Performance Considerations When Building Geoprocessing Tools

Gerhard Trichtl

ESRI EUROPEAN DEVELOPER SUMMIT

esri

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 availible 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
  - https://pro.arcgis.com/en/pro-app/tool-reference/appendices/geoprocessing-considerations-for-portal-and-service-data.htm

Cursor

arcpy.<Cursor> vs. arcpy.da.<Cursor>

- arcpy.<Cursor> – legacy – only still availible because of Backward compatiblity

> Legacy:
> The `arcpy.da` cursors (`arcpy.da.SearchCursor`, `arcpy.da.UpdateCursor`, and `arcpy.da.InsertCursor`) were introduced with ArcGIS 10.1 to provide significantly faster performance over the previously existing set of cursor functions (`arcpy.SearchCursor`, `arcpy.UpdateCursor`, and `arcpy.InsertCursor`). The original cursors are provided only for continuing backward compatibility.

- arcpy.da.<Cursor>
  - A own module within arcpy for faster access of data
  - Within module additional functions availible beside Cursors

arcpy.<Cursor> vs. arcpy.da.<Cursor>

Illustrate arcpy.<Cursor> vs. arcpy.da.SearchCursor

cursor.py - C:\Data\DevSummit2019\PerfConsider\Demo\01_Cursor\cursor.py (3.6.9)

File Edit Format Run Options Window Help

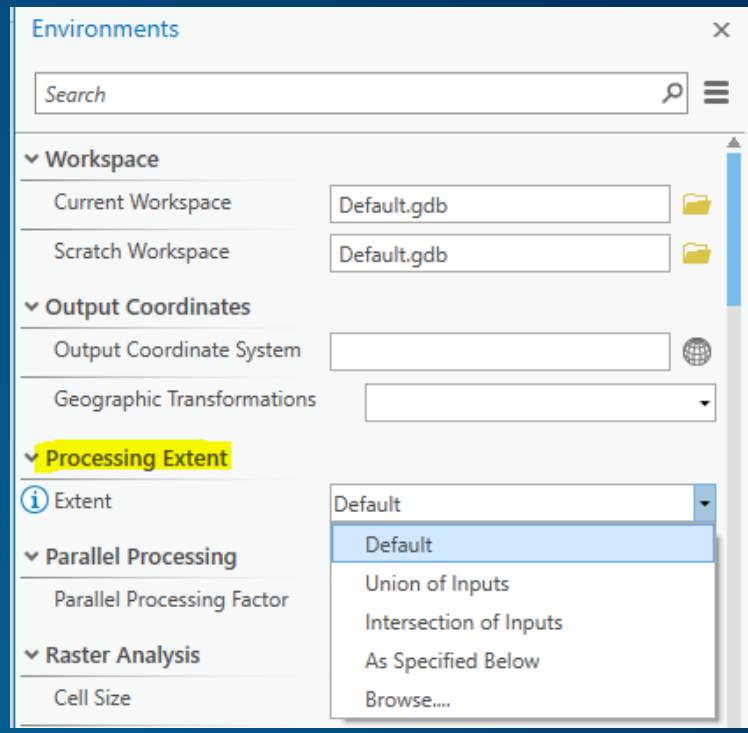
```python
import os,sys,string,time,arcpy

inFC = os.path.join(os.path.dirname(sys.argv[0]),"AnnoTest.gdb","Anno_1M")

# 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))
```

```
arcpy.da.SearchCursor:
  Time         : 0.01 min.
  Records read: 1156245
  Total Length: 22940146.54269339
arcpy.SearchCursor:
  Time         : 0.5 min.
  Records read: 1156245
  Total Length: 22940146.54269339
```

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 amout 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

Sample

SelectTime.py - C:\Data\DevSummit2019\PerformanceConsiderationsWhenBuildingGeoprocessingTools\Demo\02_Select\SelectTime.py (2.7.16)

File Edit Format Run Options Window Help

```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 NOIndex:")
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))
```

```
Select with NOIndex:
  Time        : 1.16 sec.
  Records sel.: 32369

Select with Index:
  Time        : 0.22 sec.
  Records sel.: 32369
```

