operators into data repositories to maximize use of local database capability and reduce the volume of data transfer between the data repositories and Conquest.

1 Introduction

Research interest in problems presented by geoscientific data and the unique data management requirements of scientific applications [5, 19] has significantly increased over the last few years. The importance of scientific database technology is expected to increase as geoscience studies continue to produce data from various observations, experiments, and simulations at an enormous rate. For example, NASA EOS expects to produce over 1 TByte of raw data and scientific products per day by the year 2000, and a 100-year UCLA AGCM simulation [12] running at a resolution of $1° \times 1.25°$ with 57 levels generates approximately 30 TBytes of data when the model's output is written out to the database every 12 simulated hours. Geographic information systems such as ARC/INFO, geoscientific database systems like Paradise [2], and extensible database systems exemplified by Postgres [20] have made great strides towards effective management of a wide variety of spatial and scientific data objects. It is with the proliferation of geographic applications, scientific data formats, storage systems, and hardware architectures that efficient processing of geoscientific data stored in diverse distributed repositories operating in a heterogeneous environment becomes an important, yet often overlooked, issue.

The study of geosciences often involves an elaborate process of exploratory data analysis during which application-specific procedures are applied to geoscientific datasets. As a result, it is important for a scientific query processing system to be extensible, to allow users to extend the environment by introducing complex scientific operators.

At UCLA, we are developing an extensible parallel geoscientific query processing system called *Conquest*. Conquest is designed to handle complex scientific queries on large geoscientific datasets (such as those found in geoscientific data mining [14]) in a potentially heterogeneous environment. In this paper, we discuss how the design of Conquest addresses issues of interoperability without losing sight of the need for extensibility and high-performance.

- **Data Modeling.** Conquest supports a conceptually simple yet expressive data model which captures important structural and semantic properties common to geoscientific data. A set of general purpose algebraic operators are defined for the model; among them are those whose main purpose is to resolve the heterogeneity in geoscientific data representations (e.g., interpolation).

- **Parallel Query Execution Environment.** Similar to the Volcano extensible query processing system [8], Conquest encapsulates parallel computation through the use of a "parallelizing" operator. Conquest extends the Volcano exchange operator to support multicasting of data streams. In addition, inter-process communication in Conquest's extended-exchange operator masks the heterogeneity in hardware and operating system platforms from users developing parallel geoscientific applications. Data are transparently packaged based on the schema of the data.

*This research supported by NASA under HPCC grant NAG-2225.

- **Heterogeneous Data Repositories.** To provide a convenient heterogeneous distributed scientific data processing environment to scientists, Conquest supports a uniform interface to a variety of scientific data sources including several external data formats and database servers. Among many optimization techniques, optimization of access to data managed by a remote data repository is optimized by "pushing" Conquest operators into data repositories is especially important for a large-scale heterogeneous geoscientific information system. It improves overall query performance by exploiting local database capabilities and reducing the volume of data transfer between data repositories and Conquest.

Due to the large size of geoscientific datasets, content-based indexing is an attractive technique to provide efficient access to specific parts of the dataset of interest to scientists. Unlike traditional indexes, content-based indexes to large, and usually file-based, scientific datasets are often based on complex spatio-temporal objects. For example, cyclone tracks, modeled as time-series of polygonal cells on the earth surface, can be used as content-based indexes to a sea-level pressure data field. As a result of its support for interoperability, Conquest allows these content-based index structures to be maintained by special-purpose data managers optimized for them, while still being tied to the datasets being indexed in a uniform data modeling and query processing framework.

1.1 Organization

In this paper, we describe the features of Conquest which are relevant to supporting geoscientific computation in a heterogeneous distributed environment. In the next 2 sections, we present an overview of the Conquest data model and the parallel query execution architecture respectively. Section 4 discusses techniques for optimizing access to heterogeneous data repositories. Finally, Section 5 concludes the paper with a brief summary and our outlook on future research in interoperability in geoscientific query processing.

2 Data Modeling

An important issue in heterogeneous database system research [4] is that of the choice of a canonical data model into which data in heterogeneous repositories are mapped. This issue also exists for heterogeneous scientific databases. A large variety of spatio-temporal geoscientific data, ranging from multi-dimensional arrays of floating point numbers (e.g., sea-level pressure dataset) to time series of polygons (e.g., cyclone tracks), and traditional alpha-numeric data need to be brought together and processed in a large scale geoscientific data analysis application. As a result, a goal of the data model is to provide a rich but conceptually simple data model to encompass the wide variety of scientific observation and simulation data.

