Loading Architecture for a Sensor Web Browser on Digital Earth

Chih-Yuan Huang **Steve Liang**

Geo-Sensor Web Laboratory, Department of Geomatics Engineering, University of Calgary
{huangcy, steve.liang}@ucalgary.ca

Abstract

The world-wide sensor web observes real world phenomena at a particular moment in time with a large number of geo-referenced sensors. Sensor web needs a sensor web browser for accessing distributed and heterogeneous sensor networks in a coherent frontend. The Digital Earth provides a geo-referenced three-dimensional environment for intuitively browsing and displaying sensor observations. However, the major challenge is to load the vast amount of sensor observations from servers to a sensor web browser while minimizing the delay that a user experiences. This research uses two techniques to address the challenge. First, the browser caches transmitted data onto the local hard drive to reduce redundant internet bandwidth consumption. Second, this work designs a loading architecture to decouple sensor data loading, rendering, and browsing. The proposed scheme is implemented in the GeoCENS sensor web browser. To the best of our knowledge, with the proposed loading architecture, GeoCENS is the first Digital Earth-based sensor web browser.

Background and Relevance

The Digital Earth was envisioned in Gore's 1998 speech[1]. The Digital Earth is a three-dimensional visualization model of the physical Earth, which contains high resolution imagery and digital elevation models. Users are able to intuitively interact with Digital Earth by navigation features such as flying to places or floating above the surface. In recent years more and more publicly-available Digital Earths are revealed, such as ESRI's ArcGIS Explorer[2], NASA's World Wind[3], Microsoft's Bing Map[4], and Google Earth[5]. One of the visions that Gore described is that people can access vast amount of scientific information through the Digital Earth to help them understand real world. Therefore, at the same time that Digital Earths were being developed, the deployments of the world-wide sensor web were also put into practice for observing heterogeneous, environmental phenomena. The sensor web concept originated at the NASA/Jet Propulsion Laboratory in 1997 (Delin *et al.* 2005; Liang *et al.* 2005) for acquiring environmental information by integrating massive spatially distributed consumer-market sensors. The world-wide sensor web has been applied in a range of applications, including: large-scale monitoring of the environment (Hart and Martinez 2006), civil structures (Xu *et al.* 2004), roadways (Hsieh 2004), and animal habitats (Mainwaring *et al.* 2002). Ranging from video camera networks that monitor real-time traffic, matchbox-sized wireless sensor networks embedded in the environment for monitoring habitats, sensor webs generate tremendous volumes of valuable observations, enabling scientists to observe previously unobservable phenomena. Similar to how the World

[1] http://www.isde5.org/al_gore_speech.htm
[2] http://www.esri.com/software/arcgis/explorer/index.html
[3] http://worldwind.arc.nasa.gov/java/
[4] http://www.bing.com/maps/
[5] http://www.google.com/earth/index.html

Wide Web needs an Internet browser for viewing web pages, the sensor web needs a coherent frontend for accessing the distributed and heterogeneous sensor networks. This kind of coherent frontend is called sensor web browser.

In order to achieve Gore's Digital Earth and sensor web visions, an online Digital Earth-based sensor web browser for users to browse, search, manage, and exchange sensor data is needed. However, transmitting the vast amount of sensor readings from servers to a sensor web browser with minimum delay is very challenging. Therefore, the goal of this paper is to present a sensor web data loading architecture for addressing the following two issues: (1) transmitting vast amount of sensor data and (2) minimizing the delay time that user might experience. To address the first issue, efficient data loading management utilizing a local cache is applied. To address the second issue, a loading architecture that decouples loading and browsing, a speculation mechanism, and a dynamic priority queue (Xhafa & Tonguz 2001) are applied. With the proposed scheme, efficient sensor web data loading and good user experience are attained. The sensor data used in this work are historical and can be represented as points on the Digital Earth.