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 we 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 Dekstop for faster processing, supported within ArcGIS Pro, but not for Map-Layer display (Legacy)
- MEMORY
  - New Implentation, that supports also Map-Layer-Display in ArcGIS Pro
  - Functionality get expanded with each Release

Additional Information:
https://pro.arcgis.com/en/pro-app/help/analysis/geoprocessing/basics/the-in-memory-workspace.htm

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        : 104.91 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
```

```python
import os,sys,string,time,arcpy
# Internal Disk (M.2)
filePath=os.path.dirname(sys.argv[0])
inFC = os.path.join(filePath,"GIP.gdb","GIP","Linknetz_100k")
outFC = os.path.join(filePath,"GIP.gdb","GIP","Streetbuffer")
outFC2 = os.path.join(filePath,"GIP.gdb","GIP","Streetbuffer_Memory")
memFC = os.path.join("MEMORY","Linknetz")
memFC2 = os.path.join("MEMORY","Streetbuffer")
for mem in [memFC,memFC2,outFC,outFC2]:
    if arcpy.Exists(mem):
        arcpy.Delete_management(mem)


# Now Buffer in fGDB - internal Disk
arcpy.AddMessage("Buffer within fGDB:")
ta = time.time()
t0 = time.time()
arcpy.Buffer_analysis(inFC,outFC,"10 Meters","FULL","ROUND","NONE","","GEODESIC")
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("FL","NEW_SELECTION","EDGECAT='GW'")
anz=int(arcpy.GetCount_management("FL").getOutput(0))
arcpy.AddMessage("   Selection    : " + str(round(time.time()-t0,2)) + " sec. - " + str(anz))
t0 = time.time()
fArea=0
for row in arcpy.da.SearchCursor("FL", ["SHAPE@AREA"]):
    fArea=fArea+row[0]
arcpy.AddMessage("   Get Areas    : " + str(round(time.time()-t0,2)) + " sec. - " + str(fArea))
arcpy.AddMessage("  Time          : " + str(round(time.time()-ta,2)) + " sec.")
arcpy.AddMessage("  Records       : " + str(arcpy.GetCount_management(outFC).getOutput(0)))


