
# *ArcGIS Server Performance and Scalability - Testing Methodologies*

***Andrew Sakowicz***

***Frank Pizzi***

# Introductions

- Who are we?
  - Enterprise Implementation
- Who are you?
  - New to ArcGIS Server?
  - Currently deploying 9.3?
  - Currently deploying IMS?

# Audience

- Testers
- Developers
- Project Manager

# Level

- **Beginner / Intermediate**

# Schedule and Overview


- Performance and Scalability Testing methodology
  - Web applications (step by step)
  - ArcGIS Desktop Applications
  - Exploratory tests
- Tips and Tricks

Question and answer period.

*Please complete the session survey!*

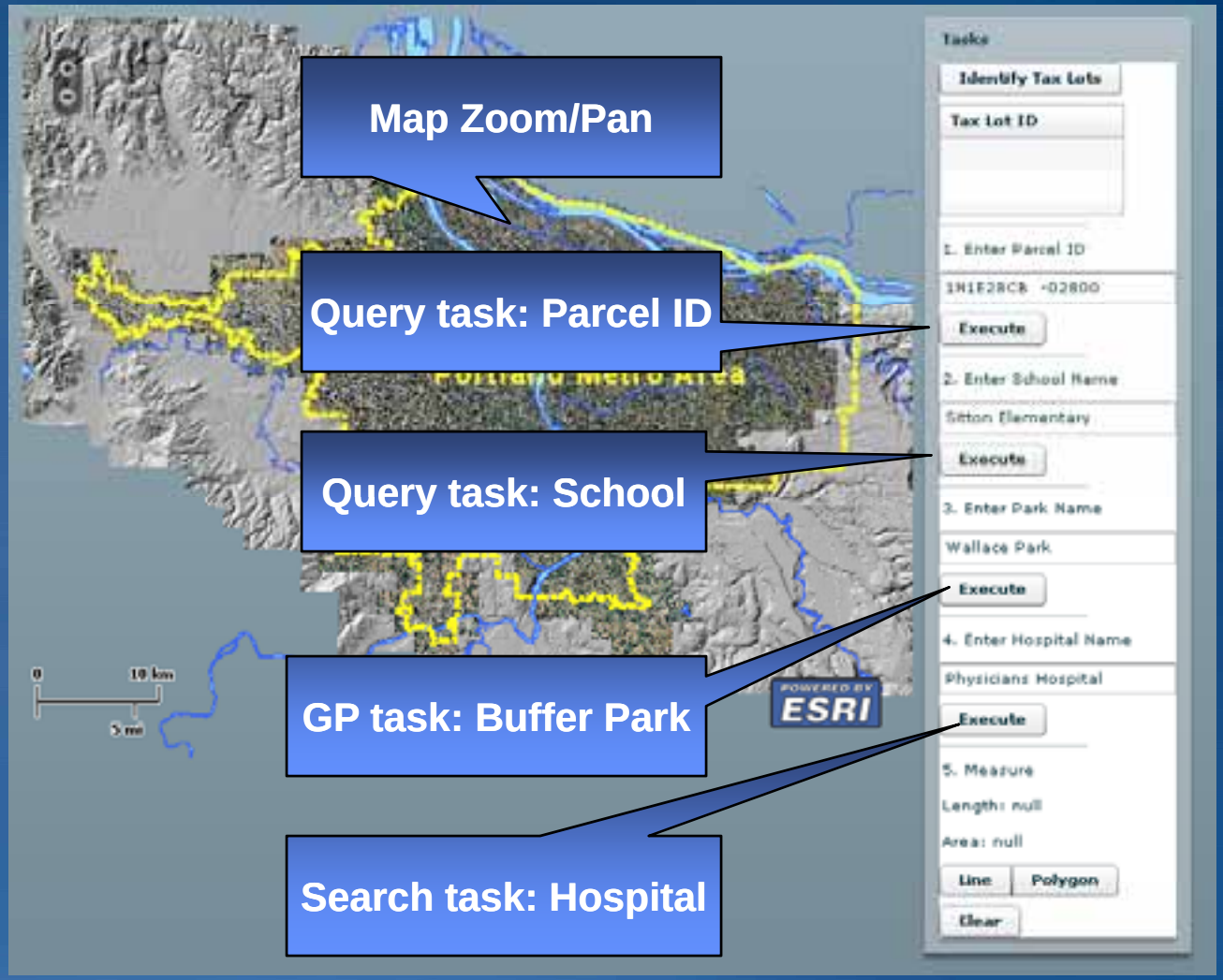
# Presentation Objectives

- Teaching Step by Step testing (a simple web Flex API)
  - Recording and creating transactions scripts
  - Creating load test
  - Executing
  - Analyzing Results
- Managing testing efforts
  - Test Plan, skills, level of effort
  - Best practices
  - Common pitfalls

# Demo

***Example of ArcGIS Server Flex application used for parcel management by local government***

**Flex application with dynamic data. Application provides the search and query functions as well as Geoprocessing**



# Demo
## Define user workflow, transactions and think time

- Define user workflow to be tested
- Define and name GIS transactions
  - Initial page of application
  - Zoom map 1
  - Zoom map 2
  - Pan map 1
  - Pan map 2
  - Query tax lot with zoom to tax lot
  - Query school with zoom to school
  - Query hospital with zoom to
  - Geoprocessing
- Define user think time between transactions

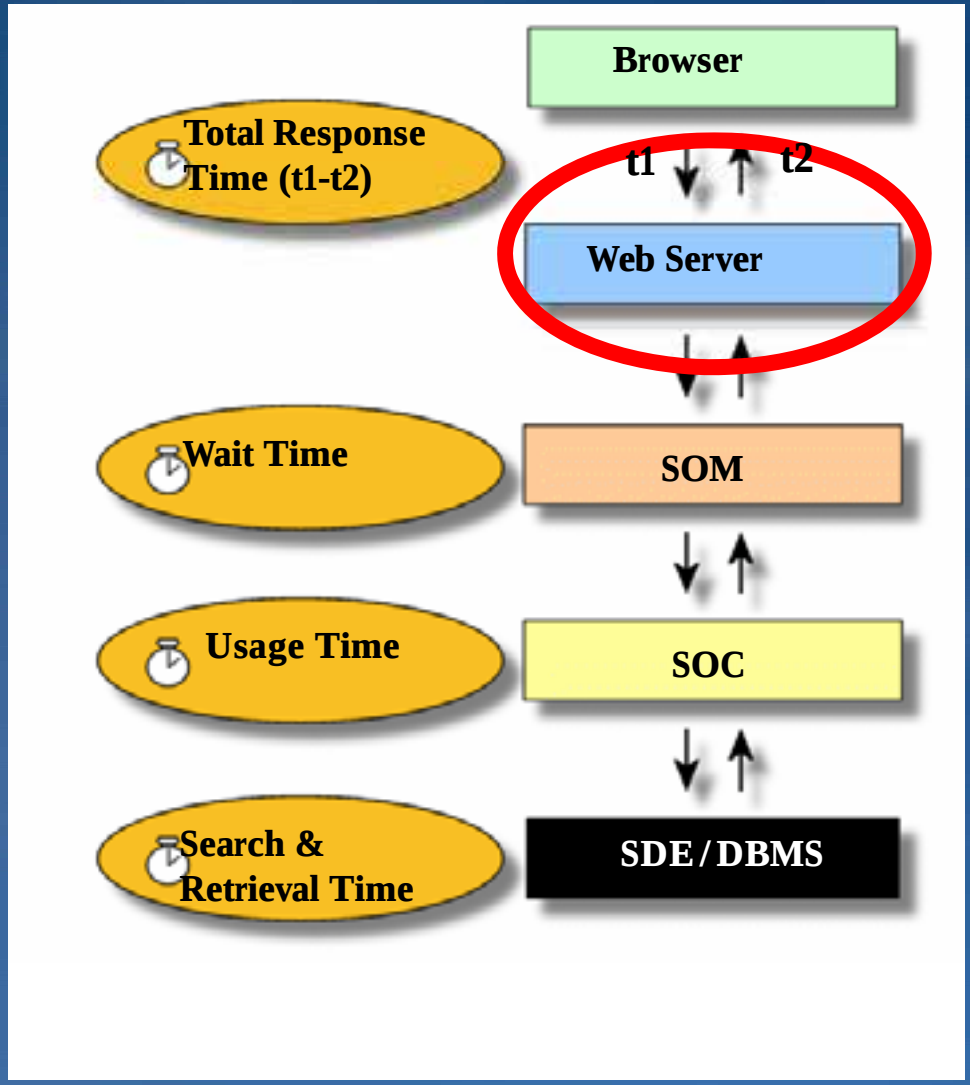
# Demo – Testing tasks

- 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 variable load test variables
- Execute

# Demo

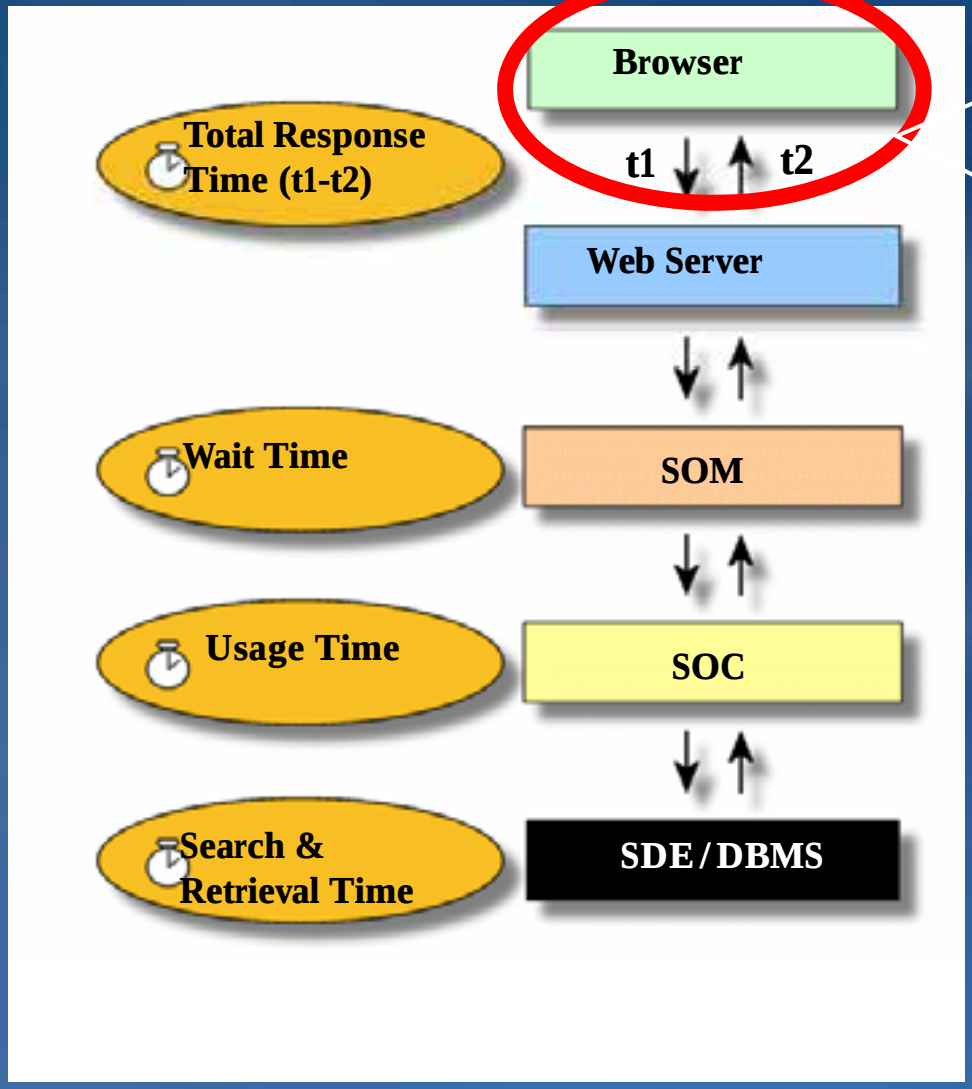
# What is measured during a typical web load test

