GeoServer Focus Group Final Report

Yong Wook Kim

Jong Lee

Rob Kooper

Todd Nicholson

Maxwell Burnette

Chen Wang

GeoServer

- an open source software server written in Java that allows users to share and edit geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards.
- serves data using standard protocols established by the Open Geospatial Consortium

Table of Contents

GeoServer Deployment Options (Docker, Kubernetes)

Native

- basic option https://docs.geoserver.org/stable/en/user/installation/index.html
- install GeoServer package and create standalone server.

Docker

- Here is the link to geoserver docker instructions: https://opensource.ncsa.illinois.edu/confluence/display/CATS/Deploying+Geoserver
- pull request for adding yml files : https://github.com/clowder-framework/clowder/pull/204
- run geoserver and extractors in docker.

Kubernetes

- deploy by creating deployment, service, and ingress
- use helm chart created by Rob Kooper https://artifacthub.io/packages/helm/ncsa/geoserver

GeoServer Services

services are the primary way of supplying geospatial information

- Web Map Service (WMS)
- Web Feature Service (WFS)
- Web Coverage Service (WCS)
- Web Processing Service (WPS)
- Web Map Tiling Service (WMTS)
- Catalog Services for the Web (CSW)

Web Map Service (WMS)

- supports requests for map images (and other formats) generated from geographical data.
- provides a standard interface for requesting a geospatial map image.
- clients can request images from multiple WMS servers, and then combine them into a single view for the user.
- images can all be overlaid on one another as they actually would be in reality.
- GeoServer supports WMS 1.1.1, the most widely used version of WMS, as well as WMS 1.3.0.
- PNG, PNG8, JPEG, JPEG-PNG, JPEG-PNG8, GIF, TIFF, TIFF8, GeoTIFF, GeoTIFF8, SVG, PDF, GeoRSS, KML, KMZ, OpenLayers, UTFGrid
- Example

    - https://<geoserver_url>/geoserver/incore/wms?SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&FORMAT=image%2Fpng&TRANSPARENT=true&tiled=true&name=tiledLayer&STYLES=&LAYERS=<layer_name>&WIDTH=256&HEIGHT=256&SRS=EPSG%3A4326&BBOX=-90%2C34.98046875%2C-89.82421875%2C35.15625

Web Feature Service (WFS)

- supports requests for geographical feature data (with vector geometry and attributes).
- defines the framework for providing access to, and supporting transactions on, discrete geographic features
- users have access to the source spatial and attribute data in a manner that allows them to interrogate, style, edit (create, update, and delete), and download individual features.
- requests
    - GetCapabilities
        - request to a WFS server for a list of the operations and services, or *capabilities*, supported by that server
    - DescribeFeatureType
        - requests information about an individual feature type before requesting the actual data.
        - request a list of features and attributes for the given feature type, or list the feature types available.
    - GetFeature
        - returns a selection of features from the data source
        - output formats: GML2, GML3, Shapefile, JSON, JSONP, CSV
    - examples
        - Filtering for selection
            - https://<geoserver_url>/geoserver/<workspace>/ows?service=WFS&version=1.0.0&request=GetCapabilities&CQL_FILTER=<field_name>=%27<value>%27
        - Output as Json Format
            - https://<geoserver_url>/geoserver/<workspace>/ows?service=WFS&version=1.0.0&request=GetCapabilities&CQL_FILTER=<field_name>=%27<value>%27&outputFormat=JSON
        - Output as Shapefile
            - https://<geoserver_url>/geoserver/<workspace>/ows?service=WFS&version=1.0.0&request=GetCapabilities&CQL_FILTER=<field_name>=%27<value>%27&outputFormat=shape-zip

Web Coverage Service (WCS)