2.1 Related Work

The formulation of the Conquest data model is similar to existing geoscientific data models (e.g., the Spatial Archive and Interchange Format [18], and those defined in popular scientific data analysis and visualization systems such as IDL [17]) since our ultimate goal is for the model to be used as a canonical model in a heterogeneous geoscientific query processing environment. Most notably, the differential geometry notion of *fiber bundle* has been recognized by several researchers as a useful abstraction for an object-oriented scientific visualization data model [1]. In brief, a fiber bundle is a space constructed by attaching a copy of a fiber space to each point in a base space. The field model borrowed this concept of space "bundling". However, research in scientific visualization modeling concentrates mainly on the mathematical basis of data models and typically ignores physical data management issues like query optimization, parallelization, and storage layout.

The Open Geodata Interoperability Specification (OGIS) [16] specifies a data model for geographic data, as well as an application programming model aiming to enable communication and interoperability between geographic applications. It defines an object-oriented model of geographic information which covers at most 3 spatial dimensions and a temporal dimension. Again the formulation of a field in the Conquest data model is similar to that of an OGIS *geographic region* in which non-overlapping, space-filling extents map to dependent data values. However, OGIS is firmly grounded in geographic applications; its specification is limited to 3-dimensional geographic space and currently does not cover other types of dimensions such as time.

2.2 The Conquest Data Model

A characteristic of geoscientific datasets is that measurements are often recorded over a multi-dimensional space. As a result, the central idea behind the Conquest data model is that of the *field* (see Figure 1). A field is a logical construct modeling common scientific datasets in which measurement of scientific parameters are recorded for *cells* in a multi-dimensional *coordinate space*. All cells in a field for which variable values are recorded belong to the same geometric object type in the field's coordinate space. On the other hand, values recorded for cells lie in another multi-dimensional *variable space*.

Cells can be of various geometric object types of different dimensionality including points, lines, polygons, and volumes. The type of the cells and hence the coordinate space they lie in are determined by *coordinate attributes*. Note that many different cell types could be associated with the same coordinate space. For example, possible cell types in a 2-D space include points, and various region types.

Similarly, the type of variable values that can be associated with a cell in the coordinate space of a field is dictated by *variable attributes*. The variable value recorded for a cell is not limited to simple data types; it can belong to a complex spatio-temporal data type or even be another field. The types of the coordinate and variable attributes combine to define the type of a field.

Not all cells in the coordinate space of a field are associated with variable values. We define a field's *cell*

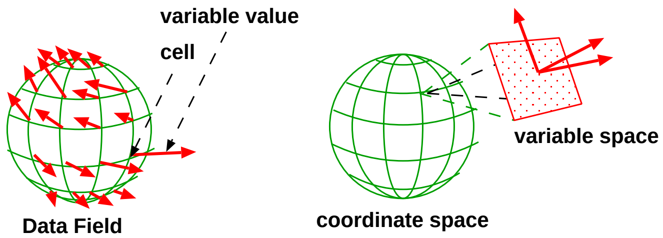

Figure 1: Example Geographic Data Field

coverage (or simply coverage) to be the set of distinct (but not necessarily disjoint) cells in its coordinate space for which variable values are recorded. Given its cell coverage, the field's *cell mapping* maps each cell in its cell coverage to a variable value. The cell coverage and cell mapping logically defines a field.

Since each cell in a field's coverage is associated with a dependent variable value, we often need to refer to both a cell and the variable value associated with it in the same context together as a single data item. For each field, a *cell record* at a cell is defined as the tuple formed by the cell in a field's cell coverage and its associated variable value.

Coordinate attributes specify the dimensions in a field's coordinate space. In addition, the actual extent within the coordinate space in which cells lie is often needed to construct data structures for storing and supporting efficient access to data. For example, we often need to know the range of coordinate values in all dimensions (i.e., the bounding box of all coordinate values) in the coordinate space to construct a hierarchical spatial data structure like the quadtree. The *cell coverage extent* of a field is determined by the extents of its coordinate attributes' values in the field. The extent of a coordinate attribute covering n dimensions is specified as a n-dimensional bounding box containing all coordinate attribute values.

2.3 Conquest Algebraic Operators

An important feature of the model is the inclusion of an extensible algebra which allows algebraic transformation techniques similar to that used in many relational DBMSs to be applied to geoscientific queries [21]. The algebra contains a base set of general purpose logical data manipulation operators, while users are allowed to introduce operators corresponding to application-specific algorithms. It allows scientists to conveniently express their intentions by functionally combining complex scientific data manipulation operators within the Conquest algebra framework.