***Performance measured at web server tier, not a web browser – not a true user experience***



# How to measure true web user performance experience?

***Measure at web browser, e.g. stop watch or programmatically using API events***

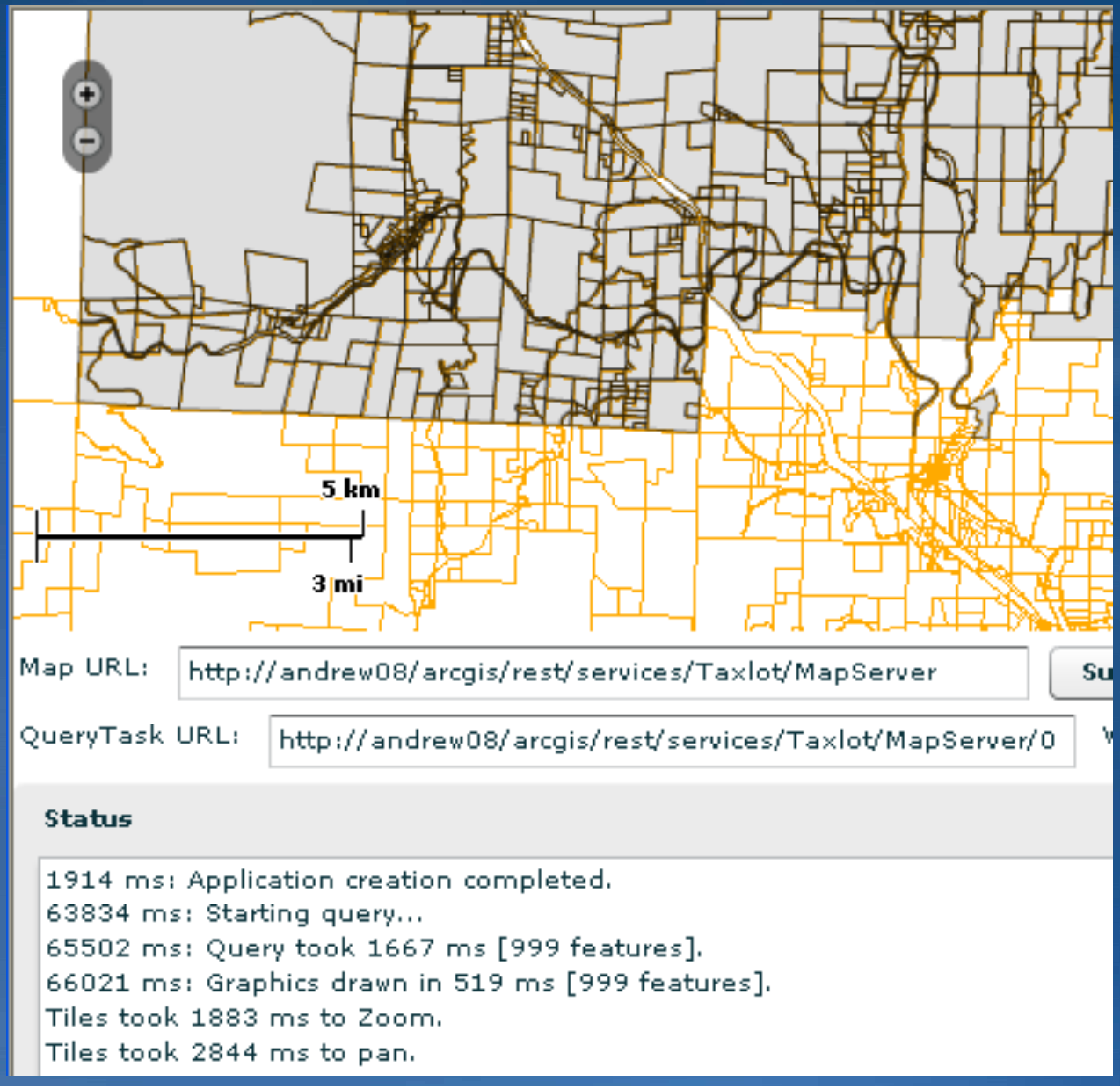



A test is executed at the web browser.

It measures web browser call's elapsed time (roundtrip between browser and data source)

# How to measure true web user performance experience?

***Measure at web browser, e.g. stop watch or programmatically using API events***



# How to measure true web user performance experience?

***Measuring map rendering at web browser using API events***

```
private function onCreationComplete():void
{
    logger.text += getTimer() + ' ms: Application creation completed.\n';
    myThematicLayer.addEventListener(Event.COMPLETE, onTilesLoaded)
    myMap.addEventListener(PanEvent.PAN_START, onPanStart);
    myMap.addEventListener(ZoomEvent.ZOOM_START, onZoomStart);
}
```

```
private function onTilesLoaded(event:Event):void
{
    if (panStartFlag){
        TimeTileComplete = getTimer() - TimeTileStart ;
        TimeTileStart = 0
        logger.text += "Tiles took " + TimeTileComplete  + " ms to pan.\n";
        panStartFlag = false;
    }
    if (zoomStartFlag){
        TimeTileComplete = getTimer() - TimeTileStart ;
        TimeTileStart = 0
        logger.text += "Tiles took " + TimeTileComplete  + " ms to Zoom.\n";
        zoomStartFlag= false;
    }
}
```

# How to measure true web user performance experience?

***Measuring client-side graphics at web browser using API events***

```
private function doThematicQuery():void
{
    myGraphicsLayer.clear();
    queryTask.url = textQueryURL.text.toString();
    logger.text += getTimer() + " ms: Starting query...\n";
    TimeQueryStart = getTimer();
    queryTask.execute( query, new AsyncResponder( onResult, onFault ));
    myGraphicsLayer.addEventListener(FlexEvent.UPDATE_COMPLETE,graphicsLayerUpdated);
    function onResult( featureSet : FeatureSet, token : Object = null ) : void
    {
        TimeQueryDone = getTimer();
        var TimeQueryDiff:uint = getTimer() - TimeQueryStart;
        logger.text += getTimer() + " ms: Query took " + TimeQueryDiff + " ms";
        logger.text += " [" + myGraphicsLayer.numGraphics + " features].\n";
    }
```

# Resources

*http://resources.esri.com/help/9.3/arcgisserver/apis/flex/samples/index.html*

- The ESRI Flex and JavaScript APIs are straight forward to test
- Flex and JavaScript APIs utilize the REST API to access GIS data
- Understanding the REST API Interface will provide insight in how to approach testing
  - Map Service
  - Geocode Service
  - GP Service
  - Geometry Service
  - Image Service


*Good resource to practice test scripting*

# Managing testing efforts

# Performance Testing Objectives

- Benchmarking to demonstrate performance improvements
- Part of formal acceptance testing
- Capacity planning (server sizing)
- Validating system architecture design

# Performance Testing Benefits

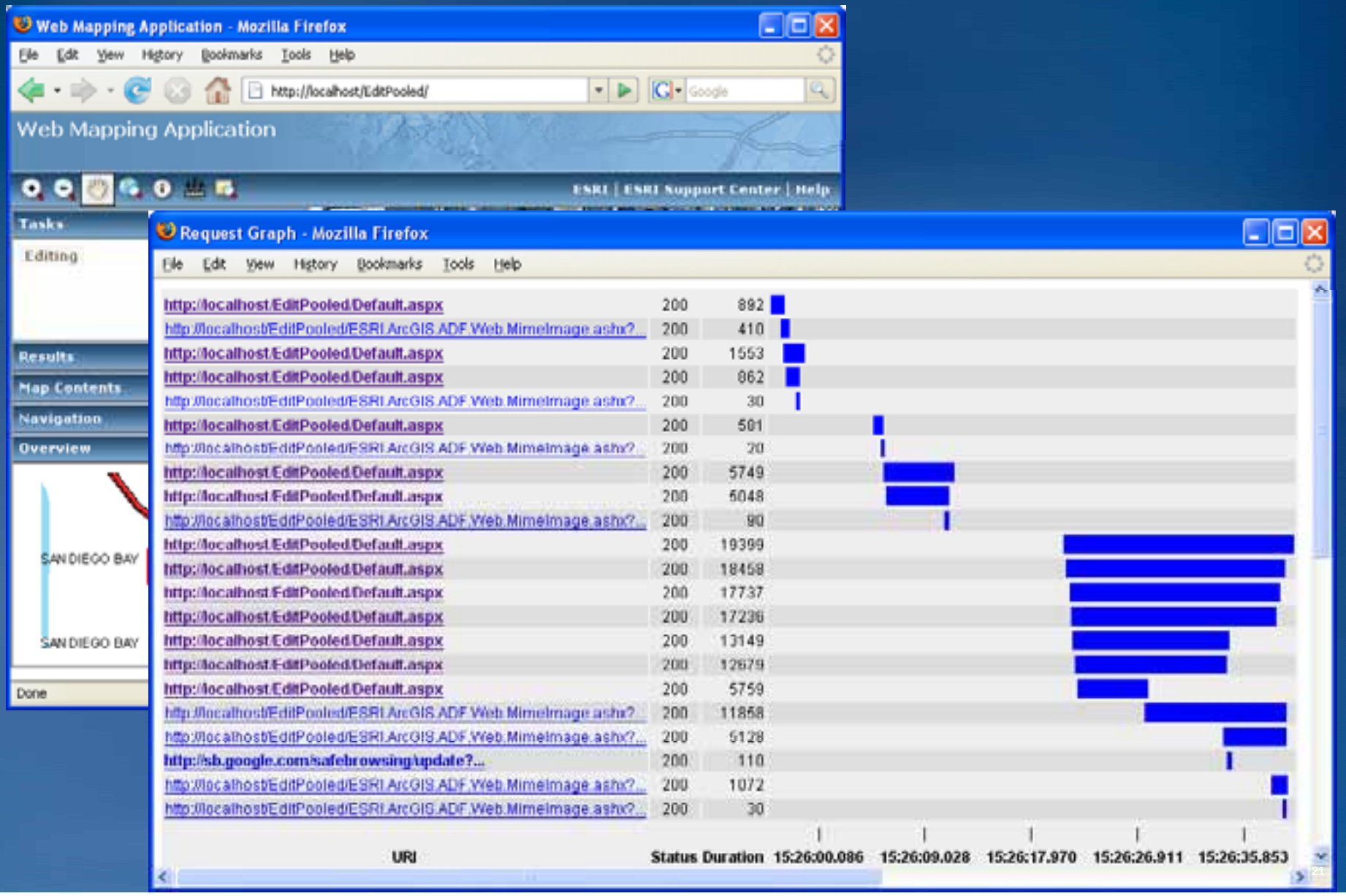
- Less costly to fix the problems at the earlier development stage
- Proactive resolution of issues before they impact on end users
- Identify bottlenecks and/or underutilized resources for redeployment

# Performance Testing Methodology

- Ensure application functional testing is complete
- Tune your system prior to testing
- Develop test plan
- Create test scripts
- Analyze results

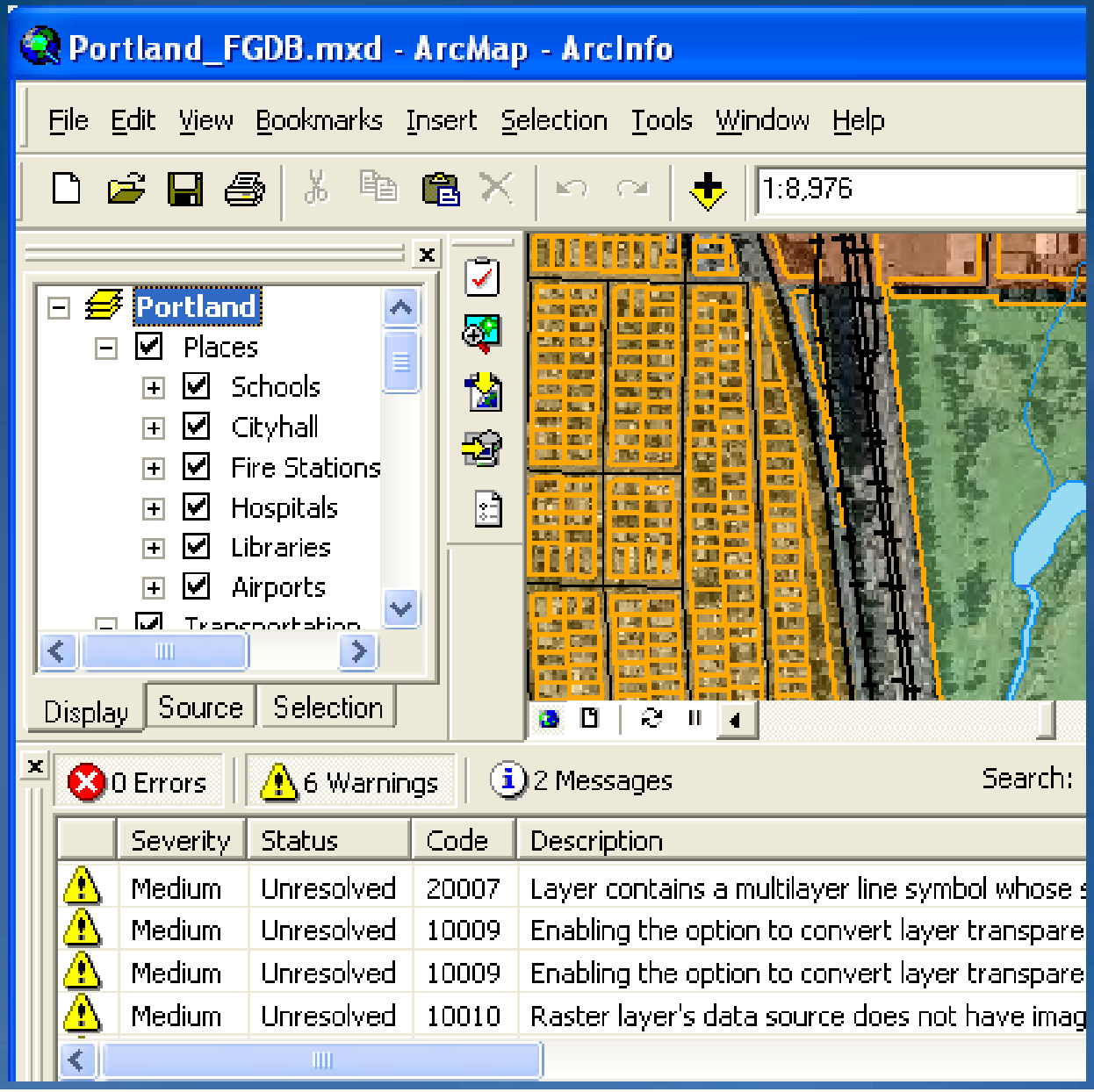
# Performance Testing Methodology – Tuning

