Performance improvement techniques for geospatial web services in a cyberinfrastructure environment – A case study with a disaster management portal

Wenwen Li a,⁎, Miaomiao Song a,⁎⁎, Bin Zhou b, Kai Cao c, Song Gao d

a GeoDa Center for Geospatial Analysis and Computation, School of Geographical Sciences and Urban Planning, Arizona State University, Tempe, AZ 85287-5302, United States
b Institute of Oceanographic Instrument, Shandong Academy of Sciences, Qingdao, Shandong 266001, China
c Department of Geography, National University of Singapore, 117570, Singapore
d Department of Geography, University of California, Santa Barbara, CA 93117, United States

A R T I C L E   I N F O

Article info:
Received 24 June 2014
Received in revised form 28 April 2015
Accepted 30 April 2015
Available online xxxx

Abstract

High population growth, urbanization, and global climate change drive up the frequency of disasters, affecting the safety of people’s lives and property worldwide. Because of the inherent big-data nature of this disaster-related information, the processes of data exchange and transfer among physically distributed locations are increasingly challenging. This paper presents our proposed efficient network transmission model for interoperating heterogeneous geospatial data in a cyberinfrastructure environment. This transmission model supports multiple data encoding methods, such as GML (Geography Markup Language) and GeoJSON, as well as data compression/decompression techniques, including LZMA and DEFLATE. Our goal is to tackle fundamental performance issues that impact efficient retrieval of remote data. Systematic experiments were conducted to demonstrate the superiority of the proposed transmission model over the traditional OGC Web Feature Service (WFS) transmission model. The experiments also identified the optimized configuration for data encoding and compression techniques in different network environments. To represent a real-world user request scenario, the Amazon EC2 cloud platform was utilized to deploy multiple client nodes for the experiments. A web portal was developed to integrate the real-time geospatial web services reporting with real-time earthquake related information for spatial policy analysis and collaborative decision-making.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

High population growth, urbanization, and global climate change drive up the frequency of disasters, affecting the safety of people’s lives and property worldwide. For example, in China 74% of state capitals and over 62% of counties are located in earthquake risk zones with potential for earthquakes larger than magnitude 7. Additionally, regions with high risk of natural disasters contain half of China’s population, who live within 70% of the urban centers where 75% of the national gross domestic product are distributed. Disaster management aims at alleviating the effects of disasters by supporting timely collection of disaster-related data, estimation of damage, evacuation routes planning and effective resource scheduling (Auf der Heide, 2002; McEntire, 2002; Goodchild, 2006; Alinia and Delavar, 2011). More specifically, a management system should be able to coordinate disaster-related data, most of which may be heterogeneous across geographically dispersed government agencies. Also, the system should provide an efficient transmission model for rapid response of end users’ spatial information requests. Lastly, the disaster management system should provide a user-friendly, and responsive web portal to facilitate human–computer interaction for successful decision-making purpose.

The emerging geospatial cyberinfrastructure (GCI; Yang et al., 2011) is a promising instrument for building a disaster management system by harnessing tremendous advances in computer hardware, GIS middleware, network and sharable geospatial web services. GCI is a descendant of Spatial Data Infrastructure (SDI). It focuses on providing better organization, integration, computation and visualization of institutionally scattered geospatial resources through the development of computationally efficient middleware. Within the context of GCI or SDI, service-orientation is a well-accepted strategy to improve the integration and exchange of heterogeneous geospatial data (Li et al., 2011). Using an Earthquake study as an example: to study the correlation between the location and magnitude of earthquake events and...
the mortality rate caused by earthquake-induced disasters, location data may come from a vector data model, such as an ESRI shapefile, whereas, the mortality data may cover a continuous surface, in a raster format. A service-oriented approach enables the conversion of various raw data types into a commonly understandable format to improve geospatial interoperability. Some web service solutions, such as Open Geospatial Consortium (OGC) Web Map Service (WMS), enhance remote interoperation by converting raw data into static images. The conversion comes at the cost of losing substantial attribute information from the original data. The OGC Web Feature Service (WFS), in comparison, is capable of preserving actual data, but it generates very large file while serializing the geospatial and attribute data. This leads to a long delay in data transfer in a client–server model. In this paper, we introduce a network transmission model that improves the performance in remote data transfer in a cyberinfrastructure environment by combining multiple data encoding and compression techniques. This model is successfully integrated into a GCI portal for efficient disaster data management.

