ArcGIS Server Performance and Scalability-Optimization and Testing

Andrew Sakowicz, Esri Professional Services
Objective

• Overview:
  - Key performance factors
  - Optimization techniques
  - Testing Methodology
Audience

- Audience
  - Testers
  - Developers
  - System Architects

- Level:
  - Intermediate
Performance Factors in ArcGIS Server Components

- ArcGIS Server services
- Data Sources
- ArcGIS Server Framework
- Client applications (Flex, Silverlight, JS, ADF)
- Hardware

Goal: Deliver highly performing deployments.
Performance Factors: ArcGIS Server Services
Performance Factors: ArcGIS Server Services

*Map Service – Source document optimization*

- Keep map symbols simple
- **Scale dependency**
- Optimize spatial index
- Simplify data
- Avoid re-projections on the fly
- Optimize map text and labels for performance
- Use annotations
- Avoid wavelet compression-based raster types (MrSid, JPEG2000)
- Use fast joins (no cross db joins)
Performance Factors: ArcGIS Server Services

Map Service – Output image format choices

- **PNG8/24/32**
  - Transparency support
  - 24/32 good for anti-aliasing, rasters with many colors
  - Lossless: Larger files (> disk space/bandwidth, longer downloads)

- **JPEG**
  - Basemap layers (no transparency support)
  - Much smaller files
Performance Factors: ArcGIS Server Services

Map Service – Source map document optimizations, scale dependency

- Performance linearly related to number of features
Performance Factors: ArcGIS Server Services

**Map Service – Cache generation**

- Optimized services create caches significantly faster than “Classic”
Performance Factors: ArcGIS Server Services

Geoprocessing Service

- Pre-compute intermediate steps when possible
- Use local paths to data and resources
- Avoid unneeded coordinate transformations
- Add attribute indexes
- Simplify data

Detailed instructions on the Resource Center
Performance Factors: ArcGIS Server Services

Image Service

- Tiled, JPEG compressed TIFF is the best (10-400% faster)

- Build pyramids for raster datasets and overviews for mosaic datasets (new at 10.0)

- Tune mosaic dataset spatial index.

- Use JPGPNG request format in Web and Desktop clients
  - Returns JPEG unless there are transparent pixels (best of both worlds).

Help Topic: “Optimization Considerations for ArcGIS Image Server”
Performance Factors: ArcGIS Server Services

**Geocode Service**

- Use local instead of UNC locator files.

- Services with large locators take a few minutes to “warm-up”

- New 10.0 Single Line Locators offer simplicity in address queries but might be slower than traditional point locators.
Performance Factors: ArcGIS Server Services

Mobile Service

- **Document Preparation**
  - Minimize operational layers
  - Cache basemap layers

- **Service Configuration**
  - Try to keep total service cache size under 250 MB

- **Usage considerations**
  - Avoid batch postings in favor of frequent updates
Performance Factors: ArcGIS Server Services

*Feature/Geodata Service – Database maintenance is key*

- **Database Maintenance/Design**
  - Keep versioning tree small, compress, schedule synchronizations, rebuild indexes and have a well-defined data model

- **Geodata Service Configuration**
  - Server Object usage timeout (set larger than 10 min default)
  - Upload/Download default IIS size limits (200K upload/4MB download)

- **Feature Service**
  - Trade-off between client-side rendering and sending large amounts of data over the wire.
Performance Factors: Machine Architecture

*Select adequate hardware to support desired performance/load*

- **CPU**
  - Select for intended use
    - Mapping: highest Baseline CINT Rate/Core
    - **GP:** highest Baseline CFP Rate/Core
  - **Sizing**
    - Published CPU benchmarks: [http://www.spec.org/cpu2006/results/cint2006.html](http://www.spec.org/cpu2006/results/cint2006.html)
    - Published CPU-limited ESRI benchmarks: [http://resources.esri.com/enterprisegis/index.cfm?fa=codeG](http://resources.esri.com/enterprisegis/index.cfm?fa=codeG)

\[
\# CPU_t = \frac{ST_b \times TH_t \times 100}{3600 \times \% CPU_t} \times \frac{SpecRatePerCPU_b}{SpecRatePerCPU_t}
\]

*subscript t = target, subscript b = benchmark;\nST = CPU service time, TH = throughput, \%CPU = percent CPU*
Performance Factors: Data Sources

