Brad Gellerstedt and Mark Harley

ArcSDE Data Preparation Techniques to Optimize Map Rendering for Time-Critical Applications

ABSTRACT

GIS software is commonly used to generate maps for interactive applications. Speed is often critical in such situations. For example, people browsing the web may lose interest and click elsewhere if they don't notice a prompt response. In certain situations it can be advantageous to pre-calculate and store selected information, rather than requiring the GIS software to generate it on-the-fly for each map draw. This paper discusses the evaluation and identification of candidates for data pre-calculation in an ArcSDE server context. Benchmarks are shown, contrasting system performance and describing the up-front processing required to achieve the demonstrated run-time performance gains.

OVERVIEW

When designing a GIS application, system response time is frequently an important factor. Often, maps are viewed interactively and the ability to display complete images quickly can be vital. This paper explores several methods for improving map drawing speed through the use of data preparation techniques.

At the core of this discussion is a classic trade-off, frequently encountered in the field of software engineering and database design. It involves a choice to calculate values in real-time, as they are needed, compared with pre-generating and storing certain information ahead of time.

Ideally, one would only process information as needed, using a master copy of the data as the sole source. Practically speaking, for performance reasons, it is sometimes advantageous to calculate certain data off-line. Spending a relatively large amount of time pre-processing can pay big dividends during an interactive session. The cost to do this includes time spent on evaluation and planning, running the data preprocessing, adjusting the application software to utilize the modified information and allocating storage space to hold the supplemental data.

ANALYSIS

The first step in considering the use of data preparation techniques is to decide where, and even if they would be beneficial. In many cases, performance turns out to be adequate, using the data directly, in an unaltered form. If early tests suggest that the performance will meet the system requirements, no further work is necessary. While speed can probably be improved through the use of data preparation techniques or other system tuning, time may be better spent on other activities.

Performance requirements for a new system should be stated during the design phase - for example, specifying that "a typical map must render in an average of five seconds, but take no longer than ten seconds, worst-case." Given a set of performance expectations, a prototype or working model of the data-intensive portions of a proposed GIS system should then be built. It ought to include both sample data and application code, constructed to simulate the expected operation of the final system. Common-sense analysis can assist in the design of appropriate tests. For example, if a particular data layer is quite dense, and it will be used extensively in the final system, it may become the focus of the initial analysis. If the system will serve a large number of simultaneous users, a prototype application may be built to simulate heavy activity near the peak anticipated system load.

If initial benchmarks show that drawing is too slow and the application is likely to fail to meet the system performance requirements, additional work should then proceed. The analysis may identify methods for utilizing data preparation techniques or performing other

system tuning to achieve the desired results. The remainder of this paper suggests some of the techniques which may be used to improve system performance through data preparation, which is also known as pre-processing, pre-formatting or non-real-time processing.

COMPUTING ENVIRONMENT USED FOR DATA EVALUATION

The benchmarking examples presented in this paper were performed using the following computing environment. The server hardware was an HP C5600, with dual 550mhz PA/RISC CPUs, and 2gb of RAM. Running on the server was ArcSDE 8.3 using Oracle 9i Release 2 as the relational database and an HP-UX 11.0 operating system. The client hardware was a Dell Latitude C600 with a 1ghz processor and 512mb of RAM. The client software was ArcInfo version 8.3 running on a Windows 2000 system.

The tests used a variety of datasets based on GDT's Dynamap/2000 products; data was loaded for the entire state of California for most tests. Testing was performed using ArcMap to render a map at a variety of different scales, usually with downtown Los Angeles at the center. Display times were determined using a simple timer control, developed in Visual Basic using ArcObjects. Tests were run a minimum of three times in succession to compute an average display time; in most instances, the times were extremely consistent.

DATA PREPARATION TECHNIQUES

The techniques described here have been used to achieve improved performance in certain specific ArcSDE system installations. Note that all methods are not suitable for every situation. It is important to measure performance, quantify the need, form a hypothesis, evaluate potential approaches and apply any of them where it is found to be appropriate.

Feature Grouping or "Multipart" Features

When a large number of features must frequently be rendered as part of a map image, feature grouping can be used to improve performance. The concept is to combine features to reduce the number of individual rows which must be retrieved from the relational database used by ArcSDE for data storage.

First, here is a quick review of how ArcSDE usually stores features. Consider a simple feature class consisting of line features with associated attributes. In the Oracle and SQL Server implementations, ArcSDE uses three primary tables to hold the feature class data. The first is a feature, (or "F") table. Each row in the feature table stores the geometry for a line. The second table is known as the "business table", where the associated attributes are stored. For example, if a line segment represents a street, the business table could hold fields with the name of the street, its length in miles, and the average traffic volume measured for each month of the year (or whatever information was present about the streets in that particular feature class). The third table is the spatial index, (or "S") table, containing information about the feature's location, instructing the application software on when it is necessary to retrieve the feature.

Figure 1 below shows the three tables containing one row each to represent MAIN ST as a simple line segment.

