Creating Geoprocessing Services
Scott Murray, Monica Joseph
Three case studies:

1. Upload data and analyze
2. Analysis with large datasets
3. Using Feature service with Geoprocessing service
Three steps for each case study

- Step 1: Create a Geoprocessing tool in ArcGIS Desktop
- Step 2: Publish the tool as a Geoprocessing service in ArcGIS Server
- Step 3: Create a web application using JavaScript web API.
Upload data and Analyze

1. Upload a CSV
2. Interpolate features
3. Visualize results
1. Create a Geoprocessing tool

<table>
<thead>
<tr>
<th>Geoprocessing tools</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy Rows</td>
<td>Converts CSV to Geodatabase table</td>
</tr>
<tr>
<td>Make XY Event Layer</td>
<td>Creates Feature Layer from Geodatabase table using its Latitude and Longitude columns</td>
</tr>
<tr>
<td>Empirical Bayesian Kriging (EBK)</td>
<td>Creates an Interpolated Raster layer from feature layer.</td>
</tr>
</tbody>
</table>
2. Publishing geoprocessing service
Publishing Workflow

- Run the tool to create Geoprocessing result
- A result is a footprint of a tool’s execution
- Provides a template to create the geoprocessing service such as the default parameter value, schema, output symbology etc.
Publishing Workflow

- **Service Properties**
  - Synchronous vs. Asynchronous
    - Synchronous
    - Asynchronous for long running tasks and Result Map Service
  - Result Map Service
    - Return results as a map service
    - Must have to visualize raster output, large data and advanced symbology
  - Message Level
    - None (default), Error, Warning, Info
  - Uploads Capability
    - Upload local files to server
Publishing Workflow

- Service Properties – Best practices
  - Number of instances per machine = number of cores (**never more than number of cores**)
  - Have at least 2GB memory per instance, more better.
  - If execution time > 10 minutes, increase the max execution time
  - If uploading large files or raster (>45 GB), increase heap size

More Info on Service Properties
- Pooling: [http://esriurl.com/Pooling](http://esriurl.com/Pooling)
Publishing Workflow

- **Task Parameters**
  - **Parameter Name**
    - Update name of the parameter
  - **Required or optional**
    - Optional parameters can be converted to required.
  - **Input parameter value:**
    - Pick one of the three options for user to input:
      - **User Defined**: User provides data
      - **Choice List**: User selects data from a pre-determined list provided by the service
      - **Constant**: Hide the parameter from user and use a constant value
Publishing Workflow

- Analyzes data access for ArcGIS Server
- Verifies the integrity of scripts and models
- Verifies ArcGIS Server License to run the tool

- Creates a new tool based on the parameter configurations
- Copies data to server
- Update data paths in models and scripts
- Creates a web service
3. Create a Web Application

- A geoprocessing service has one or more tasks
- Each task has a REST URL to interact with web applications (http://<server>/arcgis/rest/<gpservice-name>/GPServer/<gptask-name>)
- Geoprocessing tasks can be accessed through all ArcGIS Web APIs in three simple steps
  1. Initialize Tasks
  2. Setup Task parameter
  3. Call the task
  4. Get Results from the task.
Step 1. Initialize task

- Use the task url to create a geoprocessing task.

```javascript
// Step 1: Initialize GP Task
dojo.require("esri.tasks.gp");
var gpTask;

function init(){
    gpTask = new esri.tasks.Geoprocessor
        ("http://<rest-url>/<service-name>/GPServer/<task-name>");
}
```
Step 2: Setup Parameters

- Get values for each input parameters of the task

```javascript
// Step 2: Setup task input parameters
// Accessing files hosted on a web server
var params = { "InputFile":
    {"url":"http://<my-web-server>/inputfile.csv" }
};
```

- How to upload local files?

  1. Upload local files to ArcGIS Server and get id.
  2. Use uploaded file id as input to Geoprocessing tasks

***Upload capability must have been enabled on the service to upload files. Supported in 10.1 & upwards***
Uploading Local Files to ArcGIS Server

Upload url for Geoprocessing Services

Example: http://<gp-service-url>/uploads/upload

// Upload file to ArcGIS Server
function uploadFile() {
    var uploadRequest = esri.request(
        form: dojo.byId("uploadForm"),
        load: uploadSucceeded,
        error: uploadFailed
    });
}

// get itemID from successful response
function uploadSucceeded(response, io) {
    var itemID = response["item"].itemID;
    var params = {
        "InputFile": {"itemID": itemID}
    }; 
    runGpTask(params);
}
Step 3: Run GP Task

- Task supports two kinds of operations
  1. Execute: Synchronous execution
     http://<rest-url>/<service-name>/GPServer/<task-name>/execute
  2. Submit Job: Asynchronous execution
     http://<rest-url>/<service-name>/GPServer/<task-name>/submitJob