The rest of the paper is organized as follows: Section 2 reviews recent literature in Cyberinfrastructure and geospatial interoperability. Section 3 describes the architecture of a disaster response system. Section 4 discusses the solution techniques to accelerate geospatial processing in terms of vector data encoding and transmission in a service-oriented cyberinfrastructure environment. Section 5 demonstrates the performance of proposed methods through a series of experiments. Section 6 demonstrates a Graphic User Interface (GUI) for real-time disaster analysis. Finally, Section 7 concludes the work and proposes future research directions.

2. Related work

2.1. Service-oriented geospatial cyberinfrastructure

In a disaster management scenario, the required data (e.g., satellite imagery showing the change before and after a disaster) are often geographically separated from (1) the web server portal on which the data is processed, and (2) where the decision-making takes place. This scenario requires the adoption of a decentralized and interconnected architectural design: distributed geoprocessing capabilities need to be supported and distributed resources should be reused and integrated easily. A service-oriented GCI fits right into this vision (Foster, 2005). However, existing researchers mostly focus on a single aspect of technological advancements in a GCI portal, such as service access, integration or high performance geospatial computing. For example, Li, Yang, and Yang (2010) and Lopez-Pelllicer, Florczyk, Béjar, Muro-Medrano, and Zaragaza-Soria (2011) adopted large-scale web crawling to discover scattered geospatial web services to enhance accessibility and to foster better geospatial data usage. Mansourian, Rajabifard, Valadán Zоеj, and Williamson (2006), Wei, Santhana-Vannan, and Cook (2009) and Li et al. (2011) proposed implementations of service-based spatial web portal to integrate distributed web services and visualize composite maps from data hosted through these services. Wang, Armstrong, Ni, and Liu (2005), Wei et al. (2006), Yang, Li, Xie, and Zhou (2008), Zhang and Tsou (2009) proposed grid-enabled cyberinfrastructure with a geoportal to speed up computational-intensive tasks. Though local processing performance is greatly improved, these processes are implemented in a standalone application, rather than in widely adopted OGC web services. Therefore their reusability is limited. To address this issue, Wang (2010) proposed a CyberGIS framework to synergize advancement in both cyberinfrastructure and geospatial sciences. Accordingly, this project seeks to provide parallel data processing through standardized geospatial web services.

2.2. Geospatial web services and their performance issues

Geospatial web services, which foster the interoperability among heterogeneous data and computing resources, are a key component of service-oriented GCI (Li, Li, Goodchild, & Anselin, 2013; Li, Goodchild, Anselin, & Weber, 2015). Since the 1990s, a number of government agencies, research institutes, and non-profit organizations have been collaborating to foster interoperability for geospatial data. For example, the standards organization OGC has released a number of specifications allowing uniform requests and exchange of geographic features over the Internet. These service standards have been intensively used to search, analyze, and update crucial disaster related information for disaster management (Weiser & Zipf, 2007). Among over 60 standards, the most widely adopted services are OGC Web Map Service (WMS; de La Beaujardière, 2002) and Web Feature Service (WFS; Vretanos, 2005a). A WMS allows the request of geo-referenced, raster, imagery over the Internet. When a GetMap request arrives at the server side through HTTP (Hypertext Transfer Protocol), a dynamic rendering process is triggered to generate a static image from vector feature data. This process is easier when hosting raw raster data since raster data can be tiled and cached in advance. Once completed, the map image is returned to a client for visualization with other resources, e.g. base maps or images layers. A WFS service, in comparison, allows the retrieval of actual feature data. When a GetFeature request is received by a WFS server, the vector feature geometries are selected (Vretanos, 2005b), encoded usually by the OGC Geographical Markup Language (GML; Cox et al., 2002) and returned to the client. Hence, WFS delivers actual data to the client and allows users to perform spatial analyses in addition to the visual display.

Despite their popularity, WMS and WFS suffer from performance bottlenecks. Performance improvement techniques for OGC WMS have been widely discussed, such as that in Yang, Wong, Yang, Kafatos, and Li (2005), Mikula, Trots, Stone, and Jones (2007), Baumann (2001), Hu et al. (2010), primary due to its easy implementation and visualization. However, in a WFS environment, the transmission of actual data poses big challenges in making data exchange among WFS servers efficient. Server-side encoding, data transmissions, client-side decoding and client side rendering remain the primary WFS performance bottlenecks. This may be the reason why WFSs have not yet been adopted as widely as WMSs. Zhang, Zhao, and Li (2013) describes a technique to improve the query performance of WFS using a Voronoi diagram indexing and parallel task scheduling. Yang et al. (2011) conducted some preliminary study on encoding data by a binary XML with some compression techniques, which form the basis of this work.