# Buffer in MEMORY from internal Disk
arcpy.AddMessage("\nBuffer in Memory:")
ta = time.time()
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,memFC2,"10 Meters","FULL","ROUND","NONE","","GEODESIC")
arcpy.AddMessage("   Buffer       : " + str(round(time.time()-t0,2)) + " sec.")
t0 = time.time()
arcpy.MakeFeatureLayer_management(memFC2,"ML")
arcpy.AddMessage("   MakeLayer    : " + str(round(time.time()-t0,2)) + " sec.")
t0 = time.time()
arcpy.SelectLayerByAttribute_management("ML","NEW_SELECTION","EDGECAT='GW'")
anz=int(arcpy.GetCount_management("ML").getOutput(0))
arcpy.AddMessage("   Selection    : " + str(round(time.time()-t0,2)) + " sec. - " + str(anz))
t0 = time.time()
fArea=0
for row in arcpy.da.SearchCursor("ML", ["SHAPE@AREA"]):
    fArea=fArea+row[0]
arcpy.AddMessage("   Get Areas    : " + str(round(time.time()-t0,2)) + " sec. - " + str(fArea))
t0 = time.time()
arcpy.CopyFeatures_management(memFC2,outFC2)
arcpy.AddMessage("   Copy from Mem: " + str(round(time.time()-t0,2)) + " sec.")
arcpy.AddMessage("  Time          : " + str(round(time.time()-ta,2)) + " sec.")
arcpy.AddMessage("  Records       : " + str(arcpy.GetCount_management(outFC2).getOutput(0)))
```

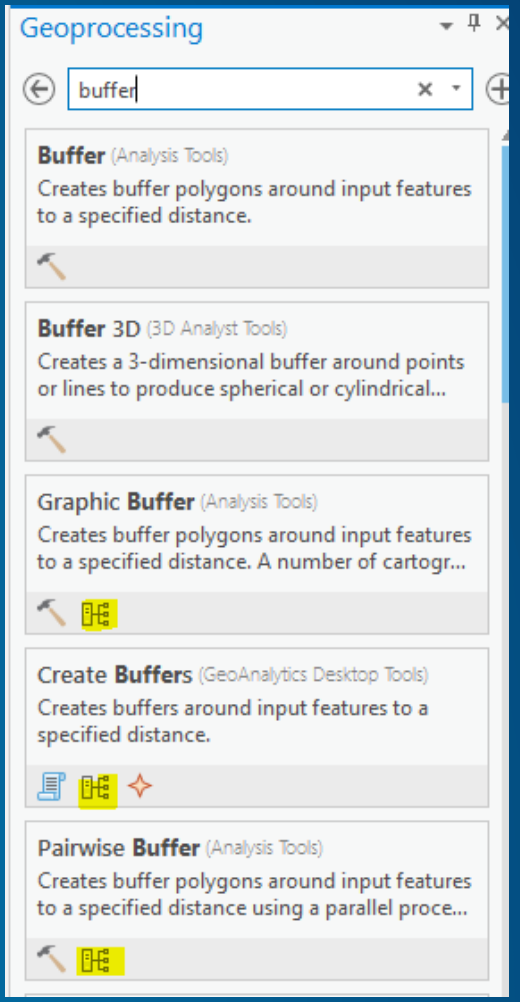
Multiprocessing

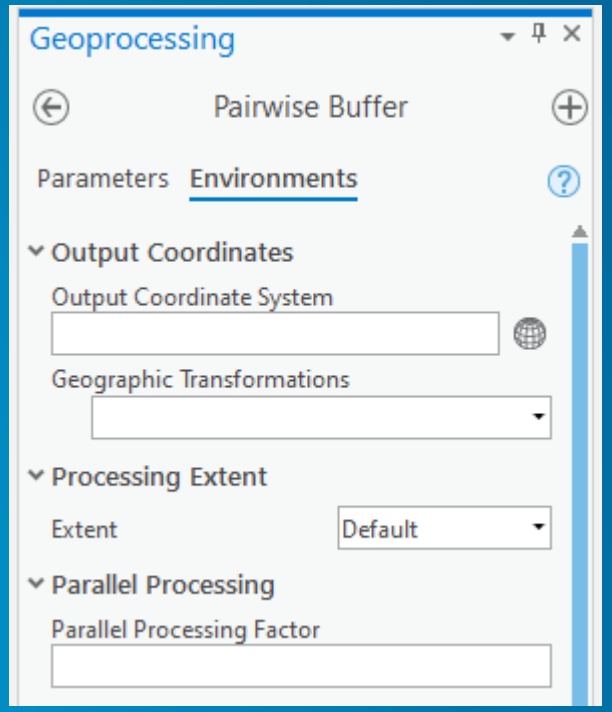
What is multiprocessing

- 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 -

- https://pro.arcgis.com/en/pro-app/tool-reference/environment-settings/parallel-processing-factor.htm#GUID-85BB7B69-C105-4F5D-B86C-B5142133DBF4

Usage notes

Note:
Individual tools may modify how this environment is used. See the tool documentation for any information that supersedes the information below.

- The value of this environment determines the number of processes across which a tool spreads its operation. Those processes will be divided between hardware cores (processors) built into the machine. The number of hardware cores does not change based on this setting.
- Each tool that honors this environment has a built-in default for the number of processes given a particular machine. You can change this based on your data, operation, and available resources.