***Analyze  performance of each request***



# Performance Testing Methodology – Tuning example

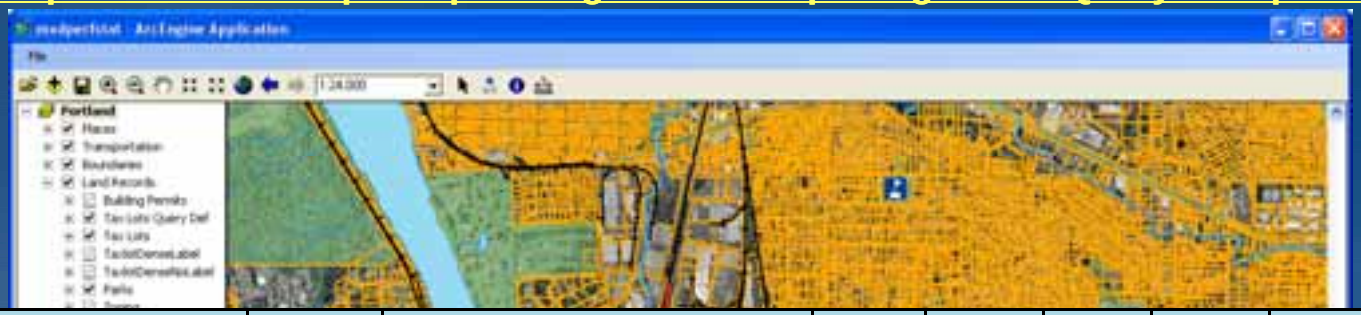
***Optimize mxd with new ArcMap 9.3.1 Analyze Tool***



# Performance Testing Methodology – Tuning example

***Optimize mxd with mxdperfstat***

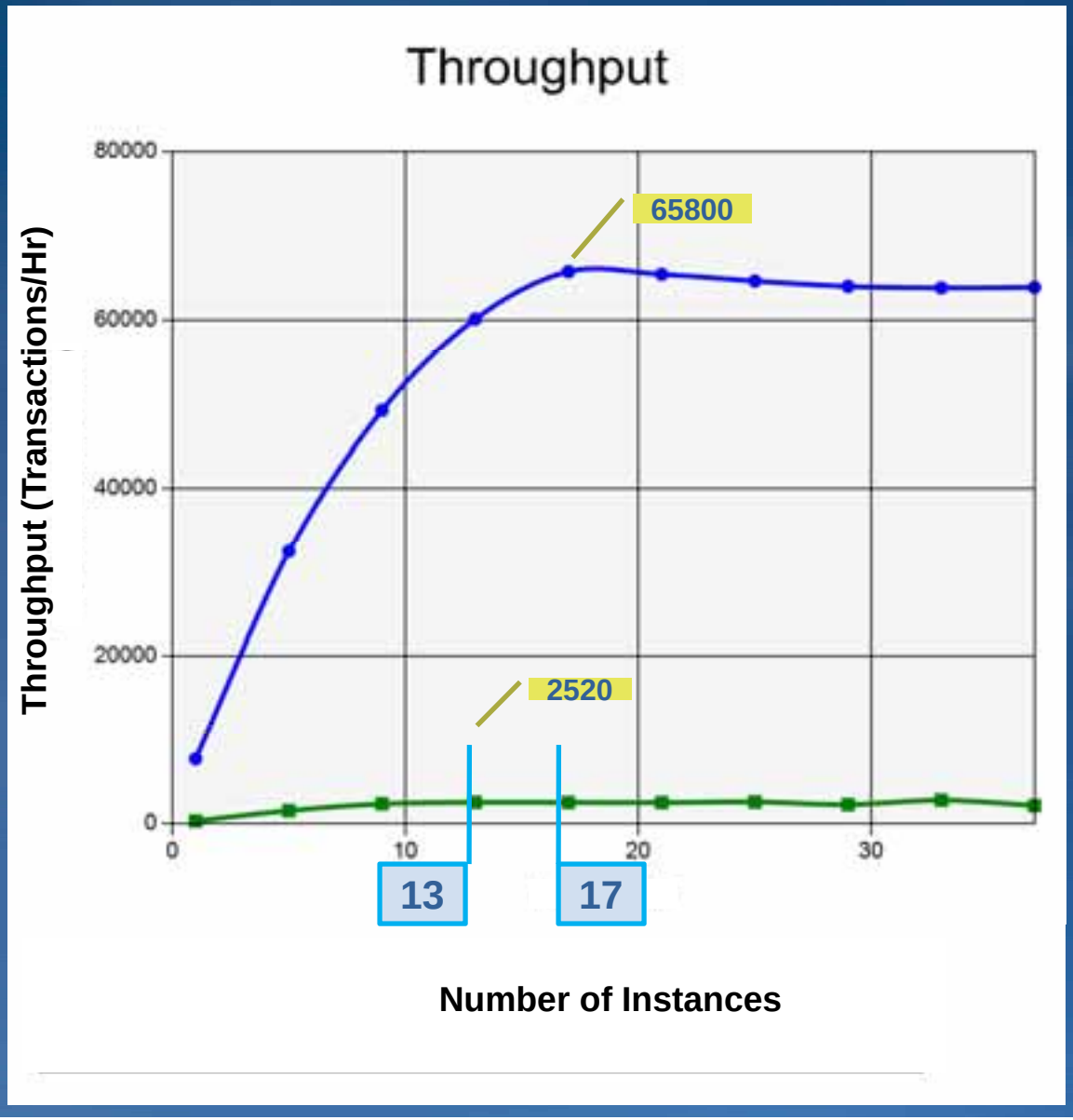
**http://arcscripts.esri.com/scripts.asp?eLang=&eProd=&perPage=10&eQuery=mxdperfstat**



| Item | At Scale | Layer Name | Refresh Time (sec) | Recommendations | Features | Vertices | Labeling | Geography Phase (sec) | Graphics Phase (sec) | Cursor Phase (sec) | DBMS CPU | DBMS LIO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 41 | 1,000 | TaxlotDenseLabel | 1.93 | 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; | 1 | 200,001 | TRUE | 0.45 | 1.42 | 1.04 | 0.02 | 266 |
| 42 | 1,000 | TaxlotDenseNoLabel | 0.53 | simplify geometry: vertices fetched=200001; | 1 | 200,001 | FALSE | 0.45 | 0.02 | 0.9 | 0.02 | 140 |

# Performance Testing Methodology – Tuning example

***Optimize ArcMap document and configure correct number of ArcGIS Server instances***




**Corrected**
**(Unloaded TT: 0.34 sec)**
**(2.1 Instances/core)**

**Uncorrected**
**(Unloaded TT: 11.97 sec)**
**(1.6 Instances/core)**

# When?

- Start performance testing early in the development cycle
  - Unit test

# Test Plan

- Understand the effort required to develop and ensure testing is valid:
  - Domain experts
  - Hardware
  - Time
  - Skills
- Document user workflows
- Document configuration

# Selecting Web Load Test Tool

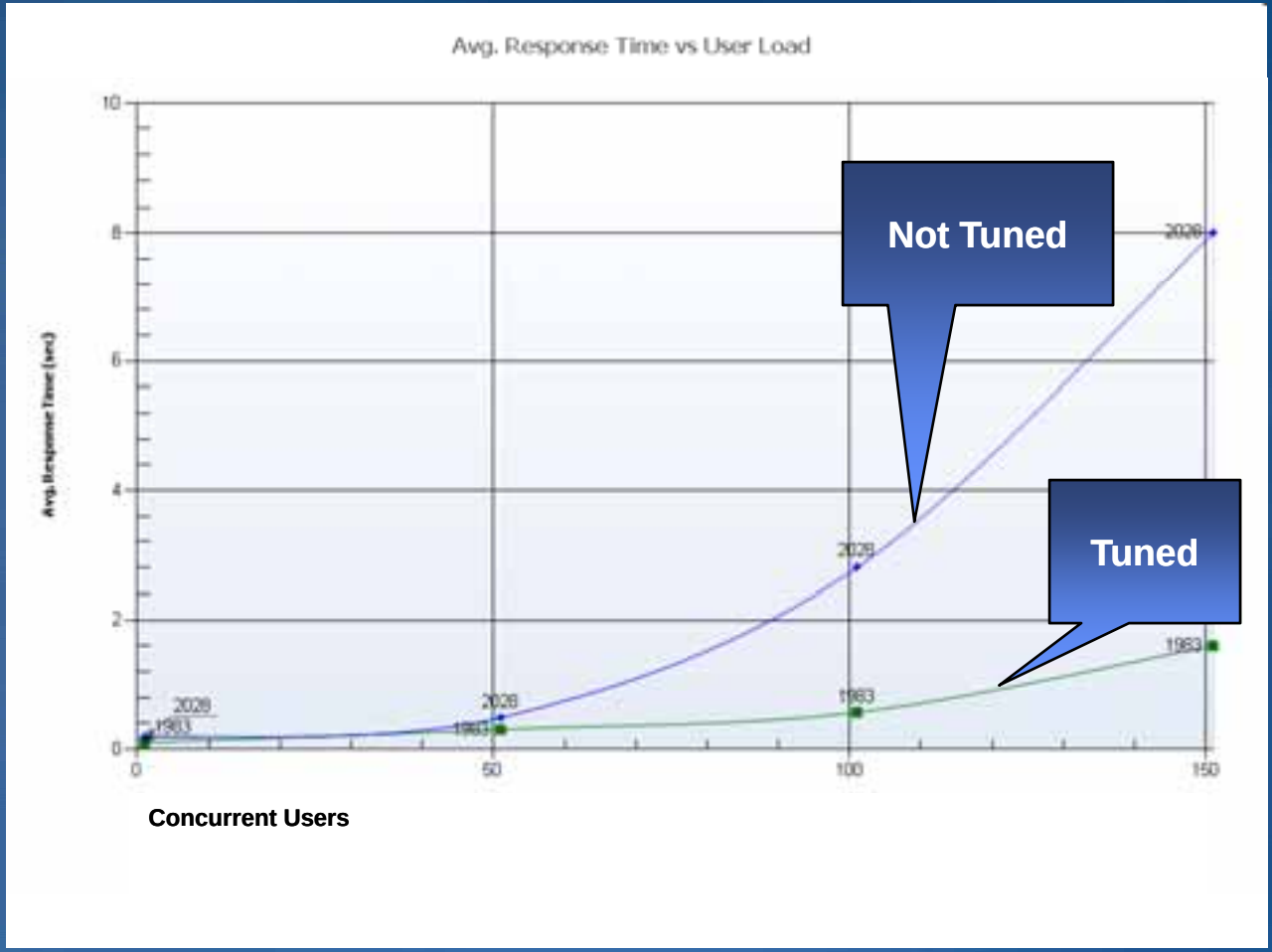
| Test Tools | Open Source | Pros | Cons |
|---|---|---|---|
| LoadRunner | No | •Industry Leader<br>•Automatic negative correlations identified with service level agreements<br>•Http Web Testing<br>•Click and Script<br>•Very good tools for testing SOA<br>• Test results stored in database<br>•Thick Client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

# Execute

- Ensure
  - Only target applications are running
  - Application data is in the same state for every test
  - Good configuration management is critical to getting consistent load test results