ArcSDE Database Tables

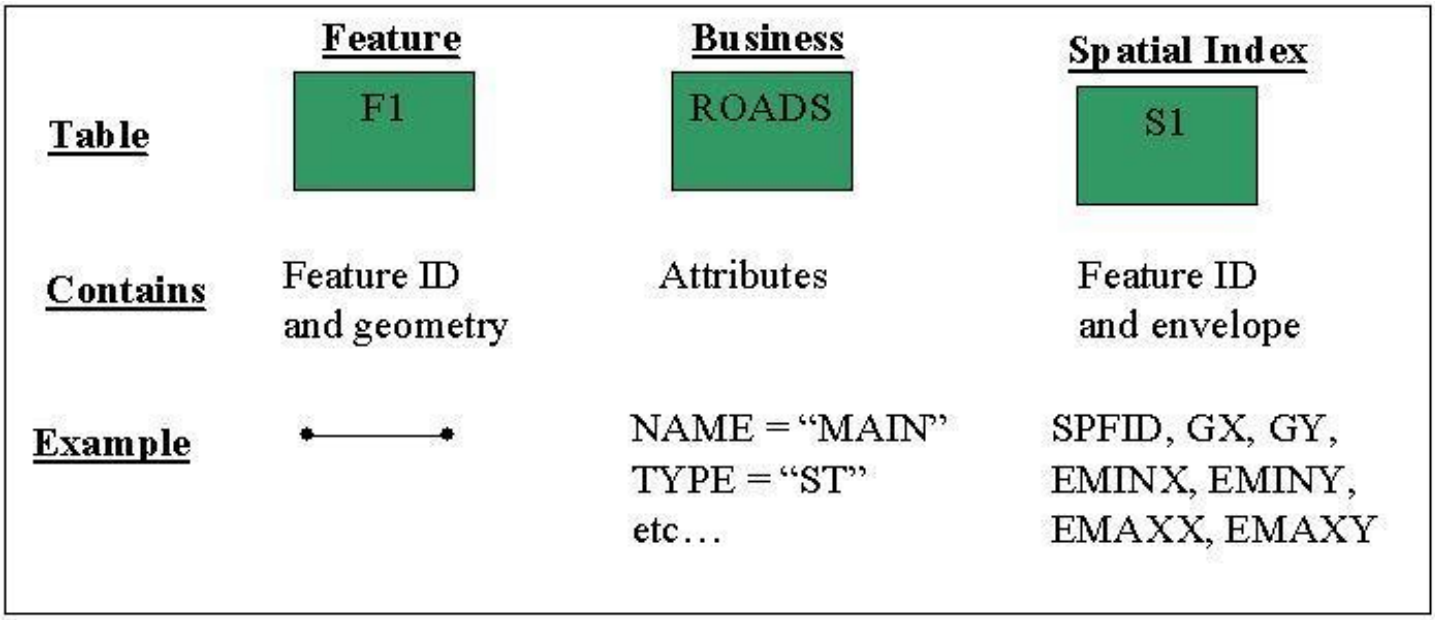

Figure 1

When drawing a map containing a large number of features, rows in each database table must be accessed for each feature. To draw the geometry only (that is, no attributes), the "S" table records must first be accessed to obtain a list of feature identifiers present in the window of interest. Then, the "F" table records must be retrieved for each and every feature in that window. In a dense map, this can represent accessing a very large number of rows in the database, sometimes resulting in slower than desired performance.

Processing the data ahead of time into grouped or "multipart" features can reduce database accesses and improve performance. The concept is to combine the geometry from many features into single, much larger features, which still contain all of the geometry of the individual features. Since there is only one business table entry describing this new "mega-feature", different individual attributes must be lost. Any attributes common to the entire combined set of features can be retained.

Figure 2 shows the same three tables defining a data layer in ArcSDE, but containing one row of multipart line segments. Note that the street geometry for a large area can be stored in one F-table entry.

