Introduction to Web-based Geoprocessing

Geospatial Sensor Web Conference 2017
Benjamin Pross, Benedikt Gräler, Christoph Stasch
52° North GmbH
Motivation

• Previous Introduction:
  - Standardized access to observations/sensor metadata

• Now:
  - How to process data? How to generate maps/statistics, etc.?
  - How to exchange/share geoprocessing tools among a group/company/globally?
Agenda (i)

- Introduction/Overview:
  - OGC Web Processing Service
  - 52° North Web Processing Service
  - Clients
  - Tools for statistical analysis
  - Project Examples
Agenda (ii)

• Demo: Spatial interpolation of air temperature observations
  - 52N WPS process
  - ArcGIS WPS client
  - WPS JS
  - SOS4R/Rstudio/sensorweb4R/Shiny
Standardized Geoprocessing - Why and how?
Motivation

OGC Web Processing Service

*Standardized Web Service Interface*
- Description and Execution of Geoprocessing Tools
- Common definition of inputs and outputs

Desktop Apps

Web Apps

SDIs, SOA, ROA

Model Implementations

Geoprocessing Tools/Libraries, Large Datasets, Cloud Environments

Processing Scripts, e.g. Python/R
Standardized Geoprocessing - Why and How?

- Data is getting bigger
- Massive processing power on standalone servers/in the cloud
- Are you going to process huge datasets on your desktop computer?
Standardized Geoprocessing - Why and How?

• Process geodata using Web services
• Process the data where it is stored
• Generate views on huge datasets
• Visualize/Download the final result
Standardized Geoprocessing - Why and How?

- Standardized service interface?
  - easier to integrate into several components
  - Common description of processes with inputs/outputs

- Client software can connect to different services

- Processing logic can be easily re-used
OGC Web Processing Service
OGC Web Processing Service Standard

- **WPS**
  - Version 2.0 published 2015

- **Operations**
  - GetCapabilities
  - DescribeProcess
  - Execute
  - GetStatus
  - GetResult

- **Access via KVP or POX**

http://www.opengeospatial.org/standards/wps
OGC WPS - Conceptual Model

(Source: WPS Specification, p. 15)
OGC WPS - Process Model

(Source: WPS Specification, p. 18)
WPS - Steps to execute a process

Clients
- Desktop Apps
- Web Apps
- SDIs, SOA, Big Data

WPS
- GetCapabilities
  - List of processes
- DescribeProcess
  - Detailed process description
- Execute
  - Processing result

SimpleBuffer Algorithm

Process 1

Process n
Synchronous vs. asynchronous execution

(Source: WPS Specification, pp. 21/22)
WPS Operations - GetCapabilities

- GetCapabilities (HTTP GET and POST)
  - General Service information
    - Provider, Contact, Security Infos, etc.
  - List of process summaries (identifier, options)
WPS Operations - DescribeProcess

- DescribeProcess (HTTP GET and POST)
  - Specify process identifier
  - Retrieve detailed process information:
    - identifier
    - textual description
    - inputs/outputs
      - supported data formats
    - execution mode (sync/async)
WPS Operations - Execute

- Execute (HTTP POST)
  - Execute a certain process
  - Specify
    - Identifier
    - inputs/outputs (reference/value) with data formats
    - execution mode
  - Returns
    - job identifier in async mode
    - Results in sync mode
WPS Operations - GetStatus

- GetStatus (HTTP GET and POST)
  - Get status information for processes running in async mode
  - Specify job identifier
  - Returns
    - Status of process:
      - Process accepted
      - Running
      - failed/succeeded
    - Estimated percentage of completion
WPS Operations - GetResult

- GetResult (HTTP GET and POST)
  - Specify job identifier
  - Get the processed result
    - Results can either be retrieved
      - In the response (inline)
      - Or by reference, e.g. through a dataset in an SOS
52° North Web Processing Service
52N WPS - General Overview

- Full java-based Open Source implementation

- [https://github.com/52North/WPS/releases](https://github.com/52North/WPS/releases)

- Releases available
  - 3.6.1 → stable
  - 4.0.0 → beta
  - javaPS → planned for end of year
52n WPS - Version 3.6.1