Select data storage format that provides optimal performance

Data storage format

![Low Complexity Map: Throughput vs. data source](image-url)
Performance Factors: Data Location

*Select data location that provides optimal performance*

- Local to SOC machine
- UNC (protocol + network latency/bandwidth penalties)

- All disks being equal, locally sourced data results in better throughput.
Performance Factors: DB Management

Optimize DB configuration and conduct maintenance

- DBMS configuration
- Create and maintain (rebuild) attribute indexes
- Updating DBMS statistics
- Versioning management
  - Reconcile and post
  - Compress

*Non-optimal* DBMS may be a source of significant performance degradation
Performance Factors: ArcGIS Server Framework

**Web Service Components – Web Service Handlers**

- Under high load, easily overloaded by MIME data (REST/SOAP returning MIME)
  - Network saturation
    - 72 cores can approach 40% Gigabit utilization
    - Network I/O (full-duplex)
  - CPU utilization

- Use multiple Web Instances
Performance Factors: ArcGIS Server Framework

Web Service Components – Virtual Directories

- Under high load, virtual output directory bottlenecks
  - Network saturation
  - Disk performance

- Use multiple output directories on different machines.
Performance Factors: ArcGIS Server Framework

Web Service Components - MIME vs. URL return type

- For serving maps, MIME can scale better than URL
- Disk/UNC shares often bottleneck before network bandwidth

MIME vs. URL

- MIME performs better with smaller images
Performance Factors: ArcGIS Server Framework

Web Services - Security

- SSL
- LSASS
  - [Link](http://support.esri.com/index.cfm?fa=knowledgebase.technicalarticles.articleShow&d=32620)
- Web Tier Security
  - User/Role Stores
  - Token Server
Performance Factors: ArcGIS Server Framework

**SOC**

Optimal number of instances/core departs from CPU-limited value of 1 by choice of source data type/location.

![Variance of Service Instances by Source Data Type](chart)
Tuning Primer
Performance tuning

• Benefits
  - Improved performance - user experience
  - Optimize the use of critical resources – scalability

• Tools
  - Fiddler
  - Mxdperfstat
  - Map Service Publishing Toolbar
  - DBMS trace
A test is executed at the web browser. It measures web browser call's elapsed time (roundtrip between browser and data source).
Tuning Primer

Web diagnostic tools: Fiddler, Tamperdata, Yslow
Tuning Primer

Web diagnostic tools: Fiddler

- Can validate image returned
Tuning Primer

Web diagnostic tools: Fiddler

• Understand each request URL
• Verify cache requests are from virtual directory, not dynamic map service
• Validate host origin (reverse proxy)
• Profile each transaction response time
Tuning Primer

Web diagnostic tools: Fiddler

HTTP Request

HTTP Response

Inspector Tab
Tuning Primer

Web diagnostic tools: Fiddler (NeXpert Report)
Analyze AGS context server statistics using ArcCatalog, Manager or logs. They provide aggregate and detailed information to help reveal the cause of the performance problem.
Tuning Primer

Analyze SOM/SOC statistics

- ArcCatalog

- Detailed log - set to verbose

```xml
<Msg time="2009-03-16T12:23:22" type="INFO3" code="103021" target="Portland.MapServer" methodName="FeatureLayer.Draw" machine="myWebServer" process="2836" thread="3916" elapsed="0.05221">Executing query.</Msg>


```
<table>
<thead>
<tr>
<th>Item</th>
<th>At Scale</th>
<th>Layer Name</th>
<th>Refresh Time (sec)</th>
<th>Recommendations</th>
<th>Features</th>
<th>Vertices</th>
<th>Labeling</th>
<th>Geometry Phase (sec)</th>
<th>Graphics Phase (sec)</th>
<th>Cursor Phase (sec)</th>
<th>DBMS CPU</th>
<th>DBMS LIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>1,000</td>
<td>TaxlotDenseLabel</td>
<td>1.93</td>
<td>Simplify labeling, symbology: GraphicsPhase=1.42; simplify geometry and/or set label scale; convert polygon to polyline: vertices fetched=200001; simplify geometry and/or set label scale: vertices fetched=200001;</td>
<td>1</td>
<td>200,001</td>
<td>TRUE</td>
<td>0.45</td>
<td>1.42</td>
<td>1.04</td>
<td>0.02</td>
<td>266</td>
</tr>
<tr>
<td>42</td>
<td>1,000</td>
<td>TaxlotDenseNoLabel</td>
<td>0.53</td>
<td>simplify geometry: vertices fetched=200001;</td>
<td>1</td>
<td>200,001</td>
<td>FALSE</td>
<td>0.45</td>
<td>0.02</td>
<td>0.9</td>
<td>0.02</td>
<td>140</td>
</tr>
</tbody>
</table>
Tuning Primer