ArcSDE Database Tables with Multipart Features

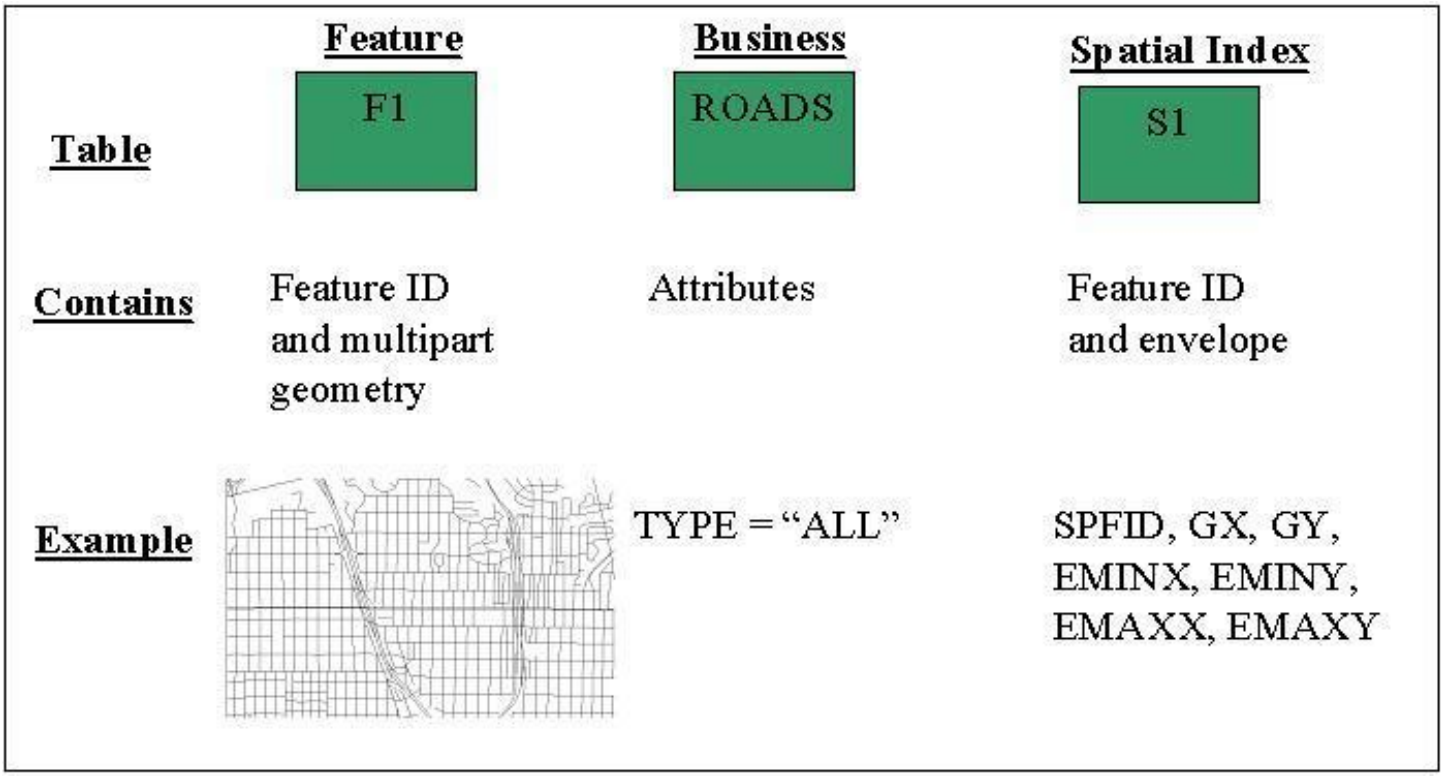

Figure 2

A detailed example may help to describe this data preparation method more clearly. Consider a street network for a large metropolitan area. If each side of each city block in the entire region is stored as a separate line segment, there may be tens or hundreds of thousands of lines used to represent the street network for the entire area. Drawing a map zoomed out to a fairly large scale can sometimes be slow as each individual line feature (or side of a city block) must be retrieved. To process the features into a "multipart feature" layer, groups of several thousand small street segments can be combined and stored as a single multipart feature. In other words, it is possible to store several hundred or even several thousand line segments as pieces of a single multipart feature, using one "F" table entry, with one corresponding row in the business table. When drawing large sections of the metropolitan area, instead of retrieving thousands of small features (by reading thousands of rows in the "F" table), a few multipart "F" table rows can be retrieved, containing the geometry for the entire map.

When combining streets, note that most of the individual attributes will be lost. If though, streets of the same type are grouped together, they can be drawn using appropriate symbology. In other words, if the freeways were grouped into one set of multipart features, major arteries into another and neighborhood roads into a third, each road type could be drawn using differing line widths and/or colors to indicate the type. Although the individual attributes (like "street name") are not present on the multipart feature class, the map image will look correct. Discreet attributes, like street names can still be made available to an application. If both the individual streets and multipart streets are loaded into separate ArcSDE feature classes, individual street segments with their attributes could still be queried when selected. Also, when drawing few enough features, due to being zoomed in to a close enough scale, the fully-attributed layer can be used for all drawing and querying. This would allow the application software to have access to road names, to label streets on the map, for example.

Figure 3 illustrates the results of tests run to measure the difference in draw times between a regular street layer and a multipart feature street layer. To perform the test, streets for the entire state of California were loaded into an ArcSDE 8.3 database in two formats, both as individual line features and the same geometry stored as multipart line features. The multipart feature layer was created with an application written in C using the SDE API; similar layers can also be created via ESRI's sdegroup command. Each map was centered on downtown Los Angeles, zooming out to the scales shown on the x-axis of the chart.

Note that when zoomed in fairly tightly, ArcSDE is able to retrieve individual line features at about the same rate as fetching the multipart line features representing the same area. As the map is zoomed out, however, the street layer draw time increases from a few seconds to several minutes, while the multipart street layer renders dramatically faster. The streets were grouped in blocks of 0.1 x 0.1 degrees.

