Computers & Geosciences 54 (2013) 66–74

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Study on generation and sharing of on-demand global seamless data—Taking MODIS NDVI as an example

Dayong Shen, Meixia Deng, Liping Di*, Weiguo Han, Chunming Peng, Ali Levent Yagci, Genong Yu, Zeqiang Chen

Center for Spatial Information Science and Systems (CSISS), George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 8 June 2012
Received in revised form
7 November 2012
Accepted 9 November 2012
Available online 19 November 2012

Keywords:
BigTIFF
GDAL
WMS
WCS
On-demand
NDVI
MODIS

ABSTRACT

By applying advanced Geospatial Data Abstraction Library (GDAL) and BigTIFF technology in a Geographical Information System (GIS) with Service Oriented Architecture (SOA), this study has derived global datasets using tile-based input data and implemented Virtual Web Map Service (VWMS) and Virtual Web Coverage Service (VWCS) to provide software tools for visualization and acquisition of global data. Taking MODIS Normalized Difference Vegetation Index (NDVI) as an example, this study proves the feasibility, efficiency and features of the proposed approach.

1. Introduction

NDVI was one of the most successful attempts to simply and quickly identify vegetated areas and their conditions, and has been widely used in agriculture, forestry, animal husbandry, ecology, climate, hydrology and environmental science (Di et al., 1994, 2011; Wade et al., 1994; Grist et al., 1997; Oesterheld et al., 1998; Azar et al., 2008; Lin et al., 2008; Hamel et al., 2009; Zhong et al., 2010; Jain et al., 2011; Fang et al., 2011; Mueller-Warrant et al., 2011; Deng et al., 2011; Spruce et al., 2011; Omute et al., 2012).

Kriegler et al. (1969) were the first to propose NDVI which is calculated by subtracting the red channel from the near-infrared (NIR) channel and dividing their difference by the sum of the two channels i.e.,:

$$NDVI = (NIR - RED)/(NIR + RED) \quad (1)$$

where RED is the red portion of the electromagnetic spectrum and NIR is the near infrared portion of the electromagnetic spectrum.

In general, NDVI values range from −1.0 to 1.0. Daily/weekly/biweekly/other composite NDVI data can be derived using surface reflectances from NASA MODIS, NOAA AVHRR, SPOT-4 VEGETATION and Landsat-7 ETM+ (Table 1). Even though Landsat-7 ETM+ surface reflectances have 30 m resolution, it is challenging to create cloud-free NDVI products from ETM+ data as Landsat-7 orbit repeats every 16 days. In terms of spatial resolution, there are two kinds of AVHRR NDVI data: one from 1982 to present at an 8-km re-sampling grid covering the entire planet, and the other from 1989 to present at a 1-km resolution for the conterminous United States (Prasad et al., 2008). As a major instrument for NASA's Earth Observing System (EOS) missions, MODIS is currently operating on-board the EOS Terra and Aqua spacecraft, launched in December 1999 and May 2002, respectively. There are more than 10 years of global Terra and Aqua MODIS NDVI data sets, respectively, with three nadir spatial resolutions: 250 m, 500 m, and 1 km (Xiong et al., 2009). In terms of data quality, MODIS NDVI data perform better than AVHRR NDVI and Landsat NDVI (Beck et al., 2011). However, existing NDVI data services are generally limited by spatial scales e.g., local boundary segmentation, etc. Moreover, the existence of huge amounts of data as well as the rapid growth of the image data raises serious challenges on disk storage. Taking global 250-m MODIS NDVI images in GeoTIFF format as an example, the size of one image is ~11 G, so it will take $\sim 10 \times 11 \times 365 = 40{,}150$ G disk space to store 10-year daily images. If both Terra and Aqua data are considered, it will take $\sim 40{,}150 \times 2 = 80{,}300$ G disk space to store just daily MODIS NDVI data. Note tile-based source input data in HDF-EOS format are not included yet. Due to the existence of huge amounts of data, we need on-demand data services. Here on-

* Corresponding author. Tel.: +1 703 993 6114; fax: +1 703 993 6127.
E-mail addresses: dayong_shen@yahoo.com (D. Shen), mdeng@gmu.edu (M. Deng), ldi@gmu.edu (L. Di).

0098-3004/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cageo.2012.11.011

Table 1
Features of data that are used for NDVI derivation.

Sensor	RED (μm)	NIR (μm)	Resolution (m)
NASA MODIS	0.62–0.67	0.84–0.88	250
NOAA AVHRR/2	0.58–0.68	0.73–1.10	1090
NOAA AVHRR/3	0.58–0.68	0.73–0.98	1090
Landsat-7 ETM+	0.63–0.69	0.75–0.90	30
SPOT-4 VEGETATION	0.61–0.68	0.79–0.89	1165

demand is relative to on-premise. However, there are some challenges to on-demand services, e.g., requirements of rapid data display, less storage capacity, and efficient data processing functionalities integrated in a Web GIS for data generation and sharing.

