Performance Considerations When Building Geoprocessing Tools

Gerhard Trichtl

ESRI EUROPEAN DEVELOPER SUMMIT
Agenda

• General Considerations

• Cursor

• Query

• Memory-Processing

• Parallel and Multiprocessing
General Considerations
Why considering Performance considerations

• Tasks needs to be executed repeatedly

• Needs to be finished in a time window

• Needs to be optimized using the hardware
General Considerations

• Working with databases
  - Use tools, which supports multiple operations at once
    - E.g. Use AddFields-Tool instead of AddField-Tool
  - When possible work directly with SQL to calculate/update values
    - Several Tools would be available within ArcGIS Pro in future releases to support this pattern better

• Working with Data over the network
  - If a lot of operations happens, copy it locally and make operations locally (specially for intermediate data)

• Working with WebLayer
Cursor
arcpy.<Cursor> vs. arcpy.da.<Cursor>

- arcpy.<Cursor> – legacy – only still available because of Backward compatibility

- arcpy.da.<Cursor>
  - A own module within arcpy for faster access of data
  - Within module additional functions available beside Cursors
arcpy.<Cursor> vs. arcpy.da.<Cursor>
Illustrate `arcpy.<Cursor>` vs. `arcpy.da.SearchCursor`

```python
import os, sys, string, time, arcpy

inFC = os.path.join(os.path.dirname(sys.argv[0]), "AnnoTest.gdb", "Anno_LM")

# Show time with arcpy.da.SearchCursor
arcpy.AddMessage("arcpy.da.SearchCursor:")
tLen=0
tRec=0
t0 = time.time()
for row in arcpy.da.SearchCursor(inFC, ['SHAPE_Length']):
    tRec=tRec+1
    tLen=tLen+row[0]
arcpy.AddMessage(" Time : "+ str(round((time.time()-t0)/60,2)) + " min."")
arcpy.AddMessage(" Records read: "+ str(tRec) + "\n Total Length: "+ str(tLen))
# Show time with arcpy.SearchCursor

arcpy.AddMessage("arcpy.SearchCursor:")
tLen=0
tRec=0
t0 = time.time()
rows = arcpy.SearchCursor(inFC,fields="SHAPE_Length")
for row in rows:
    tLen= tLen + row.getValue("SHAPE_Length")
    tRec= tRec + 1
arcpy.AddMessage(" Time : "+ str(round((time.time()-t0)/60,2)) + " min."")
arcpy.AddMessage(" Records read: "+ str(tRec) + "\n Total Length: "+ str(tLen))
```
Query
Query Considerations

• Processing Data
  - Limit only to your data you need for the analysis
  - Extract data with Clip, you need to process, or set Processing Extent
  - Use Selection (attribute/spatial) to reduce amount of processed features

• Attribute-Selection
  - Fields, which would be used for selection should be indexed
  - When joins included, also the join fields should be indexed
    - Would be checked when created within the UI and prompt for creating

• Spatial-Selection
  - Spatial Index should exists on all layers
  - Avoid projections/Transformation in operations – all data should be in the same projection
Index vs. NoIndex
```python
import os, sys, string, time, arcpy

inFC = os.path.join(os.path.dirname(sys.argv[0]),"GIP.gdb","GIP","Linknetz")

if arcpy.Exists("sellayer"):
    arcpy.Delete_management("sellayer")
    arcpy.MakeFeatureLayer_management(inFC,"sellayer")

# Now get count on field without Index
arcpy.AddMessage("Select with N0Index:")
t0 = time.time()
arcpy.SelectLayerByAttribute_management("sellayer","NEW_SELECTION","LENGTH_NOINDEX > 1")
anz = int(arcpy.GetCount_management("sellayer").getOutput(0))
arcpy.AddMessage(" Time : " + str(round(time.time()-t0,2)) + " sec.")
arcpy.AddMessage(" Records sel.: " + str(anz))

# Show time with arcpy.SearchCursor
arcpy.AddMessage("\nSelect with Index:")
t0 = time.time()
arcpy.SelectLayerByAttribute_management("sellayer","NEW_SELECTION","LENGTH_INDEX > 1")
anz = int(arcpy.GetCount_management("sellayer").getOutput(0))
arcpy.AddMessage(" Time : " + str(round(time.time()-t0,2)) + " sec.")
arcpy.AddMessage(" Records sel.: " + str(anz))
```
Additional Considerations with Index

- When working with Enterprise Databases and several fields will be used
- Instead of Individual Index on each fields use combined index
  - Order of fields in Index must match the field-order in the Query