A Comparison of Draw Times Between Streets and Multipart Feature (MPF) Streets

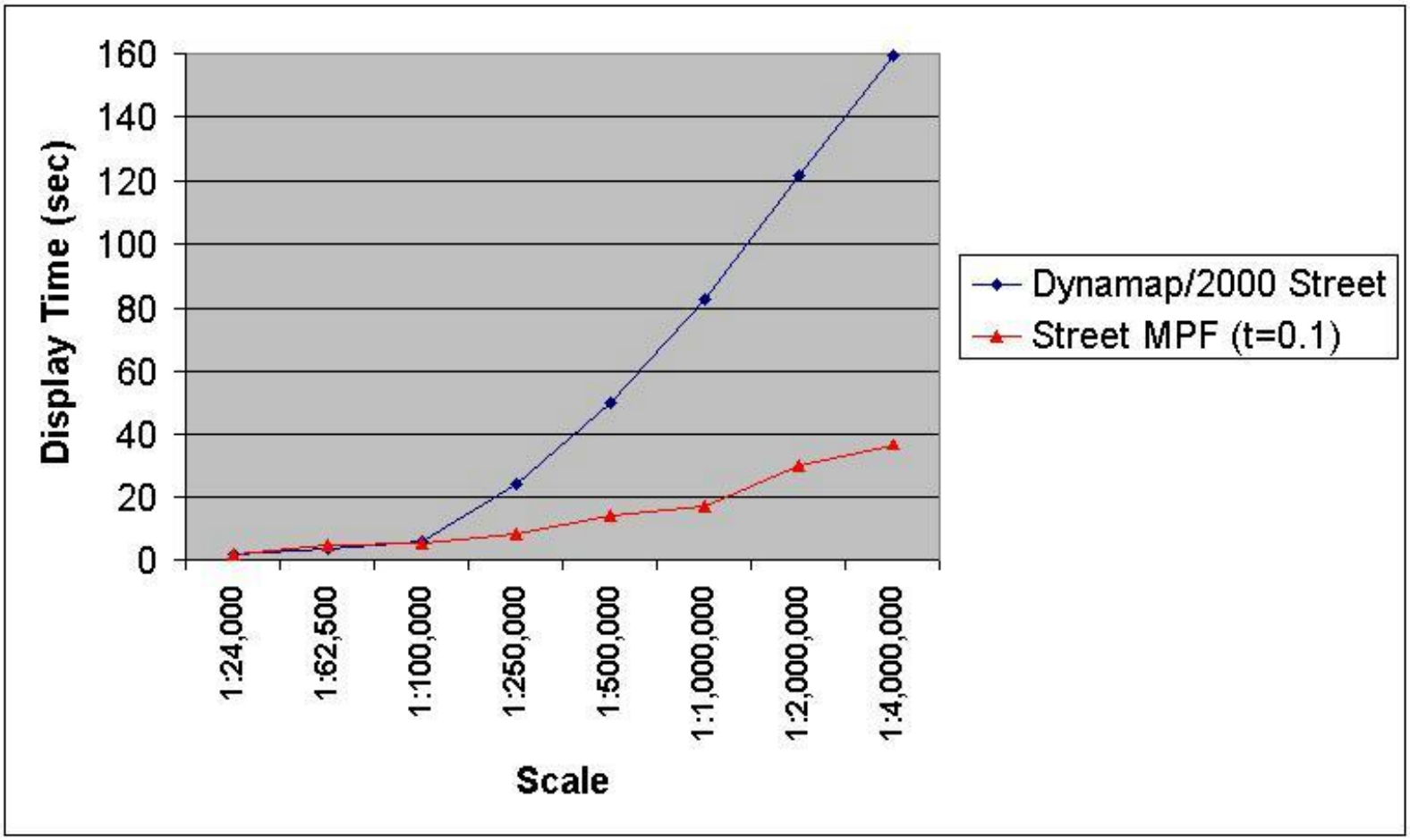

Figure 3

One thing to be aware of when creating multipart feature (or group) layers is the effect of the parameters used when creating them. For example, in the test described above, the multipart feature layer used was created using a “tile” size of 0.1 degrees. This means that when it was created, features were aggregated within areas of 0.1 x 0.1 degrees at a time. A similar multipart feature layer was also created using a tile size of 0.5 degrees. The same timing tests were performed on this layer. The results when compared with the 0.1 degree multipart feature layer are shown below in Figure 4. In this particular situation, for example, grouping with a tile size of 0.5 degrees yielded better display times when zoomed out to the farthest levels, with 0.1 degrees producing faster drawing at closer scales. The scales to be used most frequently by an application should be used to determine the optimal tile value.

A Comparison of Draw Times Between Multipart Feature Streets at Different Tile Sizes