- If you specify a percent value (using the % symbol), the number of processes used will be the specified percentage of the number of cores on the machine, rounded to the nearest integer. For example, on a 4-core machine, the percentage and number of process is as follows:
  - Setting 50% means the operation will be spread over 2 processes (50% * 4 = 2).
  - Setting 66% means the operation will be spread over 3 processes (66% * 4 = 2.64 which rounds to 3).
  - Setting 100% means the operation will be spread over all 4 processes (100% * 4 = 4).
- SQL Server Express allows a maximum of three connections at a time. Each processing CPU requires a connection to the server. Additionally, the software running the tool, such as ArcGIS Desktop, counts as one connection process, leaving only two worker connection processes available for parallel processing.
- Specifying more processes than your machine has cores may incur a performance penalty. This is because multiple processes will compete for resources on one core. To specify the environment in a way that avoids this competition, you can use either a percent value less than 100% or a number of processes less than the number of cores on your machine.
  However, for cases where all your processes are I/O bound to a disk or to an enterprise database connection, you may get better performance by specifying more processes than you have cores. For example, the Add Rasters to Mosaic Dataset tool is I/O bound when the mosaic dataset is stored in an enterprise database. Also, the Build Overviews tool is primarily I/O bound to the disk. You can use more processes than your machine has cores by specifying either a percent value greater than 100% or a number of processes greater than the number of cores on your machine. For example, if you have a 4-core machine, specifying 8 or 200% will spread operations over 8 processes.

Buffer vs. Pairwise Buffer

Small Test – 600 Features

- Standard Buffer ~ 10 sec.
- Pairwise Buffer ~ 3 sec.

