Optimization of Multiple Continuous Queries over Streaming Satellite Data

Quinn Hart
CalSpace
University of California at Davis
qjhart@ucdavis.edu

Michael Gertz
Dept. of Computer Science
University of California at Davis
gertz@cs.ucdavis.edu

ABSTRACT

Remotely sensed data, in particular satellite imagery, play many important roles in environmental applications and models. In particular applications that study (rapid) changes in the environment require frequent access to these data. For continuous data products, users are often interested in formulating continuous queries that deliver results for each incoming image. In the presence of multiple continuous queries, there is clearly an opportunity to share common intermediate data and thus, increase the overall processing speed of the entire system.

Based on the widely used Geographic Resources Analysis Support System (GRASS), this paper describes a system that realizes multiple query processing using two major components. A query optimizer maintains the current set of active continuous queries. Queries are organized into a single processing pipeline designed to share intermediate results. For each new image from the stream, the optimizer generates an execution plan that is specific to the active queries. The query executor then rewrites this plan into a set of geospatial processing steps and executes the plan.

We detail experiments using data from NOAA's Geostationary Operational Environmental Satellite (GOES). Continuous queries are defined in a way similar to the OGC Web Map Services (WMS) query specification. Using predicted query patterns over the visible hemisphere of GOES experimental results indicate that multi-query optimized plans can improve performance significantly when compared to queries that are executed separately.

Categories and Subject Descriptors

J.2 [**Computer Applications**]: Physical Sciences and Engineering—*Earth and atmospheric sciences*

General Terms

Geographic Information Systems (GIS)

Keywords

Query Optimization, Remotely Sensed Imagery, GRASS

ACM-GIS '06 Arlington, VA, USA

1. INTRODUCTION

There are many roles for remotely sensed data, in particular satellite imagery, in environmental applications and models [8]. Simple, convenient access to Remotely-Sensed Imagery (RSI), however, still is a barrier to effective research and applications. Access to this imagery still largely centers on choosing coarse grained, standard data products for specific regions and times. However, applications that study and monitor (rapid) changes in the environmental landscape require frequent, often continuous access to these data, and the temporal discontinuity in access methods can force complicated preprocessing and synchronization steps between the data provider and the data user. This is primarily a data processing artifact as the RSI itself is often acquired and transmitted continuously.

The *GeoStreams* project [7] is developing a data stream management system (DSMS) specifically designed to process diverse types of continuous, spatio-temporal queries over RSI. So far, the *GeoStreams* project has addressed models for query composition [3], spatial indexing schemes for continuous queries over streaming RSI data [6], visualization of real-time satellite data [14], and aggregation schemes for streaming raster image data [16]. Much of the concentration has been on processing data in real-time, consistent with the input stream from the GOES satellite [4], often processing incoming data one row of pixels at a time. With this approach, however, it is difficult to fully take advantage of existing geo-processing applications. This is because existing applications typically deal with complete images and also assume and take advantage of random access to the image data. The well-ordered arrangement of streaming images like GOES data allow natural divisions of the stream into separate images for different spectral channels and times. Using these divisions, the stream can be separated into files corresponding to images.

The concept of GIS image processing pipelines can be applied to streams of images with a number of advantages. First, it relates to common current practice and lends itself to simple implementations using existing applications. It results in processing methodologies that are strongly connected to current GIS practice, including models to develop pipelines within these systems. ESRI's model builder [11] is an example of such a system. It is also easier to consider retrospective queries in such a system, as the mechanism to process continuous queries is no different than retrospective queries against previously acquired data. Finally, delivery mechanisms to clients launching the queries can be simplified in this environment as well.

There are also disadvantages. Using traditional GIS applications implies the creation of many intermediate query data layers and results. This results in a large increase in the overall storage space requirement to process individual queries as intermediate images are created and used in query processing. Because these intermediate images rely on secondary storage, the system can be significantly slower than a complete in-memory stream processing system.

1.1 Objectives and Approach

The goal of this paper is to take a widely used GIS application environment (here GRASS) and investigate query processing techniques and savings by combining multiple continuous queries into a single execution plan, which realizes an image processing pipeline for streaming satellite imagery. Queries operate on data from NOAA's Geostationary Operational Environmental Satellite (GOES) and describe spectral image channels and derived products from these channels. The individual queries are relatively simple and limited in expressiveness, but are designed to satisfy the majority of practical RSI query needs, with a particular focus on spatio-temporal operations. They are defined in a way similar to the OGC Web Map Services (WMS) query specification [1].

The overall processing strategy is divided into two major components, the query optimizer and the executor. The *query optimizer* maintains the current set of active continuous queries. On the acquisition of a new frame of satellite images, the optimizer produces a dynamic execution plan that is specific to the active queries in the system. The queries are organized into a single processing pipeline designed to share intermediate results between the queries. This plan is a directed acyclic graph defining an order on the processing steps, and includes specific spatial parameters defined for each processing step.

The *query executor* rewrites a plan into a set of geospatial processing steps and executes those steps. Geographic Resources Analysis Support System (GRASS) is used to do the actual processing steps. The system performance is tested in a number of experimental designs. All experiments are based on query patterns meant to represent typical accesses to weather satellite data. The experiments are designed to test some key areas related to the optimization of multiple queries, including tests on spatial and temporal restrictions, derived image data products, and ensembles of queries.