To overcome the WFS performance bottleneck, three research questions arise concerning the implementation of a high performance disaster management system: (1) how can spatial data be efficiently encoded and transmitted to support near real-time, remote data retrieval, (2) is a single pre-defined encoding and transmission strategy suitable for diverse hardware and network environments encountered by users? and (3) what form does an extendable architecture take such that distributed and heterogeneous geospatial services can be seamlessly integrated? In this paper, we will discuss our solution to the above questions toward building a service-oriented high-performance cyberinfrastructure for rapid disaster response and decision-making.

3. Architecture

Fig. 1 demonstrates the service-oriented architecture for disaster management. From bottom to top, a disaster processing system...
is divided into five tiers. Tier 1 is the data layer, in which large amount of geospatial data are located and managed in either a database management system (DBMS) or a file system. These data sources directly interact with a spatial data engine that contains a series of data adapters to operate data embedded in different DBMSs and their spatial query servers through supported APIs (Application Programming Interfaces). On top of the spatial data engine lays Tier 2, the web service engine, composed of three components: filter engine, encoding engine and rendering engine. The filter engine is responsible for sending spatial filtering requests to the database; the encoding engine is in charge of wrapping spatial vector data into an intermediate format, e.g. GML for easy data exchange; and the rendering engine is responsible for rendering the spatial data from its original format into static images for a vector-based WMS. Once data are generated according to OGC web service standards, they are pushed up to web service container at Tier 3 for standardized spatial data handling. These services interact with clients and handle clients' request through HTTP (HyperText Transfer Protocol). The web service container also supports the configuration of map styles for returned images, including symbolization, color ramps, and line width through SLD (Style Layer Descriptor). Tiers 1–3 contain nested encapsulations of data, processing, and service interfaces, each based upon the tier below. Tier 4 is the frontend application. In a browser, it provides a user interface for web service query, integration, and visual display of maps from a remote server. For system administrators, it also provides a management interface to upload or update spatial data in real-time and to monitor running status of web services. The uppermost layer, Tier 5, is the application layer. It contains the client tools customized to meet the requirements of specific decision-making purposes, e.g. early-warning systems or disaster assessment systems.

As discussed earlier, the system has bottlenecks shown with red arrows. In Tier 2, the bottlenecks are due to expensive data encoding in a web service engine. Between Tier 3 and Tier 4, vector data transmission acts as second a bottleneck, which needs to be resolved in order to further improve system performance. The next section discusses proposed solutions to address these challenges. The vector-based environment – the WFS will be our focus.

4. An efficient network transmission model for WFS

In general, a WFS network transmission model adopted to handle an incoming request involves the process of: pulling data from the backend database, filtering them according to the given spatial and temporal constraints, encoding them into an intermediate format, and then sending the data back to the client for visualization (grey modules in Fig. 2). To resolve the performance bottleneck when large data is being transferred, we extend the regular model by integrating multiple encoding and compression techniques. Red modules in Fig. 2 illustrate the proposed modules.

4.1. Vector data encoding for network transmission

A WFS facilitate data exchange across different GIS platforms by encoding raw spatial data into a common file format. GML Simple Feature Profile is used as the primary encoding format for a WFS (Burggraf, 2006). GML is based upon XML (Extensive Markup Language) – a well-known text-based markup language with rich expression capability for complex data. GML defines the encoding formats for simple points, lines, polygons and other complex spatial data structures, such as multi-points, multi-part polygon etc. Using GML to encode spatial geometries has the following advantages: first, GML is a text-based data format and is machine-processable in a cross-platform environment; second, GML is carried by HTTP, so it can easily cross firewalls, making GML one of the ideal candidates to carry, share, and interoperate spatial data among disparate GIS systems. Despite its advantages, GML also receives criticisms for its redundant tagging strategy. Also, due to its text-based
nature, encoding a large number of features can result in huge GML files.

To enable a light-weighted encoding method, in this WFS request/response workflow, we also enable the GeoJSON encoding. GeoJSON (Butler et al. 2008) is built upon JSON (Javascript Object Notation), which uses key: value pairs rather than open and close tags to encode geospatial data. JSON is both computer parsable and human readable, and is capable of describing complex data structures. GeoJSON extends the JSON format to include geometric features including Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. Each feature in GeoJSON has two parts: vector and attribute data, describing spatial extent and non-spatial properties of a feature.