ArcGIS Services

Heat Map based on response times from ArcGIS Server
Observe correlation between feature density and performance
Tuning Primer

Data sources

- Total Response Time (t1-t2)
- Wait Time
- Usage Time
- Search & Retrieval Time

Browser → Web Server → SOM → SOC → SDE/DBMS
Tuning Primer

Data Sources – Oracle Trace

```sql
select username, sid, serial#, program, logon_time from v$session where username='STUDENT';
```

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>SID</th>
<th>SERIAL#</th>
<th>PROGRAM</th>
<th>LOGON_TIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td>132</td>
<td>31835</td>
<td>gsrvr.exe</td>
<td>23-OCT-06</td>
</tr>
</tbody>
</table>

SQL> connect sys@gis1_andrews as sysdba

Enter password:

Connected.

SQL> execute sys.dbms_system.set_ev(132,31835,10046,12,'');

DBMS trace is a very powerful diagnostic tool
Private Sub OracleTrace_Click()
    ... 
    Set pFeatCls = pFeatLyr.FeatureClass
    Set pDS = pFeatCls
    Set pWS = pDS.Workspace
    sTraceName = InputBox("Enter <test_name><email>")
    pWS.ExecuteSQL ("alter session set tracefile_identifier = '" & sTraceName & "'")
    pWS.ExecuteSQL ("ALTER SESSION SET events '10046 trace name context forever, level 12'"")
    ... 
End Sub
Tuning Primer

Data Sources – Oracle Trace (continued)

SQL ID : 71py6481sj3xu

```sql
SELECT  1 SHAPE, TAXLOTS.OBJECTID,  TAXLOTS.SHAPE.points,TAXLOTS.SHAPE.numpts,
        TAXLOTS.SHAPE.entity,TAXLOTS.SHAPE.minx,TAXLOTS.SHAPE.miny,TAXLOTS.SHAPE.maxx,TAXLOTS.SHAPE.maxy,TAXLOTS.rowid
FROM  SDE.TAXLOTS TAXLOTS WHERE SDE.ST_EnvIntersects(TAXLOTS.SHAPE,:1,:2,:3,:4) =  1
```

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.07</td>
<td>0.59</td>
<td>115</td>
<td>1734</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>242</td>
<td>0.78</td>
<td>12.42</td>
<td>2291</td>
<td>26820</td>
<td>0</td>
<td>24175</td>
</tr>
</tbody>
</table>

```

total  243  0.85  13.02  2406  28554  0      24175
```

Elapsed times include waiting on following events:

```

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>242</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>db file sequential read</td>
<td>2291</td>
<td>0.39</td>
<td>11.69</td>
</tr>
<tr>
<td>SQL*Net more data to client</td>
<td>355</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>242</td>
<td>0.03</td>
<td>0.54</td>
</tr>
</tbody>
</table>
```

 ********************************************************************************
• Definitions
  - Elapsed time [sec] = (CPU + wait event)
  - CPU [sec]
  - Query (Oracle blocks e.g. 8K read from memory)
  - Disk (Oracle blocks read from disk)
  - Wait event [sec], e.g. db file sequential read
  - Rows fetched
• Example (cost of physical reads):
  - Elapsed time = 13.02 sec
  - CPU = 0.85 sec
  - Disk = 2291 blocks
  - Wait event (db file sequential read ) = 11.69 sec
  - Rows fetched = 24175
## Tuning Primer