Buffer (Analysis Tools)

Completed.

Started: Today at 19:15:19
Completed: Today at 19:15:29
Elapsed Time: 10 Seconds

Errors and warnings

Parameters

Input Features	BufferSoruce
Output Feature Class	MEMORY\Standard
Distance [value or field]	10 Meters
Side Type	FULL
End Type	ROUND
Dissolve Type	NONE
Dissolve Field(s)	
Method	GEODESIC

Environments

Default environment settings were used.

Messages

There are are no messages.

Pairwise Buffer (Analysis Tools)

Completed.

Started: Today at 19:14:38
Completed: Today at 19:14:41
Elapsed Time: 3 Seconds

Errors and warnings

Parameters

Input Features	BufferSoruce
Output Feature Class	MEMORY\BufferSoruce_PairwiseBuffer
Distance [value or field]	10 Meters
Dissolve Type	NONE
Dissolve Field(s)	
Method	GEODESIC
Maximum Offset Deviation	0 Meters

Environments

Default environment settings were used.

Messages

There are are no messages.

Much more Input Lines ~ 100.000

- **Standard Buffer ~ 169 sec.**
- **Pairwise Buffer ~ 18 sec.**

Buffer (Analysis Tools)

Completed.

Started: Today at 19:25:38
Completed: Today at 19:28:27
Elapsed Time: 2 Minutes 49 Seconds

Errors and warnings

Parameters

Input Features	Buffer_100k
Output Feature Class	MEMORY\Standard_100k
Distance [value or field]	10 Meters
Side Type	FULL
End Type	ROUND
Dissolve Type	NONE
Dissolve Field(s)	
Method	GEODESIC

Environments

Default environment settings were used.

Messages

There are are no messages.

Pairwise Buffer (Analysis Tools)

Completed.

Started: Today at 19:24:52
Completed: Today at 19:25:10
Elapsed Time: 18 Seconds

Errors and warnings

Parameters

Input Features	Buffer_100k
Output Feature Class	MEMORY\Pairwise_100k
Distance [value or field]	10 Meters
Dissolve Type	NONE
Dissolve Field(s)	
Method	GEODESIC
Maximum Offset Deviation	0 Meters

Environments

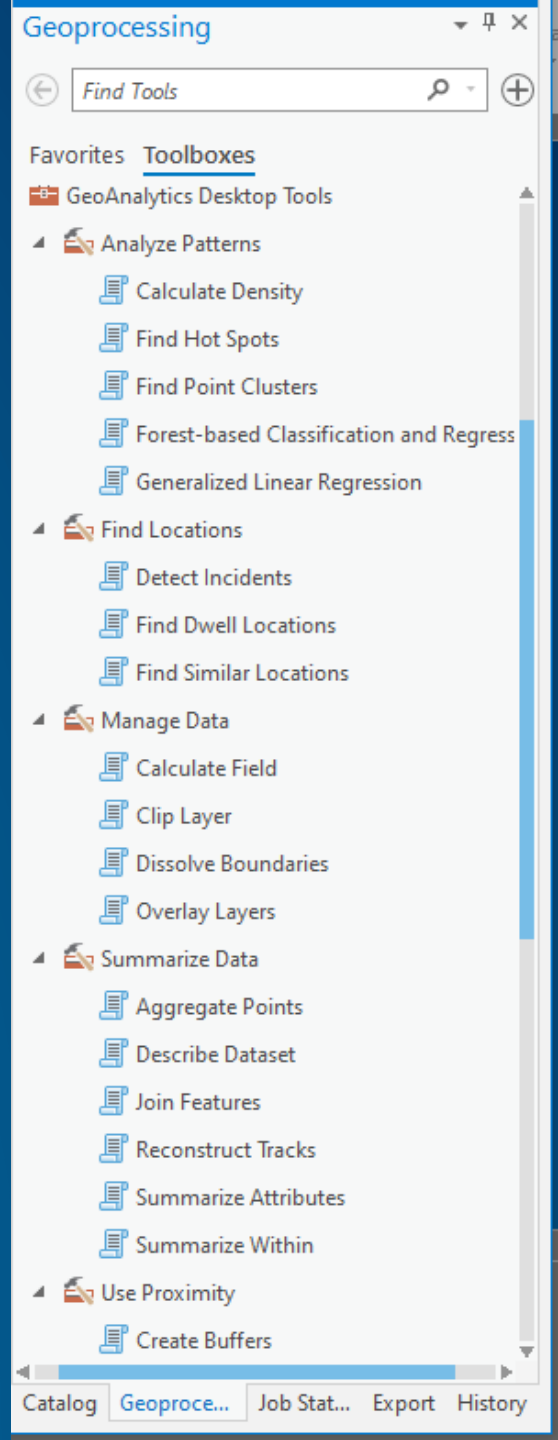
Default environment settings were used.

Messages

There are are no messages.

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 indiviual results into a single output

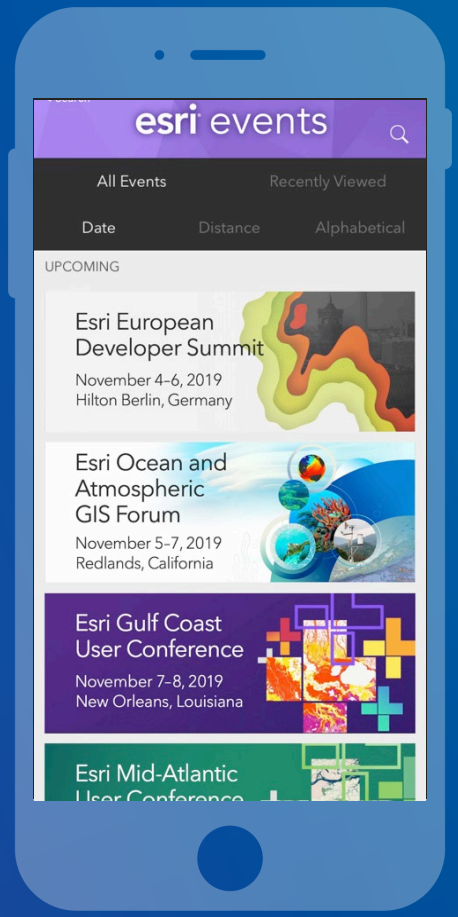
Multiprocessing Sample