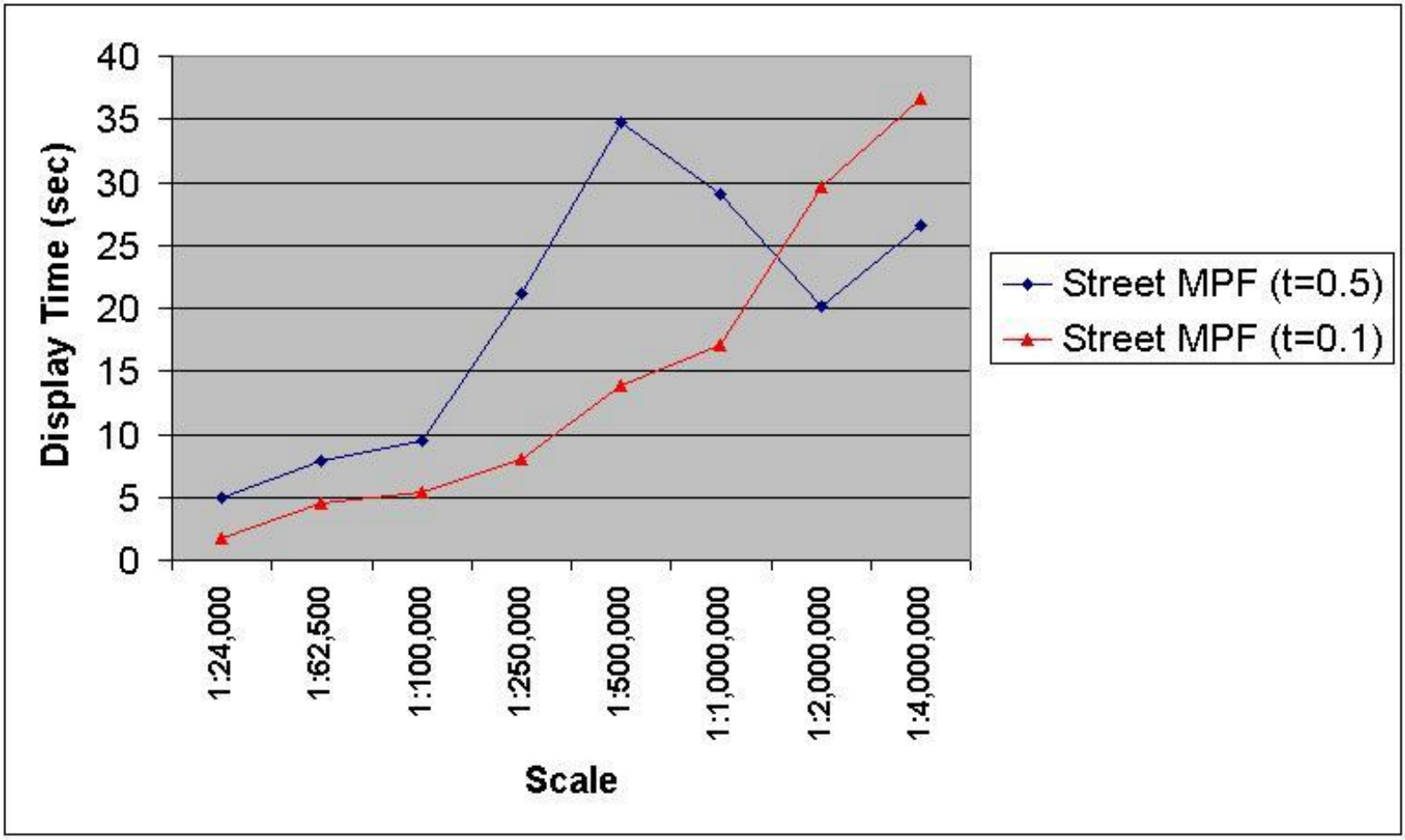

Figure 4

Pre-calculated Annotation

ArcMap has the ability to calculate annotation on-the-fly as a map image is rendered. This could mean drawing label text for one or more feature classes, in a way that uniquely identifies the features for the human reader, without obstructing other important information on the map. With certain applications, particularly when the display is dense and the map is zoomed out fairly far, the time required to generate annotation as it is needed can slow the response time to an unacceptable level.

This data preparation method involves writing software to generate annotation ahead of time and store it in an ArcSDE layer. It can be stored as a text string, including the font size and angle of rotation at which is must be displayed. To work effectively, each zoom scale at which the annotation will be displayed, and all of the visible layers to be enabled, must be determined ahead of time. This allows for advance calculation of label placement to select reasonable locations for the label text.

The primary advantage of this technique is the time saved in rendering maps containing a large amount of annotation. A disadvantage is that the annotation is pre-set for certain scales and combinations of visible layers only. Therefore, it can be difficult to use pre-calculated annotation in highly-interactive situations involving a lot of finely detailed zooming in and out or if the set of visible layers needs to be altered frequently. Also, it may be necessary to write some application software to retrieve, place and draw the pre-calculated labels, a fairly simple programming task. Pre-calculated labels can also be stored as an annotation feature class in a geodatabase. In some situations automatic label generation may produce text which is poorly sized or placed. Use of an annotation feature class allows human editing of the pre-calculated labels to produce a more desirable result.

Using a method similar to the pre-calculated annotation data, candidate locations for highway shields can be generated and stored in an ArcSDE feature class ahead of time. As with the annotation, the potential shield locations must be calculated for a particular set of displayed data, for use at a specific, small range of zoom scales. Application software is necessary to select one or more locations per highway and to render the shields on the finished map.

A test was performed using custom annotation point layers similar to those described in this section. These custom layers were created for two scales: 1:31,250 and 1:62,500. Each contains labels for several layers including streets, landmarks, water, etc. to be displayed on the map. These custom point layers were then rendered as blank points, with ArcMap's labeling engine displaying the text using the supplied angle. Drawing annotation from these layers proved to be significantly faster than allowing ArcMap's labeling engine to generate labels for each of the layers on-the-fly. The test results are presented in Figure 5 below. Similar performance improvements can be achieved by using annotation feature classes in a geodatabase.

A Comparison of Draw Times Between Labeling with Custom Point Annotation Layers and Labeling On-the-Fly