A wide variety of spatio-temporal geoscientific datasets (e.g., multi-dimensional arrays, time series, thematic maps) and traditional data types (e.g., relations) can be modeled in the Conquest data model by combining different coordinate and variable types and attaching to them special structural and semantic properties. Some of these structural and semantic properties provide important hints towards data representation, physical data organization, and query optimization and parallelization. At the same time, they are also used to model the numerous variations in representation (e.g., coordinate systems, geographic projections, georeferencing systems, regularity, unit of measure) in which data can be recorded. As a result, among the set of base operators in the algebra are a few that support the resolution of heterogeneity in data representation. These operators coerce "incompatible" data fields into formats acceptable by operators manipulating and correlating data fields. Some of the operators that fall into this category include:

- **Interpolation.** Geoscientific datasets are often sparse and irregular due to the placement and characteristics of sensors (e.g., the location of an in-situ sensor, the footprint of a satellite-based sensor). There are many ways the measured variable values can be distributed in the coordinate space; the distribution can be regular, rectilinear, or structured (logically a regular grid subjected to a non-linear transformation). The *sample* operator derives variable values at a user-specified set of cells in the coordinate space of a field with an interpolation function. By imposing a regular grid on a field, sampling (often referred to as gridding) presents a structured view of the data by deriving variable values at regular grid points through interpolation.

- **Conversion of coordinates.** The type and value of a field's cells can be changed by mapping each cell in the coverage into another cell (e.g., convert from sample number in an experiment to actual time, convert one map projection to another, or "move" cells relative to their current positions by translation or rotation, change the frame of georeferencing). The coordinate attribute conversion operator *coordinate_convert* allows the above operations to be expressed using a coordinate conversion function which maps each cell in a field's coverage to a different cell.

- **Variable value conversion.** The variable attribute conversion operator *variable_convert* derives new variable values for all cells in the coordinate space of a field by applying a user-defined variable conversion function to each cell record. One use of variable attribute conversion is to perform aggregation on related variable values, after they are collected by grouping operators.

3 Extensible Parallel Query Execution

Geoscientific datasets are extremely large and queries against them often involve the application of computationally expensive operations. Parallelization techniques are commonly used to remove bottlenecks in I/O and computation and improve query performance. Pipeline processing and partitioning are the two main dimensions in which a query execution plan can be parallelized.

Pipeline processing (or dataflow processing) supports "vertical" inter-operator parallelism in which 2 connecting operators in a query execution plans are assigned to different processors and which execution can overlap. Each operator consumes data arriving through a stream from its producer and feeds its output to an output stream until it blocks (e.g., when the stream buffer is full). Pipeline parallelism naturally supports stream query processing techniques which take advantage of data ordering to deliver excellent performance for many queries. A well-known class of such queries includes various parallel temporal join operations on multi-processor systems [10]. They provide good response time and minimize intermediate storage during parallel query execution. Most geoscientific data have a time component, and scientists are often interested in studying the variation of data in time. As a result, dataflow processing promises to be an attractive evaluation paradigm for scientific queries. The benefit of stream processing parallelism is especially obvious when a scientific query is coupled with a visualization routine which consumes results from the query. In a dataflow processing environment, query results are continually returned without having to wait for the completion of the evaluation of the whole query, reducing the response time and allowing visualization to overlap with query evaluation.

Intra-operator parallelization provides another dimension of parallelism. Partitioned parallelism provides opportunities for performance improvement by spreading I/O and computation across multiple processors or storage servers. It is achieved by dividing an input stream or dataset among a set of independent operators, each responsible for processing or retrieving a fragment of the data. Effective data partitioning is an important factor in parallelization of scientific query processing. Various data partitioning schemes (e.g., hash, range, and hybrid-range partitioning) have been developed to fragment relational data among multiple disks to reap the benefits of partitioned parallelism for different access patterns and to reduce the I/O bottleneck by allowing reading and writing of data in parallel.

3.1 The Conquest Extensible Parallel Query Execution System

The design of the Conquest Parallel Query Execution System is based on that of the Volcano extensible query execution system [8]. Conquest adopts Volcano's hybrid demand- and data-driven dataflow paradigm; it takes advantage of data-driven communication between processes to allow effective overlapping of computation while allowing operators to be invoked by consumer operators within the same process address space on demand to minimize the overhead of communication.

3.1.1 Data Exchange

In Volcano, the exchange operator is introduced to isolate interprocess communication involved in pipeline and partitioned parallelism from the implementation of data manipulation operators. The operator contributes to system extensibility since a user no longer has to worry about parallelization issues when implementing an operator. It allows a query execution plan to be easily parallelized through its insertion at various points in the execution plan. Process groups can be formed by surrounding a query execution plan fragment (in the form of a dataflow diagram) with the exchange operator. The processes within the process group combines to logically evaluate the query execution plan fragment by each executing the plan fragment on a portion of the input stream (see Figure 2). Conquest's *extended-exchange* operator extends the functionality of the exchange operator in several ways.