### Data Sources – SQL Profiler

#### ArcSDE_trace (ANDREW52)

<table>
<thead>
<tr>
<th>EventClass</th>
<th>Login...</th>
<th>Application...</th>
<th>TextData</th>
<th>CPU</th>
<th>Duration</th>
<th>RowCounts</th>
<th>Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Start</td>
<td>sde</td>
<td>SDE:5912</td>
<td>&lt;showPlanXML xmlns=&quot;<a href="http://schemas....%3E">http://schemas....&gt;</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP:StmtCompleted</td>
<td>sde</td>
<td>SDE:5912</td>
<td>SELECT state_id,owner,creation_time...</td>
<td>10</td>
<td>0</td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td>SHOW PLAN XML Statistics</td>
<td>sde</td>
<td>SDE:5912</td>
<td>&lt;showPlanXML xmlns=&quot;<a href="http://schemas....%3E">http://schemas....&gt;</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP:StmtCompleted</td>
<td>sde</td>
<td>SDE:5912</td>
<td>SELECT lineage_name, time_last_mod...</td>
<td>0</td>
<td>0</td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td>SHOW PLAN XML Statistics</td>
<td>sde</td>
<td>SDE:5912</td>
<td>&lt;showPlanXML xmlns=&quot;<a href="http://schemas....%3E">http://schemas....&gt;</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP:StmtCompleted</td>
<td>sde</td>
<td>SDE:5912</td>
<td>SELECT s_eminx,s_eminy,s_emamax,s_...</td>
<td>0</td>
<td>0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>SHOW PLAN XML Statistics</td>
<td>sde</td>
<td>SDE:5912</td>
<td>&lt;showPlanXML xmlns=&quot;<a href="http://schemas....%3E">http://schemas....&gt;</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SP:StmtCompleted</td>
<td>sde</td>
<td>SDE:5912</td>
<td>SELECT s_eminx,s_eminy,s_emamax,s_...</td>
<td>521</td>
<td>2624</td>
<td>36851 11...</td>
<td></td>
</tr>
</tbody>
</table>

#### Index Scan

- **Scan a nonclustered index, entirely or only a range.**

<table>
<thead>
<tr>
<th>Physical Operation</th>
<th>Index Scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Operation</td>
<td>Index Scan</td>
</tr>
<tr>
<td>Actual Number of Rows</td>
<td>51829</td>
</tr>
<tr>
<td>Estimated I/O Cost</td>
<td>1.1442x</td>
</tr>
<tr>
<td>Estimated CPU Cost</td>
<td>0.0395x</td>
</tr>
<tr>
<td>Estimated Operator Cost</td>
<td>1.29763 (33%)</td>
</tr>
<tr>
<td>Estimated Subtree Cost</td>
<td>1.29763</td>
</tr>
<tr>
<td>Estimated Number of Rows</td>
<td>21.77</td>
</tr>
<tr>
<td>Estimated Row Size</td>
<td>596</td>
</tr>
<tr>
<td>Actual Rebinds</td>
<td>0</td>
</tr>
<tr>
<td>Actual Rewinds</td>
<td>0</td>
</tr>
<tr>
<td>Ordered</td>
<td>False</td>
</tr>
<tr>
<td>Node ID</td>
<td>3</td>
</tr>
</tbody>
</table>
Testing
Testing

**Key Tasks**

- Record user workflow based on application user requirements
- **Create single user web test**
  - Define transactions
  - Set think time and pacing based on application user requirements
  - Parameterize transaction inputs
  - Verify test script with single user
- **Create load test**
  - Define user load
  - Create machine counters to gather raw data for analysis
  - Explain load test variables
- **Execute**
Visual Studio Quick Introduction – Load Test

**Scenarios:**
- Test Mix (WebTest or Unit Test),
- Browser Mix,
- Network Mix,
- Step Loads

**Perfmon Counter Sets:**
Available categories that may be mapped to a machine in the deployment

**Run Settings:**
Counter Set Mappings – Machine metrics
Test duration
Testing
Test Data – Attribute Data

Export Feature Attribute to ASCII

Input Feature Class
C:\Data\Riverside\Riverside.gdb\Addresses

Value Field
- OBJECTID
- ADPTNUM1
- STNAME
- ADDR

Delimiter
- Comma

Output Ascii File
C:\Documents and Settings\From444\Desktop\TestData.csv
Testing
Test Data – Bbox (Using Fiddler)