Methodology

Issue 1: Transmitting Vast Amount of Sensor Data

To mitigate this issue, we can adopt strategies from Web browsers and how they are able to minimize redundant transmissions. Web browsers cache data, which means they store web page requests and associated responses as key-value pairs on the local disk in order to reduce unnecessary and redundant transmissions, and to improve end-to-end latency. Similarly, this work implements a caching strategy to reduce the unnecessary transmission of sensor data in the sensor web. However, sensor web browsers' requests and sensor observations are different from web browsers' requests and web pages. We cannot simply manage sensor web requests and responses as key-value pairs in web caches. A sensor web request can be interpreted as asking for sensor observations of a certain phenomenon in a set of spatio-temporal cubes. The spatio-temporal cubes can be distributed irregularly in space and time. An effective sensor web caching strategy requires efficient management of these spatio-temporal cube requests. Therefore, we develop a new spatio-temporal indexing structure, LOST-Tree (LOading Spatio-Temporal indexing tree) to manage sensor data loading with a local cache.

A LOST-Tree manages sensor data loading of one phenomenon. A LOST-Tree consists of two techniques. First, instead of indexing massive amounts of raw sensor data, a LOST-Tree will index requests (*i.e.*, spatio-temporal cubes). Because no actual data stored in LOST-Tree, it is small and able to fit into memory for efficient processing. Second, a LOST-Tree applies any two regular and aggregatable structures onto the spatial and temporal domains for transforming irregular spatio-temporal cubes into regular cubes. For instance, this work implements with a Quadtree (Finkel & Bentley 1974) and the Gregorian calendar. These two structures are integrated by using temporal structure (*i.e.*, the Gregorian calendar in this work) as main structure and embedding spatial structure (*i.e.*, Quadtree in this work) in nodes of temporal structure. In this way, a record in LOST-Tree represents a spatio-temporal cube, which allows simple look-up searches (rather than range query) for determining unloaded cubes. Records are inserted into LOST-Tree only if the corresponding cubes have been loaded. Since both structures are aggregatable, the more cubes are loaded, the fewer records remain. Therefore, a LOST-Tree is scalable, light weight, and able to efficiently identify unloaded potions in sensor web browser. By incorporating LOST-Trees into a sensor web browser, previously loaded sensor data can be retrieved from the local cache. The end-to-end latency can be reduced and the Internet bandwidth can be efficiently utilized on transmitting the data that has not been loaded.

Issue 2: Minimizing the delay time that users experience
In order to minimize the delay users might experience when loading sensor web data, we need to decouple data loading, rendering, and browsing. We implemented the following performance improvement strategies in the GeoCENS sensor web browser (Liang *et al.* 2010). (1) *Checker*: In order to reduce the requesting frequency, instead of executing loading processes whenever the Digital Earth is moving, loading processes start when the earth has stopped moving for more than a defined period of time (e.g., 500 milliseconds). This follows the assumption that a user's area of interest corresponds to where the user stops moving the Digital Earth. (2) *Wrapper*: After a user requests a spatio-temporal cube, a thread is trigged to determine the portions that are not available in the local cache (by using a LOST-Tree). The *wrapper* also filters out the requests that have been issued, and composes new loading tasks. (3) *Loader*: After the loading tasks are created, these tasks are stored into a queue waiting for being sent to the servers. A thread-pool with pre-defined number of threads polls these loading tasks from the queue and issues requests based on the loading tasks to the servers. After the data are transmitted to the browser, a thread is trigged to parse the data and store them in local cache. (4) *Poller*: The *checker*, *wrapper*, and *loader* are for the data loading, and the *poller* is for rendering. A timer runs periodically to check if user moves the Digital Earth. If not, a thread will then be trigged to retrieve data from the local cache and render the data on Digital Earth.