- supports requests for coverage data (rasters)
- provides a standard interface
    - how to request the raster source of a geospatial image.
    - return an image only used as an image
- results can be used for complex modeling and analysis
    - contains more information.
- allows more complex querying
    - can extract just the portion of the coverage.
    - output formats
        - JPEG, GIF, PNG, Tiff, BMP
        - GeoTiff, GTopo30, ArcGrid, Gzipped ArcGrid
- requests
    - GetCapabilities
        - request a list of what operations and services (“capabilities”) are being offered
    - DescribeCoverage
        - request additional information about a Coverage
        - returns information about
        - crs, the metadata, the domain, the range and the formats.
    - GetCoverage
        - requests the actual spatial data.
        - can retrieve subsets of coverages
        - result can be either the coverage itself or a reference to it.
        - most powerful thing is an ability to subset domains (height and time) and ranges.
        - can also do resampling, encode in different data formats, and return the resulting file in different ways
- examples
    - GetCapabilities
        - https://<geoserver_url>/geoserver/<workspace>/ows?service=WCS&version=2.0.0&request=GetCapabilities
    - DescribeCoverage
        - http://<geoserver_url>/geoserver/<workspace>/ows?service=WCS&version=2.0.0&request=DescribeCoverage&coverageId=<coverage_id>
    - GetCoverage
        - http://<geoserver_url>/geoserver/<workspace>/ows?service=WCS&version=2.0.0&request=GetCoverage&coverageId=<coverage_id>

Web Processing Service (WPS)

- an OGC service for the publishing of geospatial processes, algorithms, and calculations
- available as an extension for executing operation for data processing and geospatial analysis.
- results of a process can be stored as a new layer in the GeoServer catalog.
- WPS acts as a full remote geospatial analysis tool, capable of reading and writing data from and to GeoServer.
- https://docs.geoserver.org/stable/en/user/services/wps/index.html
- requests
    - GetCapabilities
    - http://<geoserver_url>/geoserver/ows?service=WPS&version=1.0.0&request=GetCapabilities
    - DescribeProcess

    - Execute

Web Map Tiling Service (WMTS)

- an OGC standard currently undergoing ratification
- Geoserver has the ability to proxy a remote Web Map Tile Service (WMTS)
- Loading a remote WMTS is useful for many reasons.
- If you don't manage or have access to the remote WMTS, you can now manage its output as if it were local.
- Even if you don't have any control on the remote WMTS, you can use GeoServer features to treat its output (watermarking, decoration, printing, etc).

Catalog Services for the Web (CSW)

- supports retrieving and displaying items from the GeoServer catalog using the CSW service
- requests
    - GetCapabilities
    - GetRecords
    - GetRecordById
    - GetDomain
    - DescribeRecord
- examples
    - GetCapabilities
        - https://<geoserver_url>/geoserver/ows?service=csw&version=2.0.2&request=GetCapabilities
    - GetRecords
        - https://<geoserver_url>/geoserver/ows?service=CSW&version=2.0.2&request=GetRecords&typeNames=gmd:MD_Metadata&resultType=results&elementSetName=full&outputSchema=http://www.isotc211.org/2005/gmd
    - GetRecordById
        - https://<geoserver_url>/geoserver/ows?service=CSW&version=2.0.2&request=GetRecordById&elementsetname=summary&id=<dataset_id>&typeNames=gmd:MD_Metadata&resultType=results&elementSetName=full&outputSchema=http://www.isotc211.org/2005/gmd
    - GetDomain
        - https://<geoserver_url>/geoserver/ows?service=csw&version=2.0.2&request=GetDomain&propertyName=Title
    - DescribeRecord
        - https://<geoserver_url>/geoserver/ows?service=CSW&version=2.0.2&request=DescribeRecord&typeName=gmd:MD_Metadata

Basic manipulation using Python

Gsconfig-py3

- https://pypi.org/project/gsconfig/
- python3 library for manipulating a GeoServer instance via the GeoServer RESTConfig API
- installation
    - Pip
        - pip install gscofig-py3
    - Manual installation

```
git clone git@github.com:boundlessgeo/gsconfig.git
cd gsconfig
python setup.py develop
```