Area of Interest

Selected Extent From HTTP Debugging Proxy

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>image</td>
</tr>
<tr>
<td>dbi</td>
<td>96</td>
</tr>
<tr>
<td>transparent</td>
<td>true</td>
</tr>
<tr>
<td>format</td>
<td>png8</td>
</tr>
<tr>
<td>bboxSR</td>
<td>2230</td>
</tr>
<tr>
<td>imageSR</td>
<td>2230</td>
</tr>
<tr>
<td>size</td>
<td>1222,782</td>
</tr>
</tbody>
</table>
One simple example of Python script to generate Bboxes

```python
import random

def generateBoxes(fullExtent, gridControl):
    bBoxes = []
    bBoxes.append(fullExtent)
    width = fullExtent[2] - fullExtent[0]
    for grid in gridControl:
        nWidth = width/grid
        nHeight = height/grid
        for row in range(0, grid):
            for column in range(0, grid):
                minX = fullExtent[0] + (column*nWidth)
                minY = fullExtent[1] + (row*nHeight)
                maxX = minX+nWidth
                maxY = minY+nHeight
                bBoxes.append([minX, minY, maxX, maxY])
    return bBoxes

def writeTuple(path, arr):
    try:
        f = open(path, 'w')
        for item in arr:
            f.write(','.join([str(x) for x in item])) + '
'
        f.close()
    except IOError, (errno, strerror):
        print path
        raise

if __name__ == '__main__':
    extent = [6219593.737018972, 2303882.765275147, 6231478.586248391, 2311468.277524415]
    grid = [2, 8, 16]
    bBoxes = generateBoxes(extent, grid)
    for item in bBoxes:
        print item
    writeTuple("C:\test.csv", bBoxes)
```
Testing Tools
## Testing - Selecting Web Load Test Tool

<table>
<thead>
<tr>
<th>Test Tools</th>
<th>Open Source</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadRunner</td>
<td>No</td>
<td>• Industry Leader&lt;br&gt;• Automatic negative correlations identified with service level agreements&lt;br&gt;• Http Web Testing&lt;br&gt;• Click and Script&lt;br&gt;• Very good tools for testing SOA&lt;br&gt;• Test results stored in database&lt;br&gt;• Thick Client Testing&lt;br&gt;• Can be used for bottleneck analysis</td>
<td>• High Cost&lt;br&gt;• Test Development in in C programming language&lt;br&gt;• Test metrics difficult to manage and correlate&lt;br&gt;• Poor user community with few available examples</td>
</tr>
<tr>
<td>Silk Performer</td>
<td>No</td>
<td>• Good solution for testing Citrix&lt;br&gt;• Wizard driven interface guides the user&lt;br&gt;• Can be used for bottleneck analysis</td>
<td>• Moderate to High Cost&lt;br&gt;• Test metrics are poor&lt;br&gt;• Test Development uses proprietary language&lt;br&gt;• Test metrics difficult to manage and correlate&lt;br&gt;• Poor user community with few available examples</td>
</tr>
<tr>
<td>Visual Studio Test Team</td>
<td>No</td>
<td>• Low to moderate cost&lt;br&gt;• Excellent Test Metric reporting&lt;br&gt;• Test Scripting in C# or VB .NET&lt;br&gt;• Unit and Web Testing available&lt;br&gt;• Blog support with good examples&lt;br&gt;• Very good for bottleneck analysis</td>
<td>• No built in support for AMF&lt;br&gt;• No Thick Client options&lt;br&gt;• Moderate user community</td>
</tr>
<tr>
<td>JMeter</td>
<td>Yes</td>
<td>• Free&lt;br&gt;• Tool</td>
<td>• Provides only response times&lt;br&gt;• Poor User community with few available examples</td>
</tr>
</tbody>
</table>
Validate Test
Tips and Tricks
Results Validation

• Compare and correlate key measurements
  - Response Time (increasing, higher than initially profiled for single user)
  - Throughput
  - CPU on all tiers
  - Network on all tiers
  - Disk on all tiers
  - Passed tests
  - Failed test
Results Validation

- Lack of errors does not validate a test
  - Requests may succeed but return zero size image
  - Spot check request response content size
Results Validation - Reporting

- Exclude failure range, e.g. failure rate > 5% from the analysis
- Exclude excessive resource utilization range
Results Validation

Incorrect CPU and Response Time Correlation

Unexpected curve shape:
Response time should be increasing.
Likely root cause:
failed or 0 size image requests.
Results Validation

Correct CPU and Response Time Correlation

Max CPU utilization

Response Time increase correlates with:
- User load
- CPU utilization
Results Validation

Test failure due to w3wp memory bottleneck

Root cause: W3wp Web Server process

Symptom: System available memory is decreasing
Results Validation

*Unexpected CPU utilization while accessing cache services*

Unexpected bottleneck in cache application:
- 100% CPU utilization.