Besides applying the decoupling architecture, two additional mechanisms are implemented for improving a user's experience. They are speculation and dynamic priority queue. Firstly, speculation means the system issues requests before a user issues the requests. For instance, we expect users will browse the nearby regions around their initial area of interest. Therefore, when a user requests a spatio-temporal cube, the spatial component (e.g., bounding box) of the cube is then expanded. Secondly, users are allowed to navigate freely within the Digital Earth environment. A user may decide to move to other places before the previous loading tasks are digested. As a result, it is important to prioritize the loading tasks dynamically. For example, if we manage the loading tasks with a first-in-first-out (FIFO) strategy, new requests will not be executed until the old requests are finished. In this case, users may feel slower system performance because the system is background loading data they aren't expecting. Therefore, instead of following a FIFO queue, we apply a dynamic priority queue (Xhafa & Tonguz 2001) to prioritize the loading tasks. Whenever a user moves the Digital Earth, the priorities of loading tasks in the queue will be re-assigned with the distance between the current area of interest and the tasks' loading area. By using a dynamic priority queue, loading tasks that are close to the current area of interest will be requested from server first.

Conclusions

This paper presents a sensor web data loading architecture for constructing a sensor web browser. This work applies a caching mechanism that eliminates redundant transmissions and reduces end-to-end latency. Figure 1 depicts the end-to-end latency before and after implementing local caching. The cache mechanism significantly improves system performance. In addition, this work presents a loading architecture that decouples data loading, rendering, and browsing in order to minimizing the delay time that a user might experience while loading data from servers. Two additional mechanisms also improve user experiences: speculation and dynamic priority queue. With these mechanisms, users can efficiently and smoothly interact with a sensor web browser while transmitting vast amount of sensor web data in the background. We implemented the loading architecture in the GeoCENS sensor web browser (Liang *et al.* 2010). The GeoCENS sensor web browser can be accessed at http://www.geocens.ca. Based on our testing experience, the proposed loading architecture successfully addresses the challenges of integrating the Digital Earth and the world-wide sensor web. To our best knowledge, GeoCENS is the first Digital Earth-based sensor web browser.

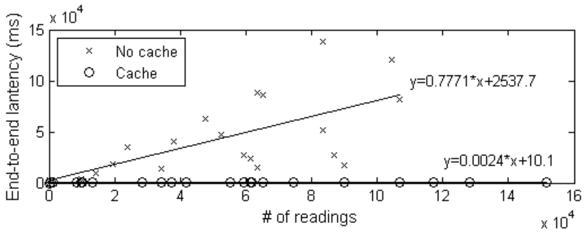

Figure 1. End-to-end latency before and after applying local cache

References

Delin, K.A., Jackson, S.P., Johnson, D.W., Burleigh, S.C., Woodrow, R.R., McAuley, J.M., Dohm, J.M., Ip, F., Ferre, T.P.A., Rucher, D.F., & Baker, V.R. (2005) Environmental Studies with the Sensor Web: Principles and Practice. *Sensors*, **5**, 103-117.

Finkel, R. & Bentley, J.L. (1974) Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, **4(1)**, 1–9.

Hart, J.K. & Martinez, K. (2006) Environmental Sensor Networks: A revolution in the earth system science? *Earth Science Reviews*, **78**, 177-191.

Hsieh, T.T., (2004) Using Sensor Networks for Highway and Traffic Applications. *IEEE Potentials*, **23(2)**, 13-16.

Liang, S., Chang, D., Badger, J., Rezel, R., Chen, S., Huang, C.Y., & Li, R.Y. (2010) GeoCENS: Geospatial Cyberinfrastructure for Environmental Sensing. In *Sixth international conference on Geographic Information Science*, Zurich, Switzerland.

Liang, S.H.L., Croitoru, A., & Tao, C.V. (2005) A distributed geospatial infrastructure for Sensor Web. *Computers and Geosciences*, **31(2)**, 221-231.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D., & Anderson, J. (2002) Wireless Sensor Networks for Habitat Monitoring. In *2002 ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, United States.

Xhafa, A. & Tonguz, O.K. (2001) Dynamic Priority Queuing, Channel Bormwing, and First In First Out Handoff Schemes: A Performance Comparison. *IEEE Yehiculor Technology Confirence Spring*, **2**, 951-955.

Xu, N., Rangwala, S., Chintalapudi, K.K., Ganesan, D., Broad, A., Govindan, R., & Estrin, D. (2004) A Wireless Sensor Network for Structural Monitoring. In *Conference on Embeded Networked Sensor Systems*, Baltimore, MD, United States.