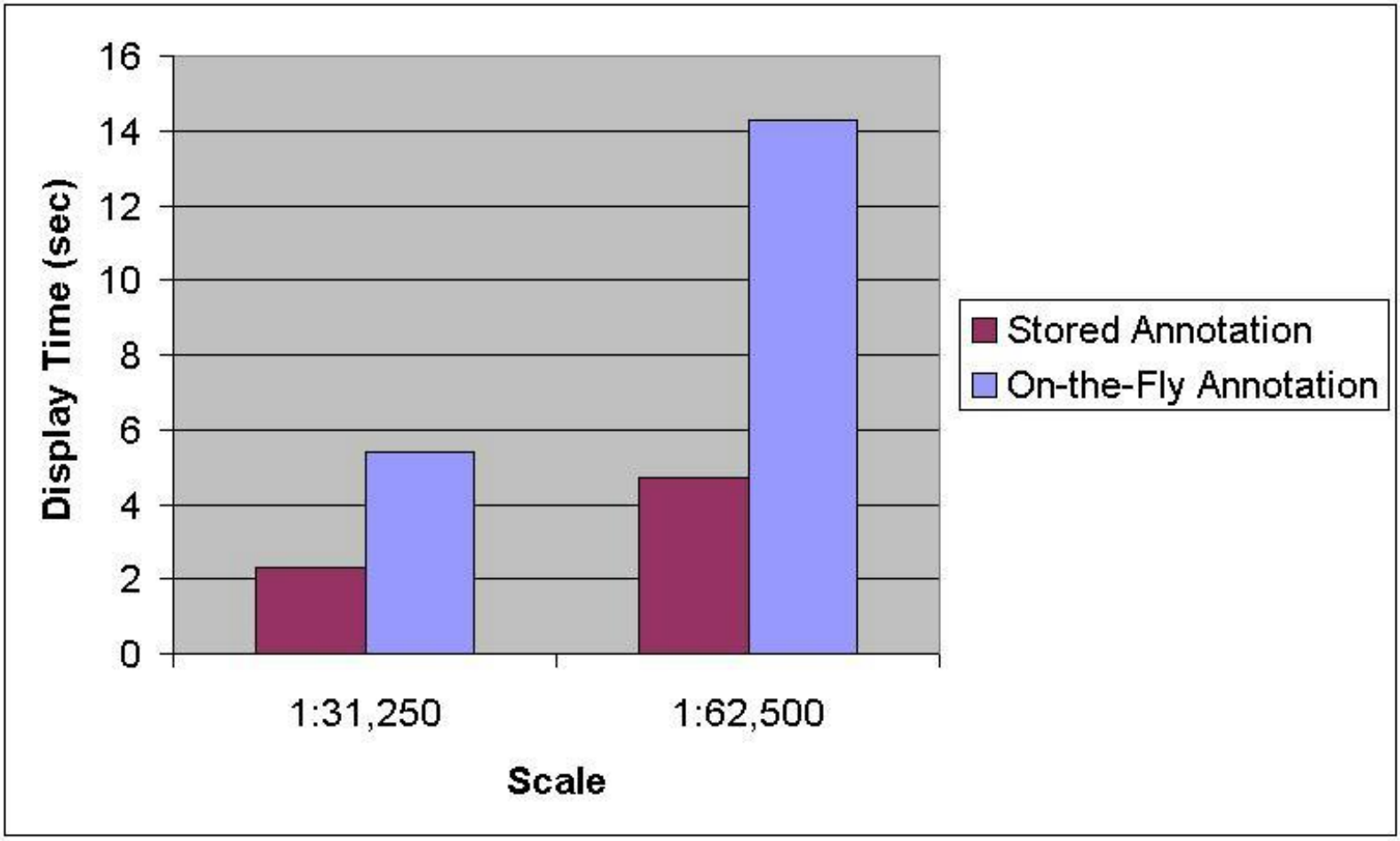

Figure 5

Data Generalization for Scale-Dependent Drawing

Scale-dependent drawing, that is, selection of appropriate layers to be included in maps based on the current zoom scale has been used with ESRI software for many years. The technique of pre-calculating generalized data is straightforward and definitely worth considering. This is most useful when dealing with data which must be displayed at a wide range of zoom scales and precisions. A good example to illustrate this concept is the drawing of state boundary polygons for the United States. Consider a situation where an end-user was interested in examining data along the state boundary near St. Louis, MO where the Mississippi River divides Missouri and Illinois. An application might want to initially display the entire USA when zoomed way out. This allows the user to locate the general area of Missouri and Illinois. After zooming in once or twice, the map might then be displaying eastern MO and western IL in the general St. Louis area. Zooming in further could result in the high level of detail desired, showing relatively few streets, parcels or buildings -- close to the river and including the state boundary.

When zoomed in to the local level, a highly-detailed, local state boundary data should be used to represent the state boundary. When drawing the initial "full USA" map though, the fine detail of the highly-accurate layer would be lost in rendering the big picture, and the data retrieval would likely be extremely slow. An appropriate data preparation technique would be to pre-calculate several generalized

state boundary layers and enable appropriately-detailed data at different scales. The Generalize tool in ArcMap uses the Douglas-Poiker algorithm to simplify the geometry of a selected feature. This method could be used to remove some of the points and create state boundary layers which contain an appropriate level of detail for the zoom scales further out. It would draw much more quickly than attempting to use the ungeneralized state boundary feature class to show a map of the full United States.

Figure 6, below, shows the difference in draw times between a generalized and ungeneralized set of state boundaries for the United States. The number of polygon features is not large, though some of the more detailed states contain several thousand points. Notice that when zoomed in to a fairly small area, the performance boost is relatively small. When drawing the full outline for all of the states though, the generalized map renders in a fraction of a second, where the detailed version takes nearly fifteen seconds. In an interactive situation, it is clearly preferable to use the generalized boundary when zoomed way out.

