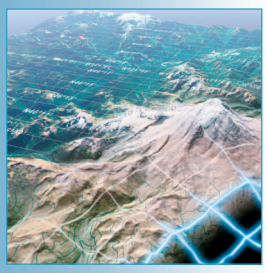

A Distributed Geotechnical Information Management and Exchange Architecture

Web services can help facilitate the exchange and utilization of geotechnical information. Such data is of critical interest to a growing number of municipal, state, and federal agencies as well as private enterprises. However, the lack of service infrastructures among heterogeneous data sources operating under different administrative organizations or agencies hampers the full use of geotechnical information. This article describes a Web-services-based way to manage geotechnical data via XML.

Roger Zimmermann, Wei-Shinn Ku, Haojun Wang, Amir Zand, and Jean-Pierre Bardet
University of Southern California

Geotechnical information is critical for many infrastructures. Universities, companies, and local, state, and federal agencies need information on soil deposits, for example, for a variety of civil engineering applications, including land usage and development and to map natural hazards such as soil liquefaction and earthquake ground motions. Geotechnical boreholes – vertical holes drilled in the ground to obtain soil and rock samples and determine stratigraphy, groundwater conditions, and soil properties – are the primary sources of geotechnical information.[1] Significant amounts of geotechnical borehole data are generated in the field from engineering projects each year. These data range from basic logs containing visual inspection reports of soil cuttings to sophisticated composites that combine visual inspection and in situ laboratory tests.

As data collection technologies improve, researchers produce (or convert) increasing amounts of geotechnical borehole data from the field and laboratory in a digital format. Moreover, with the recent ubiquity of communication networks – particularly the Internet – the trend toward electronic storage and exchange of geotechnical borehole data has accelerated. However, a significant constraint to sharing this information is that a multitude of private and public agencies, such as the US Geological Survey (USGS), the California Department of Transportation (CalTrans), and others, collect and manage it. Several pilot efforts currently under way aim to facilitate electronic access to geotechnical information: the Resolution of Site

 Published by the IEEE Computer Society 1089-7801/06/$20.00 © 2006 IEEE

Related Work in Distributed GIS

The Open Geospatial Consortium (OGC, www.opengeospatial.org) recently proposed the OGC Web Services (OWS) standard to facilitate the integration of geographic information systems (GISs) and location services. OWS is a general framework that enables distributed geoprocessing systems to communicate with each other through well-defined interfaces. One of the driving forces behind the current major revision of the framework — OWS-3 — is to provide interoperability between different implementations. Specifically, it targets wide-ranging applications from geospatial sensors to geospatial decision support.

Our framework's basic Web services use different functions than OWS does. Although OWS is an excellent framework, we chose a strategy based on the Web services standards from W3C. Although OWS and the W3C Web services standards evolved somewhat in parallel, they aren't currently compatible. OWS focuses specifically on geospatial applications, whereas the W3C's Web services — which use the Web Services Description Language (WSDL) to describe interfaces and SOAP for data transport — target more general applications. As a result, many of the popular development tools and technologies such as Eclipse, .NET, and Java NetBeans provide W3C Web service integration. The choices for OWS development are somewhat more limited. Moreover, because of W3C's wide support, our implementation gives us the flexibility to experiment with the newest techniques such as asynchronous data access.

Large bodies of work exist in several related subfields, such as spatial query processing, distributed query processing, and geospatial Web services, but a very limited subfield combines all these topics. Clement T. Yu and colleagues,[1] for example, surveyed various techniques for optimizing queries in distributed databases. They assumed a relational model and queries expressed in a QUEL-like tuple relational calculus. Sheng-Sheng Wang and colleagues[2] proposed a new qualitative spatial relation model and a solution to its consistency problem. Specifically, they described a new method to deduce the constraints of spatial queries, thus saving query-processing time in distributed GISs. Finally, Efrat Jager[3] presented an approach to compose various geospatial Web services through generic visual interfaces and scientific workflow tools. The framework encompassed registering, discovering, composing, and executing Web services to support distributed geospatial data processing.