1.2 Structure of the Paper

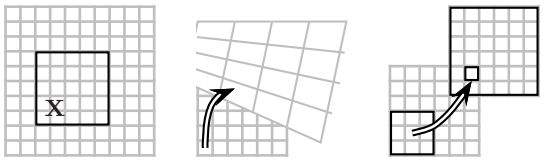
The remainder of the paper is structured as follows. Section 2 defines the concept of images and image queries. Section 3 describes the methods used to optimize multiple continuous queries. Section 4 presents a prototype implementation of the system and a number of experiments used to quantify the effectiveness of optimizing over multiple queries. Section 5 describes research related to this topic. Section 6 concludes the paper with general observations and future directions.

2. IMAGE QUERIES

Images can be compactly described using image algebra [13]. Image algebra is a many-valued algebra that includes *Point Sets*, *Value Sets* and *Images*. Point sets correspond to the spatio-temporal location of the values in a (raster) image. Since the location may include a temporal dimension, there is a common notation between continuous queries and image manipulation for a specific time. Point sets are denoted with bold capital letters and points within a point set are denoted with lower case bold letters, i.e., $\mathbf{y} \in \mathbf{X}$.

A value set $\mathbb{F}$ encompasses values associated with the points in a point set and is described by a homogeneous set of operands, typically a set like integers, $\mathbb{Z}$, or real numbers, $\mathbb{R}$. The notation $\mathbb{F}^{\mathbf{X}}$ describes the set of all functions $\{f \in \mathbb{F}^{\mathbf{X}} : f \text{ is a function from } \mathbf{X} \text{ to } \mathbb{F}\}$. An *image* is such a function. An equivalent, more familiar notation for an image $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$ is the *data structure representation*, $\mathbf{a} = \{(\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{x} \in \mathbf{X}\}$. Here the pair $(\mathbf{x}, \mathbf{a}(\mathbf{x}))$ is a *pixel* of the image. The first component $\mathbf{x} \in \mathbf{X}$ is the spatio-temporal *pixel location*, and the second component $\mathbf{a}(\mathbf{x}) \in \mathbb{F}$ is the *pixel value* at location $\mathbf{x}$.

Image operations are the building blocks for queries. These operations include functional operations, spatio-temporal image restrictions to specific point sets, spatial transforms on images from one point set to another, and neighborhood operations where multiple pixels are combined to a single value (see also [3]). Figure 1 gives pictorial examples these operations; the specific operator notations are discussed below and shown in Table 1.



(a) $\mathbf{a}|_{\mathbf{X}}$ (b) $\mathbf{a} \circ f$ (c) $\mathbf{a} \bigoplus N$

Figure 1: Image Operations: restriction (a), transform (b), neighborhood operation (c)

Operations on or among images include any operation that operates on the value set $\mathbb{F}$, which induces a natural operation on $\mathbb{F}$-valued images. For example, the addition of two images can be defined as $\mathbf{a} + \mathbf{b} = \{(x, a(x) + b(x)) : x \in \mathbf{X}\}$.

Image restrictions return images restricted to a given point set and thus realize spatio-temporal selections. If $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, then the restriction $\mathbf{a}|_{\mathbf{Z}}$ is defined as $\mathbf{a}|_{\mathbf{Z}} \equiv \mathbf{a} \cap \mathbf{Z} \times \mathbb{F} = \{(x, a(x)) : x \in \mathbf{Z}\}$. Restrictions are simply image subsets. They are possibly the most important of all operations in the context of query optimizations where restrictions on images can dramatically reduce access and storage costs.

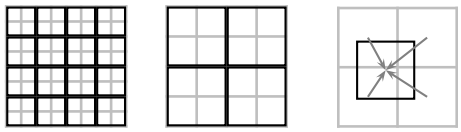
Spatial transformations map an image from one point set to another. In general, for any function $f : \mathbf{Y} \to \mathbf{X}$ between two point sets and an image $\mathbf{a} \in \mathbb{F}^{\mathbf{X}}$, the spatial transform is defined as $\mathbf{a} \circ f = \{(y, a(f(y))) : y \in \mathbf{Y}\}$.

The most common spatial transformation converts the satellite point set to the point set grids requested by the queries. Each query can specify pixel locations and resolutions that differ from the input images. Transformations then perform this conversion. As can be seen from the transformation definition, this requires that the image $\mathbf{a}$ be defined at arbitrary points, not just at the original satellite pixel locations. Bi-linear interpolation is used to provide these inter-pixel values.

Neighborhood operations allow for multiple pixels from an image to be combined to create a single pixel value in a new image. Neighborhoods allow for aggregation functions

like averaging, edge detection, speckle removal, and other operations. Neighborhood operations are also used for averaging satellite images into lower resolution pixels. Often users desire images at a coarser resolution than the raw satellite data. Since pixel values correspond to average radiance values over an area of the surface, averaging neighborhood pixels provides satellite data at other resolutions. In our model, coarser resolutions are created by progressively averaging images in 2×2 grids. For an individual query with its associated pixel resolution, the closest smaller average grid to the desired resolution is chosen and used in the interpolation.