# Analyze Results

***Expected Response Time Curve***



# Analyze 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**

# Analyze Results

***Validation***

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

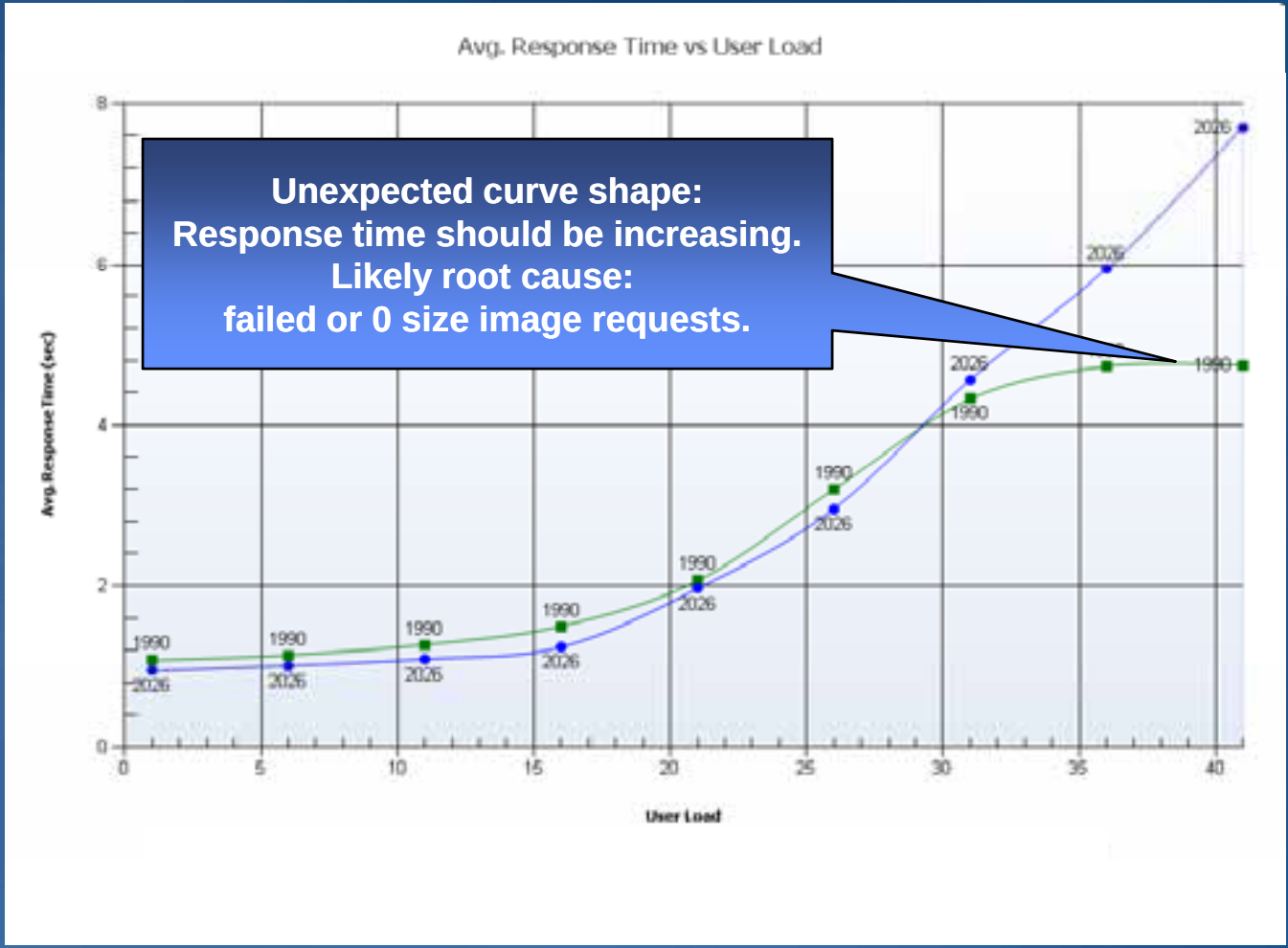
# Analyze Results

***Reporting and Analysis***

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

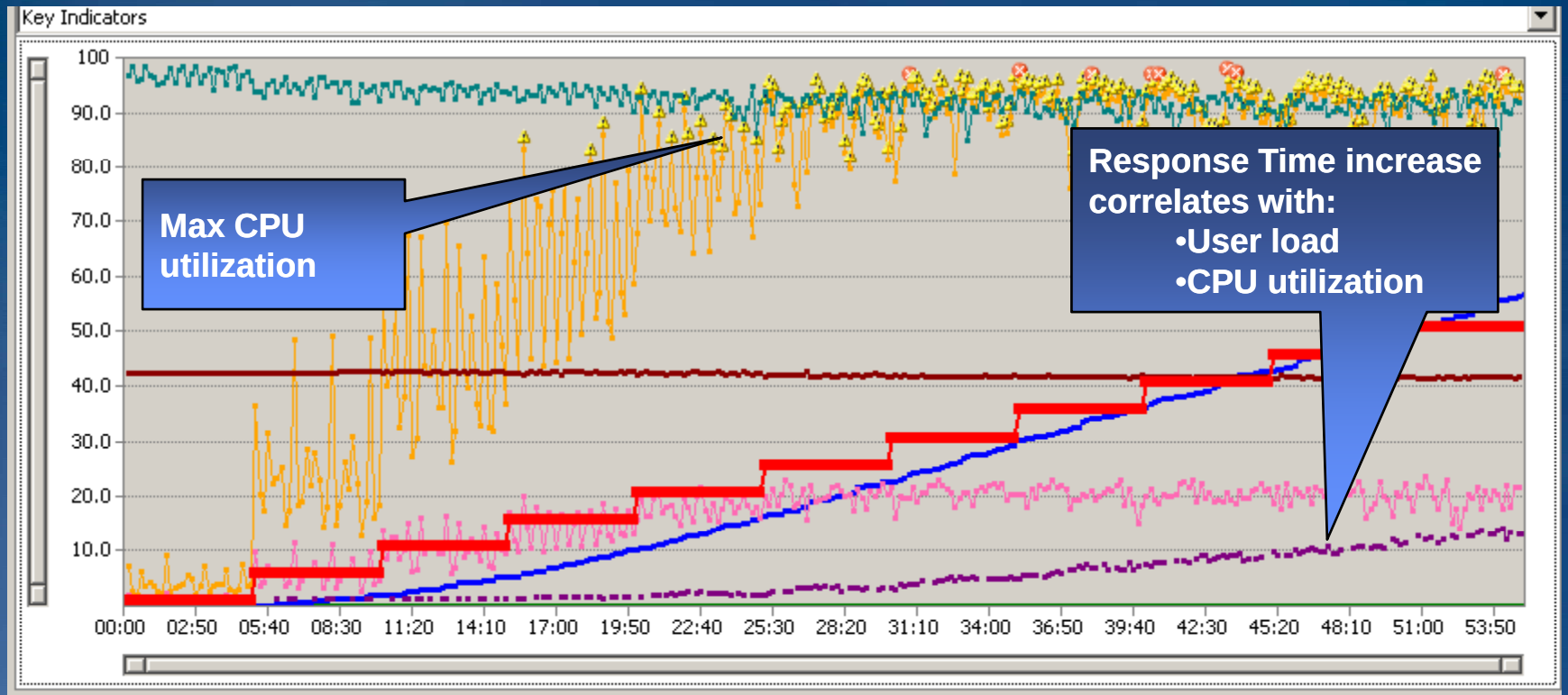
# Analyze Results

***Validation Example***



# Analyze Results

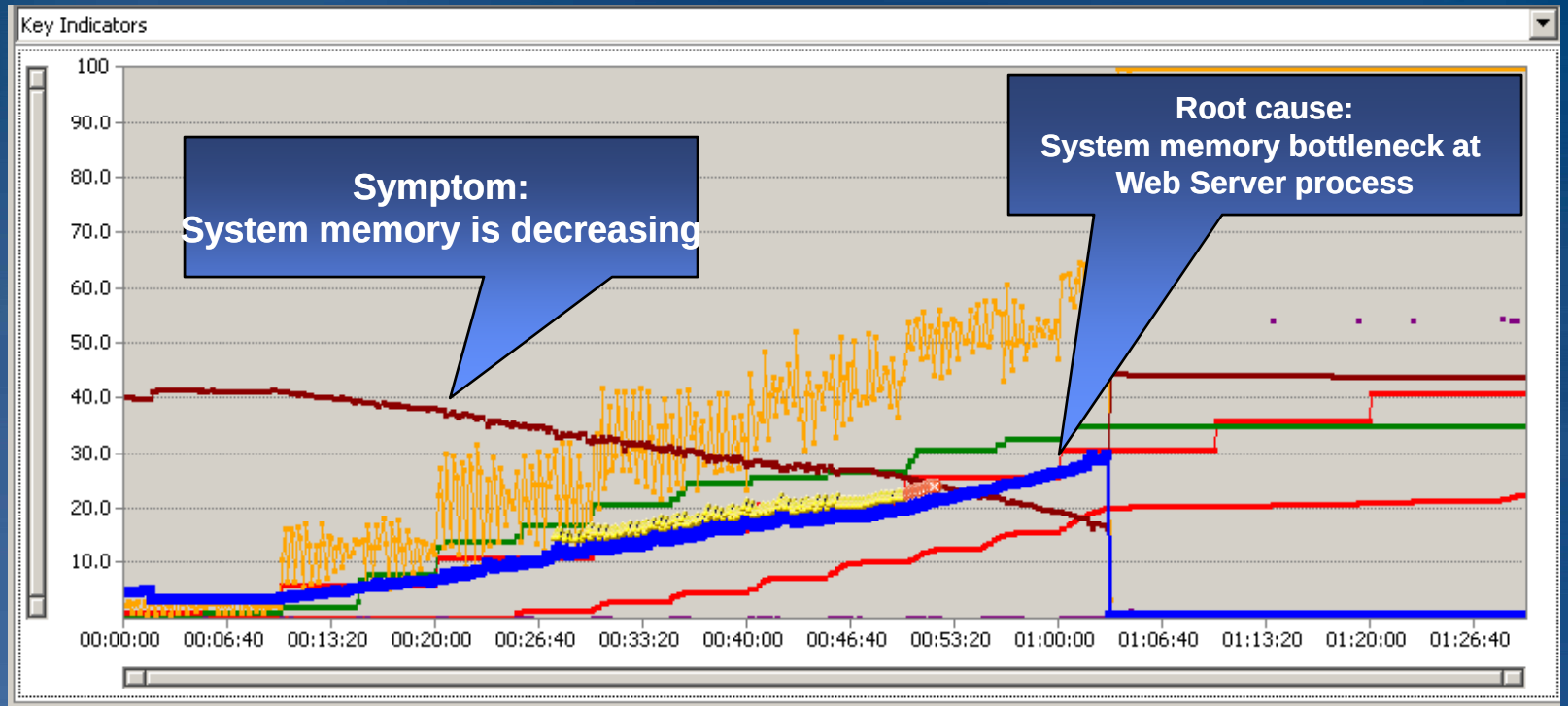
***Expected results example***




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg |
|---|---|---|---|---|---|---|---|---|
| ☑ User Load | _Total | LoadTest:Scenar | ESLWS9 | | 100 | 1 | 51 | 26 |
| ☑ Failed Tests | _Total | LoadTest:Test | ESLWS9 | | 0 | 0 | 0 | 0 |
| ☑ Passed Tests | _Total | LoadTest:Test | ESLWS9 | | 1,000 | 0 | 568 | 223 |
| ☑ Avg. Response Time | _Total | LoadTest:Transa | ESLWS9 | | 100 | 1.27 | 14.2 | 6.13 |
| ☑ % Processor Time | _Total | Processor | ESLSRV9 | | 100 | 2.27 | 96.2 | 68.4 |
| ☑ % Idle Time | _Total | PhysicalDisk | ESLSRV9 | | 100 | 82.6 | 98.9 | 93.0 |
| ☑ Bytes Received/sec | Broadcom BCM57 | Network Interfac | ESLSRV9 | | 1,000,000 | 6,632 | 238,324 | 158,332 |
| ☑ Available MBytes | - | Memory | ESLSRV9 | | 10,000 | 4,172 | 4,320 | 4,242 |