Table 1 illustrates GML and GeoJSON representations of point, line and polygon features. As shown, a GML point feature will be enclosed by embedded open and closed tags <gml:Point> and </gml:Point>, while GeoJSON uses a flat structure with keys of 'type' and 'coordinates'. In a GML format, line or polygon features' coordinates are separated by a space. In GeoJSON format, coordinates are also separated by commas but each vertex is represented as a collection and is enclosed by "[" and "]". The geometry information is stored in the "geom" section of a WFS request, together with any other feature attributes. As shown in Table 1, encoding vertex-based data with GeoJSON generates more compact files than using GML, because the GeoJSON structure does not require long tags (such as <gml:coordinates> and </gml:Point> ). However, GML is more efficient, than GeoJSON is, at representing line and polygon data when the features contain multiple vertices. This observation is validated in the experiment section. (Note that we only consider WFS GetFeature Response which contains features of a single type, either of the type of point, or line, or polygon. A hybrid geometry collection is not considered).

### Table 1

<table>
<thead>
<tr>
<th>Feature type</th>
<th>GML</th>
<th>GeoJSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td><a href="">gml:Point</a></td>
<td>{ &quot;type&quot;: &quot;Point&quot;, &quot;coordinates&quot;: [44.371525, 22.172413] }</td>
</tr>
<tr>
<td>LineString</td>
<td><a href="">gml:LineString</a></td>
<td>{ &quot;type&quot;: &quot;LineString&quot;, &quot;coordinates&quot;: [44.371525, 22.172413] }</td>
</tr>
<tr>
<td>Polygon</td>
<td><a href="">gml:Polygon</a></td>
<td>{ &quot;type&quot;: &quot;Polygon&quot;, &quot;coordinates&quot;: [44.371525, 22.172413] }</td>
</tr>
</tbody>
</table>

#### 4.2. Introducing compression strategy to the network transmission model of a WFS

Despite GML and GeoJSON’s popularity in a variety of GIS applications, the text-based nature of GML and GeoJSON results in the storage of redundant information during the data encoding process. Therefore, large, text-based files are created when large, complex datasets are requested. To reduce the amount of data transferred over the Internet and to reduce client-waiting time, our framework introduces a compression/decompression module (red1 module in Fig. 2). After a dataset is encoded, its size is reduced in the compression module before being transmitted to the client. The compression module supports two compression algorithms: Lempel–Ziv Markov-chain Algorithm (LZMA; Morse, 2005) and DEFLATE algorithm (Deutsch, 1996). These two algorithms are chosen to integrate into the transmission model because of their popularity (Dorward & Quinlan, 2000), because they both are lossless compression algorithms and they represent two typical compression techniques: DEFLATE achieves faster compression speed but relatively lower compression rate. The LZMA is the opposite.

Given a sequence of characters in a string X, the LZMA algorithm (Fig. 3) is capable of completing the compression through only one pass of the data. As a string X is read, it is partitioned into non-overlapped substrings (Xr). For any Xr, and Xs in X, if r1 < r2, Xs must not be the same as Xr. Each Xr is then encoded into a pair of characters (X', Y'), where X' is the index of Xr which appears...
Algorithm 1.1: LAMA2

Input: $X$: a sequence of characters $x_1, x_2, \ldots, x_n$

Output: a sequence $(i_r, y_r)$ representing the encoded data, both of $r$ and $y_r$ are binary numbers, $r=1,2,3,\ldots$

$z^0 \leftarrow \emptyset$

read characters sequentially from $X$

divide $X$ into substrings $z^1, z^2, \ldots, z^r$ where $z^r \leftarrow x_{n(r)} \ldots x_{n(r+1)-1}$

for $z^r$ do

convert $z^r$ into $(i_r, y_r)$, where $i_r$ represents the sequence of the unique substring $z^{ir}$ identified in $X$, and $y_r$ is a unique character $x_k$ in $X$

$i_r \leftarrow \text{binary representation of } i_r$

$y_r \leftarrow \text{binary representation if } y_r$

Fig. 3. LZMA Algorithm.

as the part of the current string and $y'$ is the rest part of the string and is one element in the alphabet in $X$. For the final step of the encoding process, the pair of characters are converted into binary numbers. In a specific implementation, the dictionary for storing occurrences in the data stream and is contained in storing $z$ elements in a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time. In a specific implementation, the dictionary for storing compression time.

A DEFLATE algorithm, in comparison, combines two stages: duplicate string elimination and bit reduction, to compress a data stream. The duplicate string elimination (stage1) tries to use a shorter code to represent recurring patterns in the data. Two moving windows of the same size are stored during the compressing process: $w_1$ storing $n$ characters backward from current location and $w_2$ storing length-$n$ characters forward from current location. If a substring in $w_2$ is contained in $w_1$, this substring is replaced by a pair of numbers: the length of the substring and the offset referring how far back this substring appears in $w_1$. The first stage describes a classic LZ77 lossless compression algorithm.