On the commercial side, professional GIS Web service platforms, such as ESRI's ArcWeb Services (www.esri.com/arcwebservices) and Microsoft's MapPoint Web Service (www.microsoft.com/mappoint), provide solutions to GIS development problems. Our example implementation is leveraging open-source components, but proprietary platforms can be attached via middleware wrappers.

References

1. C.T. Yu and C.C. Chang, "Distributed Query Processing," *ACM Computing Survey*, vol. 16, no. 4, 1984, pp. 399–433.
2. S.-S. Wang and D.-Y. Liu, "Spatial Query Preprocessing in Distributed GIS," *Proc. Int'l Conf. Grid and Cooperative Computing*, LNCS 3251, Springer, 2004, pp. 737–744.
3. E. Jaeger et al., "A Scientific Workflow Approach to Distributed Geospatial Data Processing using Web Services," *Proc. Scientific and Statistical Database Management Conf.*, IEEE CS Press, 2005, pp. 87–90.

Response Issues from the Northridge Earthquake (ROSRINE) project, for example, has produced an integrated system based on a relational database management system (RDBMS), a geographic information system (GIS), and an Internet map server (IMS) to disseminate geotechnical data via the Internet.[2] The USGS is also publishing seismic cone penetration test (CPT) data through a Web-based system managed by the Earthquake Hazards Program.[3]

Our Geotechnical Information Management and Exchange (GIME) project's goal is to overcome the challenges inherent in data sharing among heterogeneous database repositories under different administrative control. Several features and goals guided our design:

- *Autonomy*. Each archive contains data maintained by a specific organization. For organizational rather than technical reasons, it isn't desirable to replicate or cache data sets at other participating archives.
- *Standardized access*. It's desirable to allow direct, programmatic access to distributed data sets from user applications. To hide the data sources' heterogeneity, Web services provide a standardized interface. They build on the idea of accessing resources from a local machine on a powerful remote computer.
- *Cooperative and efficient query processing*. When presented with a query at a participating database node, the overall system must cooperatively execute the request and return all relevant data. For this to happen, an efficient access method rapidly decides which nodes contain potentially relevant data and which don't.

Here, we describe the distributed GIME infrastructure's design and implementation.