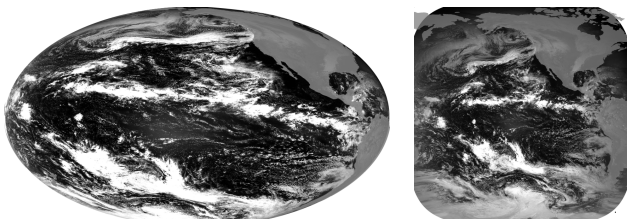
Figure 2 shows both neighborhood operations and interpolation techniques. Involved in the neighborhood operations is an implicit transformation to a point set that is one fourth the size of the original. The pixel locations are also moved to the new center of the averaged pixels. Although there are other methods of averaging pixels, this successive averaging is used as a simple and computationally effective method. Bi-linear interpolation extends an image to supply arbitrary pixel values based on a linear combination of the surrounding 4 pixels in the image.



(a) $(\mathbf{a} \bigoplus \mathbf{N_{2\times 2}})|_{/\mathbf{2}}$ (b) $\mathbf{a} \circ f$

Figure 2: Image averaging (a) and interpolation (b)

Spatial transformations are also used for projecting satellite data. Data as it is received from the GOES satellite is in it's own near-sided perspective projection. Many applications and users prefer data to be delivered in a more standard Earth coordinate system. Figure 3 shows a comparison of the GOES visible channel, in both it's native projection and projected to longitude-latitude (equidistant cylindrical).



(a) GOES-10 (b) Latitude Longitude

Figure 3: GOES satellite projection compared to an equidistant cylindrical projection.

Symbol	Operation	Description
$\mathbf{a}\|_{\mathbf{X}}$	Restriction	Restrict to new point set
$\frac{\mathbf{a}-\mathbf{b}}{\mathbf{a}+\mathbf{b}}$	Derived	Image channel ratio
$\mathbf{a} \bigoplus \mathbf{N_{2\times 2}}$	Neighborhood	Pixel averaging from 2×2 grid
$\mathbf{a} \circ LL$	Transform	Project to Longitude/Latitude

Table 1: Image Algebra operator notations

Queries are declared with image algebra expressions using a simple interface. Consider a query for a normalized difference ratio on two satellite bands, a common index for RSI. The query is to continuously receive this index for a particular region, projected to some convenient coordinate system, e.g., longitude/latitude. In image algebra this query is represented as

$$\left(\frac{\mathbf{a}-\mathbf{b}}{\mathbf{a}+\mathbf{b}} \circ LL\right)|_{\mathbf{X}} \tag{1}$$

where $((\mathbf{a}-\mathbf{b})/(\mathbf{a}+\mathbf{b}))$ represents the index, $\circ LL$ represents a function that maps from the satellite image to a new coordinate system, and $|_{\mathbf{X}}$ represents a spatial restriction to a new spatial extent.

The specific query declaration uses a formulation inspired by the WMS specification [1]. Users specify a specific data product, coordinate system, spatial extent and pixel size via the resultant image height and width. Although temporal restrictions can be identified too, they are not included in the experiments below. Queries like a ratio index can be specified as long as the index itself is identified as a specific data product. The specification further simplifies query formulation by standardizing and simplifying both spatial transforms and restrictions to a limited but well-defined subset. More formally, these queries can be identified as particular expressions in image algebra. In general, the WMS specification limits queries to the form $\mathbf{a} \bigoplus N \circ f|_X$, where $\mathbf{a}$, N, f, and X are specified in a simple standard way, as described above.

This simple "interface" does not allow for a sophisticated set of user queries, but it does investigate the most basic requirements of serving many spatial restrictions and geometric transformations to many clients.

3. QUERY OPTIMIZATION

The goal of query optimization is to minimize the processing (or response) time for all active queries in the system. For the types of queries described in the previous section, optimization is primarily concerned with two goals: (1) query rewrite rules to reduce the amount of work done for each individual query, and (2) exploiting common (intermediate) result sets among individual queries.

3.1 Single Query Optimization

Consider again query (1), $(\frac{\mathbf{a}-\mathbf{b}}{\mathbf{a}+\mathbf{b}} \circ LL)|_{\mathbf{X}})$. This is a natural way to represent the query, but does not directly translate into an efficient evaluation method. As written, the index and spatial transform are performed on the entire domain (point set) of the image, most of which is discarded in the final spatial restriction. Moving restrictions to the front of the query improves efficiency. Restrictions can be reordered over spatial transforms by transforming the restriction point sets as well. For example, given $\mathbf{Y} = \{LL(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$, the above query can be rewritten as

$$((\mathbf{a}-\mathbf{b})/(\mathbf{a}+\mathbf{b}))|_{\mathbf{Y}} \circ LL \quad \text{or} \quad ((\mathbf{a}|_{\mathbf{Y}}-\mathbf{b}|_{\mathbf{Y}})/(\mathbf{a}|_{\mathbf{Y}}+\mathbf{b}|_{\mathbf{Y}})) \circ LL \tag{2}$$

Simple heuristics on processing individual queries work well, especially in the case where queries are limited in complexity, as they are with the WMS query interface.