To further reduce the size of data, a Huffman encoding is introduced as the second stage of DEFLATE algorithm. This encoding reduces the average code length by using the shortest code for the most frequently occurring character and the longest code for the least frequently occurring character. A binary coding tree is constructed where each character in the original data flow is represented as a leaf node. The left child node is encoded as 0 and right child node is encoded as 1. The final code of a character is a sequence of 0s and 1s on the path from the root to the leaf node.

To generate such a Huffman tree, the original data flow first needs to be scanned to count the number of occurrences of each unique character. Then characters with minimal frequencies will be selected to construct a subtree, and this process will continue until all characters have been encoded in the Huffman tree. During decoding process, the compressed data flow must first be de-serialized from its Huffman code to the original characters, and then the repeated occurrence pattern will be replaced by the original string to recover the data stream before compression.

Huffman encoding is efficient in runtime and it takes only $O(n \log n)$ operations to construct it. It also provides good compression ratio at $\sum \frac{p_i \log_2 \frac{1}{p_i}}{\log_2 k}$, where $p_i$ is the probability of a character $x_i$ appearing in the data stream and $k$ is the unique number of characters in the data stream to be compressed.

Though the above compression techniques substantially reduce the amount of data to be transferred over the Internet, they also introduce extra time for server compression and client decompression. For example, in a traditional WFS request/response flow, the wall time $t$, a.k.a, total response time, is composed by the time used for handling a HTTP request and encoding vector data into an intermediate format ($t_1$), such as GML, result transfer or client download ($t_3$) and client decoding ($t_5$).

$$t = t_1 + t_3 + t_5$$ (1)

After introducing the compression module, the total response $t'$ also includes the time for result compression on the server side ($t_2$) and client decompression ($t_4$):

$$t' = t_1 + t_2 + t_3 + t_4 + t_5$$ (2)

Although compression reduces the amount of data and time ($t_1$), it also introduces extra time $t_2$. Therefore, only when

$$t' - t = t_2 + t_4 + t_5 - t_1 < 0$$ (3)

are the advantages of compression realized. It is known that LZMA has better compression ratio than DEFLATE, however DEFLATE runs faster than LZMA, so when handling data with different repeat patterns, they will behave differently. Section 5 will compare their performance under different transmission and network conditions.

5. Experiment

In this section, we present systematic experiments to test the performance of WFSs with proposed network transmission model. To simulate a real-world scenario, we set up the following experimental environments: (1) the server hosting web services and the client application hosting the CI web portal (client) were deployed at physically distributed locations; (2) the web service engine with support of a geospatial database (PostgreSQL 9.4 + PostGIS1.5) was deployed on a Windows server located at UCSC; (2) a main client node, which is also the permanent host for the CI portal, was deployed at ASU; (3) we also deployed two additional client nodes on the cloud through Amazon EC2 platform: One cloud node is located at Singapore and another cloud node is located at Ireland.
Both of the cloud nodes were deployed as EC2 microinstances with the same hardware and network conditions. The configuration of the cloud nodes simulates users who have lower Internet speed or unstable network conditions, as might occur after major disasters. As the speed of EC2 instances is variable, each result is the averaged value among six runs. Fig. 4 displays the locations and the connection speed between the server node and the client nodes.

Sections 5.1 and 5.2 focus on comparing WFS performance with and without our proposed techniques using point and polygon data on the main client node. Section 5.3 compares the performance of WFS with the proposed model under different network conditions.

Fig. 4. Cloud deployment of server and client test nodes.

Fig. 5. Comparison WFS GetFeature responses using different encoding and compression method for point data.
5.1. WFS performance for point data on main client node

We used a 12-year earthquake location dataset to test the network transmission efficiency of the proposed model. The compression methods, encoding methods and data size all varied within the tests. Fig. 5(a)–(c) compares the differences in size of data transferred back to the client side using the combination of encoding (GML or GeoJSON) and compression techniques (LZMA or DEFLATE). The x-axis shows the number of years of earthquake data used for the test. For the four leftmost encodings (1–8 years), as the time interval doubles, the amount of data transferred almost also doubles. Fig. 5(a) shows that GeoJSON is much more efficient than GML in encoding point data. The data size after GeoJSON encoding is less than half of that in GML. This can be attributed to GML’s redundant open and close tag encoding. Fig. 5(b) and (c) demonstrate the data sizes after employing DEFLATE and LZMA compression. It can be clearly observed that: (1) no matter which compression method is used or the size of the dataset, data encoded in GeoJSON are always smaller than the data in GML. This is not surprising since the raw GeoJSON file is much smaller than GML file, according to Fig. 5(a). It is also clear in Fig. 5(b) and (c) that the differences between GML and GeoJSON