- **Heterogeneous supercomputing.** Conquest deals with hardware heterogeneity by implementing interprocess communication using the portable message passing library PVM [6]. PVM enables a network of heterogeneous Unix machines to behave as a single parallel computer by transparently resolving architecture dependence in interprocess communication. The Conquest parallel query execution server has been ported to run on the IBM SP1 and SP2, the Intel Paragon, and Sun workstation farms. Operators in a query execution plan can be executed on processors with different architectures. Applications in which different phases can take advantage of the special benefits offered by the different architectures, superlinear speedup has been reported despite latency and communication overheads [11].
- **Data marshalling.** Volcano's exchange operator assumes that data streams between operators are simple byte streams, with the interpretation left to the implementation of operators. However, complex geoscientific data structures have to be packaged before transfer to ensure correct transfer between platforms. Conquest packages complex data objects in a data stream in an architecture-neutral encoding based on their schemas (by taking advantage of PVM's support of XDR encoding). For a shared-memory environment, data transfer is accomplished through shared memory, obviating the need for packaging and architecture-neutral encoding.
- **Multicasting data streams.** Conquest's extended-exchange operator further extends the exchange operator by allowing a data stream to be multicasted to feed multiple consumer process groups (rather than just multiple copies of a execution plan fragments in processes within a process group as in exchange). This facility provides

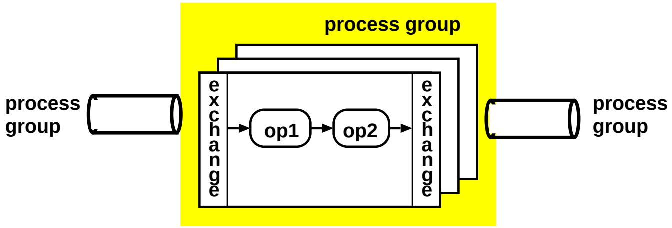

Figure 2: Parallelism in Volcano and Conquest

additional opportunity for I/O and computation to be amortized. In addition, the multicast operator explicitly controls data flow to avoid data being sent too fast from a producer operator to flood the system when the consumer operators fails to keep pace with the rate of data production.

3.1.2 Physical Data Model

Volcano is data model neutral in the sense that it does not define a data model for operators implemented in it. To take advantage of stream processing, a geoscientific data field is implemented as in Conquest as a data stream. Each data object in a data stream corresponding to a field is a cell record, which is the tuple formed by a cell in the field and its associated variable value.

Because Conquest uses the cell record as the unit of data passing between physical operators, it becomes possible to take advantage of the Conquest nest and unnest operators as a unique mechanism to control the granularity of data communication by taking advantage of the regularity in the coordinates of field types. Nesting a field causes related cell records in the original field to be grouped in a field (or a collection of cell records) in the result. For example, given a regular 3-D floating-point array measuring sea-level pressure on regularly spaced locations on the surface of the earth at regular time intervals, a cell record is a 4-tuple containing the spatio-temporal location and the floating point value recorded at the point. Nesting the spatial coordinate dimensions inside the time dimension causes the same array to be viewed as a time series of 2-D spatial array each storing the sea-level pressure values recorded on the earth surface at the corresponding time (see Figure 3). This allows cell records in the array to be implicitly referenced and hence significantly reduces the overhead required to explicitly represent the coordinate of each cell.

3.1.3 Operator Library and Interface to Heterogeneous Data Repositories

Conquest supports a library of physical operators, each of which implements some algebraic operator or performs some data manipulation task that changes the physical property of a data stream (e.g., sort). A subset of the operators provided in the GRASS library have been selectively incorporated into Conquest.

A large amount of scientific data are stored in various databases and file formats. It is important for a scientist to be able to access these diverse data sources through a uniform interface. Conquest supports access to a number of databases, file formats, and legacy data sources by means data repository specific access operators. Currently, we have implemented scan operators which interface to a number of vastly different data repositories including:

- Popular scientific data formats such as HDF [15] and netCDF containing multi-dimensional raster datasets.
- Extended relational DBMS Postgres which us used as both a storage and an external content-based index server.
- OASIS [13] CORBA distributed objects encapsulating OGIS-compliant geographic objects such as multi-dimensional cellular grid coverages (i.e., arrays) and feature buckets (which are collections of complex spatial data objects).
- A "live" scientific data generation application. For example, a UCLA climate model simulation running on an MPP can feed its output data stream into Conquest for it to be processed in "real-time" as they are being generated.

