Moving Code in SDI: Sharing Geoprocessing Tools with Web Services

Matthias Müller, Lars Bernard, Johannes Brauner
Outline

- Service-based geoprocessing
- “Moving Code” approaches
- “Moving Code” infrastructure & requirements
- Advantages in lifecycle management
- Assessment and outlook
Service-based Geoprocessing (GP)

- **Data-driven (state-of-the-art)**
  - The focus is set on data
  - GP Services provide a static set of spatial operations
  - Data is shipped around

- **Code-driven**
  - The focus is set on code
  - GP Services are supplied ad-hoc with new operations
  - GP Services receive dynamic updates
  - GP Services retrieve algorithms from different producers and locations
  - **Code-driven infrastructures require “moving code” mechanisms**
Data-driven Geoprocessing in SDI

S – Service

t_T – data transportation time

t_S – time for Web Service invocation
The “Code driven” Geospatial Calculator

Alice (Developer)

Bob (User)

Calculator Memory

Custom Constants / Data

Custom Code

Created by Alice

Used by Bob

Ad-hoc data
"Moving Code" for Geoprocessing – A Classification

<table>
<thead>
<tr>
<th>Execution Scheme</th>
<th>Data Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tightly coupled</td>
</tr>
<tr>
<td><strong>Instant execution</strong></td>
<td>Execute(Code)</td>
</tr>
<tr>
<td><strong>Permanent deployment</strong></td>
<td>Deploy(Code)</td>
</tr>
<tr>
<td></td>
<td>Execute() [multiple]</td>
</tr>
</tbody>
</table>
### “Moving Code” for Geoprocessing – A Classification

#### Data Coupling

<table>
<thead>
<tr>
<th>Tightly coupled</th>
<th>Loosely coupled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instant execution</td>
<td>Filter Encoding</td>
</tr>
<tr>
<td>Web Coverage Processing Service</td>
<td>Stored SQL query</td>
</tr>
<tr>
<td>Filter Encoding</td>
<td>(Transactional WPS)</td>
</tr>
<tr>
<td>SQL statements</td>
<td>Grid infrastructures</td>
</tr>
<tr>
<td>Permanent deployment (Transactional WPS)</td>
<td></td>
</tr>
</tbody>
</table>
A Simple “Moving Code” Architecture
Workspaces to Cache Tightly Coupled Data

- Workspaces contain code and some (constant) data
- No need to ship data around that is required in every execution ("caching")
- Workspaces have to be designed to run on each intended platform

Sharing Tools with Workspaces in ArcGIS (Taken from the ESRI Help)
Linking WPS Process Interfaces, Algorithms and Workspaces
Linking WPS Process Interfaces, Algorithms and Workspaces

- We lack a generic Geoprocessing algebra …
  - How to communicate the Algorithm and the Workspace to the Service?
  - The number of Geoprocessing Systems is limited …
  - Can we instead describe the required runtime environment?
  - What about the parameter mapping?

<table>
<thead>
<tr>
<th>AlgorithmDescriptionSchema</th>
</tr>
</thead>
<tbody>
<tr>
<td>+algorithmWorkspaceLocation[1]</td>
</tr>
<tr>
<td>+algorithmExecutableLocation[0..1]</td>
</tr>
<tr>
<td>+algorithmExecutableTypeURN[1]</td>
</tr>
<tr>
<td>+processingSystemURN[1..*]</td>
</tr>
<tr>
<td>+algorithmParameterValue[1..*]</td>
</tr>
</tbody>
</table>

Where to find the Code / Data Package and the executable.
- The required runtime environment.
- The legacy parameters.
Deployment

• Develop algorithm

Process Description
Algorithm Description
Workspace
Life Cycle Management

Static Part

- WPS offers Process
- Process has Process Description
- Process Description references Algorithm Description
- Algorithm Description links to Code / Data Package
- Code / Data Package represented as Workspace

Dynamic Part
Life Cycle Management

- Develop algorithm
- Modify / update algorithm

Execute

- OWS Operator Clients
- Developer Clients

Upload when ready

Holds reference to Algorithm Description

Process Description

Algorithm Description

Workspace

Data Services

Algorithm Registries

Code / Data Packages

WPS Instances

Algorithm Descriptions

Local Workspaces

Reload & update on demand / scheduled
Life Cycle Management

- Develop algorithm
- Modify / update algorithm
Code-driven vs. Data-driven

- ‘Moving code’ approaches are beneficial if:
  - Algorithms are frequently changed and evolved
  - Identical algorithms have to be deployed at or shared among several service instances
  - A substantial amount of data can be shipped with the algorithm and stored prior to execution
  - Some tightly coupled data sets can be used to increase performance (caching impact)
  - Algorithms have to be placed at a processing service that resides “close” to the data (bandwidth impact)
Code-driven vs. Data-driven

- Data-driven approaches are beneficial if:
  - All required data sets change frequently
  - One-time assembly and execution of workflows
  - Real-time response for complex service chains is not required
  - The required atomic operators are available at processing services or
  - The required simple operators are available at the data service level
What’s next?

- Progress in the design of well defined geoprocessing algebras
- Evaluate potential for parallelization
- Create Service Grids
- Evolve the Standards