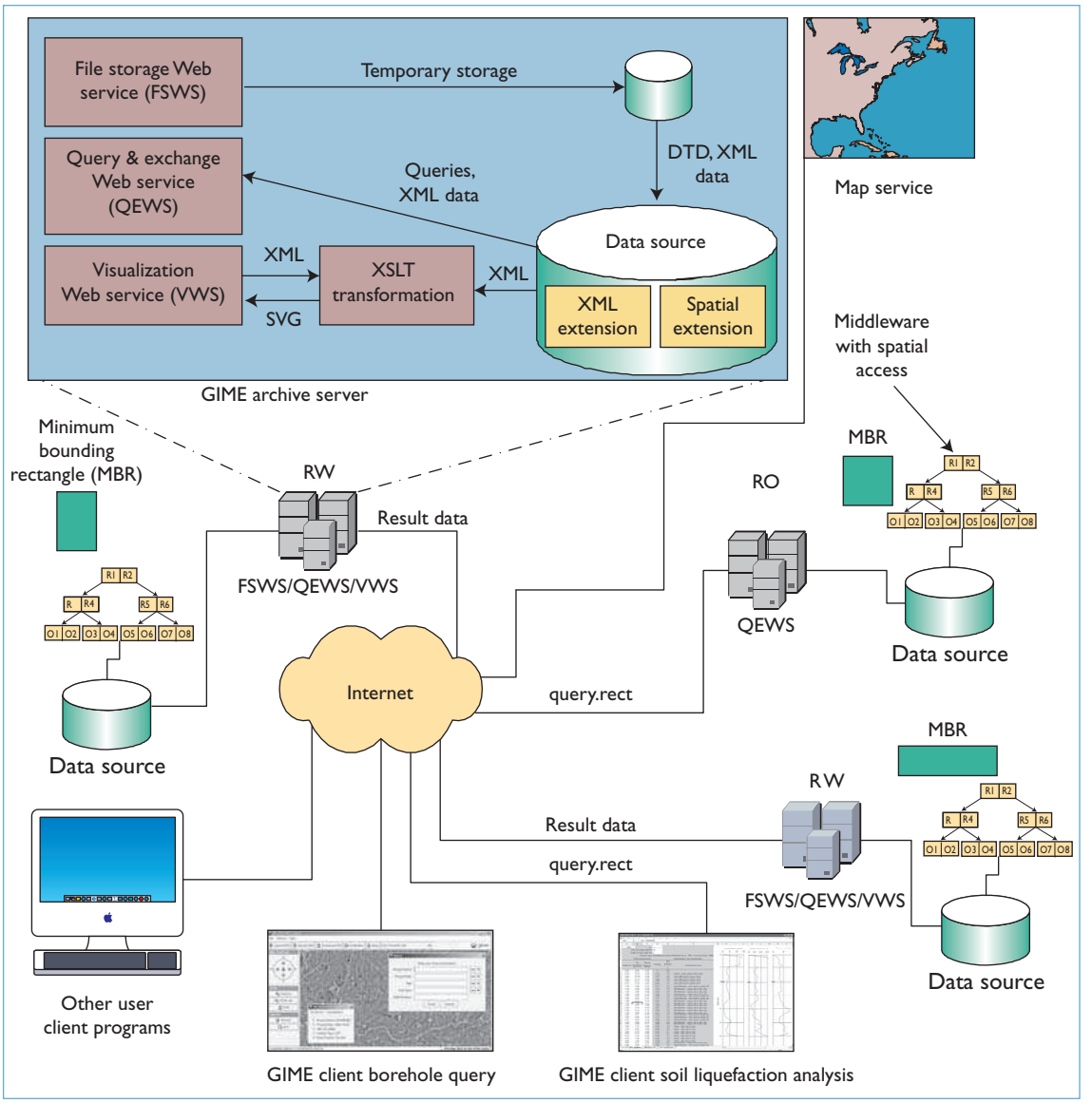

Figure 1. Geotechnical Information Management and Exchange (GIME) architecture. Some of the distributed archives are read-only (RO), whereas others allow read-write (RW) access. Each archive contains middleware that uses replicated spatial index structures.

GIME

Figure 1 illustrates the GIME system's architecture. GIME users can access multiple, distributed geotechnical data archives via the Internet, and GIME provides services such that practitioners in the field can directly store newly acquired data in a repository while users uniformly access these data sets. By adopting a Web services infrastructure, GIME users can build multiple applications, such as a soil liquefaction analysis or a borehole visualization, in a modular fashion.

GIME distinguishes two types of archives on the basis of the data access allowed: read-write (RW) or read-only (RO). RW archives host three geotechnical Web services to provide the interface for distributed applications to store (file storage Web service; FSWS), query and retrieve (query and exchange Web service; QEWS), and visualize (visualization Web service; VWS) geotechnical information. RO archives implement only the QEWS Web service. In this case, an on-site database administrator can insert the data directly into the local database. The upper left corner of Figure 1 illustrates an RW archive's components.

Finding the relevant data sets required for a specific application among all the geotechnical

archives can be a daunting task. To conveniently process the spatial queries and locate the relevant information, we designed an efficient query-routing algorithm for GIME that automatically forwards queries to other known archives and collects the results before returning the data to the application. Such forwarding mechanisms are quite effective, as demonstrated by the SkyQuery project[4] and our own distributed query-routing techniques.[5]

Geotechnical borehole data is complex and sophisticated in that it contains both well-structured and semi-structured elements; thus, we need an efficient data format for storage and exchange to handle the data's diversity. In GIME, we use XML as the preferred container format for both functions.[6] XML offers many advantages over other data formats,[7] and its tree structure and flexible syntax are ideally suited for describing constantly evolving and annotated information. It also lends itself to an automated visualization capability that converts XML geotechnical data into a graphical view similar to the traditional hardcopy format. The output can be presented in scalable vector graphics (SVG; www.w3.org/Graphics/SVG/). To process a user-uploaded data file, GIME first places the file in a temporary space in which the file validator can validate it against the Document Type Definition (DTD) or XML Schema for geotechnical data sets. If the file is accepted, GIME then stores the data in the main database.