The optimizations used here are very straightforward and do not change from query to query. The following rules are used to optimize a single query: **(1)** The restrictions are

pushed up to the raw image channels. In the case where a different coordinate system is requested, the query describes the spatial extent in terms of that coordinate system. This means the extents need to be projected to a near equivalent GOES restriction as shown in Equation (2). **(2)** The input channel(s) are averaged to the resolution appropriate for the query. This is done with the progressive 2×2 averaging shown in Figure 2. **(3)** If the query is a derived data product, a processing node is added to perform that calculation. **(4)** The image is either projected or interpolated to the final coordinate system and point set as requested by the query. **(5)** As a final step, the data can be processed to an appropriate format suitable for delivery to clients.

3.2 Multiple Query Optimization

Once the individual queries are rewritten to optimize their individual execution, the queries are then optimized among each other. Optimization here centers around grouping similar query components into single operations that simultaneously satisfy a group of queries and operators. Execution plans in the multi-query environment simply use a single processing node for all queries that potentially share intermediate results. This includes all nodes related to resolution changes and derived operations.

Because queries are allowed to define arbitrary final point sets, the last interpolation or projection step cannot generally be shared among multiple queries. However, if queries were limited to specific point sets, for example, by limiting resolutions to integer numbers of hectares at standard postings, then an additional level of intermediate image sharing among queries would be possible.

The following rules are used to optimize all queries together. Assume there is a single execution plan that is being maintained for a set of queries. **(1)** The new query is rewritten based on the processing steps described above. **(2)** Starting from input image channels, the current execution plan is checked for the same processing step that is required by the new query. **(2a)** If the node does not exist, then it is added to the execution plan. In addition, a restriction to the new query's point set is applied before the processing node. **(2b)** If a similar processing node exists, then the restriction associated with this node is *expanded* to include the new query if necessary. **(3)** The process is repeated for the next operation in the individual query, starting from this new current node. This process continues until the final interpolation step, which is always added into the execution plan as it is not shared between multiple queries.

The optimization is shown with a simple example using four queries defined in Table 2. Figures 4, 5 and 6 progressively show the construction of the query plan.

Q	Product	Region	Projection	Resolution
Q1	C1	Mexico	GOES	~[1]km^2
Q2	C1	N. America	Lat/Long	~[4]km^2
Q3	$f(C1, C2)$	N. America	GOES	~[4]km^2
Q4	$f(C1, C2)$	Hemisphere	Lat/Long	~[8]km^2

Table 2: Example Queries

In Figure 4(a), the first query Q1 is for the Channel 1, GOES data over Mexico. The query resolution is approximately the same as the image stream, and no averaging is required. The query operations consist of a restriction on the Channel 1 image, and an interpolation to the query point set.

In Figure 4(b), query Q2 is added, also for Channel 1. The region of interest is North America, and here a coarser resolution is required. The Channel 1 image is reduced in resolution through two 2×2 averaging steps. The restriction on the original image is expanded to include both query regions, but the averaging is restricted to North America only. This query also projects the final product to a new coordinate system.

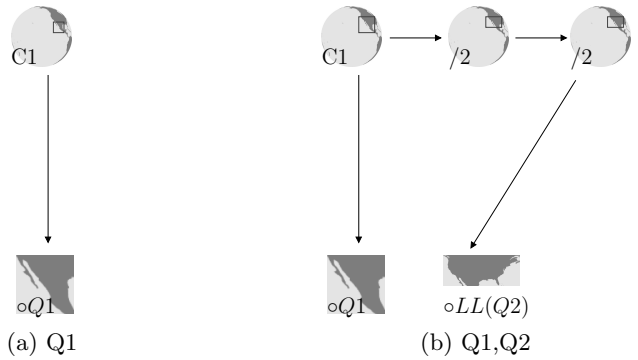


Figure 4: Execution plan for queries Q1 and Q2

Figure 5, shows another query, Q3, over North America being added, this one being a function of two input channels. The processing nodes for averaging the channel 1 image have their restrictions expanded to include this new region. In GOES data, different channels have different resolutions, and Channel 2 is already at the required resolution. The restriction on Channel 2 only needs to encompass this last query. A new processing node is added to perform the derived operation on the two images.

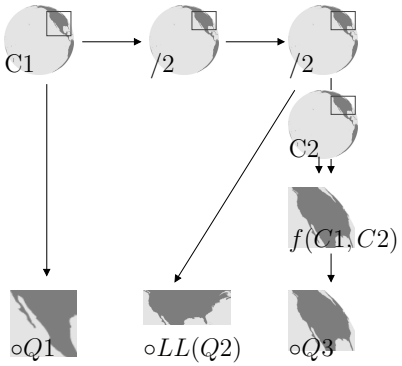


Figure 5: Execution plan for queries Q1, Q2, Q3

Figure 6 shows query Q4 requesting a nearly global coverage for the same image function as query Q3. This requires the restrictions for all previous nodes to be expanded for this final query. Since two queries are interested in the same function, these results are shared between the queries.

The final query processing pipeline combines all four queries using some data of the Channel 1 node; all but query Q1 use a common set of averaging functions as well. Queries Q3 and Q4 also share all of the nodes of Channel 2, as well as the output of the derived image function.