The main objective of this study is to generate and share global data taking MODIS NDVI as an example. The web-based data services are expected to provide seamless datasets covering the globe and thus will overcome the spatial limitations of other existing NDVI data services. Moreover, the data services will be on-demand, so it will significantly save data storage space while time efficiency will be still high by applying advanced computing and geospatial technology. The remainder of this paper is organized as the following: Section 2 presents methodology, Section 3 describes implementation and results and Section 4 is summary and outlook.

2. Methodology

The advanced computer and geospatial technology applied in this study make the generation and sharing of on-demand global datasets possible.

2.1. Data processing

The total size of a classic TIFF file is limited to 4 gigabytes, i.e., 2^{32} bytes (Aperio, 2011). In this study, the size of a global 250-m MODIS NDVI images is ~11 G, so newer, more advanced technology is necessary. BigTIFF file format and Geospatial Data Abstraction Library (GDAL) APIs (Open Source Geospatial Foundation, 2012) are utilized for creating global seamless image data in this study.

2.1.1. BigTIFF and GDAL

The Tagged Image File Format (TIFF) is a widely accepted standard, supported by many applications on a wide range of platforms due to its capabilities and flexibility.

BigTIFF closely resembles TIFF. However, by using 64 bit offsets, BigTIFF files are no longer restrained by the 4-gigabyte limitation from which classic TIFF using 32 bit offsets suffers. The total size of a classic TIFF file is limited while for a BigTIFF file there is no practical limit to the size of a file (Aperio, 2011).

Table 2 compares the classic TIFF file header and the BigTIFF file header. Note the last members in both variants of the structure point to the first Image File Directory (IFD). This IFD can be located anywhere in the file and every page in a multi-page TIFF or BigTIFF is represented by exactly one IFD.

The libtiff library is an open-source cross-platform library which enables applications to read and write images stored in TIFF files. Since libtiff version 4.0, the library supports processing BigTIFF file format.

GDAL is a translator library released by the Open Source Geospatial Foundation for processing raster geospatial data formats. The latest version is GDAL 1.9.2. When built with internal libtiff (4.0 or above), GDAL can support reading and writing BigTIFF files. With the command -co "BIGTIFF=YES", GDAL also allows creating BigTIFF files.

Moreover, GDAL comes with a variety of useful commandline utilities for data processing (Open Source Geospatial Foundation, 2012). For example, gdalinfo is used for reporting information about a file; gdal_translate is used for copying a raster file, with control of output format; gdaladdo is used for adding overviews to a file; gdalwarp is used for warping an image into a new coordinate system and gdal_merge.py is used for building a mosaic from a set of images. In this study, experiments show that it is very time consuming when using gdal_merge.py to mosaic tile-based global images. Therefore, we developed a customized tool based on C–C++ platform and the mosaic speed has been significantly accelerated after the tool is utilized. The pseudo code of the customized tool is as follows:

Step 1: Create a global new image.
Step 2: Start a For loop.
Step 3: Load tile-based HDF-EOS files.
Step 4: Extract layers of data from the HDF-EOS files.
Step 5: Image mosaic by writing data into the global image tile by tile.
Step 6: End the For loop.

As shown in Fig. 1, the source tile-by-tile MODIS data are available in HDF-EOS format. For each tile, there is local boundary. The boundary can be eliminated by global mosaicking using BigTIFF and GDAL technology. In terms of data visualization, the global image data is seamless and thus overcomes the limitation of local boundary segmentation of other existing NDVI data services.

2.1.2. Optimization

Both running speed and allocated memory can be optimized to enhance the program performance.

(1) **Code optimization**
   (1) Apply unsigned int instead of int if the value will never be negative since processors can handle unsigned integer arithmetic considerably faster than signed integer arithmetic (Ghosh, 2004).
   (2) Replace the division by a multiplication. The division function takes a constant time plus a time for each bit to divide. Time (numerator/denominator)$=C0+C1\times\log_2$ (numerator/denominator)$=C0+C1\times(\log_2$ (numerator)—$\log_2$ (denominator)). As an expensive operation, it is desirable to avoid the division where possible. For example, $(a/b)>c$ can be rewritten as $a>(c\times b)$ if it is known that b is positive and $(b\times c)$ fits in an integer.
   (3) Use the most appropriate type for variables, since it can reduce code and data size and increase performance considerably.
   (4) Apply simple conditional execution. Conditional execution is disabled for code sequences which contain function calls. It is therefore beneficial to keep the bodies of if-and-else statements as simple as possible, so that they can be conditionalized. Relational expressions should be grouped into blocks of similar conditions.
   (5) Utilize fast loops and inline functions. Loops are a common construct in most programs; a significant amount of the execution time is often spent in loops. It is therefore worthwhile to optimize time-critical loops. Meanwhile, even though the code sizes increase if an inline function is used, there is no function call overhead and the overhead