Geotechnical Web Services Functionalities

Web services commonly operate from a combination of a Web and an application server; they can be implemented via many existing tools. Our local GIME prototype testbed uses the open-source software components Apache Tomcat (Web server) and Apache Axis (application server). The application code specific to GIME is embedded within Axis, which proves convenient: when Tomcat starts, Axis automatically compiles the application codes located in its working directory and generates the necessary object class files.

The three main geotechnical Web services provide several specialized methods for programmatic access to geotechnical data. The FSWS provides client programs with an interface to upload their XML data files into the main database. During the upload process, GIME extracts and stores the metadata, including the specific elements that facilitate querying. The QEWS's main purpose is to facilitate the dissemination of valuable geotechnical data sets and encourage their use in broad and novel applications. XML borehole files, although easy for computers to read, become meaningful to geologists and civil engineers only after they're rendered into images, so the VWS's main purpose is to generate SVG files.

GIME also has several API methods:

- `GeoPutDTD()`: uploads and stores a new DTD file on the server (FSWS);
- `GeoGetDTD()`: retrieves the current DTD file (FSWS);
- `GeoPutXMLFile()`: uploads an XML borehole data file and stores it on the server (FSWS);
- `GeoQuery()`: executes a query expression and returns a list of unique identification numbers, one for each borehole file in the result set (QEWS);
- `GetXMLFile()`: retrieves an XML borehole data file based on a unique identification number (QEWS);
- `GeoVisualization()`: transforms the XML borehole file selected with the unique identification number into SVG format on the server (VWS); and
- `GeoGetSVGFile()`: retrieves an SVG borehole file based on a unique identification number (VWS).

These methods allow uniform access to the GIME functionality while hiding the heterogeneous resources.

Efficient Query Routing with Spatial Indexing

Given a federation of independently managed spatial database servers, one research challenge is the efficient querying of the distributed infrastructure. Complicating matters, the data set in each repository could be disjoint or overlap with other archives. To keep an application from contacting each and every repository, we implemented a distributed query mechanism that automatically forwards queries to other known archives and collects the results before returning the data to the application. Repository data sets are spatially indexed at a middleware[5] layer via replicated R-trees or Quadtrees.

Exhaustive query routing. Geotechnical data sets are generally large and valuable and thus professionally managed. We consider this a stable environment in which occasionally, but not very