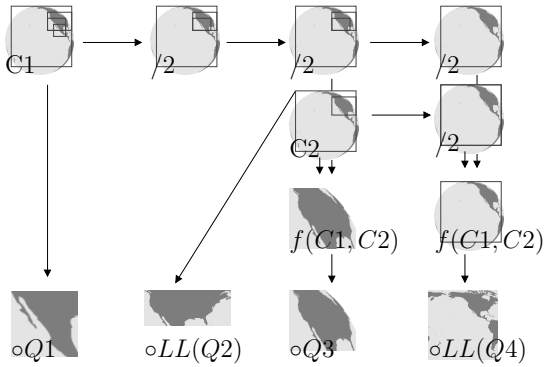


Figure 6: Execution plan queries Q1, Q2, Q3, Q4

This example also shows that the inclusion of large spatial extent queries, such as query Q4, can lead to large spatial extents on many intermediate nodes. While driving up the total processing time, it still does allow for sharing among many queries.

4. PROTOTYPE AND EXPERIMENTS

The multiple query processing framework described above has been implemented using a combination of the GRASS [5] Geographic Information Systems (GIS) application and supporting *nixtools. GRASS is a procedural GIS application that is especially well suited for raster image data manipulation. GRASS is developed as a series of stand-alone programs that can linked together for more complex operations.

GRASS includes the notion of an *active region*. This basically implies an implicit restriction for most operations in GRASS. Because GRASS is implemented as a group of *nixtools, the default processing environment is the shell, and standard *nixtools are available. In particular, this application uses a sophisticated `makefile` as the query plan executor. `Make` is designed to execute Directed Acylic Graphs (DAGs), and lends itself well to organizing the GRASS processes. The query optimizer is a separate program that organizes the multiple queries into a single processing pipeline.

The RSI data used in this study is the continuous weather imagery from GOES [4]. Data from GOES is received directly from a receiving station co-located with the computer running the query application. GOES offers a continuous stream of image data for regions ranging from the continental United States to a hemisphere centered near Hawaii.

The data stream transmits at approximately 2.1 Mb/sec. The GOES satellite continuously scans a hemisphere of the Earth with two sensors, the Imager and the Sounder. The instrument for this study, the Imager, has 5 spectral channels with resolutions ranging from about $1\,\text{km}^2$ to $32\,\text{km}^2$. GOES scans various sections of the Earth's surface about once every 15-30 minutes in specific frames. A single frame varies in size from about 100 MB to 440 MB, depending on the region scanned.

The framework described in Section 3 was tested against a similar system running queries that were individually optimized only. Between 5 and 100 queries were posed against the system and performance measured as the total processing time for each MB of data delivered to queries. Various combinations of restrictions and derived operations were performed on either single channels or all channels in the system. For simplicity, all queries were executed on the default GOES coordinate system.

Most experiments were run against a single set of GOES Imager data corresponding to a full disk input set taken at mid-day. Channel 1 was about 440 MB in size, with 10828 rows and 20792 columns in the input raster image. The other channels have coarser resolution and are about a quarter of that size.

Query Regions		Region Variations	
Global	GL	Area	0.8-1.2
Northern Hemisphere	NH	Aspect	±10%
United States	US	Center	±10%
Western Coastal	WC		
Mexico	MX		
South America	SA		
Random	RN		

Table 3: Query Region Parameters

The experiments were developed to replicate sets of queries made on GOES Imager input data and realistic regions of interest for the continuous queries. Rather than randomly locate these regions throughout the image domain, they were preferentially located around a small number of "hot spots" in the hemisphere. This more closely relates to real world scenarios where specific parts of the RSI data are requested by a large numbers of users. Although the general area of the queries was fixed to a small number of locations, the individual region centers, total sizes and aspect ratios were varied for each region. This corresponds to many users requesting slightly different particulars for a general region of interest. Some random regions were also included in the experimental setup. Table 4 describes the parameters for the query regions. The query region parameters table lists the various general regions used in the setup. Each region was given equal weight and the queries were randomly distributed among them. The region variation table shows the range of variation on the image size, location, and aspect with respect to the selected "hot spot".

Figure 4 shows the distribution of the input query regions over the hemisphere. Lighter areas on the map indicate more queries for that region. Areas along the western seaboard are most queried. For 100 queries, individual pixels participate in an average of about 28 queries, with participation in up to 66 queries for some pixels. The experiments were run using 5, 20, 50, and 100 individual queries.

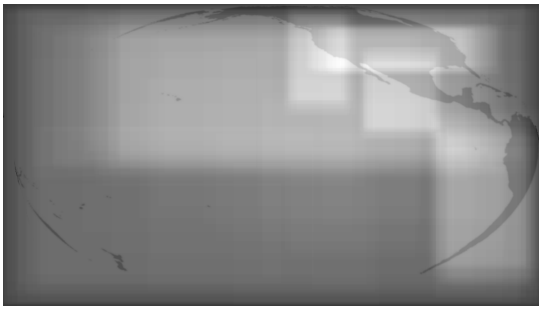
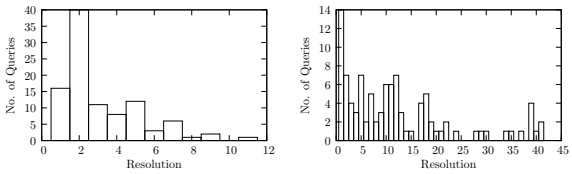


Figure 7: Query density

As the experiments show, the processing time is related also to the final resolution of images requested by the queries.