![Figure 6a](image1.png)  
(a) Response time for GeoJSON encoding with DEFLATE and LZMA compression

![Figure 6b](image2.png)  
(b) Response time for GML encoding with DEFLATE and LZMA compression

Fig. 6. Comparison of total response time for a WFS GetFeature request of earthquake point data.
encoded files are small after either compression method, because the majority of the content (coordinates) are the same for both files. (2) LZMA yields better compression rate than DEFLATE. When compressing GML file, LZMA achieves on average a 29-time compression rate and it was only 24 for DEFLATE. Similarly, LZMA compresses a GeoJSON file at a compression ratio of 16:1, and DEFLATE only receives a value of 10. (3) GML has more redundancy than GeoJSON when encoding point data, reflected by its higher compression ratio using either DEFLATE or LZMA than GeoJSON.

In addition to data size, we also compared WFS efficiency in terms of total response time $t^*$ of a WFS GetFeature Request using the proposed transmission model. Fig. 6 compares the response time for GML and GeoJSON encoding combined with different compression methods. As the amount of requested data increases, the proposed transmission model presents different performance. First, no matter which text encoding method is used, GeoJSON or GML, DEFLATE compression presents the fastest response time. Using DEFLATE, the transmission model can be more than twice as fast as compression-free transmission. When more data is requested, DEFLATE achieves even greater speed up. For instance, the speed up of total response time using DEFLATE compression can reach up to 2.9 for GeoJSON encoding (Fig. 6a) and 3.7 for GML encoding (Fig. 6b) using the 12-year data. In contrast, LZMA does not perform as well as DEFLATE. For GeoJSON data, the WFS response time using LZMA compression is slower than when no compression is used. For transmitting GML data, the LZMA compression is faster than the compression-free transmission, but it is still slower than DEFLATE compression. This is because although LZMA compression can achieve better compression ratio, it takes longer for server-side compression and client-side decompression. However, it may be noted that this time is significantly longer than transmitting the uncompressed data over the Internet would be, given the high-speed connection between the ASU client and the UCSB server. For the GML case, as the GML file is really large, much larger than GeoJSON for encoding same amount of data, the transmission time over the Internet is long enough to make the LZMA approach demonstrate better performance than the regular WFS transmission model.

Through the above experiments, we show that when the client–server maintains a high connection speed, in another words, when the network transmission is not a bottleneck: (1) the transmission model using DEFLATE compression is the most efficient among all cases; (2) GeoJSON format is more effective in encoding point data than GML; and (3) LZMA has better compression ratio than DEFLATE for compressing either GeoJSON or GML files, and this compression technique performs better than the regular WFS model when the requested data is in GML and the requested data size is large.

5.2. WFS performance for polygon data on main client node

This section compares the performance of the proposed techniques for polygon data. US 2010 census tract data is used in the study. The same set of experiments was conducted here as those in Section 5.1. Fig. 7 shows the data size for 5 increasing samples of data after LZMA and DEFLATE compression. Fig. 7(a) shows the raw data size for GML and GeoJSON. This is the data volume

![Comparison in data size after GeoJSON and GML encoding](image1)

(a) Comparison in data size after GeoJSON and GML encoding with no compression

![Comparison in data size after GeoJSON and GML encoding with DEFLATE compression](image2)

(b) Comparison in data size after GeoJSON and GML encoding with DEFLATE compression

![Comparison in data size after GeoJSON and GML encoding with LZMA compression](image3)

(c) Comparison in data size after GeoJSON and GML encoding with LZMA compression

Fig. 7. Comparison WFS GetFeature response using different encoding and compression method for polygon data.
transferred in a regular WFS transmission model. It can be seen that GML is more effective in encoding polygon data than GeoJSON. This is not surprising since the analysis of the GML and GeoJSON data structures indicated that GeoJSON used an extra ‘[‘ and a comma to separate coordinates in its presentation. For a polyline or a polygon feature, which usually has multiple vertices, the redundancy caused by the extra separators in GeoJSON substantially increases the final data size.