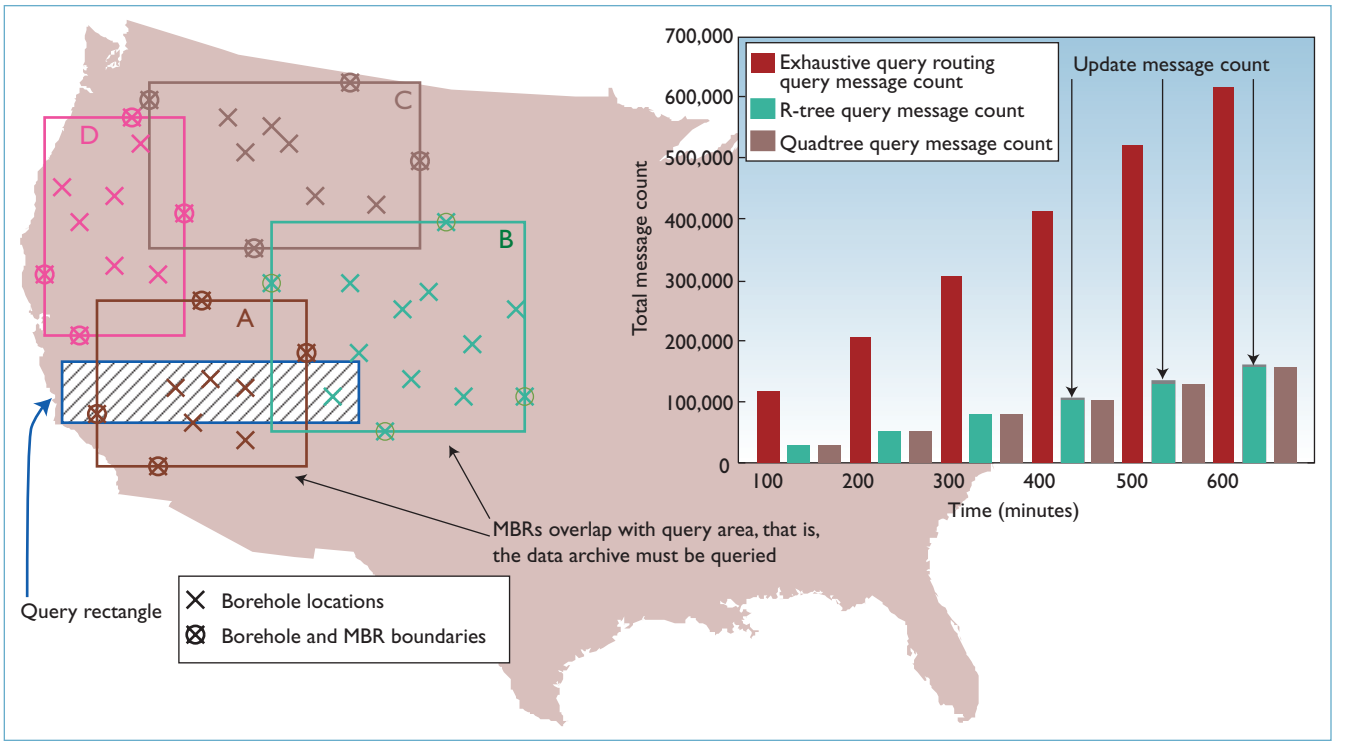

Figure 2. Example of query routing. The blue query rectangle intersects with two minimum bounding rectangles (MBRs) (brown and green), which results in those archives being queried for data. The message traffic is greatly reduced (shown on the right) when only relevant archives are contacted.

frequently, a repository leaves or joins the collective. Consequently, we can compile a list of all participating archives. This list might not be completely up to date at a specific time instance, but should be accurate enough to result in few disruptions. The list is distributed to every archive; when a query q arrives at a specific node, the node forwards it to all the other nodes for exhaustive processing. Although this naïve method — called exhaustive query routing (EQR) — is inefficient, it's useful as a baseline mechanism.

The metric we use for comparison is the total number of messages created in the system to execute a query q and collect the results. A lower number of messages reduces network traffic and indicates better system scalability. We can represent the number of messages generated by queries with EQR as $M = 2 \times Q \times (N - 1)$: the overall number of messages M is the product of the total number of queries Q and the number of archives N in the system. The total is doubled because an equal number of result messages are generated.

Query routing with R-trees and Quadtrees spatial indexing. The R-tree[8] and Quadtree[9] families of algorithms are well established for spatial data indexing. Both build a tree-structure that partitions the overall space into successively smaller areas at lower levels of the index hierarchy, and both are very useful in the core engines of spatial database systems. We use them in a novel way in GIME as index structures across multiple spatial databases to decrease the query-forwarding traffic. Specifically, we insert the minimum bounding rectangle (MBR) of each archive's data set into a global R-tree or Quadtree. Because we prefer to avoid a centralized index server, we further distribute copies of this global index structure to each archive. During query processing, an archive intersects each query rectangle with the MBRs stored in the global index. The query is then forwarded only to candidate archives whose MBR overlaps with the query rectangle, immediately reducing internode message traffic significantly. Figure 2 shows an example, with a blue query rectangle intersecting with a green and a brown MBR, respectively. Forwarded queries are marked to show that they originated from a server rather than a client to avoid query loops. The results of forwarded queries are returned to the initially contacted server, which aggregates them and returns the set to the client.

