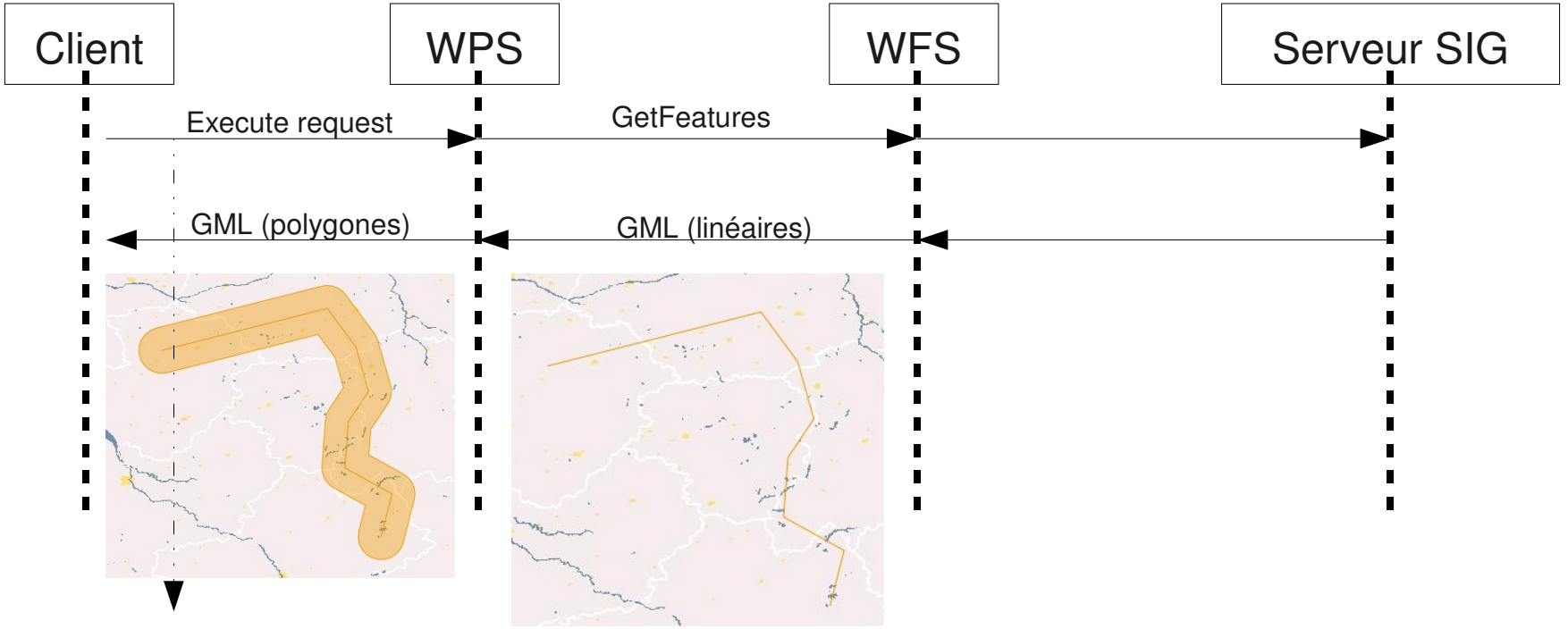
Les Web Processing Services

- Dans la lignée des Web Services de l'OGC :
  - WMS : afficher de la donnée
  - WFS : transmettre de la donnée
  - WPS : traiter de la donnée
- Introduction aux spécifications du WPS :
  « *Un WPS fournit à un client un accès via le réseau à des modèles de calcul qui opèrent sur des données géoréférencées.* »
- Traitements géométriques (union, intersection, buffer...), topologiques, plus court chemin, tournée, traitements raster (visibilité, route...), interpolation... Tous les types de traitements des données géographiques sont concernés.