- catalog
    - Set up catalog

```
from geoserver.catalog import Catalog
cat = Catalog("http://geoserver_url/geoserver/rest/", "admin", "geoserver")
note /rest/ at the end of the url
```

- workspace

```
# get all the workspaces
cat.get_workspaces()

#filter which workspace to return
cat.get_workspace("any workspace name")

#filter which stores to return
cat.get_store("store name", "workspace name")
```

- upload shapefile

```
# construct geoserver catalog
cat = Catalog(gs_url, gs_user, gs_pw)

# set workspace, if the workspace is not there, it has to be created
worksp = cat.get_workspace(gs_workspace)

# create the list object related to shapefile (including shp, prj, shx, dbf)
shapefile_plus_sidecars = gs_util.shapefile_and_friends(filename)

# upload shapefile
ft = cat.create_featurestore(filename, shapefile_plus_sidecars, worksp)
```

- upload raster

```
# construct geoserver catalog
cat = Catalog(gs_url, gs_user, gs_pw)

# set workspace, if the workspace is not there, it has to be created
worksp = cat.get_workspace(gs_workspace)

# upload raster data
cat.create_coveragestore(out_name, in_tif, worksp)
```

Using GeoPackage in GeoServer

GeoPackage

- https://www.geopackage.org/
- The GeoPackage Encoding Standard describes a set of conventions for storing the following within an SQLite database:
  - vector features
  - tile matrix sets of imagery and raster maps at various scales
  - attributes (non-spatial data)
  - Extensions
- OGC GeoPackage specification 1.0.1 published in year 2015
- Latest version 1.3.0 in year 2020 (https://www.geopackage.org/spec130/index.html)

Implementations

- GDAL
  - Vector Features and Tiles (raster) since v2.0
- QGIS
  - Vector Features (read/write) since 2.10.1
  - Tiles(read)since2.18
- Geoserver
  - GeoPacakge plugin
  - Handle both Vector Features and Tiles
- ESRI ArcGIS
  - Vector features (since ArcGIS 10.2.2)
  - Tiles (since ArcGIS 10.3
- GeoTools
  - Vector Features and Tiles (raster) since 11.0
  - Recently added GeoPackage R-Trees
- NGA open source lib
  - http://ngageoint.github.io/GeoPackage/
  - OGC certified
- https://www.geopackage.org/implementations.html

A Quick Comparison

ESRI Shapefile	GeoPackage
Multiple files (.shp, .shx, .dbf, .prj)	Single file
Limitation because of DBF (10 ch length for column names)	No limitations like DBF
1 Shapefile has1 Feature Type (Road.shp has “road” feature type)	1 GeoPackage could have multiple Feature Types

Can’t contain Raster data	It contains raster data and other attribute tables
Galveston buildings shapefile (40.7 MB)	Galveston buildings geopackage (3.66 MB)

GPKG Support in Geoserver

- GeoPackage is Core Functionality of Geoserver
  - Vector: https://docs.geoserver.org/latest/en/user/data/vector/geopkg.html
  - Raster: https://docs.geoserver.org/latest/en/user/data/raster/geopkg.html
- GeoPackage Extension (Community plugin)
  - https://docs.geoserver.org/latest/en/user/community/geopkg/
  - The GeoServer GeoPackage extension also allows to create a completely custom made GeoPackage with multiple layers, using the GeoPackage process.