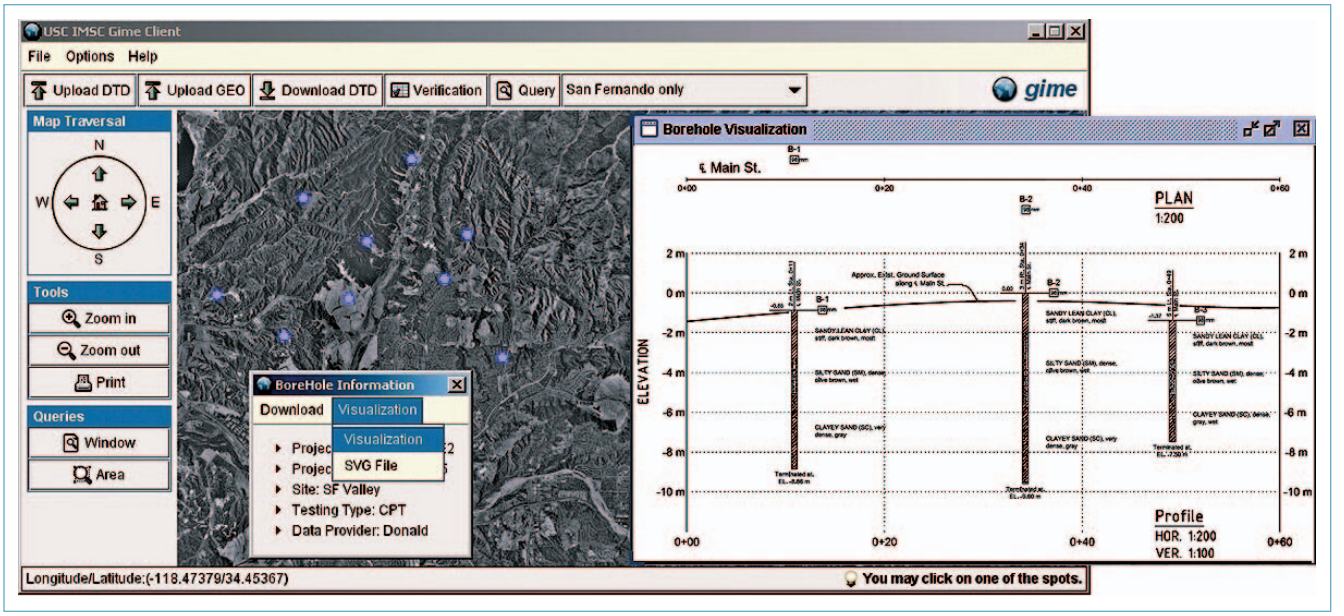

Figure 3. A sample borehole query and drafting client application. After a query is issued, a small pop-up display (the left window) shows a borehole's metadata. The fence diagram rendering (the right window) is generated from three selected XML data files.

This design requires GIME to synchronize the global index structures and keep them consistent, which introduces overhead in terms of both communication cost and implementation complexity. However, one characteristic of this technique greatly reduces overhead and makes it an attractive solution: because the global index structures manage bounding rectangles and not individual data points, changes to the data set of any specific archive only result in index updates if the MBR changes (which happens infrequently). Let's say an archive manages 1,000 spatial-point objects in a two-dimensional space. The MBR is generally defined by four of them: the two points with the most and least x values and the two points with the most and least y values (see Figure 3). Any insertion or deletion of data objects confined within the MBR doesn't affect the global index — only changes that either stretch or shrink the MBR need to propagate.

With our spatial-indexing mechanism, the query traffic reduces by a factor equal to the query's selectivity S_Q. The selectivity $0 \leq S_Q \leq 1$ estimates what fraction of the total number of archives might need to be contacted. Similarly, the update message traffic U is diminished by $0 \leq S_U \leq 1$, which describes how many data updates (including insertions and deletions) actually result in global index changes. Consequently, the frequency of data updates in the system affects the aggregate number of messages. However, we found in our experiments that the update traffic is a negligible fraction — less than 5 percent — of the overall message traffic.

Experimental validation. To validate our query-routing design's efficiency with a controlled and large quantity of access traffic, we ported the GIME modules to a simulation environment. We implemented the query-routing component with two plug-in modules to enable either the R-tree or the Quadtree algorithms. In a distributed environment, the communication overhead between servers dominates search complexity. Therefore, our simulation's goal was to quantify the query-routing traffic generated by processing a sequence of spatial-range queries and updates. If a query window intersected with several server MBRs, then the queried server forwarded the query to each repository. Any server with updated MBR broadcast tree-update information to all servers. We performed our experiments with both synthetic and real-world spatial data sets and created a simulation module to analyze EQR-generated message traffic with the same event sequence. Consequently, we could measure performance differences between the two approaches.