Applications simples

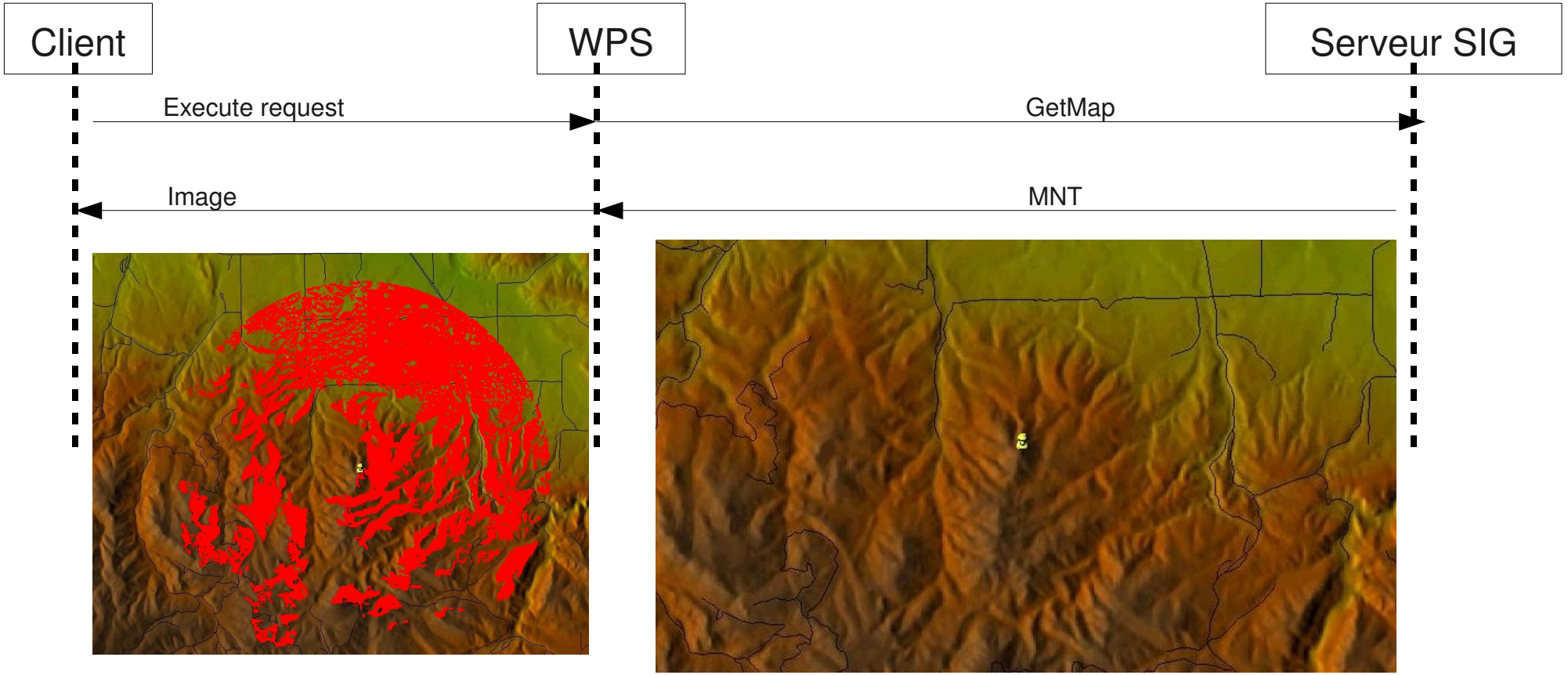
- En WPS, le GetCapabilities liste les traitements utilisables (buffer, visibilité, intersection)
- DescribeProcess : renvoie les détails d'un traitement
- Execute : lance le traitement. Exemple avec le buffer d'un flux WFS



http://serveur.fr/wps?
request="Execute"&
service="WPS"&
version="0.4.0"&
Identifier="Buffer"&
DataInputs= Buffer,http://wfs.serveur.fr/requête_WFS,BufferDistance,100