A Comparison of Draw Times Between a Generalized and Detailed Map Layer

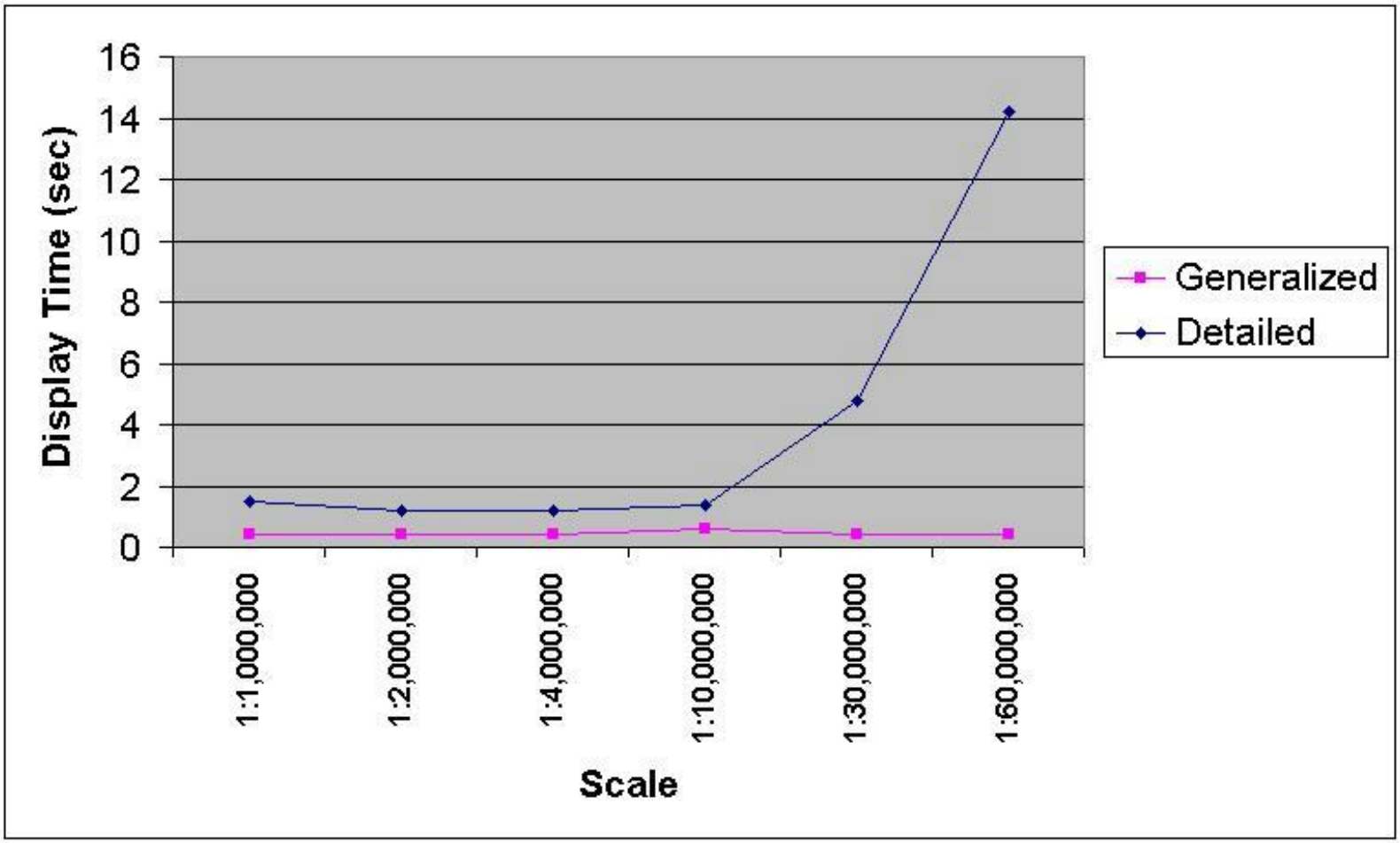

Figure 6

Re-projecting Data

ArcSDE allows data to be stored using different coordinate reference systems, or projections, in each feature class. ArcMap can be used to re-project each data layer on-the-fly to the target projection for the overall rendered map. In many situations, re-projecting on-the-fly will be sufficiently fast that conversion of the source data is not necessary. Sometimes though, this simple conversion technique can save quite a bit of map drawing time.

Often, most layers within a system will be in the projection normally used throughout a particular organization. ArcMap may only need to re-project a few layers which are not heavily used. In situations where a large database is assembled from source material in differing projections, it may be advantageous to convert each layer into a common projection. This data conversion, performed as a feature class is loaded into ArcSDE, can save time by paying for the re-projection up-front, rather than requiring a conversion as it is retrieved for drawing.

Figure 7 illustrates the draw times for both a feature class stored as projected data in UTM zone 11N, and a second layer stored in geographic projection, that is, raw, unprojected latitude and longitude coordinates which are converted to UTM zone 11N as they are drawn. Though the difference in this example is not dramatic, there is a performance gain resulting from performing the projection ahead of time.

A Comparison of Draw Times Between a Pre-Projected and a Run-Time Projected Map Layer