Figure 2 shows how our design improves query-routing performance. Tree-based designs

result in a decrease of approximately 60 to 70 percent of interserver message traffic compared with exhaustive query routing (with query window sizes of 10 to 20 percent of the data area). These results also indicate that our techniques should scale well.

During the simulation, we gained some additional insights into our techniques. For the R-tree based design, every MBR change translates into an index structure update. By contrast, we found that the index structure updates are reduced by an order of magnitude in the Quadtree design. We can conclude that the update message traffic to synchronize distributed Quadtrees is thus much lower than for R-trees because most MBR updates don't affect the Quadtree structure (meaning no index synchronization is necessary).

GIME Client Application Example

Figure 3 shows the GIME concept's feasibility in a sample client application. We wrote the borehole query and drafting (BQ&D) application the figure shows in Java; it demonstrates all of GIME's features. We assembled the background map from aerial images retrieved from the Microsoft TerraServer Web service.[10] A user can select a query window and get the metadata for matching borehole files from GIME along with their locations (displayed as dots over the background map; see Figure 3).

One advantage of a Web services infrastructure over traditional browser-based applications is that raw data can be programmatically accessed and locally processed. This integration is illustrated with a drafting capability in our client. The drafting component produces fence diagrams and a log of test borings (LOTBs) along user-defined alignments, based on data retrieved from GIME's Web services. A fence diagram is a two-dimensional interpretation of the soil stratigraphy along a (usually) vertical plane; the right-most pop-up window in Figure 3 illustrates the concept of a fence diagram. Fence diagrams and LOTBs are the fundamental tools civil engineers use to perform major geotechnical studies. Traditionally, the production of these diagrams has required significant drafting efforts. The drafting application (BQ&D) reduces this effort by automating the drafting process and directly accessing the required borehole data via the GIME infrastructure.

An earlier version of the GIME borehole visualization component generates SVG files using Apache Batik. It's available at the GIME Web site (http://datalab.usc.edu/gime/) for installation as a standalone Java program along with the Batik Squiggle package for displaying SVG files.

Asynchronous Data Access

Although the GIME infrastructure provides an efficient spatial query-routing algorithm, accessing and moving data in widely distributed environments naturally introduces communications overhead and delays between servers. Providing low system latency and good response time is especially challenging with services that move large volumes of data — for example, by retrieving map information from a Web service such as TerraServer. Web services provide a synchronous access method, but in a worst-case scenario, a service could become a bottleneck in the system. System designers must carefully consider performance issues when handling large data sets via Web services.

Novel techniques for asynchronous data access, such as Asynchronous JavaScript and XML (AJAX), aim to reduce system response latency and increase interactivity. AJAX is based on the observation that, for several tasks, only small amounts of data need to be retrieved from the server incrementally. In AJAX, a client adaptively retrieves data from a server based on, say, a user's interaction and navigation. As a result, the user experiences prompt system responses. Although AJAX is geared toward interactive applications, it's also useful for large-scale simulations that progressively access volumes of data.

In 2006, we implemented a Web browser-based prototype of the GIME client application that supports asynchronous data access. On the client side, the map data comes from Yahoo's map Web services via AJAX so that map retrieval latency is diminished. In addition, we defined the map tile area displayed on the client program as the borehole data access area. Once the user submits a spatial query, the client program only retrieves the query results that overlap with the visible map area. If the user changes the displayed area and navigates to a different location, the client program will calculate the data access area and retrieve the query result set correspondingly. In essence, GIME's data retrieval function is adaptively invoked from the user's interactions. Initial user experience has confirmed that the system's responsiveness is notably enhanced.