# Analyze Results

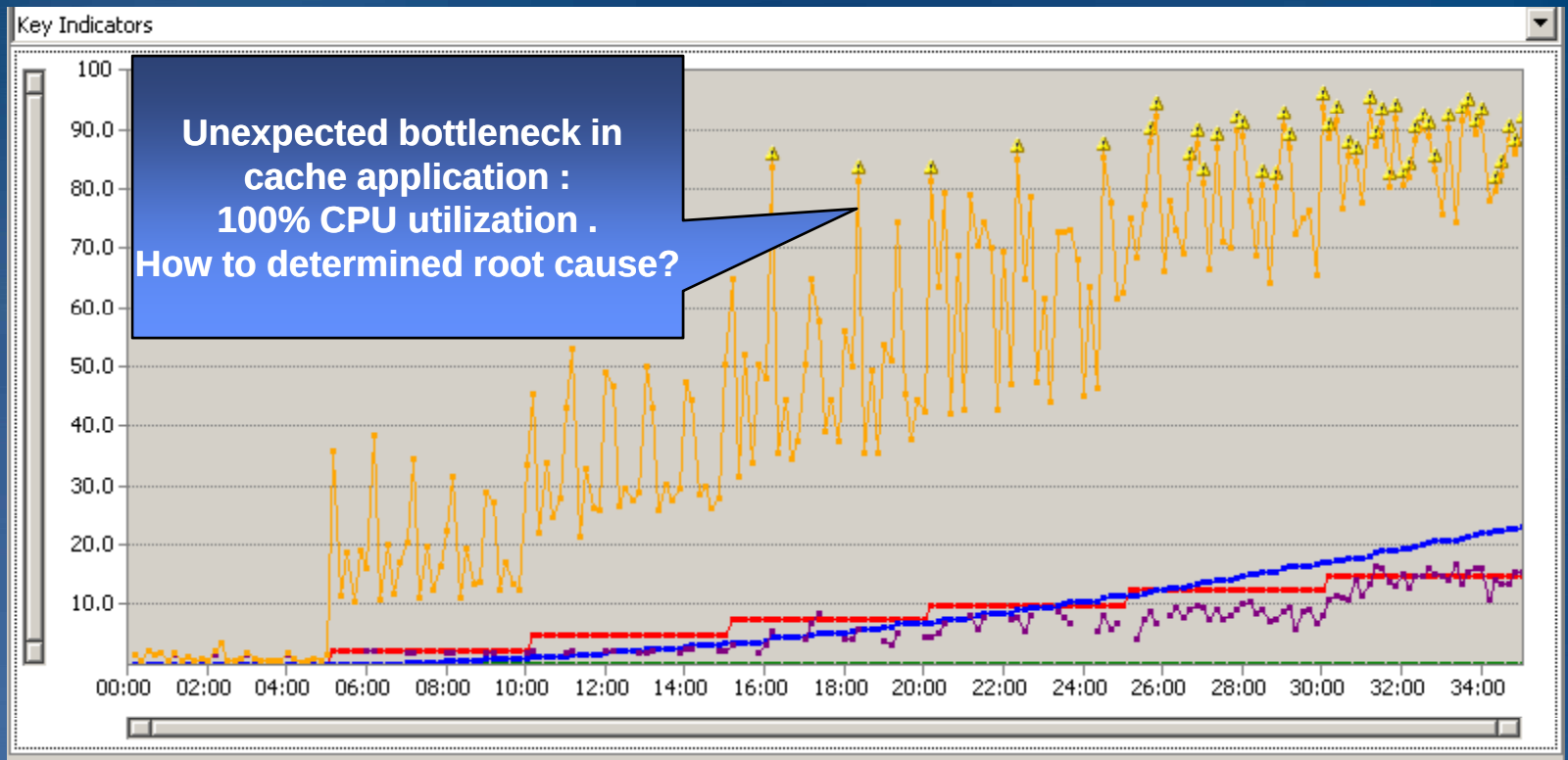
***Failing test case***




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg |
|---|---|---|---|---|---|---|---|---|
| ☑ User Load | _Total | LoadTest:Scenar | ESLIBMBL4 | | 100 | 1 | 41 | 21 |
| ☑ Failed Tests | _Total | LoadTest:Test | ESLIBMBL4 | | 1,000 | 0 | 226 | 99 |
| ☑ Passed Tests | _Total | LoadTest:Test | ESLIBMBL4 | | 100 | 0 | 35 | 23 |
| ☑ Avg. Response Time | _Total | LoadTest:Transa | ESLIBMBL4 | | 1,000 | 0.31 | 546 | 1.84 |
| ☑ % Processor Time | _Total | Processor | ESLSRV15 | | 100 | 0.94 | 100 | 49.6 |
| ☑ Available MBytes | - | Memory | ESLSRV15 | | 10,000 | 1,629 | 4,485 | 3,575 |
| ☐ Avg. Content Length | ESRI-ArcGIS-ADI | LoadTest:Reque | ESLIBMBL4 | | 1,000,000 | 5,442 | 105,690 | 98,554 |
| ☑ Private Bytes | w3wp | Process | ESLSRV15 | | 10,000,000... | 46,809,088 | 3,000,823,808 | 951,365,184 |

# Analyze Results

***Failing test case***




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg |
|---|---|---|---|---|---|---|---|---|
| ☑ User Load | _Total | LoadTest:Scena | ESLWS9 | | 1,000 | 1 | 151 | 76 |
| ☑ Failed Tests | _Total | LoadTest:Test | ESLWS9 | | 0 | 0 | 0 | 0 |
| ☑ Passed Tests | _Total | LoadTest:Test | ESLWS9 | | 10,000 | 0 | 2,338 | 753 |
| ☑ Avg. Response Time | _Total | LoadTest:Trans | ESLWS9 | | 10 | 0.19 | 1.71 | 0.87 |
| ☑ % Processor Time | _Total | Processor | ESLSRV9 | | 100 | 0.59 | 94.1 | 47.8 |

# Analyze Results

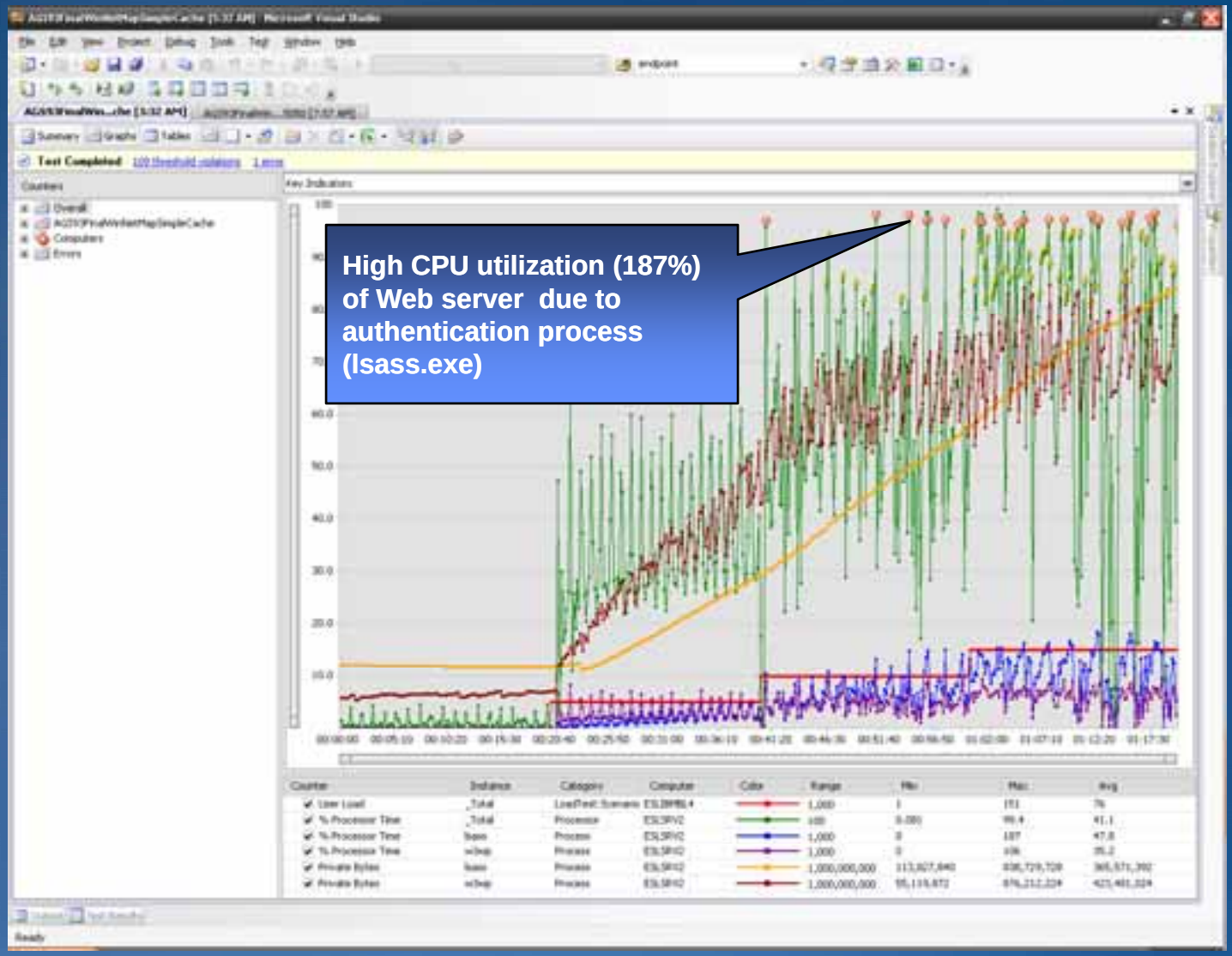
***Determining system capacity***

- **Maximum number of concurrent users corresponding to:**
  - **Maximum acceptable response time**
  - **First failure or 5%**
  - **resource utilization greater than 85%, for example CPU**
- **Different projects define acceptance criteria (performance level of service), e.g.**
  - **95% of requests under 3 sec**
  - **Max request under 10 sec**

# Common pitfalls

- 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)

# Common pitfalls – configuration problem test case



# Demo – Analyzing Test Case Results

# ESRI Flex, JavaScript, and REST Resources

- Flex API
  - http://resources.esri.com/help/9.3/arcgisserver/apis/flex/samples/index.html
- JavaScript API
  - http://resources.esri.com/help/9.3/arcgisserver/apis/javascript/arcgis/help/jssamples_start.htm
- REST API
  - http://resources.esri.com/help/9.3/arcgisserver/apis/REST/index.html

# References

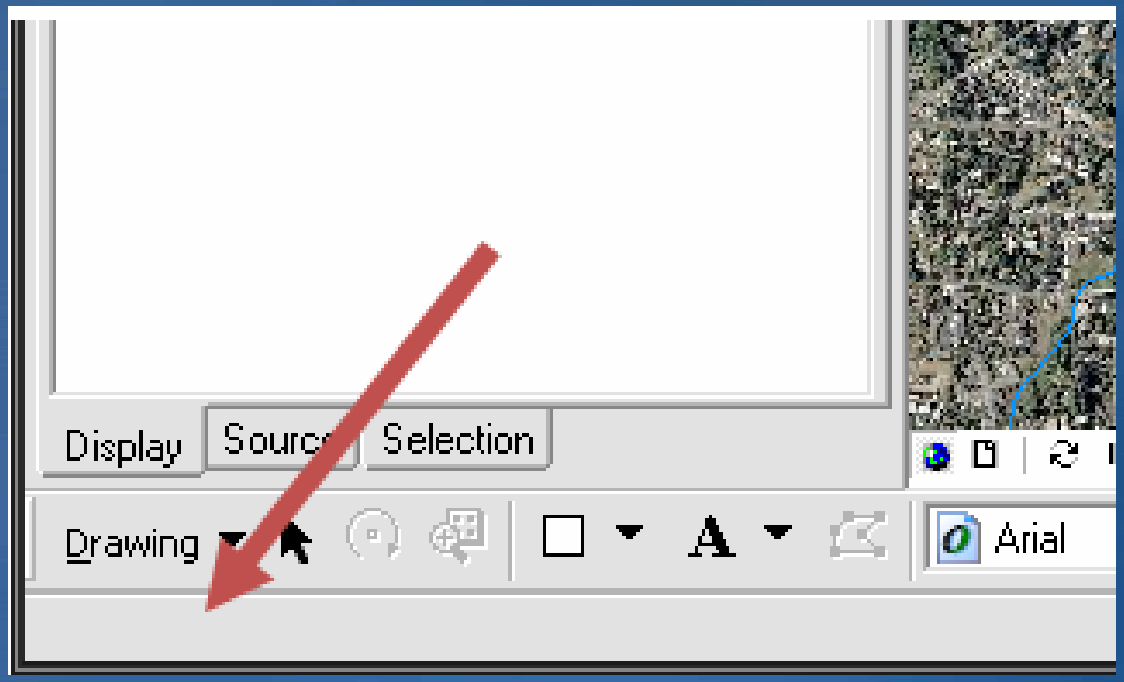
- Fiddler
  - http://www.fiddlertool.com/fiddler/version.asp
- Visual Studio 2008 Professional Edition (90-day Trial)
  - http://www.microsoft.com/downloads/details.aspx?familyid=83C3A1EC-ED72-4A79-8961-25635DB0192B&displaylang=en
- MSDN
  - Chapter 17 – Load-Testing Web Applications
    - http://msdn.microsoft.com/en-us/library/bb924372.aspx
  - Creating a Web Test
    - http://msdn.microsoft.com/en-us/library/ms182538.aspx
- BLOGS
  - Bill Barnett's blog
    - http://blogs.msdn.com/billbar/pages/load-test-api-enhancements-in-vsts-2008-sp1-beta.aspx
  - Ed Glas's blog on VSTS load testing
    - http://blogs.msdn.com/edglas/archive/2006/02/06/525614.aspx
  - Sean Lumley's Blog
  - http://blogs.msdn.com/slumley/archive/2008/12/22/modifying-request-properties-with-a-web-test-plugin.aspx

# Performance Testing ArcMap and ArcEngine Applications

- Several tools can be used to automate thick client workflows
  - HP LoadRunner
  - Borland's Silk Perfomer
  - NRG's AppLoader

# Capturing Transactions in ArcMap Using Events

Better choices are text indicators that eventually have a definitive ending (such as text stating “complete”, or no text at all)



