TerraFly Geospatial System for Disaster Mitigation

Naphtali Rishe, Xudong He, Scott Graham, Shu-Ching Chen, Armando Barreto, Malek Adjouadi
Florida International University, 11200 SW 8th ST, ECS 354, Miami, FL 33199

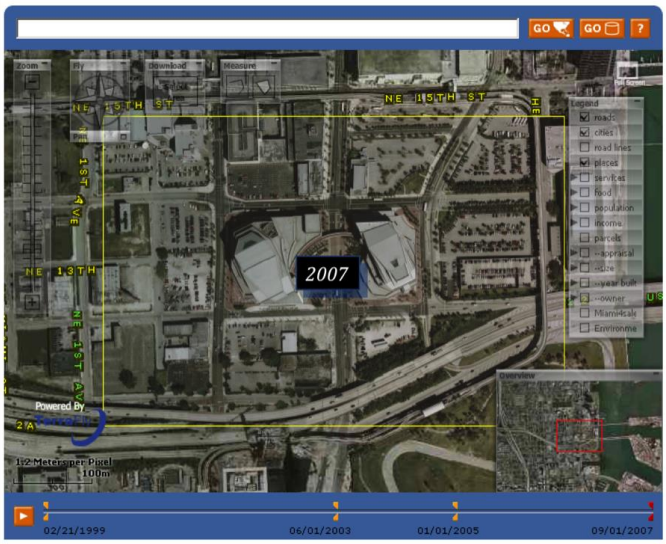
Florida International University's (FIU's) second-phase CREST, the Center for Innovative Information Systems Engineering is housed in its School of Computing and Information Sciences and its Department of Electrical and Computer Engineering. The Center's four research thrusts bring together a multidisciplinary group of researchers, large-scale collaborative relationships, and a broad ecosystem of partners to perform research that will lead to information technologies that help to solve critical societal problems of national priority.

TerraFly users visualize aerial imagery, precise street name overlays, and various other overlays. Users virtually "fly" over imagery via a web browser, without any software to install or plug in. Tools include user-friendly geospatial querying, data drill-down, real-time data suppliers, demographic analysis, annotation, customizable applications, route dissemination via autopilots, production of aerial atlases, and application programming interface (API) for web sites.

TerraFly has been featured on TV news programs (including FOX TV News), worldwide press, covered by the *New York Times*, *USA Today*, NPR, and *Science* and *Nature* journals.

The 40TB TerraFly data collection includes, among others, 1-meter aerial photography of almost the entire United States and 3-inch to 1-foot full-color recent imagery of major urban areas. TerraFly vector collection includes 400 million geolocated objects, 50 billion data fields, 40 million polylines, 120 million polygons, including: all US and Canada roads, the US Census demographic and socioeconomic datasets, 110 million parcels with property lines and ownership data, 15 million records of businesses with company stats and management roles and contacts, 2 million physicians with expertise detail, various public place databases (including the USGS GNIS and NGA GNS), Wikipedia, extensive global environmental data (including daily feeds from NASA and NOAA satellites and the USGS water gauges), and hundreds of other datasets.

TerraFly's *TimeSeries* browser and its *Rooftop Geocoder* are especially important for disaster mitigation applications. We are leveraging our NSF Cluster Exploratory award to experiment with using MapReduce techniques to enhance performance. These research areas are detailed below.



The *TerraFly TimeSeries* application is a web browser application with the distinct feature of comparing aerial and satellite images of the same geographical location, but acquired at different times. TerraFly Timeseries is able to animate the display of these images. We successfully implemented TimeSeries as described below.

First, the TimeSeries application receives the Latitude and Longitude of current TerraFly map center from the TerraFly API. Timeseries then sends the coordinates to the TerraFly imagery source service, called "SO". The SO service then returns the available imagery sources, resolutions and acquisition dates of these available sources back in a customized XML formatted string. TimeSeries parses the XML and sorts the sources by their acquisition date, counts the days between each image acquisition date, recreates the time-line panel, and finds the closest resolution to current map view for each source. With the source names, acquisition date, resolutions and coordinates of the visible area, TimeSeries is able to generate the

URLs needed to request the corresponding imagery. The TerraFly API loads the imagery from the imagery servers and fits them to the viewing area as an overlay. TimeSeries then displays the imagery by fading-in and fading-out the imagery sequence. The Fading-in/out effect is achieved by changing the transparency parameter of these images from low to high (fade-in) or high to low (fade-out), in the TerraFly API every 40 millisecond, 25 frames per second.

One of the most important TerraFly services, and of any mapping system, is the geocoder. There are standard methods to geocode street addresses and they rely on the accuracy of the underlying data and the standardization of the street numbers. In particular, Terrafly needs high precision geocoding in order to perform most of its different tasks. We purchased and are using the First American Parcel Point Nationwide Cadastre data set to produce a high precision geocoder for TerraFly services. This data set is also used to conduct research, related to information retrieval and spatial databases.

The following is a description of the First American Parcel Point dataset and how it is used to implement a precise geocoder, referred to as *Rooftop Geocoder*. The data set contains attributes that include parcel boundaries, parcel centroid, address, Assessor's Parcel Number (APN) and ownership information for selected counties. We found at least 20 million parcels without address information and with incorrect address format so we decided to clean the data, to reduce the size of the data set and to get more accurate results. Having a nationwide parcel data set allowed us to implement precise geocoding, using interpolation of street addresses and string matching algorithms.

We built our Rooftop Geocoder using spatial indexes and data structures that we already use in some of the Terrafly services. To give an example of how a query against the Rooftop geocoder works, we describe the following procedure: when the query comes in, the system geocodes the address with the usual interpolation mechanism, which gives an approximate location of the object. The system then performs a nearest neighbor query using the returned coordinates to retrieve the nearby objects. The system then performs a local search to look for the property. If a record exists in the database, the system obtains its coordinates and those coordinates are returned to the user.

The construction of our spatial keyword index (SKI) may take a substantial amount of time, depending mainly on two factors. The first factor is the size of the database. Let us assume that the database contains *N* objects. Objects' spatial attributes are used to construct SKI's R-tree modified structure (R). Thus, the number of insert operations is *O(N)*. The second factor is the size of the lexicon. We construct a modified inverted file on database textual attributes (SIF). The most expensive operation involves sorting the lexicon. The time to construct SIF is bound by $O(N + V \log(V))$, where *V* is the lexicon size.

We previously proposed one way to parallelize R-tree constructions on MapReduce [2]. This idea is leveraged in the construct of SKI's R data structure. Following these ideas, SKI's inner structures are constructed as a sequence of MapReduce jobs as follows. First, on a given spatial database, R structure is built with two MapReduce pairs as in [2]. The output includes references to R nodes as intermediate data used in the next job. Second, a MapReduce job builds the modified inverted file (SIF) on the database lexicon, considering each object as a document. The MapReduce compound uses the intermediate data generated in the first iteration.

SKI's data structures built on MapReduce, stored remotely in the Cloud, are downloaded to a local host, which serves interactive queries. Queries are processed with a modified version of the search algorithm proposed in [1]. The modification accounts for the fact that SKI data structures are partitioned as a result of parallel constructions; there are as many "smaller" SKI structures as the number of Reducers chosen in MapReduce jobs.

[1] Cary, A., Wolfson, O., Rishe, N.: Efficient and Scalable Method for Processing Top-k Spatial Boolean Queries. To appear in proceedings of SSDBM 2010 conference.

[2] Cary, A., Sun, Z., Hristidis, V., Rishe, N.: Experiences on Processing Spatial Data with MapReduce. In SSDBM, Vol. 5566, pp. 302–319, 2009.