A DISTRIBUTED LIDAR PROCESSING MODEL BASED ON OWS AND BPEL

Lingjun Kang

Qunyong Wu, Ying Yuan

Bibliography: OGC Web Service, workflow, LIDAR processing

1. PROBLEM

LIDAR (Light Detection and Ranging [1] has gained more and more popularity for its advantage in detail DEM construction, 3D surface feature building and topographic parameter extraction. Nevertheless, current LIDAR processing system is centralized. The drawback of it is obvious. First of all, shorter acquisition circle and larger volume point cloud dataset require excellent computing and storage ability for efficient post-processing. Secondly, from perspective of software engineering, workflow-like processing procedure demands of processing step separation for better maintenance and reuse. Thirdly, universal interface for LIDAR data is necessary for lowering public threshold of accessing LIDAR data. To tackle problems mentioned above, this paper proposes distributed LIDAR processing model, which abstracts LIDAR processing procedure into sequential steps, consequently, realizes these steps with GRASS module and encapsulates them with OWS (OGC Web Service) [2]standard interface, finally, chains OWS with de facto workflow language BPEL[3].

2. METHODOLOGY

2.1. LIDAR process procedure

In this paper, we generate bare ground DEM (no building and vegetation) from LIDAR raw point cloud data and generate DSM (Digital Surface Model) from LIDAR multiple return data which includes building and vegetation. Additionally, we extract topographical parameter (aspect and slope) from generated DEM. Finally, generated DEM, DSM, aspect graph and slope graph are accessible to public through internet. The whole procedure is illustrated in Fig 1 and Fig 2:

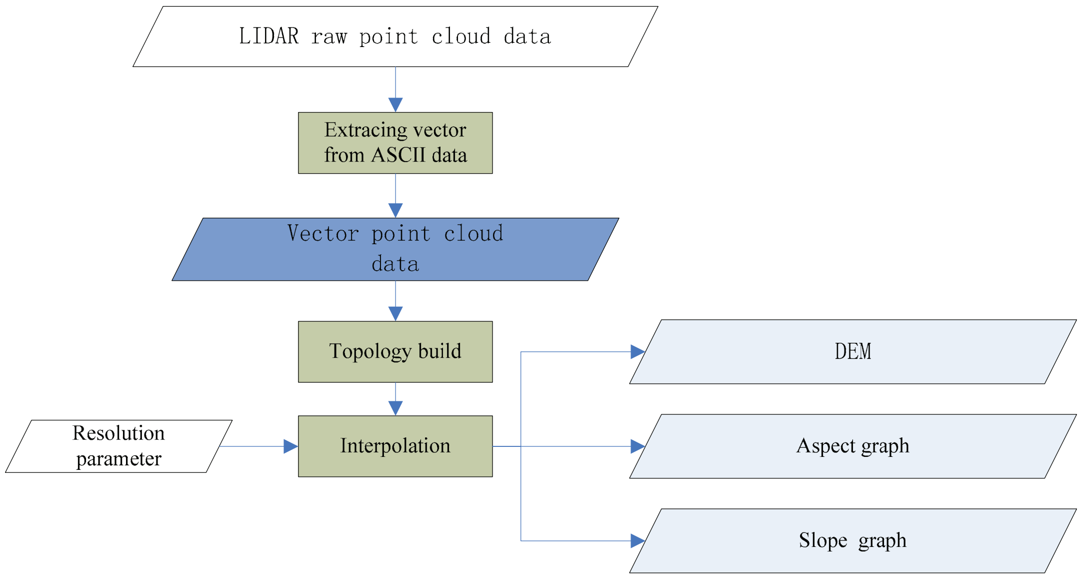

Fig 1

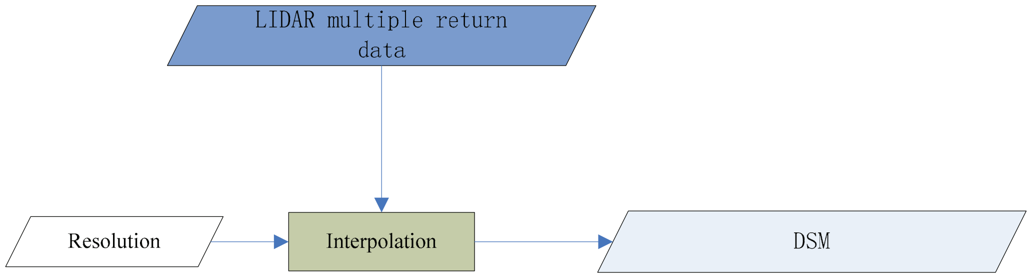

Fig 2

In above figure, white parallelogram denotes literal or ASCII format data in procedure. Deep blue parallelogram denotes vector data type. Cambridge blue denotes raster data type. There data acts as either input and output of procedure or intermediate data within procedure. The brown polygon denotes function which is to be realized. These functions are steps in procedure.

2.2. Leveraging GRASS for module realization

GRASS [4] is popular open-source desktop GIS system. It covers both common GIS module (such as vector process module and raster process module) and specific function extension (for instance LIDAR process module). GIS module in current version (GRASS6.4svn) can satisfies LIDAR process procedure from DEM and DSM generation to topographic parameter (aspect, slope) extraction.

GRASS module is written in C or shell script. To retrieve it through internet, a middleware which can not only communicate C module or shell script but also be accessed through internet is required. In the paper, we apply PyWPS [5] for this middleware. PyWPS is a framework written in Python, which makes desktop GRASS module accessible through internet. The architecture of PyWPS and GRASS module is shown in Fig 3.

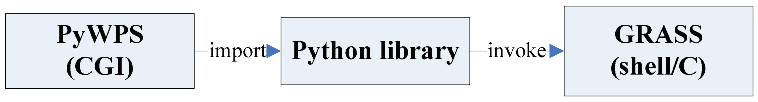

Fig 3

As shown in figure above, PyWPS is realized as CGI in application server (Tomcat 6.5.1) which is responsible for interacting with client request and response. The communication with GRASS module is accomplished by python library in PyWPS. The library translates client request to command line to invoke GRASS module.

2.3. Encapsulating service with OWS standard interface

OGC dedicates in standardizing OGC web service. In the scenario of LIDAR processing, web service in it is categorized into two kinds: process-central service and data-central service. Process-central service realizes process module of each step in workflow. We encapsulate this kind service with WPS (Web Processing Service). Data-central service serves as data retrieving. Both raw LIDAR point cloud dataset and process result are encapsulated as OWS (WFS for feature data, WMS for raster data). Additionally, we choose open-source GeoServer for data-central server. The strategy of OWS encapsulation is shown in Fig 4:

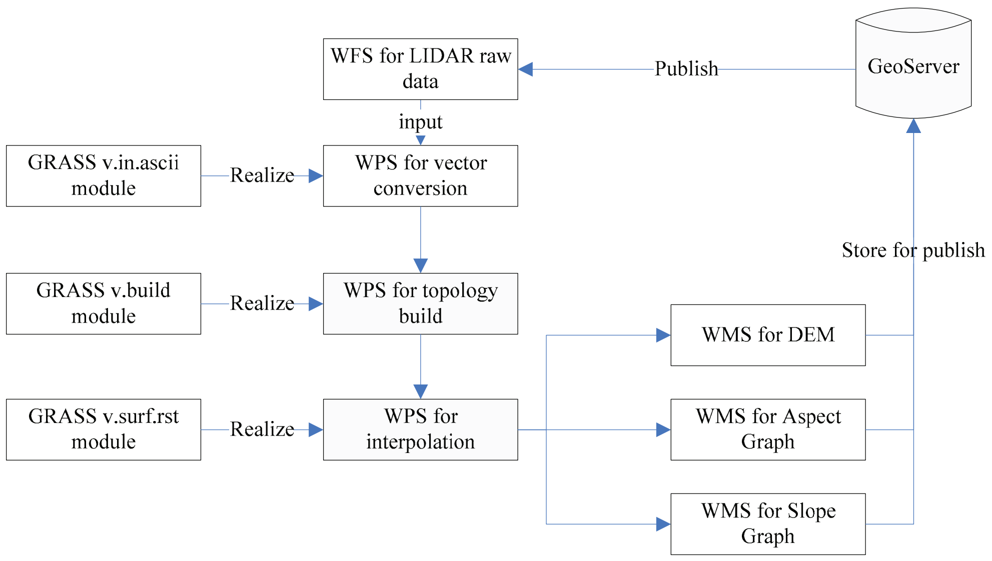

Fig 4

According to above figure, we realize three WPS which is responsible for vector conversion, topology build and interpolation respectively. These three WPS can be realized by three GRASS function module. The raw LIDAR point cloud dataset is stored in GeoServer and published as WFS, which acts as input data for WPS. Accordingly, three process results are stored in GeoServer and published as WMS for public browsing.

2.4. Chaining OWS with BPEL

BPEL acts as de facto workflow language in SOA. Its advantages lie in powerful control flow representation, multiple transportation protocol supporting and well orchestration engine supporting. The thinking in BPEL is to abstract genetic workflow. Though every step in LIDAR process procedure varies in either implementation or provider, the workflow chaining them is changeless and reusable. For workflow in LIDAR process is mature and changeless, we utilize BPEL to represent genetic LIDAR process procedure. Besides, each step is represented as activity element in BPEL which binds to URL address of OWS. we choose open-source Apache ODE as orchestration engine. The architecture of BPEL and OWS is shown in Fig 5.

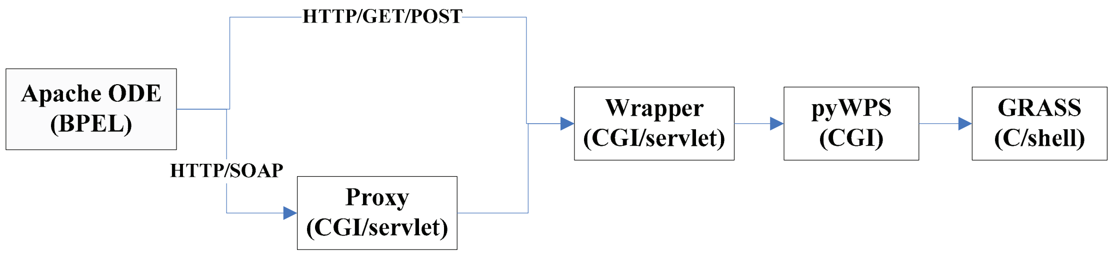

Fig 5

According to WS-BPEL2.0, BPEL communicates with service on HTTP protocol through both GET/POST way and SOAP way. For PyWPS does not support SOAP invoking directly, a proxy should be designed to extract execution command from SOAP message. Furthermore, a wrapper is also required to convert execution XML to supported data type in PyWPS.

3. CONCLOSION

Shifting centralized processing model to distributed processing model leverages computing and storage ability of node in internet. For expert user, it only needs a light-weight client to upload initial input data and sends workflow execution request. Furthermore, expert user is not necessarily worried about CPU or hard-disk limitation of PC. These resources are distributed in every service node on internet.

For LIDAR process procedure maintainer, the focus is shifted from maintaining whole workflow to every service node. In other words, change in any service node can not have effect on whole process procedure.

Distributed process model also lowers public threshold of LIDAR data processing and retrieving. For public user, it is easier to access LIDAR data through light-weight client instead of traditional desktop process system.

4. REFERENCES

[1] http://en.wikipedia.org/wiki/LIDAR
[2] http://www.opengeospatial.org/
[3] http://www.oasis-open.org/committees/wsbpel/
[4] http://grass.itc.it/
[5] http://pywps.wald.intevation.org/