- Version 3.6.1 (stable)
  - Supports all features and operations of WPS specification version 1.0.0 (document 05-007r7)
  - Build upon robust libraries (JTS, GeoTools, XMLBeans, Servlet API, Derby)
  - Web GUI to maintain the service
  - Result handling:
    - Raw (binary) data
    - Stored in WPS directly
    - Stored in GeoServer W*S
52n WPS - Version 3.6.1

- Version 3.6.1 (stable)
  - Supported data formats (selection)
    - GeoTiff
    - NetCDF
    - GML 2 and 3
    - Shapefiles
    - KML
    - WKT
    - (Geo-)JSON
52n WPS – Version 3.6.1

- Version 3.6.1 (stable)
  - Shipped process repositories:
    - WPS4R - R Backend
    - GRASS 7
    - 220+ SEXTANTE Processes
  - Upload new processes
    - Java classes
    - R scripts
52n WPS - Version 4.0.0/javaPS

- Version 4.0.0 (beta)
  - WPS 1.0.0 and 2.0 support
  - New admin application
  - Used in current projects

- javaPS (next generation geoprocessing framework)
  - WPS 2.0 support
  - Built from scratch
  - State-of-the-art technologies
  - Used in upcoming projects
WPS4R

- Wraps annotated R-Scripts in a WPS process
- Flexible re-use/exchange/standardized statistics
- Full power of R „hidden“ behind a toolbox like application
WPS Clients/Client Libraries
wps-js library

- **wps-js**
  - JavaScript API
  - Functions for Web-based access to WPS 1.0.0 and 2.0 servers

- [https://github.com/52North/wps-js](https://github.com/52North/wps-js)
wps-js client

- **wps-js-client**
  - Visual JavaScript client application based upon wps-js
  - Map area for feature creation and result visualization

- **https://github.com/52North/wps-js-client**
WPS Client for ArcGIS

- WPS Client for ArcGIS
  - Extension for ArcMap 10.x
  - Wraps WPS processes in standard ArcGIS Geoprocessing tools

- [https://github.com/52North/wpsclient4arcgis](https://github.com/52North/wpsclient4arcgis)
Statistical Computing

- **SOS4R:**
  - Full implementation of the SOS standard in R

- **sensorweb4R:**
  - Lightweight REST-API based access to the SOS

- **Shiny**
  - Framework to design interactive webpages with R
  - Visualize data/statistics in combination with SOS (REST) clients
    - http://pilot.52north.org/shinyApps/geo-bridge-stats/real-time-stoch-hydro/
Project Examples
Project Example: TaMIS
Project Example: TaMIS

Client developed by Christian Malewski (Wupperverband)
Project Example: OGC-Testbeds

- Innovation Program
- Requirements and Use Cases defined by Sponsors
- Test how OGC Web services perform in real-world scenarios
- Emphasis on how good OGC services can work together
Project Example: OGC-Testbeds

- 52n WPS was used in all Testbed since OWS-4
- Testbed-11/-12
  - WPS 2.0 was used for data fusion (conflation) and to execute legacy modelling software
  - REST binding for WPS was drafted
Project Example: OGC-Testbeds

- Currently Testbed-13
  - WPS 2.0 is used as workflow engine and for different workflow steps
  - Transactional WPS is explored
Open/future feature requests

- Tight coupling with SOS → SOPS?
- Tight coupling with statistical tools, i.e. R/RStudio Server/Shiny/visualization of complex results (interactive 3D)
- Implementation of security tools
- Geoprocessing via „micro“ services
  - Wrapping of micro services, implementation of micro services
- Provision of „standard“ processing tools
- Transactional WPS → deploying processing in different environments…
Demo
Dataset Description

- Data collected by the Wupperverband
- Stored and provided via SOS:
  - Fluggs: http://fluggs.wupperverband.de/sos2/
- Different parameters of the waterbody (discharge, fill level, ...) and air (temperature, humidity)
- Measured sub-hourly
- Aim: obtain maps of daily mean temperatures in January 2013
Spatial Interpolation of air temperature observations

- Aggregation: from sub-hourly to daily means
- Regression: regression on coordinates
- Interpolation: kriging of residuals
- Output as geotiff with one band/day
More information:

http://52north.org

Benedikt Gräler  
b.graeler@52north.org

Benjamin Proß  
b.pross@52north.org

Christoph Stasch  
c.stasch@52north.org
Acknowledgements

Parts of this work have been funded by the German Ministry of Research and Education under Grant 03G0854[A-D].