Applications simples

- Carte de visibilité

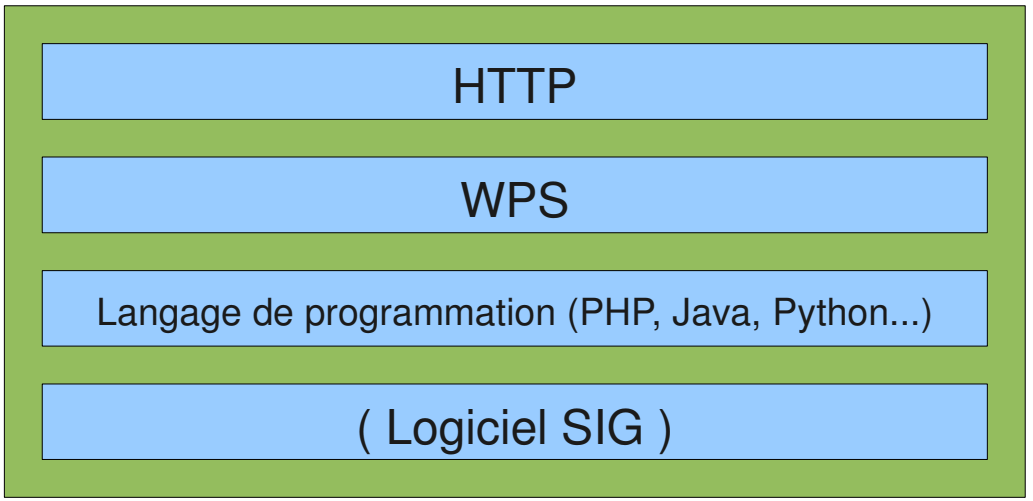


PyWPS

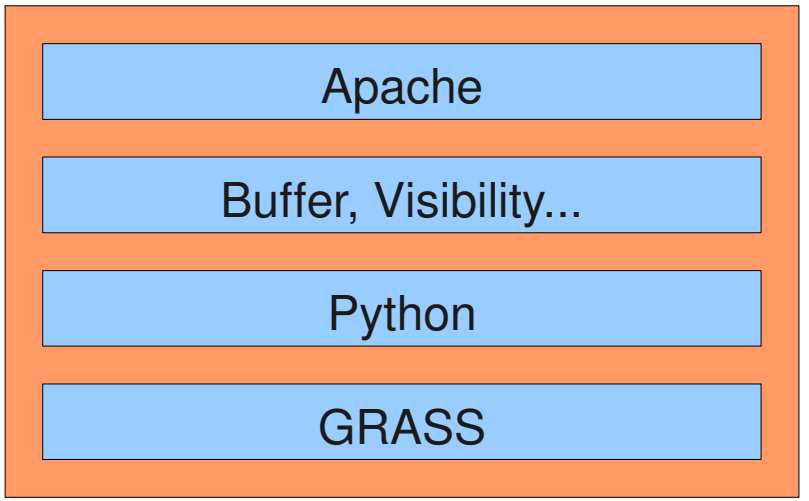
- Voir http://pywps.wald.intevation.org/index.psp
- Projet démarré en Mai 2006, licence GPL
- Implémentation du WPS de l'OGC autour d'un module écrit en Python
- Le module expose les services WPS au client (getCapabilities, Execute...) via une interface SOAP
- Le module pilote GRASS pour générer les résultats.

Architecture WPS

- Les WPS sont des applications SOAP qui encapsulent des traitements SIG classiques réalisés sur un serveur via HTTP.



Principe général



Modules mis en œuvre par PyWPS

Intérêt des WPS

- Déporter sur des serveurs de calcul des traitements complexes, tout en gardant la main sur la donnée depuis le client.
- Mutualiser l'accès à des traitements géographiques complexes.
- Permettre à des clients légers (navigateurs web) de bénéficier des résultats de réelles fonctions SIG.

Inconvénients des WPS

- Pas d'opération par défaut (équivalent du GetMap du WMS) puisqu'on peut lancer des traitements très divers
- Pas de paramétrage par défaut, puisque les paramètres dépendent du type de traitement choisi
- Pas de format par défaut, puisque celui-ci dépend du type de traitement choisi.
- Donc au final, peu de généricité, et une intégration difficile dans un client quelconque.

Lourdeur et verbosité

- Chaque traitement doit être renseigné dans le GetCapabilities, avec ses paramètres et formats.
- Le client doit présenter un formulaire adapté à chaque WPS. uDig en est capable.
- La désignation des services en SOAP est très verbeuse et passe par une nomenclature complexe :

Ici, un exemple de réponse à un DescribeProcess sur un traitement de bufferisation...