Fig. 7(b) and (c) presents the data size in GML and GeoJSON after applying the proposed compression techniques. As shown, for both GeoJSON and GML files, the LZMA compression output is half the size of the DEFLATE compression output. This observation is consistent with the point data experiment. However, there is a notable difference: the compression ratio is only about 3:1 for DEFLATE and 6:1 for LZMA, both much lower than the point data compression rates of 10:1 and 16:1. This means that point data encoding structure is less efficient than polygon encoding in both GML and GeoJSON. Another observable difference in Fig. 7(b) and (c) is that GeoJSON and GML files remain very different in size after applying the same compression technique. This indicates that current data structure of GeoJSON for encoding polygon data is not as compressible as GML.

Fig. 8 illustrates the total client wait time as the sum of: request processing time on the server end, data decompression time, and decoding time on the client side. It shows that under current network conditions, the total DEFLATE response time is faster than the GeoJSON response time by about 3:1. This is true whether the encoding is GeoJSON (Fig. 8a) or GML (Fig. 8b). The stacked bars show that the majority of the request-response cycle is monopolized by the client for on-the-fly decompression of streaming data.
For a regular WFS (the “none” category), this simplifies the data transfer time. Contrasting the earlier point geometry experiment, polygon transmission with LZMA is slower than the regular WFS model, suggesting that LZMA is not suitable for a polygonal data request in a fast network environment.

The results suggest that: (1) GML has a better encoding structure for polygon dataset than GeoJSON. As polygon and polyline encodings are very similar, this conclusion can apply to polyline data as well; (2) transmission models using DEFLATE are shown to be more efficient than regular WFS model and the LZMA model in a fast client–server network; (3) the compression ratio for both GML and GeoJSON of polygon data is smaller than those for point data, suggesting that GML and GeoJSON’s polygon encoding structure is better than their point encoding structures.

5.3. Comparison of response time for multiple, world-wide cloud nodes

This section tests the performance of the proposed network transmission model with the client application running on distributed cloud nodes. After occurrences of major disasters and infrastructure failure, remote users’ network stability may be greatly affected. Therefore, we deployed two micro-instances with identical hardware and network configuration to represent this disrupted scenario. For consistency, we used the same census tract data as the test dataset. As GML was validated to be the more effective encoding method for polygon data, we only tested the performance of the model combining GML with LZMA and DEFLATE compressions. The four columns of data (from left to right) in Fig. 9 are of sizes of 7 M, 14 M, 28 M and 56 M respectively. This figure shows that (1) as the amount of requested data doubles, the total response time almost doubles. The web server does not introduce extra complexity to process larger datasets, which reflects the scalability of the web server. (2) For the same request, it takes a longer time for the Singapore node to get the response than it takes the Ireland node. Given that they are pinging the same server, this result reveals that location matters in network transmission. In the test case, the server is located in North America, so the Singapore node may need access its data through more physical routers than the Ireland node, resulting in longer waiting period. (3) Because of this geographic inequity, different users at different locations must carefully choose the most suitable WFS transmission model according to their network condition. As shown, the total response time using different compression techniques are close in their performance. This is because although DEFLATE decompression is faster than LZMA decompression, in the network with lower speed and high latency, decompression is no longer a bottleneck. Instead, what matters is the speed for transferring the data stream. More importantly, as the data volume increases to certain amount (>28 M bytes), the model with LZMA compression shows faster response time than the DEFLATE model on the Singapore node. However, the DEFLATE model still works slightly better than the LZMA model on the Ireland node.

Through above experiments, we verified the better performance of the proposed transmission model than the regular WFS model. In next section we describe the integration of the proposed model into an operational CI system for disaster management.

6. Graphic user interface for the disaster management portal

Fig. 10 demonstrates the Graphic User Interface (GUI) of the disaster management CI portal that integrates the proposed techniques. This portal integrates data services for disaster data visualization and analysis. A backend data crawler retrieves real-time earthquake data from a USGS website (http://earthquake.usgs.gov/earthquakes/map/) and stores it into the backend geospatial database, an essential component shown in Figs. 1 and 2. A web service engine, enabled by GeoServer and deployed in the service pool of the cluster framework, pulls spatial data directly from the database. Due to this dynamic invocation mechanism, a new earthquake record is immediately reflected in the portal. The earthquake data are published as WFS enabled by the proposed network transmission model. The window in the lower-left corner of the screenshot is a configuration interface, which allows end users to select their preferred encoding and compression methods. Choosing compression methods that complement the users’ network conditions maximizes the WFS performance. Once the actual earthquake data is on the client, it can be further analyzed using statistical or spatial analytical functions. For example in Fig. 10, a bar chart displays the number of earthquakes within a space–time frame. In addition to WFS, this portal is also capable of adding distributed WMSs for correlated analysis. A WMS can be added directly by an end user, or can be searched through a CSW (Catalog Service for the Web; Senkler, Voges, & Remke, 2004) interface provided by a central repository.
located at ASU. This way, all modules in this portal, including the data discovery module, data integration, and visualization module are all web service-based.