3.1.4 Dynamic Extensibility

Similar to Volcano, Conquest operators are encapsulated by a uniform iterator interface, making it straightforward to allow new operators to be seamlessly integrated into the system. However, extensibility in Volcano is static. Introducing a new operator requires the modification of the parser to recognize the name and parameters of the operator, and recompilation is required before the new operators can be used. While such "compile-time" extensibility satisfies the requirements for traditional database developers, scientists demand a "dynamically" extensible system due to the exploratory nature of their studies. Conquest supports extensibility by a system dictionary. A user introduces a new operator into Conquest by registering its specification with the dictionary. Query initialization and execution are driven by dictionary information; Conquest dynamically loads an operator from a Unix shared library on demand during query execution.

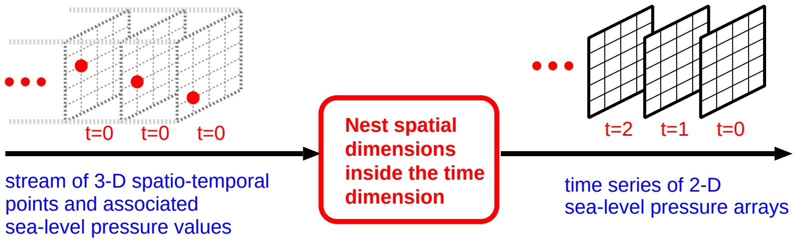

Figure 3: Using the Nesting of a Data Stream to Control Granularity of Data Communication

4 Query Optimization

Extensibility is an important requirement of a geoscientific information system. The implication of extensibility to query optimization is that the search space of parallel query execution plans for a query has to be extended as user-defined operators (either logical or physical) are introduced. Rule-based query optimization is the usual technique to construct an extensible query optimizer. Rewrite rules are repeatedly applied to a query expression to generate a set of computationally equivalent query execution plans (each of which is usually a dataflow diagram consisting only of operators defined in the execution environment). The rule base defines the search space of query execution plans for queries; the choice of rules is driven by the search strategy based on a combination of cost estimation and heuristics.

4.1 Related Work

Although a great deal of work has been done in optimization and parallelization of relational queries, relatively little has been done in studying issues specific to scientific databases. One notable exception can be found in [21] which describes a prototype scientific query optimizer constructed using the Volcano optimizer generator [9]. It focused on the optimization of computations over three bulk types: sets, time series, and spectra. Scientific operators considered include random sampling, digital filtering, interpolation, time series/spectra merging, spectra filtering, and FFT. An example optimization rule described is the algebraic equivalence of digital filtering (on time series) and spectral filtering when combined with a time series-to-spectrum conversion.

The above mentioned research attempted to demonstrate the applicability of traditional rule-based query optimization techniques to geoscientific applications. In particular, algebraic optimization at the logical level is specified by a set of *algebraic transformation rules* which define rules of algebraic equivalence between logical query expressions. In addition, a set of *implementation rules* dictates how logical query sub-expressions can be mapped into execution plans consisting of physical operators, by matching patterns of logical sub-expressions and rewriting them into sub-plans. Each operator is associated with a cost estimate function, which is used by the optimizer to determine the relative optimality of query execution plans. Enforcer operators may have to be introduced to resolve mismatches between the physical properties of the output of a query execution plan fragment and the expected properties of the consumer after implementation rules are applied. Example physical properties of a Conquest operator output include its schema as well as the sortedness and "compressedness" of data items in the stream.

From our experiences with complex geoscientific query processing, we realized that the benefit of algebraic transformation is often limited due to the application-specific nature of user-defined operators which define the semantics of the computation. Furthermore, many scientific operators are very sensitive to the accuracy and precision of its inputs; small round-off errors introduced at one point in a query execution plan may snowball as data flow up the stream through the plan and cause significant error in the result. It is unclear what the effects of query expression transformations are on the accuracy of query results even though the transformation may be valid algebraically with infinite precision.

4.2 Optimized Access to Heterogeneous Data Repositories

Because of the limitations of algebraic transformation as a means to optimize geoscientific queries, optimized access to remote and autonomous data repositories in a heterogeneous geoscientific information system becomes especially important. Special types of rules are introduced into a rule-based optimizer to handle the data distribution and take advantage of the fact that the execution of some operations are more efficient when executed in the remote data repository. Within the general rule-based optimization framework, decisions on how logical operations on logical data objects in a query can (and should) be pushed into data repositories are guided by the following set of rules.

4.2.1 Map Data Layout

A simple logical data field corresponds to a single data object in a database server or a file in a scientific data format. Because of the large size of scientific datasets, a logical object is often partitioned, with fragments stored in multiple data repositories or as separate entities in a repository.