- SELECT FIELD_mA = 12 AND FIELD_B = 34 AND FIELD_C = 56

- CREATE INDEX IDX_SELECTION ON <Table> (FIELD_A, FIELD_C, FIELD_B)
  - Index only on Field A would be used – Field B and Field C would be run with full table scan
- CREATE INDEX IDX_SELECTION ON <Table> (FIELD_A, FIELD_B, FIELD_D)
  - Index on Field_A and Field_B would be used, Field_C would be used with full table scan
- CREATE INDEX IDX_SELECTION ON <Table> (FIELD_C, FIELD_A, FIELD_B)
  - Index only on Field_C would be used – Field_A and Field_B would be used with full table scan
- CREATE INDEX IDX_SELECTION ON <Table> (FIELD_A, FIELD_B, FIELD_C)
  - Index would be used for all 3 fields
Memory-Processing
Processing in Memory

- Fastest possible access and processing of data – no disc I/O during processing

- IN_MEMORY
  - Implemented in ArcGIS Desktop for faster processing, supported within ArcGIS Pro, but not for Map-Layer display (Legacy)

- MEMORY
  - New implementation, that supports also Map-Layer-Display in ArcGIS Pro
  - Functionality get expanded with each Release

Additional Information:
General Considerations

• Machine needs to have enough Memory to hold the data in RAM
  - Delete the data, as soon you didn't need it longer

• Consider overall performance with MEMORY-Processing vs. Direct data processing
  - Load data into Memory
  - Processing in Memory
  - Write data into Geodatabase to persist the result

• Biggest difference on Data on slow disks or over slow network

• Ideally for temporary data within your processing, when you perform several operations between your input and final output
fGDB vs. MEMORY-Processing
Using in ArcGIS Pro
## fGDB vs. MEMORY

### Analysis within fGDB:
- **Buffer**: 114.63 sec.
- **MakeLayer**: 0.32 sec.
- **Selection**: 0.28 sec. - 22536
- **Get Areas**: 0.52 sec. - 366477864.88162017
- **Time**: 115.75 sec.
- **Records**: 99507

### Analysis in Memory:
- **Copy in Mem**: 2.12 sec.
- **Buffer**: 184.91 sec.
- **MakeLayer**: 0.08 sec.
- **Selection**: 0.26 sec. - 22536
- **Get Areas**: 0.5 sec. - 366477864.88162017
- **Copy from Mem**: 7.17 sec.
- **Time**: 115.04 sec.
- **Records**: 99507

### How Buffer in fGDB - Internal Disk

```python
import os, sys, string, time, arcpy
# Internal Disk (H:\)
filePath = os.path.join(os.path.dirname(sys.argv[0]), "GIP.gdb\")
outFC = os.path.join(filePath, "GIP.gdb\", "L2_100k")
outFC = os.path.join(filePath, "GIP.gdb\", "Streetbuffer")
memFC = os.path.join(filePath, "GIP.gdb\", "OTF\", "Streetbuffer")

for mem in (memFC, memFC2, outFile, outFile2):
    if arcpy.Exists(mem):
        arcpy.Delete_management(mem)

# How Buffer in fGDB - Internal Disk
arcpy.AddMessage("Buffer within fGDB.")
t0 = time.time()
for mem in (memFC, memFC2, outFile, outFile2):
    if arcpy.Exists(mem):
        arcpy.Delete_management(mem)

t0 = time.time()
arcpy.Buffer_analysis(inFC, outFile, "10 Meters", "FULL", "ROUND", "NONE", "", "MODIFIED")
arcpy.AddMessage(" Buffer : " + str(round(time.time() - t0, 2)) + " sec.")
t0 = time.time()
arcpy.MakeFeatureLayer_management(outFC, "FL")
arcpy.AddMessage(" MakeLayer : " + str(round(time.time() - t0, 2)) + " sec.")
t0 = time.time()
arcpy.SelectLayerByAttribute_management(outFC, "NEW_SELECTION", "MOREEQUAL\""
arcpy.GetCount_management(outFC).getOutput(0)
arcpy.AddMessage(" Selection : " + str(round(time.time() - t0, 2)) + " sec. - " + str(ans))
t0 = time.time()
fSync 없다:
for row in arcpy.da.SearchCursor("FL", "[SHAPE@AREA]"):
    ans = float(row[0])
arcpy.AddMessage(" Get Areas : " + str(round(time.time() - t0, 2)) + " sec. - " + str(ans))
arcpy.AddMessage(" Time : " + str(round(time.time() - t0, 2)) + " sec.")
arcpy.AddMessage(" Records : " + str(arcpy.GetCount_management(outFC).getOutput(0)))
```