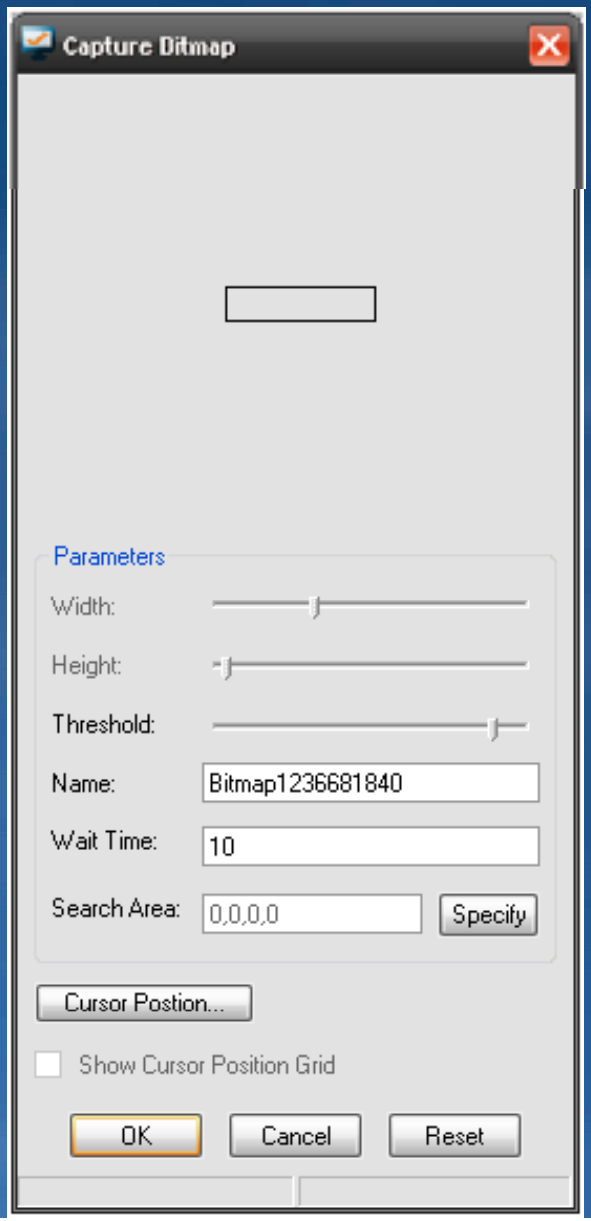
# Capturing Transactions in ArcMap Using Events

***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

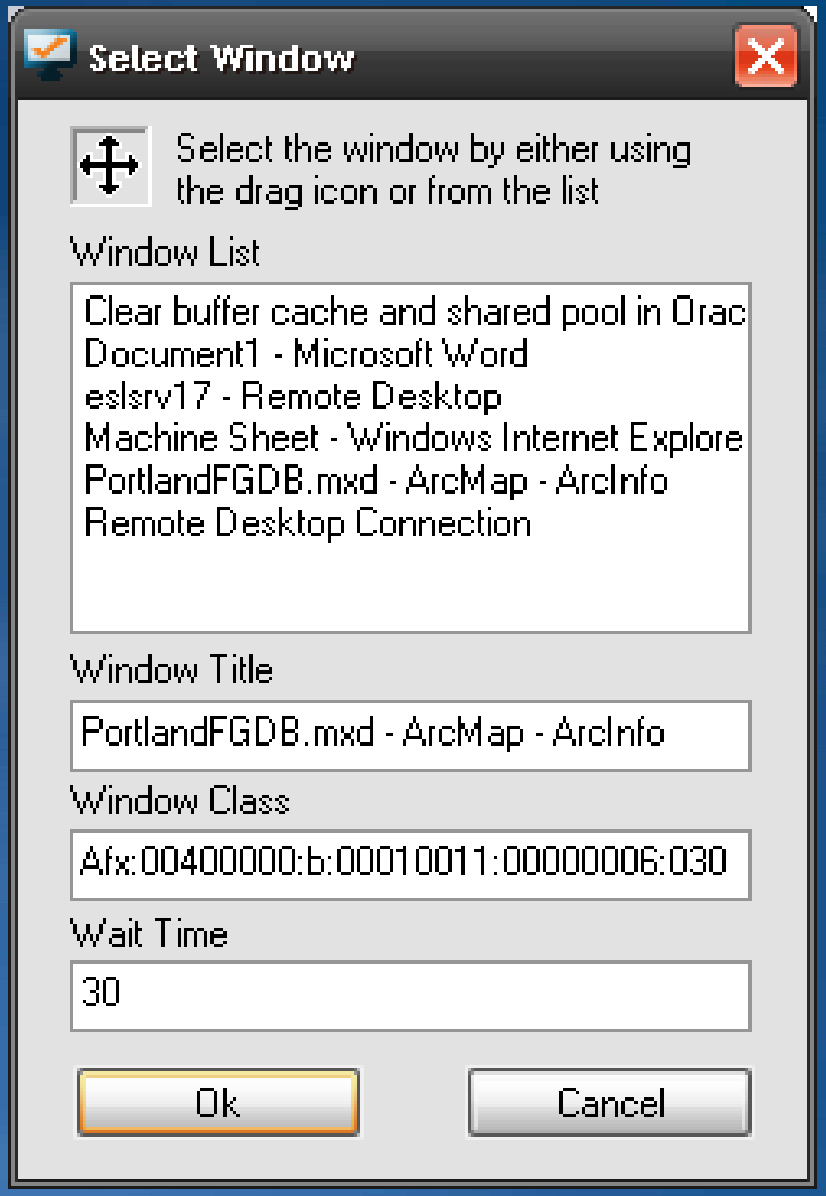
***Measuring pan and zoom transactions (Apploader tool example)***



# Capturing Transactions in ArcMap Using Events

***Measure opening ArcMap document (Apploader tool example)***

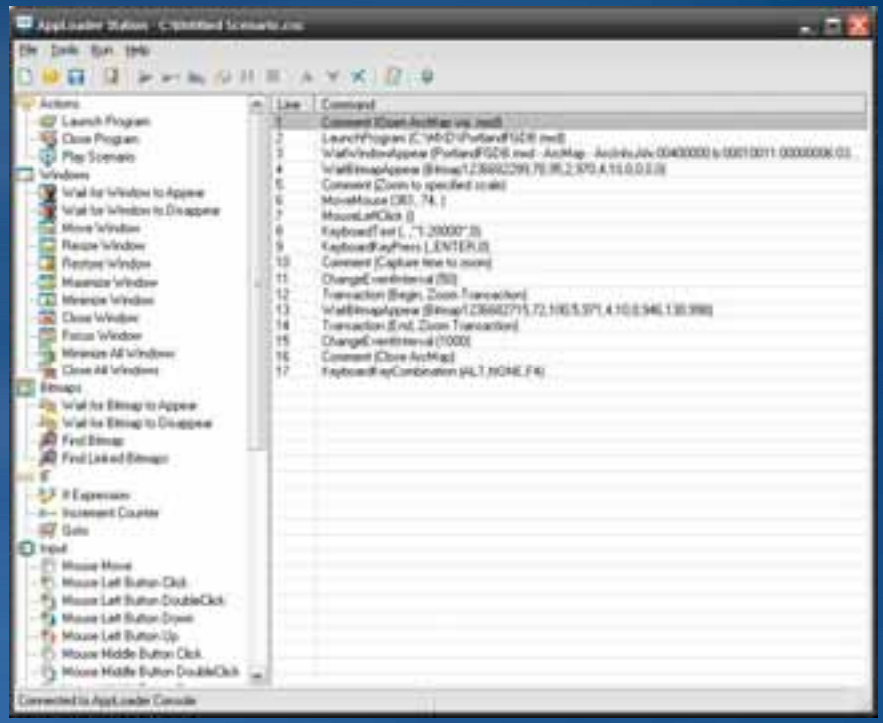
- **Wait for Window**



# Capturing Transactions in ArcMap Using Events

## *Open ArcMap, Zoom, Close ArcMap*

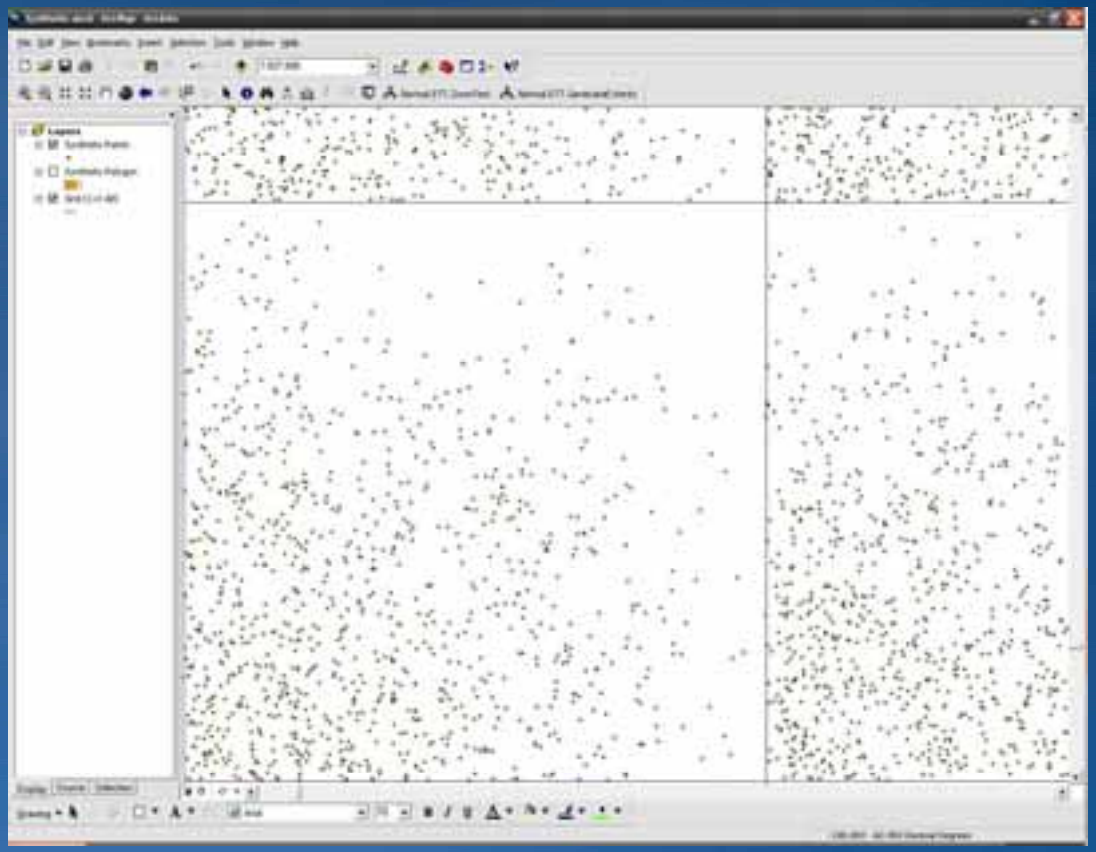
1. Comment : Open ArcMap
2. Launching ArcMap via specified MXD
3. Waiting for the ArcMap window to appear
4. Waiting for captured bitmap to appear. This bitmap is a blank status area indicating all drawing has completed.
5. Comment: Zoom
6. Move the mouse to the map scale field
7. Click the map scale 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 transaction capturing
12. Begin the transaction
13. Wait for bitmap to appear. This bitmap is the grey status area at the lower left of the screen. It will become blank when the map has finished drawing
14. End the transaction
15. Changing the Event Interval back to its default 1000ms
16. Comment: Close ArcMap
17. Closing ArcMap via shortcut