How to determine root cause?
Reporting Results

Determining system capacity

- Maximum number of concurrent users corresponding to, e.g.:
  - Maximum acceptable response time
  - First failure or 5%
  - Resource utilization greater than 85%, for example CPU

- Different ways of defining acceptance criteria (performance level of service), e.g.
  - 95% of requests under 3 sec
  - Max request under 10 sec
Configuration validation

- Application stack is not properly configured and tuned
- Single error in a test can lead to a cascade of failing requests
- Test client machine is bottleneck
  - CPU
  - Memory
  - Networks (cache test in particular)
Contact us

• Professional Services
  - [http://www.esri.com/services/professional-services/implementation/request-services.html](http://www.esri.com/services/professional-services/implementation/request-services.html)
  - profservices@esri.com

• Andrew Sakowicz, asakowicz@esri.com

• Your account manager
Questions?
Demo Related and Supplemental Slides
Jmeter
Testing with JMeter

- Riverside Electric MSD Service
  - Testing REST Export Map
Testing with JMeter– Test Data from CSV

CSV Data Set Config

- Name: CSV Data
- Comments:
- Configure the CSV Data Source:
  - Filename: RiversideElectric.csv
  - File encoding:
  - Variable Names (comma-delimited): MinX, MinY, MaxX, MaxY
  - Delimiter (use 't' for tab):
  - Allow quoted data?: False
  - Recycle on EOF?: True
  - Stop thread on EOF?: False
  - Sharing mode: All threads
Testing with JMeter – Set Thread Properties
Testing with JMeter – Create Step Load
Testing with JMeter - Results
## Testing with JMeter - Results

### Aggregate Report

<table>
<thead>
<tr>
<th>Label</th>
<th># Samples</th>
<th>Average</th>
<th>Median</th>
<th>90% Line</th>
<th>Min</th>
<th>Max</th>
<th>Error %</th>
<th>Throughput</th>
<th>KB/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 User</td>
<td>1110</td>
<td>152</td>
<td>141</td>
<td>161</td>
<td>73</td>
<td>3289</td>
<td>0.00%</td>
<td>6.5/sec</td>
<td>136.2</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1110</td>
<td>152</td>
<td>141</td>
<td>161</td>
<td>73</td>
<td>3289</td>
<td>0.00%</td>
<td>6.5/sec</td>
<td>136.2</td>
</tr>
</tbody>
</table>
Desktop Testing
Several tools can be used to automate thick client workflows:
- Citrix Edge Sight
- HP LoadRunner
- Borland’s Silk Performer
- NRG’s AppLoader
Better choices are text indicators that eventually have a definitive ending (such as text stating “complete”, or no text at all)
Capturing Transactions in ArcMap Using Events

Measuring pan and zoom transactions

- **WaitBitmapAppear:**
  - lower left status text continuously update when drawing occurs, and will become blank when ended
  - Testing software waits until the bitmap is matched before proceeding.
  - Specify a search area when capturing the bitmap.
  - Keep a search area small
  - Animated icons or other moving status indicators are particularly difficult to implement (typical example is the spinning globe in ArcMap)
  - Better choices are text indicators that eventually have a definitive ending (such as text stating “complete”, or no text at all)
Capturing Transactions in ArcMap Using Events

Measuring pan and zoom transactions (Apploader tool example)
Capturing Transactions in ArcMap Using Events

Measure opening ArcMap document (Apploader tool example)

- Wait for Window
Capturing Transactions in ArcMap Using Events

Open ArcMap, Zoom, Close ArcMap

1. Comment: Open ArcMap
2. Launching ArcMap via specified MXD
3. Waiting for the ArcMap window to appear
4. Waiting for captured bitmap to appear. This bitmap is a blank status area indicating all drawing has completed.
5. Comment: Zoom
6. Move the mouse to the map scale field
7. Click the map scale field
8. Type the scale we wish to zoom to
9. Hit enter, initiating the map zoom
10. Comment: Start Zoom Transaction
11. Changing the Event Interval to 50ms for better
ArcGIS ADF Testing
Using Visual Studio for web testing ADF