Le nombre de lignes de code du module lui-même doit être moindre !

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This example describes a buffer command that accepts polygon
coordinates in GML, and used a buffer distance in meters to produce a
buffered polygon feature, which is output in GML, in either UTF-8 or
base64 encoding. The polygon can be returned directly as output, or
stored by the service as a web-accessible resource. Ongoing processing
status reports are not available. -->
<ProcessDescriptions xmlns="http://www.opengeospatial.net/wps"
xmlns:wps="http://www.opengeospatial.net/wps"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengeospatial.net/wps
..\wpsDescribeProcess.xsd">
   <ProcessDescription processVersion="2" storeSupported="true"
statusSupported="false">
      <ows:Identifier>Buffer</ows:Identifier>
      <ows:Title>Create a buffer around a polygon.</ows:Title>
      <ows:Abstract>Create a buffer around a single polygon. Accepts
the polygon as GML and provides GML output for the buffered feature.
</ows:Abstract>
      <ows:Metadata xlink:title="spatial" />
      <ows:Metadata xlink:title="geometry" />
      <ows:Metadata xlink:title="buffer" />
      <ows:Metadata xlink:title="GML" />
      <DataInputs>
        <Input>
          <ows:Identifier>InputPolygon</ows:Identifier>
          <ows:Title>Polygon to be buffered</ows:Title>
          <ows:Abstract>URI to a set of GML that describes the
polygon.</ows:Abstract>
          <ComplexData defaultFormat="text/XML"
defaultEncoding="base64"
defaultSchema="http://foo.bar/gml/3.1.0/polygon.xsd">
            <SupportedComplexData>
              <Format>text/XML</Format>
              <Encoding>UTF-8</Encoding>
  <Schema>http://foo.bar/gml/3.1.0/polygon.xsd</Schema>
            </SupportedComplexData>
          </ComplexData>
          <MinimumOccurs>1</MinimumOccurs>
        </Input>
        <Input>
          <ows:Identifier>BufferDistance</ows:Identifier>
          <ows:Title>Buffer Distance</ows:Title>
          <ows:Abstract>URI to a GML resource file</ows:Abstract>
          <LiteralData>
            <SupportedUOMs defaultUOM="meters"/>
            <ows:AnyValue/>
          </LiteralData>
          <MinimumOccurs>1</MinimumOccurs>
        </Input>
      </DataInputs>
      <ProcessOutputs>
        <Output>
          <ows:Identifier>BufferedPolygon</ows:Identifier>
          <ows:Title>Buffered Polygon</ows:Title>