# Exploratory tests

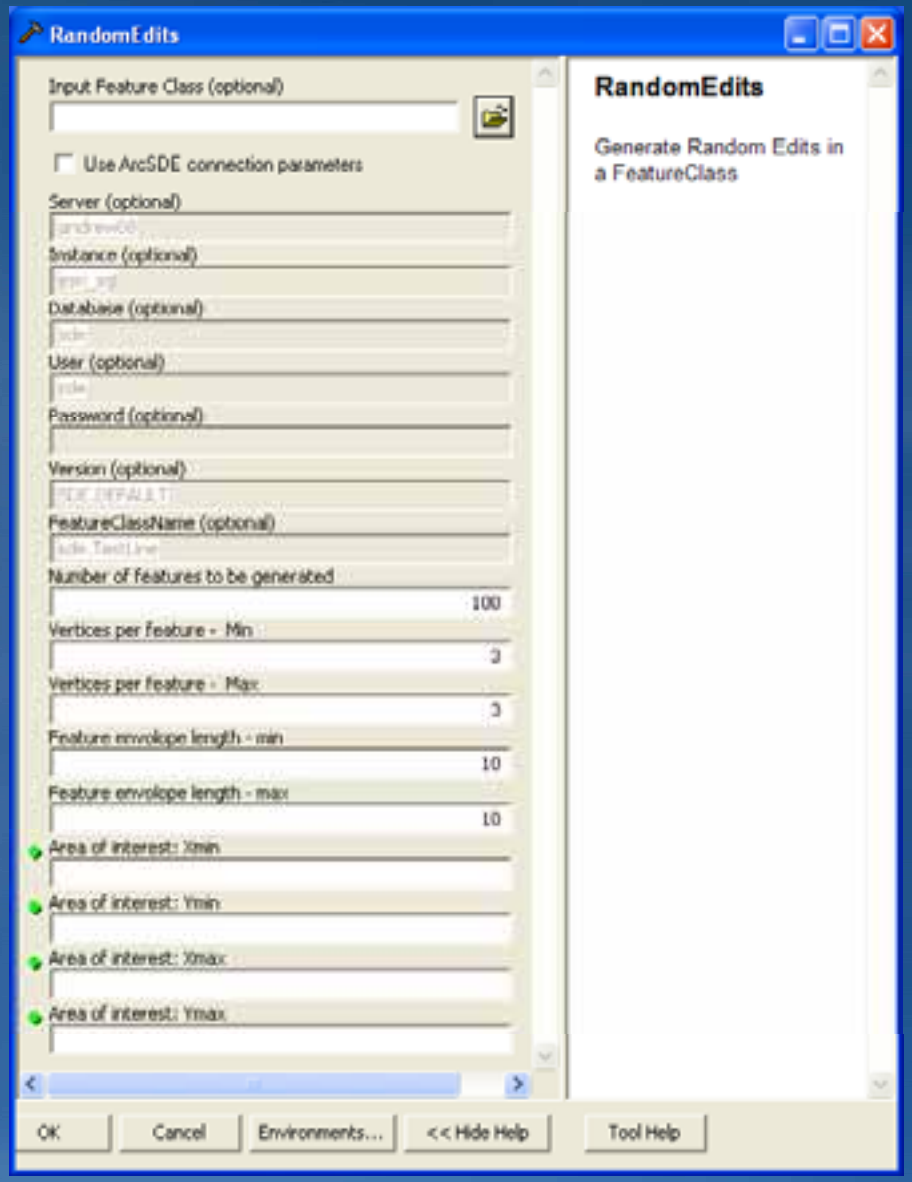
# Using synthetic data for exploratory tests

- **Benefits**
  - **Controlled density**
  - **Easy to script**



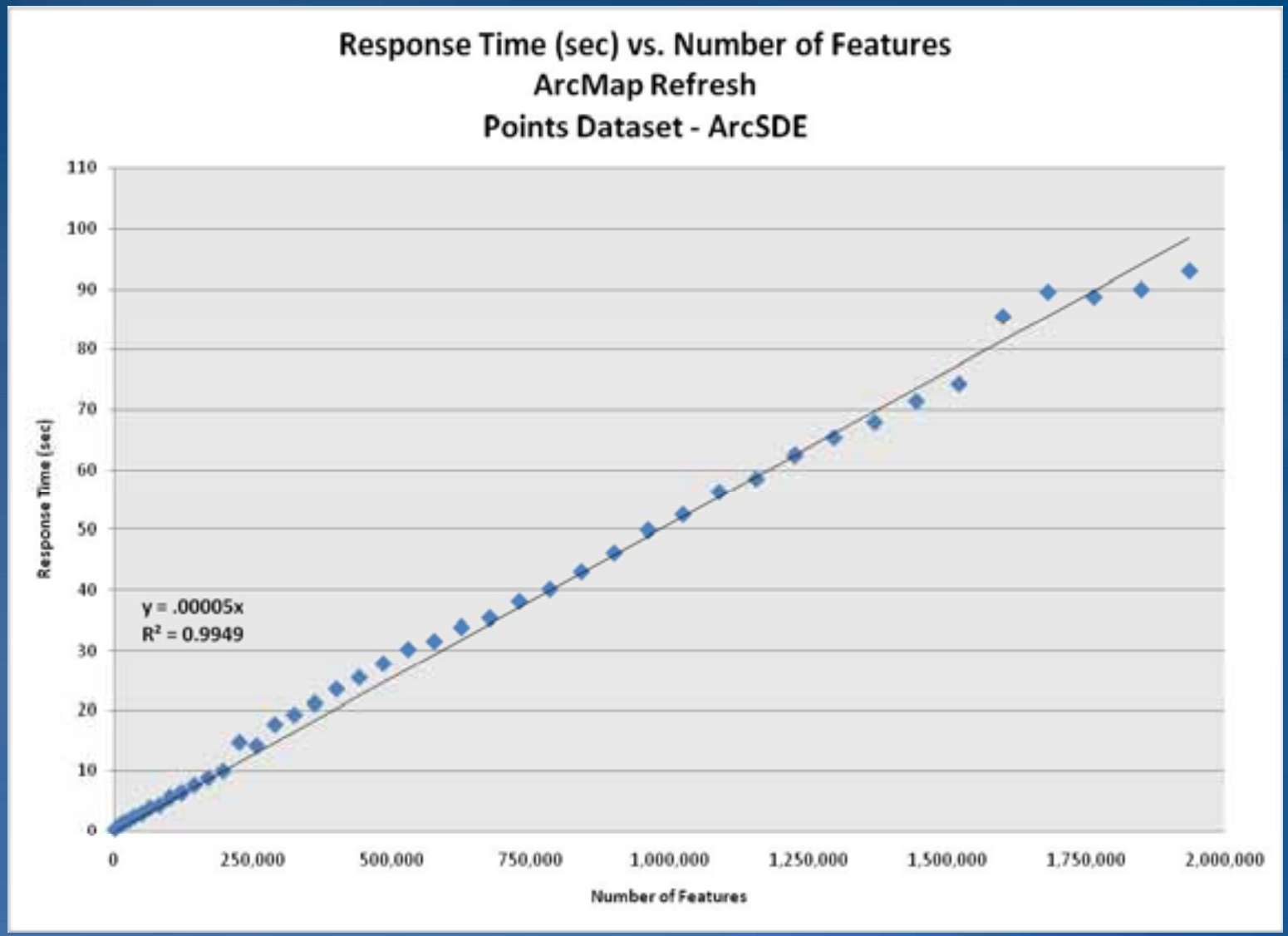
# Generating synthetic data

***RandomEdits -a custom gp tool available for download***



# Using synthetic data for exploratory tests

***Helps analyzing generic performance factors, e.g. Fetch time vs. # features***



# Summary

- Today we covered:
- Performance and Scalability Testing methodology
  - Web applications (step by step)
  - ArcGIS Desktop Applications
  - Exploratory tests
- Tips and Tricks

***Still have questions?***

# Additional Resources

***Questions, answers and information…***

- ***Tech Talk***
  - ***Outside this room right now!***
- ***Other sessions***
  - ***ArcGIS Server Performance and Scalability – Performance Factors***

- ***ESRI Resource Centers***

**PPTs, code and video**
**resources.esri.com**

- ***Social Networking***


**www.twitter.com/ESRIDevSummit**


**tinyurl.com/ESRIDevSummitFB**

# Questions?

*Frank Pizzi, fpizzi@esri.com*
*Andrew Sakowicz, asakowicz@esri.com*
*ESRI Professional Services requests:*
*John Graham, jgraham@esri.com*
*Chad Helm, chelm@esri.com*

# Demo Related and Supplemental Slides

# Using Visual Studio for web testing

- 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
       - http://msdn.microsoft.com/en-us/library/cc678655.aspx
  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 Stateless vs. Stateful applications

- Examples
  - Stateless:JavaScript, Flex, SilverLight API
  - Stateful: .NET and Java ADF
- State management
  - Stateful
    - Stores sessions in a web server process (InProc)
    - Often web server memory bound
  - Stateless
    - Manages state in a browser
- Challenges of testing
  - Stateful application more difficult to test

# Scripting 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 a 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

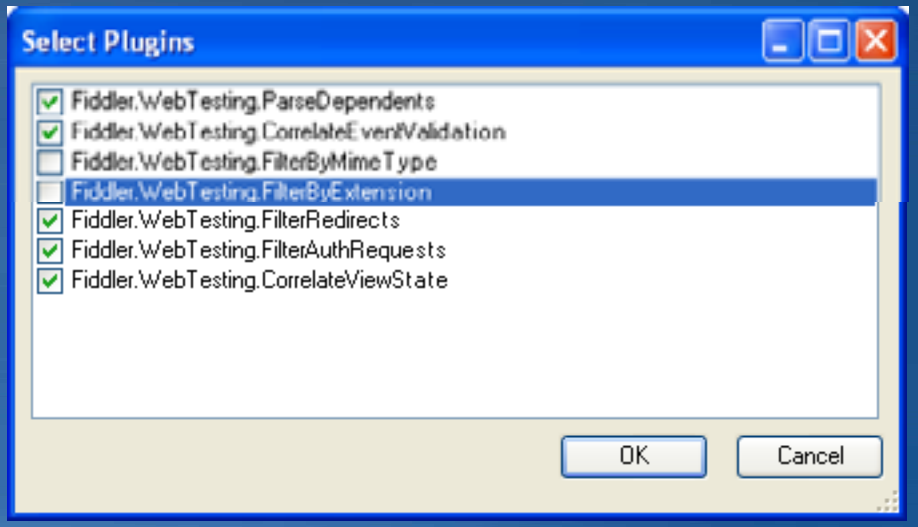
| | | | | |
|---|---|---|---|---|
| 84 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/WebResource.axd?d=InwIAjzTW5r2YzMrDnvYp-t8uIToH6LbDqCa-F |
| 85 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/WebResource.axd?d=InwIAjzTW5r2YzMrDnvYp-t8uIToH6LbDqCa-F |
| 86 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/WebResource.axd?d=InwIAjzTW5r2YzMrDnvYp-t8uIToH6LbDqCa-F |
| **87** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=d20** |
| 88 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/WebResource.axd?d=InwIAjzTW5r2YzMrDnvYp-t8uIToH6LbDqCa-F |
| **89** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=c58** |
| 90 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/images/dismiss.png |
| **91** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=5d5** |
| 92 | 502 | HTTP | INI | / |
| 93 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda3/C0000a0e5.jpg |
| 94 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda4/C0000a0e5.jpg |
| 95 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.UI.WebControls.MapHandler.ashx?MapID= |
| 96 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/Default.aspx |
| 97 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda3/C0000a0e6.jpg |
| 98 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda4/C0000a0e6.jpg |
| 99 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda3/C0000a0e7.jpg |
| 100 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L06/R0000bda4/C0000a0e7.jpg |
| 101 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/images/backward.png |
| **102** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=b2e** |
| **103** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=305** |
| 104 | 502 | HTTP | Z1 | / |
| 105 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R0001602f/C00012acf.jpg |
| 106 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016030/C00012acf.jpg |
| 107 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016031/C00012acf.jpg |
| 108 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R0001602f/C00012ad0.jpg |
| 109 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/Default.aspx |
| 110 | 200 | HTTP | eslsrv11 | /TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.UI.WebControls.MapHandler.ashx?MapID= |
| 111 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016030/C00012ad0.jpg |
| 112 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016031/C00012ad0.jpg |
| 113 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R0001602f/C00012ad1.jpg |
| 114 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016030/C00012ad1.jpg |
| 115 | 200 | HTTP | eslsrv11 | /arcgiscache/PortlandCacheNoTaxLots/Portland/_alllayers/L07/R00016031/C00012ad1.jpg |
| **116** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=210** |
| **117** | **200** | **HTTP** | **eslsrv...** | **/TemplateMixedAggregate/ESRI.ArcGIS.ADF.Web.MimeImage.ashx?ImgID=c65** |
| 118 | 502 | HTTP | Z2 | / |

# 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.MimeImage.ashx must be altered to reflect the new session
- Transaction markers must be removed