We're currently testing our Web services in realistic work environments in collaboration with municipal, state, and federal agencies, as well as international partners. We plan to extend our work further in various directions. First, we anticipate that we'll need additional Web services once GIME users deploy more sophisticated applications. One new Web service under development is for automatic format transformation (for example, taking XML to the Association of Geotechnical and Geoenvironmental Specialists [AGS]). Second, the progressive streaming of large query result sets is beneficial for some applications, so that the asynchronous processing of time-consuming simulations, for example, can start as quickly as possible. We plan to integrate the AJAX technology into existing GIME clients. You can find GIME project-related information and GIME client application examples at http://dmrl.usc.edu/gime/.

Acknowledgments

This research is funded in part by US National Science Foundation grants EEC-9529152 (IMSC ERC), CMS-0219463 (ITR), and IIS-0534761; we've received equipment gifts from Intel, Hewlett-Packard, Sun Microsystems, and Raptor Networks Technology.

References

1. R. Hunt, *Geotechnical Engineering Investigation Manual*, McGraw-Hill, 1984.
2. J. Swift et al., "ROSRINE: Resolution of Site Response Issues from the Northridge Earthquake, Field Processing Manual," USC Report CE471, 2001, p. 76.
3. T. Holtzer, "Distribution of USGS CPT Data via the Web," *Proc. COSMOS/PEER-LL Workshop on Archiving and Web Dissemination of Geotechnical Data*, COSMOS, 2001, pp. 93–96.
4. T. Malik et al., "SkyQuery: A Web Service Approach to Federate Databases," *Proc. 1st Biennial Conf. Innovative Data Systems Research* (CIDR 2003), VLDB Endowment, 2003, pp. 188–196.
5. R. Zimmermann, W.-S. Ku, and W.-C. Chu, "Efficient Query Routing in Distributed Spatial Databases," *Proc. 12th ACM Int'l Symp. Geographic Information Systems* (ACM-GIS 2004), ACM Press, 2004, pp. 176–183.
6. R. Zimmermann et al., "Design of a Geotechnical Information Architecture Using Web Services," *Proc. 7th World Multi-Conf. Systemics, Cybernetics, and Informatics* (SCI 2003), Int'l Inst. of Informatics & Systematics, 2003, pp. 394–398.
7. *Electronic Transfer of Geotechnical and Geoenvironmental Data*, 3rd ed., Assoc. Geotechnical and Geoenvironmental Specialists, 1999.
8. A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1984, pp. 47–57.
9. R. Finkel and J. Bentley, "Quadtree: A Data Structure for Retrieval on Composite Keys," *ACTA Informatica*, vol. 4, no. 1, 1974, pp. 1–9.
10. T. Barclay et al., "TerraService.NET: An Introduction to Web Services," tech. report MSR-TR-2002-53, Microsoft Research, June 2002.

Roger Zimmermann is a research assistant professor in the computer science department and a research area director in the Integrated Media Systems Center at the University of Southern California. His research activities focus on distributed and peer-to-peer systems, collaborative environments, streaming media architectures, geospatial database integration, and mobile location-based services. Zimmermann serves on the editorial boards of the ACM's *Computers in Entertainment* and the *International Journal of Multimedia Tools and Applications*. Contact him at rzimmerm@imsc.usc.edu.

Wei-Shinn Ku is a PhD candidate in the computer science department at the University of Southern California. His research interests include spatial/temporal data management, geographical information systems, network security, and peer-to-peer systems. Ku has an MS in electrical engineering from the University of Southern California. He is a member of the ACM and the IEEE. Contact him at wku@usc.edu.

Haojun Wang is a PhD student in the computer science department at the University of Southern California. His research interests include spatial data management, peer-to-peer systems, and location-based systems. Wang has an MS in computer science from the Oregon Graduate Institute. Contact him at haojunwa@usc.edu.

Amir Zand is a PhD student in the Department of Civil and Environmental Engineering at the University of Southern California. His main research field is database management systems for geotechnical and seismic information. Contact him at azand@usc.edu.

Jean-Pierre Bardet is a professor in and chairman of the civil and environmental engineering department at the University of Southern California. His research interests include geomechanics, geotechnical engineering, and granular mechanics. Bardet has a PhD in civil engineering from the California Institute of Technology. Contact him at bardet@usc.edu.