          <ows:Abstract>GML stream describing the buffered polygon
feature.</ows:Abstract>
          <ComplexOutput defaultFormat="text/XML"
defaultEncoding="base64"
defaultSchema="http://foo.bar/gml/3.1.0/polygon.xsd">
            <SupportedComplexData>
              <Format>text/XML</Format>
              <Encoding>UTF-8</Encoding>
  <Schema>http://foo.bar/gml/3.1.0/polygon.xsd</Schema>
            </SupportedComplexData>
          </ComplexOutput>
        </Output>
      </ProcessOutputs>
   </ProcessDescription>
</ProcessDescriptions>
```

Des WPS Restful ?

- Le modèle d'architecture REST a été formalisé par Roy Fielding en 2000.
- Il s'agit de considérer le contenu du web comme des ressources auxquelles on accède via des identifiants uniques, les URIs.
- On parle alors de ROA, ou architecture orientée ressources.
- Chaque ressource peut avoir *n* représentations, elle-même éventuellement composées de nouvelles URIs.

Exemple

La France : http://www.monserveur.com/france?f=png

-> renvoie une image PNG de la France

mais http://www.monserveur.com/france?f=xml

-> renvoie une liste d'URIs désignant des sous-ressources, régions, départements par exemple.

Pourtant, la ressource est la même, une table PostGIS par exemple.

De même :

http://www.monserveur.com/france/dept/29?f=png

propose une représentation image en PNG du département du Finistère.

et http://www.monserveur.com/france/dept/29?f=shp&srs=27572

renvoie un shape du finistère en Lambert II étendu.

L'API staticMap de GoogleMaps ne fonctionne pas différement :

http://maps.google.com/staticmap?center=47.15984,
2.988281&zoom=12&size=512x512&key=MAPS_API_KEY

http://maps.google.com/staticmap?center=47.15984,
2.988281&zoom=12&size=512x512&key=MAPS_API_KEY&maptype=mobile

WPS et REST

- L'approche REST supprime un intermédiaire applicatif entre la requête et la ressource : le service. Le service est directement délivré par HTTP.
- http://www.monserveur.com/WPS/

  --> Correspond à

  http://foo.bar/foo?service=WPS&Request=GetCapabilities&AcceptVersions=1.0.0&language=en-CA

  - Les propriétés par défaut peuvent être gérées directement au niveau HTTP.
  - Cette ressource renvoie la liste des URIs des services disponibles
- http://www.monserveur.com/WPS/buffer/

  --> Correspond à :

  http://foo.bar/foo?Service=WPS&Request=DescribeProcess&Version=1.0.0&Language=fr&Identifier=buffer