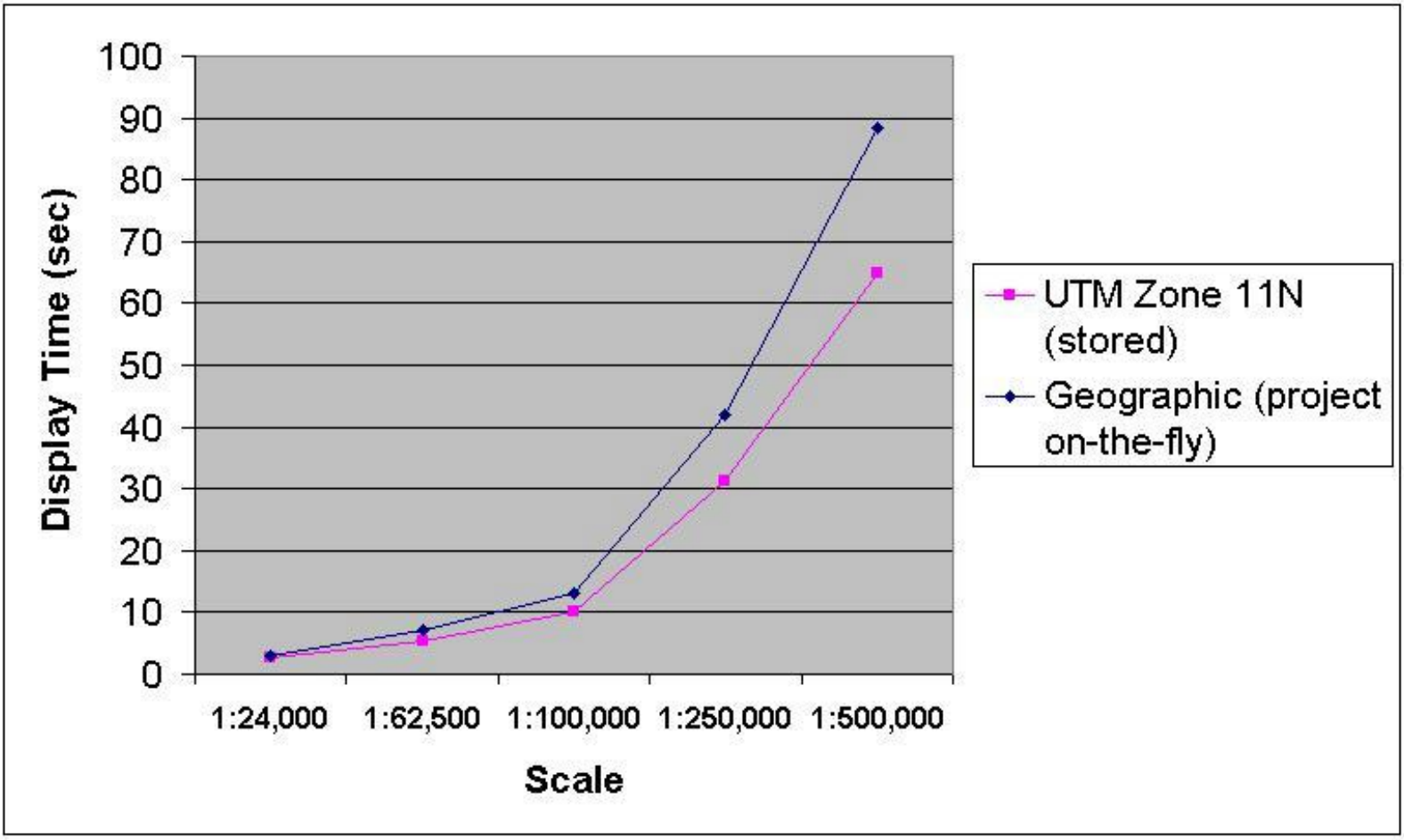

Figure 7

Database De-normalization

Database designers are trained to normalize data, that is, to move both repeating groups of items and independent fields to separate tables. The bottom line is to avoid storing multiple copies of the same information.

A database is often said to be in "proper relational form", if it follows a set of widely-accepted normalization rules, often referred to as First Normal Form (1NF), Second Normal Form (2NF) and Third Normal Form (3NF). Although it is cleaner to organize and maintain information in properly normalized databases, a side effect is that more table joins are likely to be required to satisfy certain queries.

In an SDE context, this means the Business Table may not contain all of the attributes associated with a feature. Joins to other related tables may be required to access a particular field. If required queries perform acceptably with all data normalized, by all means it makes sense to maintain the database in proper relational form.

If speed is a problem, selected de-normalization may help. For example, when storing ZIP code polygons, it would be proper normal form to maintain the ZIP code as an attribute directly on the business table and keep the (one or many) associated post office names in a separate table. If it was measured that the additional table joins required to select the primary name string for a given ZIP code was slowing down an important query significantly, the primary name string could be duplicated right in the business table.

CONCLUSIONS

When designing a GIS application, rapid system response time is often critical, particularly when the application is interactive. Data preparation techniques can be a good way to improve system performance. The specific speed increase realized from using any particular data preparation method will vary depending on the computing environment, number of simultaneous users, the precise nature of the data, and, perhaps most importantly, the way the data is typically accessed.

It is important to evaluate performance prior to investing time on data modifications. If the system performs well enough with the data in its normal format, up-front data preparation may not be warranted.

It is worth mentioning that additional elements should be considered when designing for ArcSDE system performance. System-level tuning of the interactions between the operating system, relational database, ArcSDE and the end-user application are critical. Proper server sizing, that is, adequate hardware selection, is also vital. For further information on the latter topic, an ESRI white paper found at http://www.esri.com/library/whitepapers/pdfs/sysdesig.pdf is highly recommended.

AUTHORS

Brad Gellerstedt, Senior Software Engineer
Mark Harley, Director, Consultative Services
Geographic Data Technology, Inc.
11 Lafayette St.
Lebanon, NH 03766-1445
Telephone: (603) 643-0330
Fax: (603) 653-0249
E-mail: brad_gellerstedt@gdt1.com, mark_harley@gdt1.com
Web: http://www.geographic.com