As an example of a disaster management scenario, consider a researcher who wants to analyze the correlation between earthquake locations and mortality of landslides (a disaster caused often by earthquakes) across different geographic regions. Fig. 10 presents how this example would look in the disaster management CI portal. The researcher can first retrieve any data available through the cataloged web services, including earthquake data in 2013 from an ASU WFS server, US state boundary from a UCSB WMS server, and the landslide mortality data from NASA SEDAC (Social-Economic Data Application Center) WMS server. These data can be seamlessly integrated and visualized in our CI portal. It can be seen that although California, US has a high number of earthquakes in 2013 (number is reflected by the size of the orange circles), the mortality caused indirectly by its induced disaster is low (indicated by the very sparse red color on the mortality distribution map). In contrast, places such as Northern Africa and Northwest China have suffered seriously from earthquake and landslide disasters, as indicated by the large orange circles and dark red hues on the map. Using the results from this initial investigation, further studies can then be conducted in these areas to improve disaster management, such as comparing the geology and urban disaster preparedness infrastructure.

7. Conclusion and discussion

This paper reports our proposed network transmission model to enhance the performance of geospatial web services, in particular WFS, in a Cyberinfrastructure environment. The network transmission model enables an efficient delivery of geospatial data in different network environments. Systematic experiments revealed several interesting findings. First, the proposed transmission model presents significant advantages over the regular WFS transmission model. The two compression techniques are suitable for different network environments. In a network with high speed and low latency, the DEFLATE model works better than LZMA; in contrast, in an environment with a data transmission bottleneck, the LZMA model performs better. Second, GML is more efficient at encoding polyline/polygon data than GeoJSON; and GeoJSON is more efficient than GML for encoding point data. Both methods have redundant point data encoding, reflected by their high compression ratios. This finding suggests that the renovation of the encoding structure for both GML and GeoJSON may be necessary. Third, location matters in the selection of the transmission model. Even with the same hardware, software, and network environment, different users may receive different experiences with the same web server. Therefore, service quality in terms of request-response latency needs to incorporate this location-awareness and be tailored for users in different spaces and times. These findings contribute to a comprehensive understanding of performance issues in vector-based data transmission in a service-oriented cyberinfrastructure environment, and may serve as a guide for refining existing encoding standards and data transmission middleware.

We have successfully integrated the proposed techniques into a service-oriented CI portal for disaster management. This portal provides a central access point where general public or researchers can interact with physically distributed data services, configure the best network transmission model considering his/her hardware and network environment, view statistical information, and conduct space–time queries about remote datasets. To our best knowledge, this is the first time such transmission model has been integrated into an operational system. We expect that the disaster response community can greatly benefit from the establishment of this service-oriented cyberinfrastructure for efficient spatial decision-making. Additionally, this performance boosting model for WFS and the portal techniques be applied for analyzing more than just disaster data, it can also be applied to other applications, such as emergency response, which has real-time requirements that can benefit from the efficient compilation of distributed data.

Last but not the least, per a latest study conducted Li (2014), Li, Wang, and Bhattia (2015), though the interoperability process in GIScience has been greatly advanced over the past few years (indicated by the over twenty times of increase in the number of WMSs emerging on the Web in 2014 than that in 2010), the support to
WFS remains insufficient by the community. This fact may be largely due to the challenges in WFS’s heavy data transfer process. By providing timely solutions to improve the performance of WFS, we expect this work to serve as a momentum to widen the use of WFS, and eventually broaden the adoption of open geospatial service and science. The solution technique will also be open sourced to benefit other GIScience researchers and users. In the future, we will also deploy the proposed transmission model to the GeoDa's high performance cluster to achieve high system throughput.

Acknowledgements

This paper is in part supported by National Science Foundation PLR-1349259, BCS-1455349 and PLR-1504432, as well as the Open Geospatial Consortium.

References


Please cite this article in press as: Li, W., et al. Performance improvement techniques for geospatial web services in a cyberinfrastructure environment – A case study with a disaster management portal. Computers, Environment and Urban Systems (2015), http://dx.doi.org/10.1016/j.compenvurbs.2015.04.003