### How Buffer in MEMORY from internal Disk

```python
import os, sys, string, time, arcpy
# Internal Disk (H:\)
filePath = os.path.join(os.path.dirname(sys.argv[0]), "GIP.gdb\")
outFC = os.path.join(filePath, "GIP.gdb\", "L2_100k")
outFC = os.path.join(filePath, "GIP.gdb\", "Streetbuffer")
memFC = os.path.join(filePath, "GIP.gdb\", "OTF\", "Streetbuffer")

for mem in (memFC, memFC2, outFile, outFile2):
    if arcpy.Exists(mem):
        arcpy.Delete_management(mem)

# How Buffer in MEMORY from internal Disk
arcpy.AddMessage("Buffer within Memory.")
t0 = time.time()
arcpy.CopyFeatures_management(inFC, memFC)
arcpy.AddMessage(" Copy in Mem : " + str(round(time.time() - t0, 2)) + " sec.")
t0 = time.time()
arcpy.Buffer_analysis(memFC, outFile2, "10 Meters", "FULL", "ROUND", "NONE", "", "MODIFIED")
arcpy.AddMessage(" Buffers : " + str(round(time.time() - t0, 2)) + " sec.")
t0 = time.time()
arcpy.MakeFeatureLayer_management(memFC2, "FL")
arcpy.AddMessage(" MakeLayer : " + str(round(time.time() - t0, 2)) + " sec.")
t0 = time.time()
arcpy.SelectLayerByAttribute_management("FL", "NEW_SELECTION", "MOREEQUAL\""
arcpy.GetCount_management("FL").getOutput(0)
arcpy.AddMessage(" Selection : " + str(round(time.time() - t0, 2)) + " sec. - " + str(ans))
t0 = time.time()
fSync 없다:
for row in arcpy.da.SearchCursor("FL", "[SHAPE@AREA]"):
    ans = float(row[0])
arcpy.AddMessage(" Get Areas : " + str(round(time.time() - t0, 2)) + " sec. - " + str(ans))
arcpy.AddMessage(" Time : " + str(round(time.time() - t0, 2)) + " sec.")
arcpy.AddMessage(" Records : " + str(arcpy.GetCount_management(outFC).getOutput(0)))
```
Multiprocessing
What is multiprocessing

- An operation would be executed on several Cores of a CPU
  - Process is splitted in several parts and gets executed on several cores
  - Data would be interally splitted for each indivual processing
  - After individual processing of data get's merged into a single result
ArcGIS Pro

- Several GP-Tools support Multiprocessing
  - Within Search identified by Symbol
  - On Tool within Environments you see Parallel Processing
Parallel Processor Factor -

Buffer vs. Pairwise Buffer
Small Test – 600 Features

- Standard Buffer ~ 10 sec.
- Pairwise Buffer ~ 3 sec.
Much more Input Lines ~ 100,000

- Standard Buffer ~ 169 sec.
- Pairwise Buffer ~ 18 sec.
GeoAnalytics Desktop Tools

- GeoAnalytics Tools from ArcGIS Enterprise are now also on Desktop

- Uses Spark-Technology
  - Java-JRE got's installed for SPARK
    - Esri would maintain Java-JRE-Installation

- Uses Multi-Processing-Technology
Alternative implement your own Multiprocessing-Solution

- Using Python multiprocessing

- Consider reading/writing data with geodatabase context with locking

- Processing needs to be done in a function inside the module
  - Data Preparation before executing

- Best Practice
  - Split data in chunks before executing
  - Process chunk data in multiprocessing mode
  - Merge individual results into a single output
Multiprocessing Sample
import httplib, sys, os, string, time, socket, urllib, zipfile
import re, multiprocessing

baseURL = "https://www.data.com/dop/downloads/"
ext = "_DOP_Farbe.zip"
URLList = []
freeCores = 2

def URLLoad(URL):
    # Routine to download a file
    # This would be executed on each process
    # URL = URL + "!"[0] + "/" + fileName
    print "Working on " + str(URL)
    urllib.urlretrieve(URL, os.path.join(os.path.dirname(sys.argv[0]), fileName))
    # Check if ZIP-File OK
    try:
        z = zipfile.ZipFile(os.path.join(os.path.dirname(sys.argv[0]), fileName), "r")
        z.close()
    except:
        os.system("del " + os.path.join(os.path.dirname(sys.argv[0]), fileName) + " ")

if __name__ == '__main__':
    # Create List of possible Downloads files
    x = 40101
    while x < 41899:
        fileName = str(x) + ext
        URLList.append(baseURL + str(x) + "!" + fileName)
        x = x + 1

    # Define Cores to be executed
    CPUanz = multiprocessing.cpu_count() - freeCores
    print("Processing on " + str(CPUanz) + " Cores")
    # Now span the list of tasks around the deassigned Cores
    pool = multiprocessing.Pool(CPUanz)
    pool.map(URLLoad, URLList)
    pool.close()
    pool.join()
Questions?
Please Take Our Survey on the App

Download the Esri Events app and find your event

Select the session you attended

Scroll down to find the feedback section

Complete answers and select “Submit”