Two sets of query image resolutions were used in the experiments. In both cases, the query resolution ranged from resolutions on the scale of the finest satellite image—Channel 1, approximately 1 km resolution at nadir—to a resolution corresponding to a query image width of 256 pixels, a reasonable lower bound on resolution. The distribution of resolutions were varied. Both distributions weighted the finer resolutions more than the coarser ones, but with different scales. Figure 8 shows the histogram distributions of the two example query sets.



(a) Fine Resolution (b) Coarse Resolution

Figure 8: Query resolutions

The experiments were run on a quad Intel(R) Xeon(TM) CPU 2.80GHz processor with 4GB of memory and a 3.1 TB RAID 5 filesystem using 3ware 9500S-12 SATA-RAID.

4.1 Restrictions

The first tests investigated only the effects of sharing intermediate data for spatial restrictions. In this example, all queries were limited to simple restrictions on image channel 1. For the multi-query optimization plan, this includes sharing of intermediate data sets at the various coarser resolutions of the channel 1 data. Figures 9 and 10 show the processing time normalized to the total size of all query output. In both instances, sharing intermediate data sets decreases the overall processing time of the system. For the finer resolution case, as the number of queries increases, the mix of regions in the set becomes fairly uniform, and the average processing time becomes fairly constant.

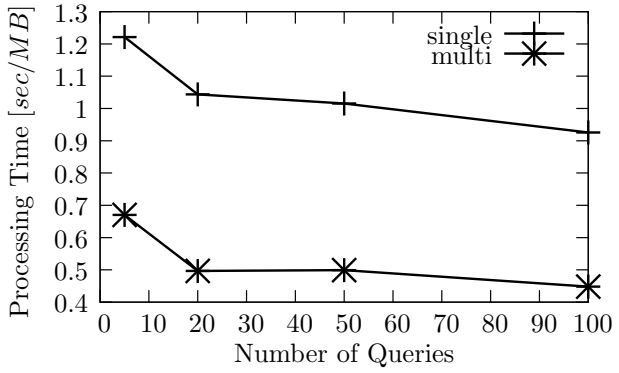


Figure 9: Restriction only, finer resolutions

The savings are more dramatic for the coarser scales, primarily because each individual query is more expensive as more averaging needs to be done. This is not true in the optimized case, as all the averaging is performed one time for all queries. It's unclear why the processing time for the non-optimized case did not stabilize to a more constant processing time in this example.

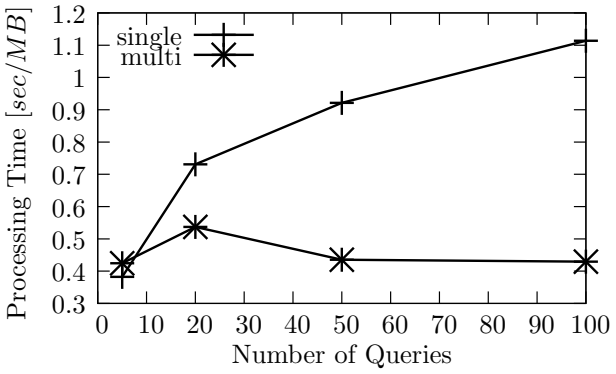


Figure 10: Restrictions only, coarser resolutions

Overall, there is about a two-fold decrease in the processing time using our proposed optimization method, compared to a single query optimization strategy.

4.2 Derived Products

If simple restriction queries show an improvement in processing time due to multi-query optimization, one would expect that derived products show even more improvement. This is because the multi-query optimization strategy saves on two levels, by avoiding image averaging on multiple channels for each query, and by sharing the derived products among all queries.

Two example derivative products were examined. The first was a normalized difference ratio, Equation (1), described in Section 3. This is a moderate computation. Figure 11 shows this experiment run on the finer resolution test. As expected, the multi-query optimization runs considerably faster than the single query optimization, by about a factor of 3.

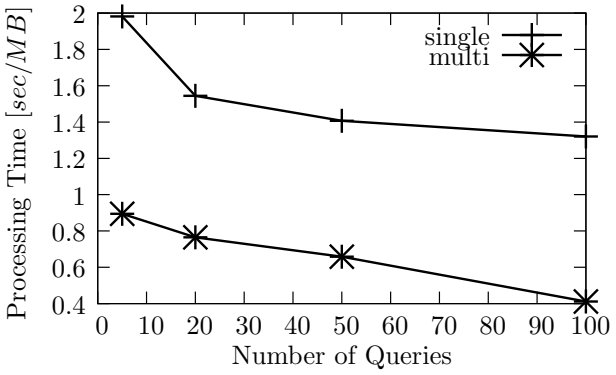


Figure 11: Image ratio, channels 1 and 2, finer resolutions

For a different type of derived product, *surface albedo* was calculated. Surface albedo is an important parameter for many applications. Albedo is calculated as the minimum pixel value over some preceding days, here up to the previous 14 days. This assumes that at some point in that time, every pixel in the image had at least one cloud free acquisition and that clouds are brighter than the surface. This is a simple calculation but expensive in that many raster images need to be accessed to generate the result. Figure 12

shows the comparison between the single and multi-query optimizations for albedo.