• Creating web test script
  1. VS Recorder
     - might not detect all parameters for ADF
     - must validate with Fiddler or other tools
     - may have to edit registry entries
  2. Fiddler and export to VS Web test
     - Deselect appropriate plug-in
  3. Automated solutions (especially useful for ADF testing), e.g. Fiddler custom plug-in

• Declarative vs. Code web test
  - Looping and if logic can be implemented only in coded web test
Scripting ADF applications - Challenges

• Stateful application
  - Test scripts must manage the state
  - Most requests contain session based or dynamic GUID
    - Map Images
    - Scale bar
    - North Arrow
    - Search task

• Parameterize request
Creating Transaction Scripts – manual approach

- Most load testing tools include an HTTP debugging proxy to model user workflows. The output from the HTTP Debugging proxy is raw text or another format that represents the Web Client requests.
- When recording user workflows, use transaction markers between the transactions. For example, initial page load, zoom, and pan are transactions.
- The transaction marker is also the correct place to insert think time as a user would naturally pause after a transaction.
Creating Transaction Scripts – manual approach

- Using the transaction markers we will define GIS transactions.
- The content returned from the initial page request will contain dynamic images that must be captured, formatted properly, and sent back to the server in the form of an HTTP GET.
- All references to HTTP GET with ESRI.ArcGIS.Web.MimelImage.ashx must be altered to reflect the new session
- Transaction markers must be removed
Creating Transaction Scripts – Fiddler manual

• Using Fiddler, an HTTP Debugging proxy, record the desired Web client workflow

• Be sure to include the transaction markers in your workflow.

• When you have completed recording the defined workflow, select all (Ctrl-A) and go to File->Save->Session(s)->as Visual Studio Web Test and save the raw HTTP traffic to fiddler.webtest. You will be prompted to select VS2008 plugins on export. Please select the following options:

![Select Plugins window](image)
Creating Transaction Scripts – VS Web Test

- Declarative webtest
Creating Transaction Scripts – VS Web Test manual

- Group requests into transactions

- Define extraction rules for each web request that returns dynamic images.
  - Extraction rules define search patterns used to parse web requests, and the results of the search are held in a variable defined in extraction rule.
Creating Transaction Scripts – VS Web Test
manual extraction Rules

Add Extraction Rule

Select a rule:
- Extract Attribute Value
- Extract Form Field
- Extract HTTP Header
- Extract Regular Expression
- Extract Text
- Extract Hidden Fields

Properties for selected rule:

Options
Context Parameter Name

Parameters
Regular Expression
Ignore Case (False)
Required (True)
Index (0)

Description for selected rule:
Extract text from the response matching a regular expression and place it into the test context.
### Creating Transaction Scripts – VS Web Test manual extraction rules for ADF testing

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContextParameter Name</td>
<td>t0-i0</td>
<td>This item represents a variable that will be set when the extraction rule is executed on the response of an HTTP request. I recommend following the convention to the left. T0 represents transaction0 and i0 represent image0. If there is more than one request coming back in the response of the server or the define transaction has multiple dynamic images the Context Parameter name will simply be advanced. For Example, t0-i2.</td>
</tr>
<tr>
<td>Regular Expression</td>
<td>ImgID=(((0-9a-fA-F){8})-((0-9a-fA-F){4})-((0-9a-fA-F){4})-((0-9a-fA-F){12})</td>
<td>Dynamic images will have this signature, and the regular expression will find multiple images if they are present in the HTTP response from the server.</td>
</tr>
<tr>
<td>Ignore Case</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>0</td>
<td>The regular expression will return an array of item that matches the search criteria. This number represents the index of the array that must be dereferenced.</td>
</tr>
</tbody>
</table>
Now, we have defined all of the extraction rules for the first transaction.

Next, we must find the static requests from the initial HTTP Debugging proxy and modify them to be dynamic.

Below is an example of a static request for a mime image that must be modified.