Algebraic transformation rules define equivalence between between a pair of query expressions by specifying a mapping from a logical query sub-expression to an equivalent expression. A *layout mapping rule* is a special algebraic transformation rule which, based on information about data distribution, maps each reference of a logical field in a query expression to a union of logical scan operators that reads the physical fragments of the field. Depending on the capabilities of the underlying data repository, algebraic operators can be pushed into the logical scan operators. Each logical scan operator is later mapped into a physical scan operator encapsulating a specific data repository by an implementation rule and assigned to be executed at a specific processor.

4.2.2 Push Algebraic Operators into Data Repositories

In our execution environment, geoscientific datasets are managed by autonomous database servers or stored on file systems in some scientific data format. The capabilities of different data repositories vary greatly. For instance, a file-system supports basic support for storage and retrieval, the library for a scientific data format (e.g., HDF) supports simple operations like clipping), while GIS servers provide integrated support for more complex operations like FFT. Most database servers and scientific data format libraries efficiently support some data manipulation and filtering operations since the implementation of these operations are closely tied to the storage management. In addition, many problems do not fit the stream paradigm (e.g., many matrix operations such as transformations used in linear algebra, Laplace or fast Fourier Transform, and slab multi-dimensional subarray extraction), and fit better into the storage management subsystem rather than the query execution engine [7]. As a result, it is often advantageous to optimize extraction of data from external data sources by pushing operations and filters into the data source to take advantage of efficient processing and reduce the amount of data that needs to be extracted out of the data source. For a fragmented field which is composed of fragments stored in different data repositories, different operations may be pushed into the scan operators for different fragments.

Data has to be extracted from autonomous data repositories before they can be processed by Conquest operators. As described earlier, layout mapping rules map each reference to a logical object in the user query into a union of logical scan operators, each of which is mapped into a physical scan operator for a data repository. All scan operators accept a query parameter that is used to constrain the data to be retrieved from the underlying data repository. The syntax of the query is specific to the particular scan operator and the corresponding data repository. For example, the query parameter of a RDBMS scan operator may be an SQL statement, while a GIS scan operator may recognize some map algebra expression.

To take full advantage of the vastly different capabilities of data repositories, each logical scan operator is associated with a set of *operator ingestion rules* which guide how operators in a query expression can be pushed into the logical scan operator for a specific data repository. Given a query sub-expression consisting of an algebraic operator connected to a logical scan operator, an operator ingestion rule may rewrite it into one containing only the scan operator, but with the algebraic operator translated into a repository-specific query parameter for the scan operator. By pushing algebraic query sub-expressions into the query parameter of a logical scan operator (and hence the physical scan operator), an operator ingestion rule models and exploits the capabilities of external data servers. In addition, users wishing to optimize access to autonomous data repositories are relieved from having to know repository-specific query languages since the generation of query parameters for scan operators is driven by operator ingestion rules.

Some work has been done on estimating the cost of query execution in autonomous database systems [3]. However, the problem remains an open one, especially for non-traditional data servers supporting spatio-temporal geoscientific data. Similar to the widely accepted relational database optimization heuristics of pushing selection down the query execution plan, a useful heuristic to use in heterogeneous geoscientific query processing is to push operators into repositories whenever it is possible.

4.3 Example

Consider a query which counts the number of cyclone tracks observed in the UCLA AGCM dataset (recorded at $4° \times 5°$ resolution every 12 simulated hour from August 1981 to the end of 1989) passing through each grid point in each month from 1982 to 1983. This query is used by scientists to validate model output by visualizing the change in the cyclone presence concentration pattern over time, and compare the pattern with that generated from observation data. Figure 4 shows a possible dataflow representation of the query.

Let us assume that cyclone tracks are stored in a Postgres database in a class with the schema

```
c_UCLA_GCM_0 (
  track = txyzv[],  /* array of 5-D points*/
  track_len = int4, /* length of track    */
  min = txyzv,      /* min and max specify*/
  max = txyzv       /*   bounding box     */
)
```

where `txyzv` is an abstract data type for the time, latitude, longitude, altitude, and the sea-level pressure value recorded for each point on a cyclone track.