```python
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
        fileName = URL.split("!")[1]
        URL = URL.split("!")[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:
                zipf = zipfile.ZipFile(os.path.join(os.path.dirname(sys.argv[0]),fileName),"r")
                zipf.close()
        except:
                os.system("del " + os.path.join(os.path.dirname(sys.argv[0]),fileName) + "  ")

if __name__ == '__main__':
        # Create List of possibile Downloadfiles
        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 definied Cores
        pool = multiprocessing.Pool(CPUanz)
        pool.map(URLLoad,URLList)
        pool.close()
        pool.join()
```

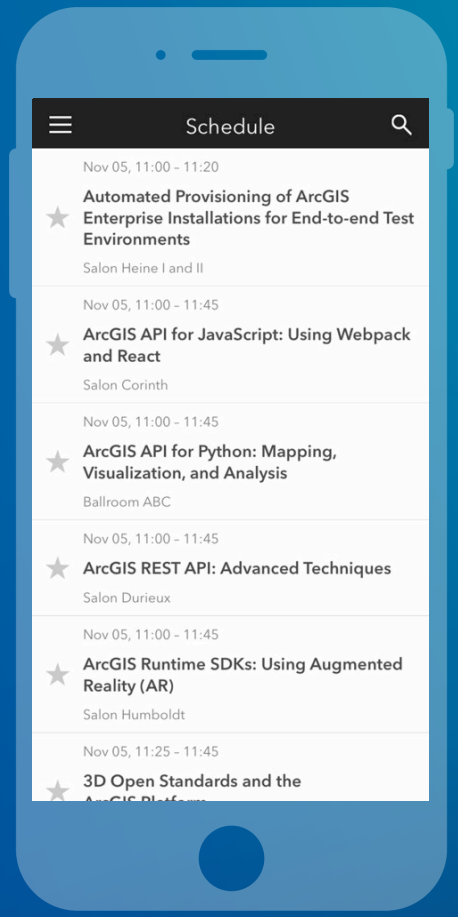
Questions?

Please Take Our Survey on the App

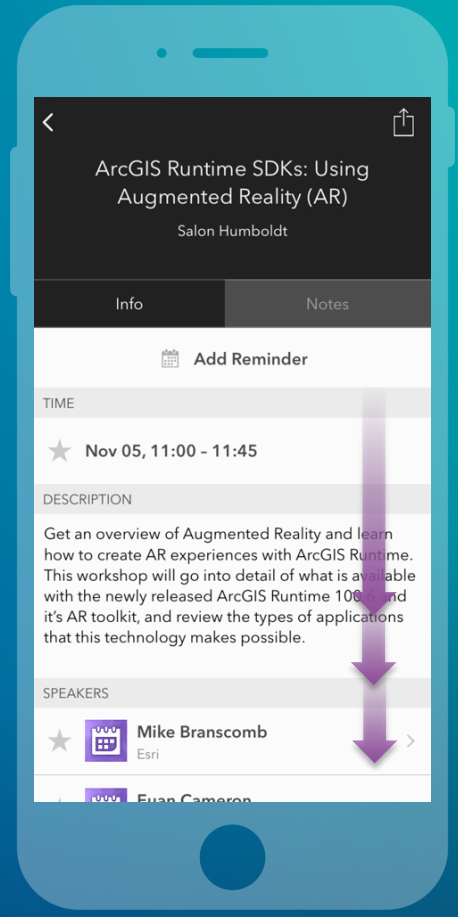
Download the Esri Events app and find your event



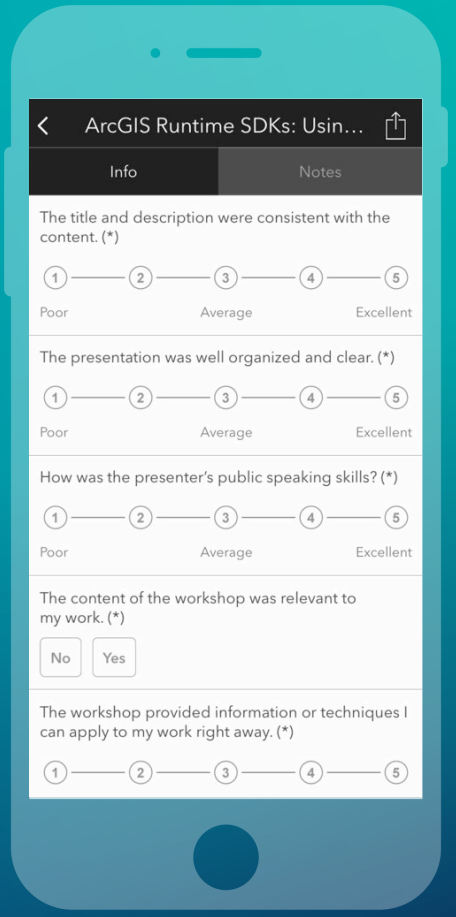
Select the session you attended



Scroll down to find the feedback section



Complete answers and select "Submit"



esri

THE SCIENCE OF WHERE