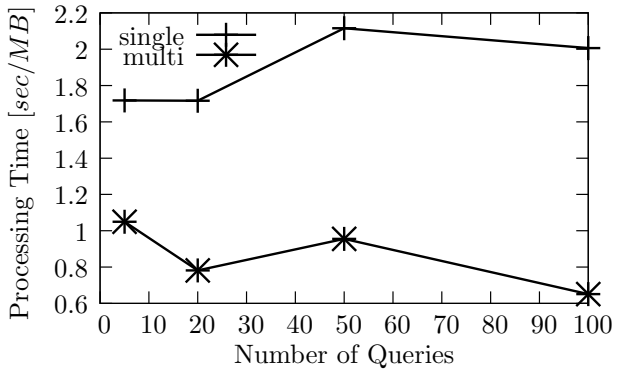


Figure 12: Surface albedo, finer resolutions

As with the other derived product, sharing intermediate results lead to an increased processing speed.

4.3 Multiple Data Products

The above examples are illustrative, but are not necessarily representative of a normal set of queries acting on such a system, because the above queries all request the same data product. A more common scenario would be where queries are spread among a larger number of possible data products. In this case, it would be expected that there is less benefit from the multiple query optimization, as less sharing goes on among the queries.

Figures 13 and 14 show the results of a set of queries that request 10 different data products. The same query regions were used, but the data products requested were uniformly distributed among the 5 Imager channels, and 5 other derived indices, all normalized ratios.

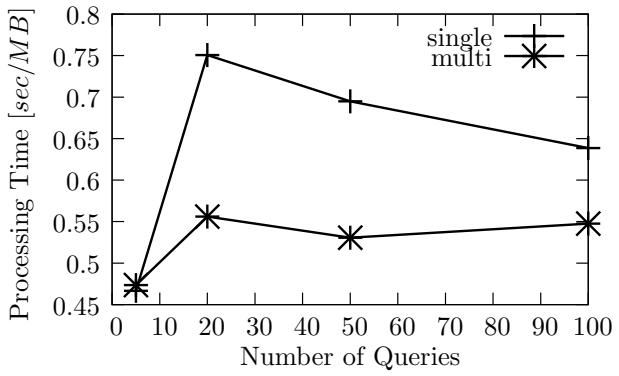


Figure 13: Queries on 10 data products, finer resolutions

In this experiment, the non-optimized solution performs much closer to the optimized version. Besides the fact that there is less chance for sharing among queries, another fact somewhat idiosyncratic to the GOES data is contributing. The non-visible channels of the GOES Imager have about a 4 times coarser resolution, compared to the visible. The previous experiments used the visible channel as it is the most popular channel, but this example spread most queries to the other channels, less averaging was required by each query. This further reduces the opportunities for queries to share intermediate data.

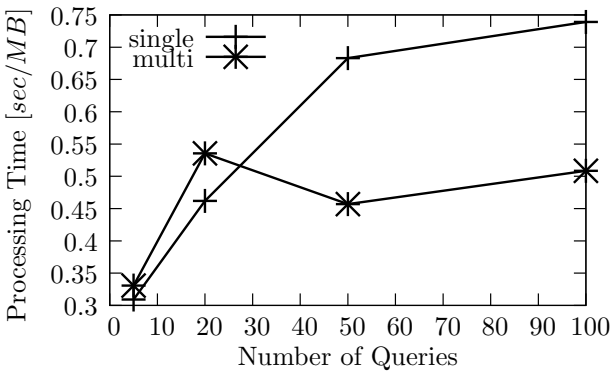


Figure 14: Queries on 10 data products, coarser resolutions

4.4 Reducing Secondary Storage Costs

Some of the tradeoffs between using existing GIS applications versus developing specific applications for streaming RSI data were discussed in Section 1. Chief among considerations was the added cost of secondary storage I/O. One method of minimizing I/O costs while allowing traditional applications like GRASS, would be to use a large RAM filesystem for processing time images. The current system does not include enough free RAM to perform the experiments described above although some preliminary experiments were undertaken. Specifically, a GRASS database of GOES data was copied to a RAM filesystem, and the 5 queries/multiple data product experiment were re-run on that filesystem, with all intermediate and result data products produced on the RAM filesystem. Comparisons of these times to the previously reported ones show only a moderate 10% decrease in the overall processing time for the RAM filesystem. While this is not necessarily an indicator of expected speeds for a specially crafted DSMS, it does indicate that other aspects of the processing besides I/O affect system performance.

5. RELATED WORK

Previous studies have recognized the critical role played by many data producers manipulating and making available RSI products to support environmental applications. One example, the Earth Science Workbench [2], describes a system for developing standardized processing pipelines on locally received satellite image data that is very similar in context to individual user queries here. Similarly, the Goddard Earth Sciences Data and Information Services Center is developing a service of virtual data products that reproduces the data from original sources dynamically based on the queries to the system [9].

None of the above systems looked at combining multiple queries and processing pipelines into more efficient query evaluation strategies. This is in part because the focus of these efforts is on traditional historical queries.