Figure 5 shows a possible query execution plan generated by the optimizer. The layout mapping for this query is trivial since the logical cyclone track collection referenced in the query maps directly into the Postgres class `c_UCLA_GCM_0`. At the same time, an algebraic transformation rule can be applied to the query expression to push the filtering of data by time (1982 to 1983) through the operator collecting cyclone presence statistics. Then an operator ingestion rule would push the operator filtering cyclone tracks based on their valid times into the scan operator for the Postgres class containing the cyclone tracks and generate the Postgres query

```
retrieve (track) from c_UCLA_GCM_0
```

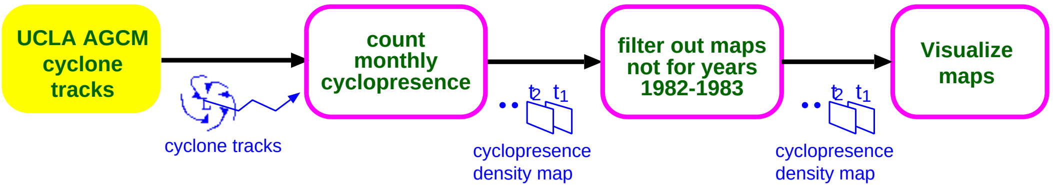

Figure 4: Cyclone-presence Visualization Query

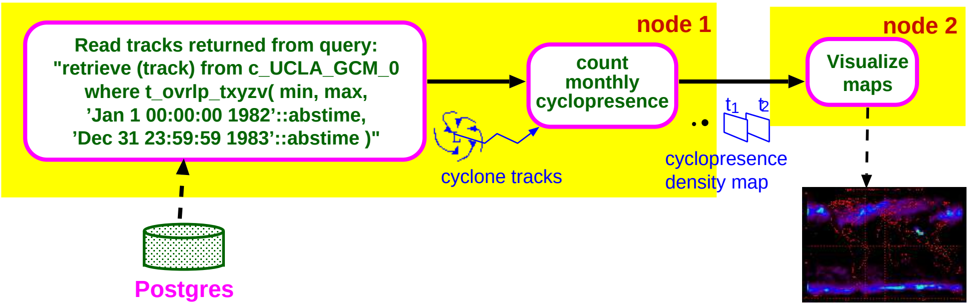

Figure 5: Query Execution Plan for Cyclone-presence Visualization Query with Operator Ingestion

```
where t_ovrlp_txyzv( min, max,
  'Jan 1 00:00:00 1982'::abstime,
  'Dec 31 23:59:59 1983'::abstime )
```

which retrieves all cyclone tracks in the Postgres class `c_UCLA_GCM_0` that overlap the years 1982 and 1983 as a parameter to the scan operator. Finally, each algebraic operator in the query expression is mapped into a Conquest operator that implements it.

Contrast the optimized query execution plan with a trivial one which is generated without using operator ingestion rules to push the filtering the cyclone tracks by time into the database storing the tracks (see Figure 6). The most significant difference between them is that in the trivial query execution plan, all cyclone tracks are retrieved from the track repository before those not covering the years 1982 to 1983 (which represent approximately 75% of all the tracks in the database) are filtered out in Conquest. On a Sun SPARC 20 (with the Postgres server running on a Sun SPARC 10), the execution times for the query execution plan with operator ingestion and the trivial plan are 32 seconds and 112 seconds respectively. The significantly shorter execution time for the first plan demonstrates the value and validity of operator ingestion as a geoscientific query optimization approach.

5 Conclusions and Future Work

In this paper, we described the features of the Conquest parallel query processing system that are relevant to geoscientific query processing in a heterogeneous environment. Conquest supports a data model designed to naturally model the logical structure of a wide variety of geoscientific datasets. The data model and algebra capture unique structural and semantic properties (e.g., regularity, ordering, access patterns) of geoscientific datasets, hence exposing opportunities for automatic query optimization and parallelization. Interfaces to heterogeneous data sources are supported in the form of scan operators, hence providing a means to process and correlate scientific data stored in autonomous data sources in our environment. Conquest has been ported to run on massively parallel processor supercomputers as well as workstation farms. The extensibility and built-in support for parallel computation make Conquest an attractive development environment for highly parallel geoscientific applications. Sequential applications can be easily parallelized taking advantage of the encapsulation of parallelization by parallelizing operators.

The current thrust of our research is on query optimization and parallelization. Some of the main issues that we are studying include automatic optimization of geoscientific queries in both MPP and heterogeneous supercomputing (mixed workstation and MPP) environments, systematic characterization and classification of user-defined scientific operators for parallelization, adaptive reconfiguration of query execution plan at run-time, and the application of Distributed Object Management technology such as CORBA in geoscientific query processing [13].

Extensibility, heterogeneity, and high performance are not characteristics unique for geoscientific database applications. Instead, they exist in a wide spectrum of advanced applications to which the approaches we develop in this research to address heterogeneous distributed geoscientific query processing should also be applicable.

References

[1] D.M. Butler and S. Bryson. Vector-bundle classes form powerful tool for scientific visualization. *Computers in Physics*, 6(6):576–584, Nov/Dec 1992.