- http://www.monserveur.com/WPS/buffer?data=http://www.monserver.com/france/dept/29&distance=100

  --> Correspond à :

  http://www.monserveur.com/wps?request="Execute"&service="WPS"&version="1.0.0"&Identifier="Buffer"&DataInputs=Buffer,http://wfs.serveur.fr/requête_WFS,BufferDistance,100

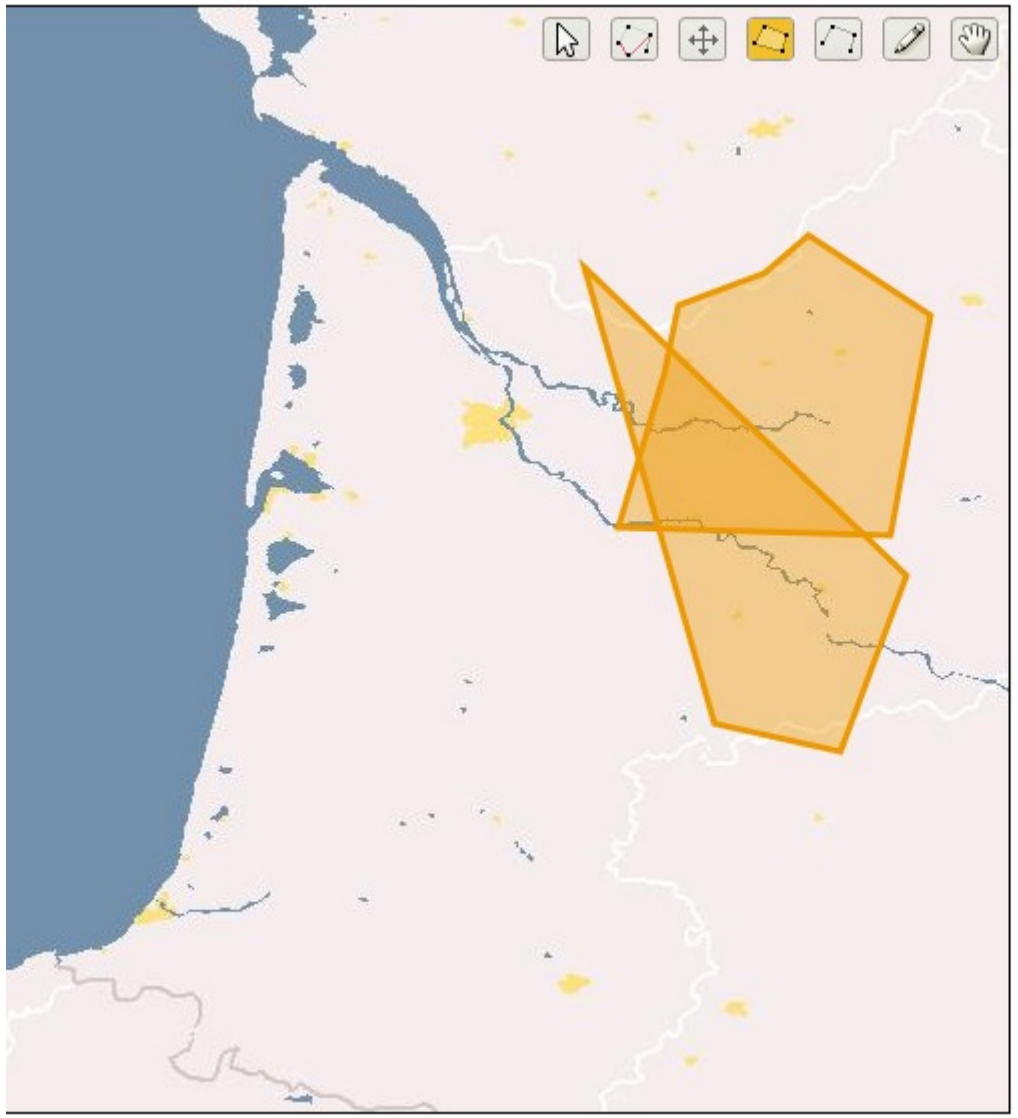
Comment implémenter des WPS RestFul ?

- En utilisant des frameworks web doués pour la cartographie, tels que GeoDjango (http://code.djangoproject.com/wiki/GeoDjango)
- En développent directement les services, basés sur des fonctions de postGIS par exemple.
- En utilisant ArcGisServer ?

WebProcessingServer

- Serveur WPS RestFull OpenSource écrit en Python et utilisant notamment la librairie OpenSource Shapely pour manipuler les données.
- Traite les données au format GeoJSON.
- Interfaçable avec le client web OpenLayers
- Parmi les opérations implémentées : Union, Différence, Intersection, Différence symétrique, Enveloppe, Centroïde, Buffer, Généralisation.
- Voir http://code.google.com/p/webprocessingserver/

Example (http://crschmidt.net/mapping/wpserverdemo/)



WPServer Demo

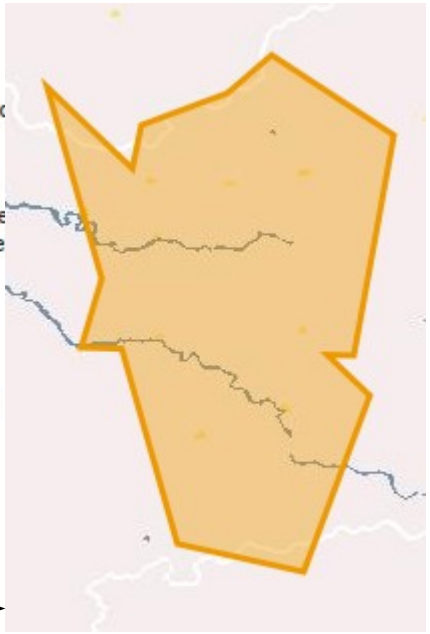
Draw features. Once you're done, choose an action -- it will send a request to the server, and redraw the features.

If you'd like, you can just select a few features, and click the processing button -- if features are selected, they will be processed rather than the whole layer.

As always, you can hold down shift to draw 'freehand', in sketch mode. This is especially useful to see the 'simplify' functionality in action. To stop drawing in freehand mode, just let go of the mouse -- in non-freehand mode, double click to stop.

Powered by OpenLayers and Web Processing Server.

- Multiple Geometry Actions:
  - Dissolve
  - Union
  - Intersection
  - Difference
  - SymmetricDifference
- Per Geometry Actions:
  - Convex Hull
  - Centroid
  - Buffer Size: 2
  - Simplify Tolerance: 1
- Feature Management:
  - FeatureServer data:
    - FS.org
      - Download last 5
      - Download by ID: 1148,1149
  - Save to local FeatureServer



Conclusion

- La donnée géographique devient abstraite car détachée du support physique qui la stocke.
- Elle s'adapte enfin à son contexte d'utilisation (stockage, visualisation, calcul...) par l'intermédiaire de services dédiés à ces usages.
- Cette approche repose essentiellement sur une architecture de services, qui font passer les logiciels clients à un second plan.