//Step 3: Run GP Task
function runGpTask(params){
    gpTask.submitJob(params, onJobComplete, onJobStatus, onJobFail)
}
Step 4 : Get results

- Using result map service
  - Result map service is a dynamic map service and supports query, dynamic legends, time etc from 10.1 onwards
  - To visualize, add result map service as dynamic layer to the webmap
  - Result map service url is dependent on the job
    - http://<rest-url>/<gp-service-name>/MapServer/jobs/<jobid>

```javascript
//Step 4: Get results when job completes
function onJobComplete(jobInfo){
    //construct result map service url using jobid from jobInfo
    var mapserviceurl = "http://<rest-url>/<gp-service-name>/MapServer/jobs"
    var mapurl = mapserviceurl +"/"+ jobInfo.jobId;
    var resultlayer = new esri.layers.ArcGISDynamicMapServiceLayer(
        mapurl, { "id": gpResultLayer,
                "opacity": 0.7 });
}
```
Upload data and Analyze

Demo
1. Create tool
2. Publish
3. Create application
Analysis with large datasets

Viewshed with world-wide raster
1. Input features from web map
2. Analyze with large dataset at the server
3. Visualize results
Step 1. Create tool

<table>
<thead>
<tr>
<th>Geoprocessing tools</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer</td>
<td>Create processing extent</td>
</tr>
<tr>
<td>Viewshed</td>
<td>Calculate viewshed</td>
</tr>
<tr>
<td>Raster Calculator</td>
<td>Get visible cells</td>
</tr>
<tr>
<td>Raster to Polygon</td>
<td>Converts raster to polygon features</td>
</tr>
</tbody>
</table>
1. Create tool - Best practices

- Pre-compute input data
- Use map layer as project data
  - Example Elevation layer
- Limit processing extent
  - Example Set Environments – Mask & Extent
- Use In-memory workspace for intermediate output
2. Publish Geoprocessing Service: Best practices

- Register large dataset location with the server data store to prevent copying large datasets during publishing.
- Two ways to create data store:
  - Share the data location between desktop and server and register the shared location (OR)
  - Copy data manually before publishing to server and register the locations with data store.
3. Creating web application

- Synchronous execution: execute operation

```javascript
// Step 1: Initialize GP Tasks
// Step 2 & 3: Setup Task parameters and run Task
// Step 4: Get results and draw them as Graphics Layer

function onTaskSuccess(result, messages){
  // create graphics layer from the result
  var taskResultLayer = new esri.layers.GraphicsLayer({id:“resultLayer”});
  var features = result[0].value.features;
  dojo.forEach(features,function(feature){
    feature.setSymbol(polygonSymbol);
    taskResultLayer.add(feature);
  });
  map.addLayer(taskResultLayer); // Add graphics layer to map
}
```
Analysis with large datasets

Demo
1. Create tool
2. Publish tool
3. Web application
Case Study 3

Using Feature Service with Geoprocessing Service

1. Collect Field Data
2. Analyze & Interpolate
3. Visualize Concentrations
1. Create tool

<table>
<thead>
<tr>
<th>Geoprocessing tools</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Layer By Location</td>
<td>Selects features in a specific area of interest</td>
</tr>
<tr>
<td>Empirical Bayesian Kriging</td>
<td>Interpolates selected features</td>
</tr>
</tbody>
</table>
2. Publish Geoprocessing Service

- Register SDE database in the data store
- Use Lead Points SDE Layer as project data in Geoprocessing task
- Also, publish Lead points layer as feature service to collect field data
2. Behind the scene

- Feature Service
- Geoprocessing Service
- Field Crew collecting data
- Interpolated Surface in Web application

Enterprise Geodatabase

Update data

Analyze

Input

Output

Interpolated Surface in Web application
### 3. Creating web application

- **Asynchronous execution : submitJob operation**

```javascript
// Step 1: Initialize GP Task
// Step 2 & 3: Setup task parameters and call Task
// Step 4: Get results when job completes

function onJobComplete(jobInfo){
  //construct result map service url using jobid from jobInfo
  var mapserviceurl = “http://<rest-url>/<gp-service-name>/MapServer/jobs”;
  var mapurl = mapserviceurl +"/”+ jobInfo.jobid;
  var resultlayer = new esri.layers.ArcGISDynamicMapServiceLayer(
    mapurl, { "id": gpResultLayer, "opacity": 0.7 });
}
```
Final Notes

• **Please fill out session surveys**
  - Helps us improve the session
  - GoTo: [http://www.esri.com/events/devsummit/session-rater](http://www.esri.com/events/devsummit/session-rater)
  - Offering ID: 226

• All sessions are recorded and will be available on the Resource Center
  - Slides and code samples will also be available

• **Still have questions?**
  - Meet us @Island in the Showcase
Understanding our world.