While multi-query optimizations have traditionally been based on finding commonality among queries (e.g., [15]), DSMS have concentrated efforts into systems with multiple continuous queries. In DSMS research, the notion of organiz-

ing multiple queries to speed up processing has been studied under multiple conceptual definitions, including grouped filters [10] and query indexing [12]. These approaches, however, concentrate exclusively on traditional (relational) data and operations.

All GIS related operations have a long history. The averaging feature of progressively rescaling imagery by halving the spatial resolution has been used in many applications, including quad trees, Haar wavelets, and image pyramids.

6. CONCLUSIONS AND FUTURE WORK

For large numbers of continuous queries against RSI, optimizations over multiple queries has been shown to be an effective method of increasing the overall system performance. How much savings can be expected depends primarily on the relationships of the queries in the system. The system described here uses a heuristic approach to developing processing pipelines for individual queries, and then combines these using a common processing template for all queries. Nodes are reused for multiple queries, with spatial restrictions modified to encompass all the spatial locations of interest to the continuous queries.

The implementation uses existing GRASS GIS applications to execute the processing steps. Remotely-sensed imagery clearly provides a great opportunity to study concepts and paradigms for the management and processing of streaming data, given the existence of various satellites that are used constantly for numerous data products. Even if new processing methods are developed to pipeline these images operations on a finer scale–for example processing rows of data at a time–the experiments presented here inform the developer on predicted levels of data sharing that will exist in such a system.

The ideas presented here are described in terms of continuous queries over RSI. However, opportunities exist in other applications. Another good candidate includes model outputs. For example, the Weather Research and Forecasting (WRF) model can be run in modes that periodically output weather predictions.

Examination of the times it takes in the current system to process only 100 relatively simple continuous queries, compared to the speed at which new spatial frames of data arrive from GOES, implies that the processing load will quickly become a problem. Future work requires investigating scale issues with load shedding, and multi-processor solutions. Load shedding in relational DSMS typically entails skipping the processing of certain input tuples in the data stream. Besides load shedding, multiple processors could be used for the query execution. However, the GRASS library was not developed to be thread safe or even safe for multiple processes to perform on the same image mapset.

As a simple example, since most of the processing of continuous queries involves processing on the most recently received image frame, one simple multi-processing scenario could simply use multiple processors and their associated local disk storage in a round-robin technique. That is, rather than processing every new frame from the GOES satellite, if their were n processors in a cluster, then each processor would only need to handle every nth frame from the GOES system. These aspects will be studied on a recently acquired multi-processor architecture.

Acknowledgments

This work is supported by the NSF grant IIS-0326517. The California Department of Water Resources provides partial support for operation of the GOES receiver.

7. REFERENCES

[1] J. de La Beaujardiere. Web Map Service. OpenGIS Implementation OGC 04-024, Open Geospatial Consortium Inc., Aug 2004.

[2] J. Frew and R. Bose. Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products. In *SSDBM 2001*, 180–189.

[3] M. Gertz, Q. Hart, C. Rueda, S. Singhal, and J. Zhang. A Data and Query Model for Streaming Geospatial Image Data. In *11th International Workshop on Foundations of Models and Languages for Data and Objects*, 2006.

[4] *GOES I-M DataBook, Revision 1*, 1996.

[5] GRASS Development Team. *Geographic Resources Analysis Support System (GRASS GIS) Software*, `grass.baylor.edu`.

[6] Q. Hart, M. Gertz, J. Zhang. Evaluation of a Dynamic Tree Structure for Indexing Query Regions on Streaming Geospatial Data. In *9th International Symposium on Spatial and Temporal Databases (SSTD05)*, LNCS 3633, Springer, 145-162, 2005.

[7] Q. Hart and M. Gertz. Querying Streaming Geospatial Image Data: The GeoStreams Project. In *SSDBM 2005*, 147–150, 2005.

[8] J.R. Jensen. Remote Sensing of the Environment: An Earth Resource Perspective. Prentice Hall, 2000.

[9] C. Lynnes and B. Vollmer. A Robust, Low-cost Virtual Archive for Science Data. In *SSDBM 2005*, 59–62, 2005.

[10] S. Madden, M. Shah, J. M. Hellerstein, and V. Raman. Continuously Adaptive Continuous Queries over Streams. In *ACM SIGMOD International Conference on Management of Data*, 49–60, 2002.

[11] J. McCoy. *ArcGIS9 Using ArcGIS Spatial Analyst.* ESRI, Redlands, CA, 2005.

[12] S. Prabhakar, Y. Xia, D. Kalashnikov, W. Aref, and S. Hambrusch. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. *IEEE Trans. on Computers*, 51(10):1124–1140, 2002.

[13] G. X. Ritter and J. N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra (2nd Ed)*, CRC Press, 2001.

[14] C. Rueda, M. Gertz, B. Ludäscher, B. Hamann. An Extensible Infrastructure for Processing Distributed Geospatial Data Streams. To appear in *18th International Conference on Scientific and Statistical Database Management*, 2006.

[15] T. K. Sellis. Multiple-query Optimization. In *ACM Trans. on Database Systems*, Vol.13(1), 23–52, 1988.

[16] J. Zhang, M. Gertz, and D. Aksoy. Spatio-Temporal Aggregates over Raster Image Data. In *12th ACM International Workshop on Geographic Information Systems (ACM-GIS)*, 39–46, 2004.