Uploading GeoPackage to Geoserver

- Using Geoserver REST endpoints
- https://docs.geoserver.org/latest/en/user/rest/index.html#rest
- Uploading geopackage (or ESRI shapefile) means creating a store at Geoserver
  - https://docs.geoserver.org/latest/en/user/rest/stores.html

```
curl-v-u<USR>:<PASSWORD> -XPUT-H"Content-type: application/zip"
--data-binary@<PATH&FILENAME.ZIP>http://<HOSTNAME>:<PORT>//geoserver/rest/<WORKSPACE>/datastores/<FILENAME>
/file.shp

curl -v -u <USR>:<PASSWORD> -XPUT -H "Content-type: application/x-sqlite3"
--data-binary @<PATH&FILENAME.GPKG> http://<HOSTNAME>:<PORT>/geoserver/rest/workspaces/<WORKSPACE>/datastores
/<FILENAME>/file.gpkg
```

- Using Jetty Httpclient
- Reason:
  - Current Geoserver manager java library is using Apache Common Http component (old version)
  - If I install another version (latest apache httpclient), there maybe a class loading issues.

```java
public static boolean uploadGpkgToGeoserver(String store, File gpkgFile) {
    String url = GEOSERVER_REST_URL+"/rest/workspaces/" + GEOSERVER_WORKSPACE + "/datastores/"
            + store + "/file.gpkg";
    URI uri = URI.create(url);
    Authentication.Result auth = new BasicAuthentication.BasicResult(uri, GEOSERVER_USER, GEOSERVER_PW);


    HttpClient httpClient = new HttpClient();
    try {
        httpClient.start();
        Request request = httpClient.newRequest(uri);
        request.method(HttpMethod.PUT);
        request.file(gpkgFile.toPath(), "application/x-sqlite3");

        auth.apply(request);
        ContentResponse response = request.send();
        int responseStatus = response.getStatus();
        httpClient.stop();

        if ( (responseStatus == HttpStatus.CREATED_201) || (responseStatus == HttpStatus.ACCEPTED_202)
            || (responseStatus == HttpStatus.OK_200)) {
            return true;
        }
    } catch (Exception e) {
        logger.error("HttpClient error", e);
        return false;
    }

    return false;
```

Using PostGIS in Geoserver

Comparison of adding a shapefile to Geoserver vs. adding a geopackage file created from QGIS vs. adding a Postgres entry (all of same large dataset with 185k+ polygons).

Performance test results:

When loading a large shapefile in entirety (i.e. display everything with no filter), the SHP was generally faster. PSQL would return an entire state of census blocks in ~900ms while the Shapefile version was around 450ms.

However, with any sort of spatial or property filtering applied, which will likely be the typical case for many projects, PSQL is significantly faster with proper index, ~161ms for a property filter vs. 500-1200ms for Shapefile.

Geopackage was generally the slowest, and I couldn't find an effective means to improve performance. Geopackage does generate a SPATIAL index that could mean good performance in tasks that are filtered by bounding box, but for our typical use case of filtering by property, performance was worse than a shapefile locally.

Repeat each x20.	SHP	Geopackage	PSQL
Full dataset, no filter	450ms	1200ms+	900ms
With filtering (bounding box, property)	500-1200ms	670ms-1200ms	160ms

Presence of indexes in PSQL database is important for performance. Also storing data in db can save disk space because shapefile is not saved in data_dir.

Uploading Styles dynamically

```
curl -u admin:geoserver -XPUT -H "Content-type: text/xml" -d "<layer><defaultStyle><name>GPP7_4km_Dekad<
/name><workspace></workspace></defaultStyle></layer>" http://127.0.0.1:8080/geoserver/rest/layers
/GPP7_4km_Dekad:GPP7_4km_Dekad.2010.04.Early.tif.xml
```

content-type xml isn't enough, you need an Accept header or to append format onto URL. .xml should be appended to end.

GPP7_4km_Dekad is the style name, and `GPP7_4km_Dekad:GPP7_4km_Dekad.2010.04.Early.tif` is the layer name. Both of these exist

Issues Found

Problems confronted in deploying Kubernetes

- Stickiness problem?
  - only single instance works
- Some containers not working with GUI
  - pagination
- Data migration
  - hard coded path in *store.xml
- Loading time when restart
  - check the log level
- Check SRS for the layers

Suggestions

Possible Question from the users
Data size

Loading number of datasets

Raster, geoserver PostGIS raster.
GeoWebcache image pyramid
Data sharing for geoserver and others.
Multiple instances