We will delete all of the query string parameters, but we must leave the headers alone.
Creating Transaction Scripts – VS Web Test
manual extraction rules for ADF testing

• Now, we will use the variable defined by the Context Parameter Name in the extraction rules

• As you can see, we added the {{t0-i0}} to the correct request parameter. This modification to the request will dereference the variable that has been set when the extraction rule is executed.
Creating load test – Visual Studio

Select a load pattern for your simulated load:

- **Constant Load:**
  - User Count: 25 users

- **Step load:**
  - Start user count: 1 user
  - Step duration: 120 seconds
  - Step user count: 1 user/step
  - Maximum user count: 4 users
Creating load test – Visual Studio

Add tests to a load test scenario and edit the test mix

<table>
<thead>
<tr>
<th>Test Name</th>
<th>%</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>PortlandDynamicTemplate-Final</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Total: 100
Creating load test – Visual Studio

Add network types to a load test scenario and edit the network mix

<table>
<thead>
<tr>
<th>Network Type</th>
<th>%</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Total 100
Creating load test – Visual Studio
While the load test is running, various system components should be monitored including the CPUs, disk subsystem, memory, and network. System monitoring applies to all the systems that play a part in the test configuration, including the database server. At a minimum, the following core components should be monitored during the tests:

- CPU Utilization
- Memory
- Disk Utilization
- Network Utilization
Creating load test – Visual Studio
Creating load test – Visual Studio
Executing load test – Visual Studio

Test in progress... 2 threshold violations... 24 errors

Summary
- Configuration: Local run
- Sampling Rate: 0.05
- Requests: Total Requests: 910, Requests/sec: 3.71, Failed Requests: 0
- Test Cases: Total Tests: 1, Total Sec: 0.0041, Failed Tests: 0

Graph: Default

% Processor Time

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:40</td>
<td>1.2</td>
</tr>
<tr>
<td>01:45</td>
<td>3.4</td>
</tr>
<tr>
<td>01:50</td>
<td>5.3</td>
</tr>
<tr>
<td>01:55</td>
<td>1.5</td>
</tr>
<tr>
<td>02:00</td>
<td>0.4</td>
</tr>
<tr>
<td>02:05</td>
<td>2.1</td>
</tr>
<tr>
<td>02:10</td>
<td>2.7</td>
</tr>
<tr>
<td>02:15</td>
<td>1.4</td>
</tr>
<tr>
<td>02:20</td>
<td>0.2</td>
</tr>
<tr>
<td>02:25</td>
<td>2.6</td>
</tr>
<tr>
<td>02:30</td>
<td>2.7</td>
</tr>
<tr>
<td>02:35</td>
<td>28.1</td>
</tr>
<tr>
<td>02:40</td>
<td>9.0</td>
</tr>
<tr>
<td>02:45</td>
<td>25.9</td>
</tr>
<tr>
<td>02:50</td>
<td>26.3</td>
</tr>
<tr>
<td>02:55</td>
<td>3.4</td>
</tr>
<tr>
<td>03:00</td>
<td>13.4</td>
</tr>
<tr>
<td>03:05</td>
<td>10.7</td>
</tr>
<tr>
<td>03:10</td>
<td>13.2</td>
</tr>
<tr>
<td>03:15</td>
<td>11.0</td>
</tr>
<tr>
<td>03:20</td>
<td>14.5</td>
</tr>
<tr>
<td>03:25</td>
<td>13.0</td>
</tr>
<tr>
<td>03:30</td>
<td>6.8</td>
</tr>
<tr>
<td>03:35</td>
<td>7.1</td>
</tr>
<tr>
<td>03:40</td>
<td>16.2</td>
</tr>
<tr>
<td>03:45</td>
<td>12.7</td>
</tr>
<tr>
<td>03:50</td>
<td>6.9</td>
</tr>
<tr>
<td>03:55</td>
<td>7.0</td>
</tr>
<tr>
<td>04:00</td>
<td>4.1</td>
</tr>
<tr>
<td>04:05</td>
<td>13.6</td>
</tr>
</tbody>
</table>

00:00 00:10 00:20 00:30 00:40 00:50 01:00 01:10 01:20 01:30 01:40 01:50 02:00 02:10 02:20 02:30 02:40 02:50 03:00 03:10 03:20 03:30 03:40 03:50 04:00 04:10 04:20