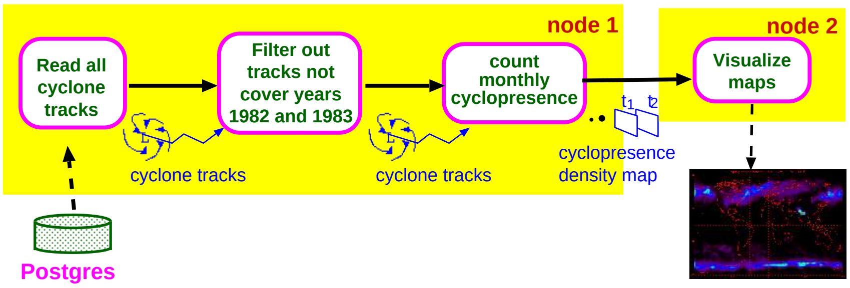

Figure 6: Trivial Query Execution Plan for Cyclone-presence Visualization Query

[2] D.J. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J. Yu. Client-server Paradise. In *Proc. 20th Int'l Conf. on VLDB*, pages 558–569, 1994.

[3] W. Du, R. Krishnamurthy, and M.C. Shan. Query optimization in heterogeneous DBMSs. In *Proc. 18th Int'l Conf. on VLDB*, pages 277–291, 1992.

[4] A. Elmagarmid and C. Pu (eds.). Special issue on heterogeneous distributed database systems. *ACM Computing Surveys*, 22(3), Sep 1990.

[5] J.C. French. Support for scientific database management. In Z. Michalewicz, editor, *Statistical and Scientific Databases*, chapter 4, pages 51–82. Ellis Horwood, New York, NY, 1991.

[6] G.A. Geist and V.S. Sunderam. Experiences with network-based concurrent computing on the PVM system. *Concurrency: Practice and Experience*, 4(4), Jun 1992.

[7] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2), Jun 1993.

[8] G. Graefe. Volcano – an extensible and parallel query evaluation system. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):120–135, Feb 1994.

[9] G. Graefe and W.J. McKenna. The Volcano optimizer generator: Extensibility and efficient search. In *Proc. 9th Int'l Conf. on Data Engineering*, pages 209–218, 1993.

[10] T. Y. C. Leung and R. R. Muntz. Temporal query processing and optimization in multiprocessor database machines. In *Proc. 18th Int'l Conf. on VLDB*, pages 383–394, 1992.

[11] C.R. Mechoso and J.D. Farrara an J.A. Spahr. Achieving superlinear speedup on a heterogeneous, distributed system. *IEEE Parallel and Distributed Technology: Systems and Applications*, 2(2):57–61, Summer 1994.

[12] C.R. Mechoso, C.C. Ma, J.D. Farrara, and J.A. Spahr. Simulations of interannual variability with a coupled atmosphere-ocean general circulation model. In *Proceedings of 5th Conference on Climate Variations*. American Meteorology Society, 1991.

[13] E. Mesrobian, R.R. Muntz, E.C. Shek, S. Nittel, M. LaRouche, and M. Kreiger. OASIS: An open architecture scientific information system. In *Proc. of Sixth International Workshop on Research Issues in Data Engineering: Interoperability of Nontraditional Database Systems*, 1996.

[14] E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.Y. Chien, C.R. Mechoso, J.D. Farrara, P. Stolorz, and H. Nakamura. Exploratory data mining and analysis using Conquest. In *Proc. of IEEE Pacific Rim Conference on Communications, Computers, Visualization, and Signal Processing*, 1995.

[15] National Center for Supercomputing Applications. *HDF User's Guide, Version 3.2*, 1993.

[16] The OGIS Project Participants. The Open Geodata Interoperability Specification. Technical report, OGIS, 1994.

[17] Research Systems, Inc. *IDL User's Guide (Version 3.1)*, 1993.

[18] SAIF. Spatial archive and interchange format: Formal definition (release 3.1). Technical report, Surveys and Resource Mapping Branch, Ministry of Environment, Lands and Parks,, British Columbia, Canada, Apr 1994.

[19] A. Shoshani. Properties of statistical and scientific databases. In Z. Michalewicz, editor, *Statistical and Scientific Databases*, chapter 1, pages 11–34. Ellis Horwood, New York, NY, 1991.

[20] M. Stonebraker and G. Kemnitz. The Postgres next generation database management system. *Communications of the ACM*, 34(1), Jan 1991.

[21] R. H. Wolniewicz and G. Graefe. Algebraic optimization of computations over scientific databases. In *Proc. 19th Int'l Conf. on VLDB*, 1993.