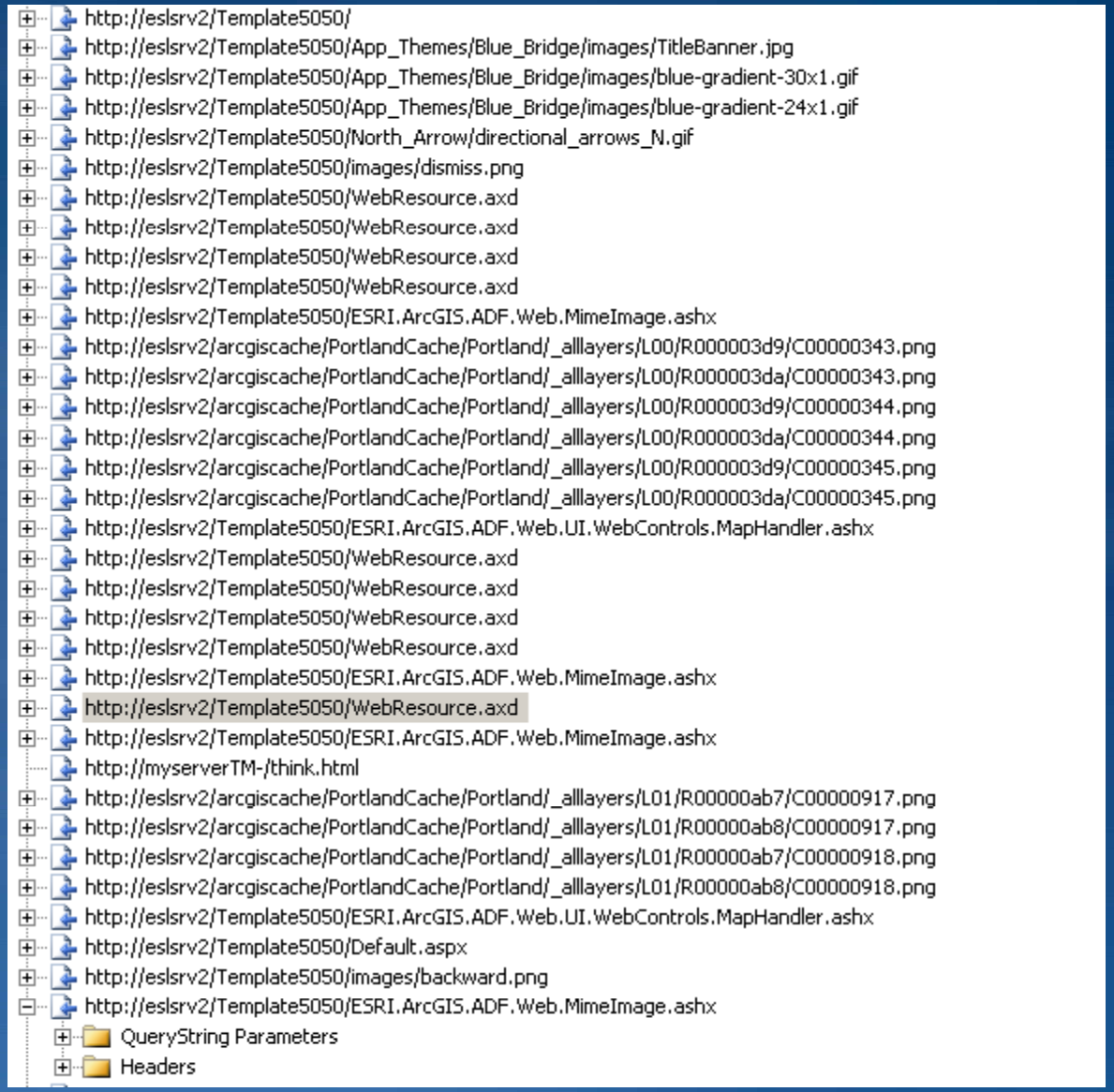
# Creating Transaction Scripts – Fiddler manual

- 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-



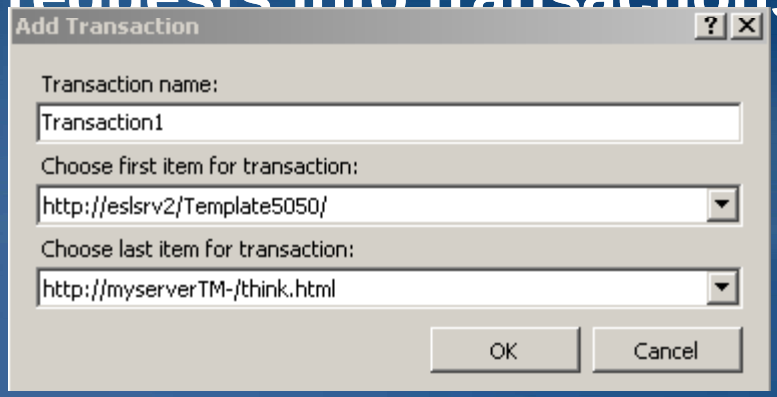
# Creating Transaction Scripts – VS Web Test

- **Declarative webtest**



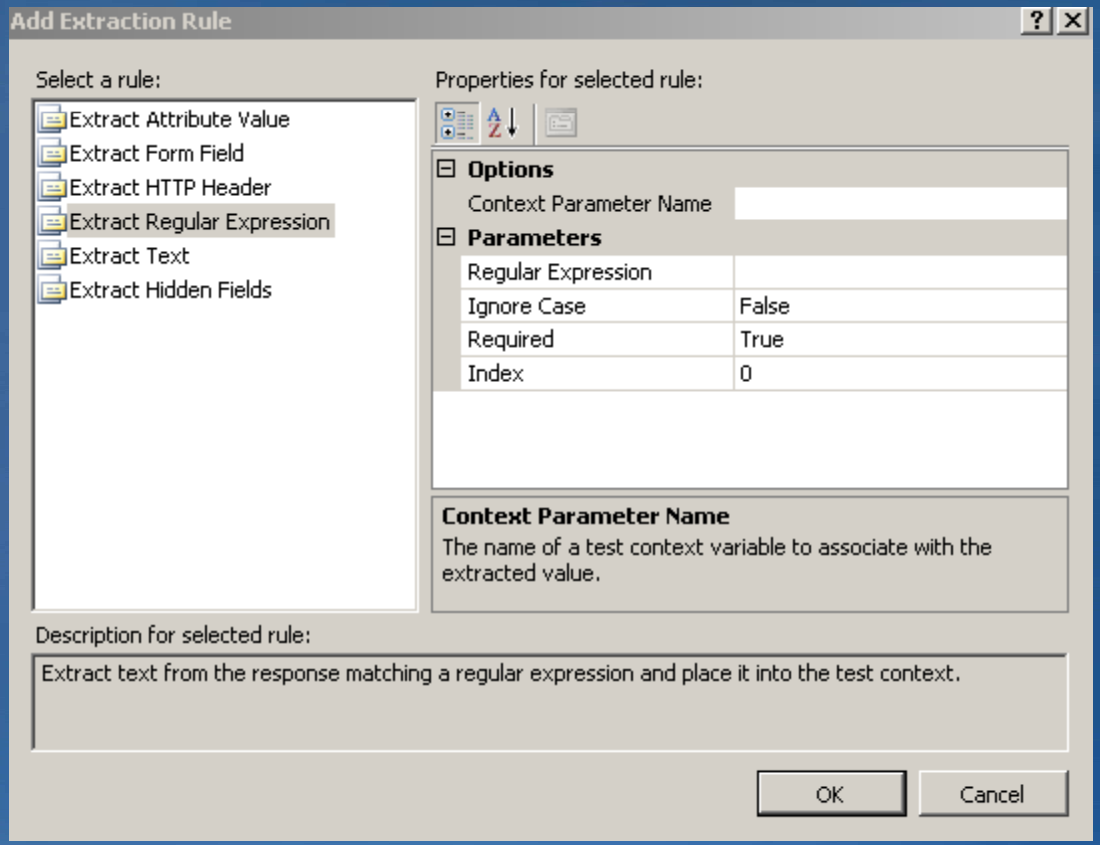
# Creating Transaction Scripts – VS Web Test manual

- 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

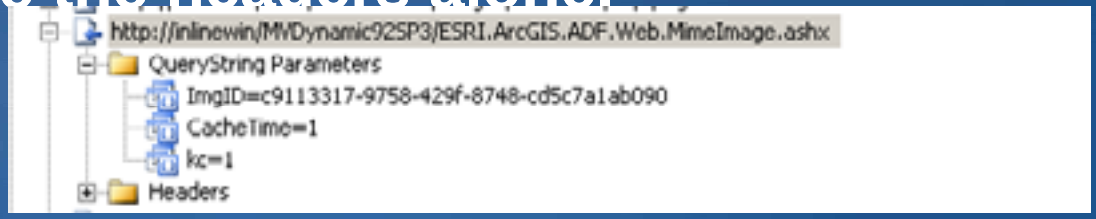


# Creating Transaction Scripts – VS Web Test manual extraction rules for ADF testing

| Parameters | Value | Notes |
|---|---|---|
| Context Parameter Name | t0-i0 | 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. |
| Regular Expression | ImgID=((([0-9a-fA-F]){8}-([0-9a-fA-F]){4}-([0-9a-fA-F]){4}-([0-9a-fA-F]){4}-([0-9a-fA-F]){12})\|([0-9a-fA-F]){32}) | 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 |
| Ignore Case | False | |
| Required | True | |
| Index | 0 | 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. |

# Creating Transaction Scripts – VS Web Test manual extraction rules for ADF testing

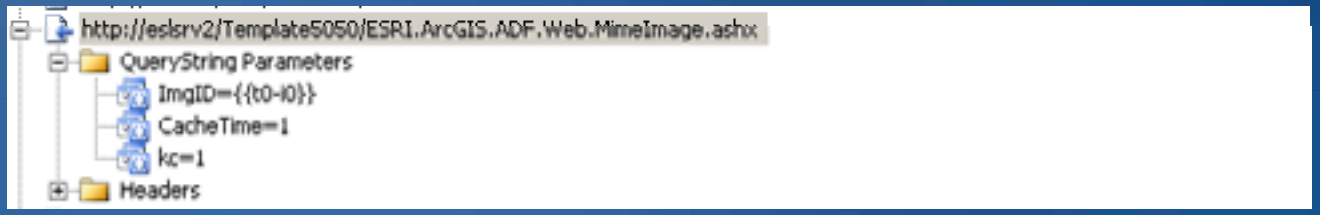
- Now, we have defined all of the extraction rules for the first transaction.
- Next, we must find the static requests from the initial HTTP Debugging proxy and modify them to be dynamic.
- Below is an example of a static request for a mime image that must be modified.
- We will delete all of the query string parameters, but we must leave the headers alone.



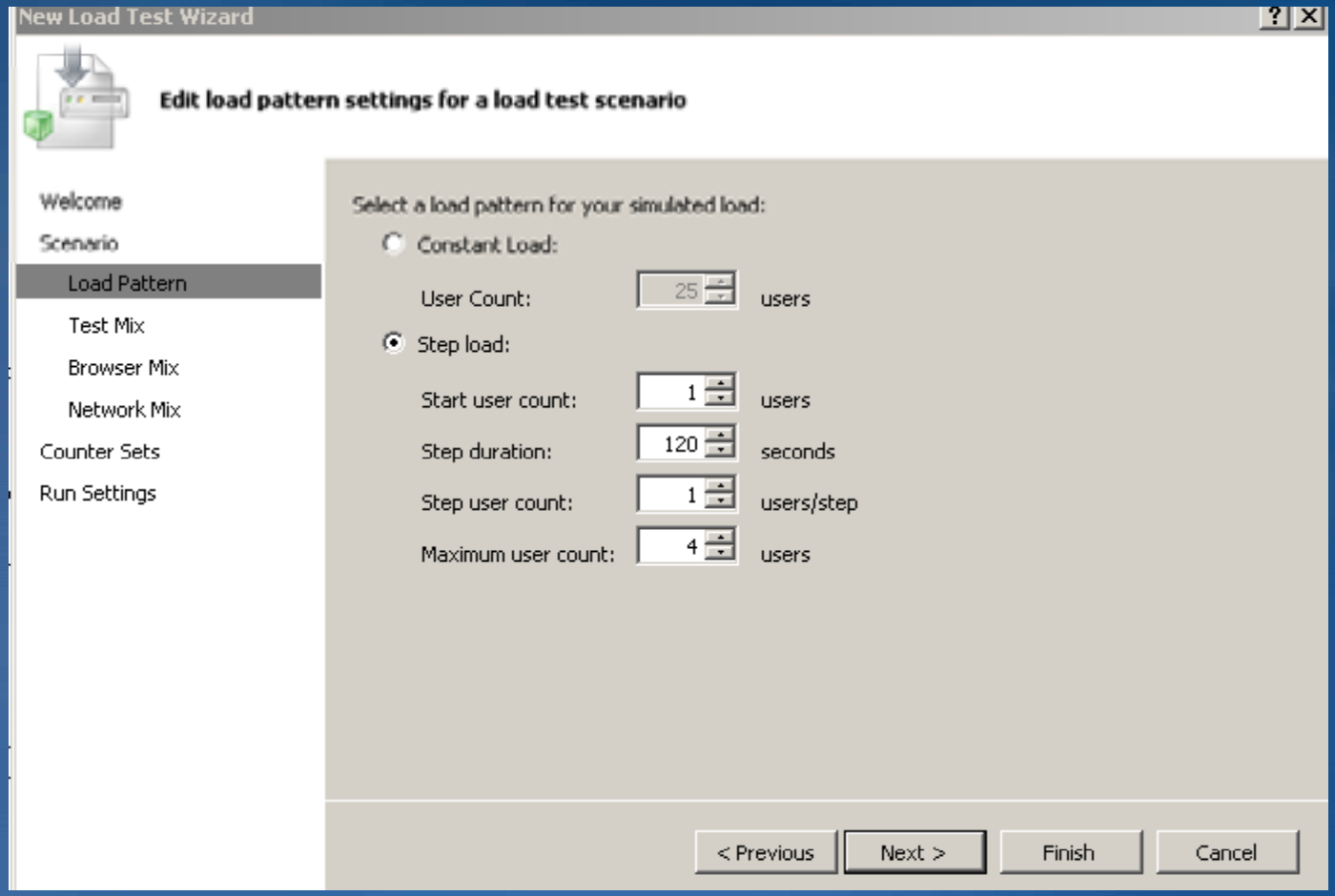
# Creating Transaction Scripts – VS Web Test manual extraction rules for ADF testing

- 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.

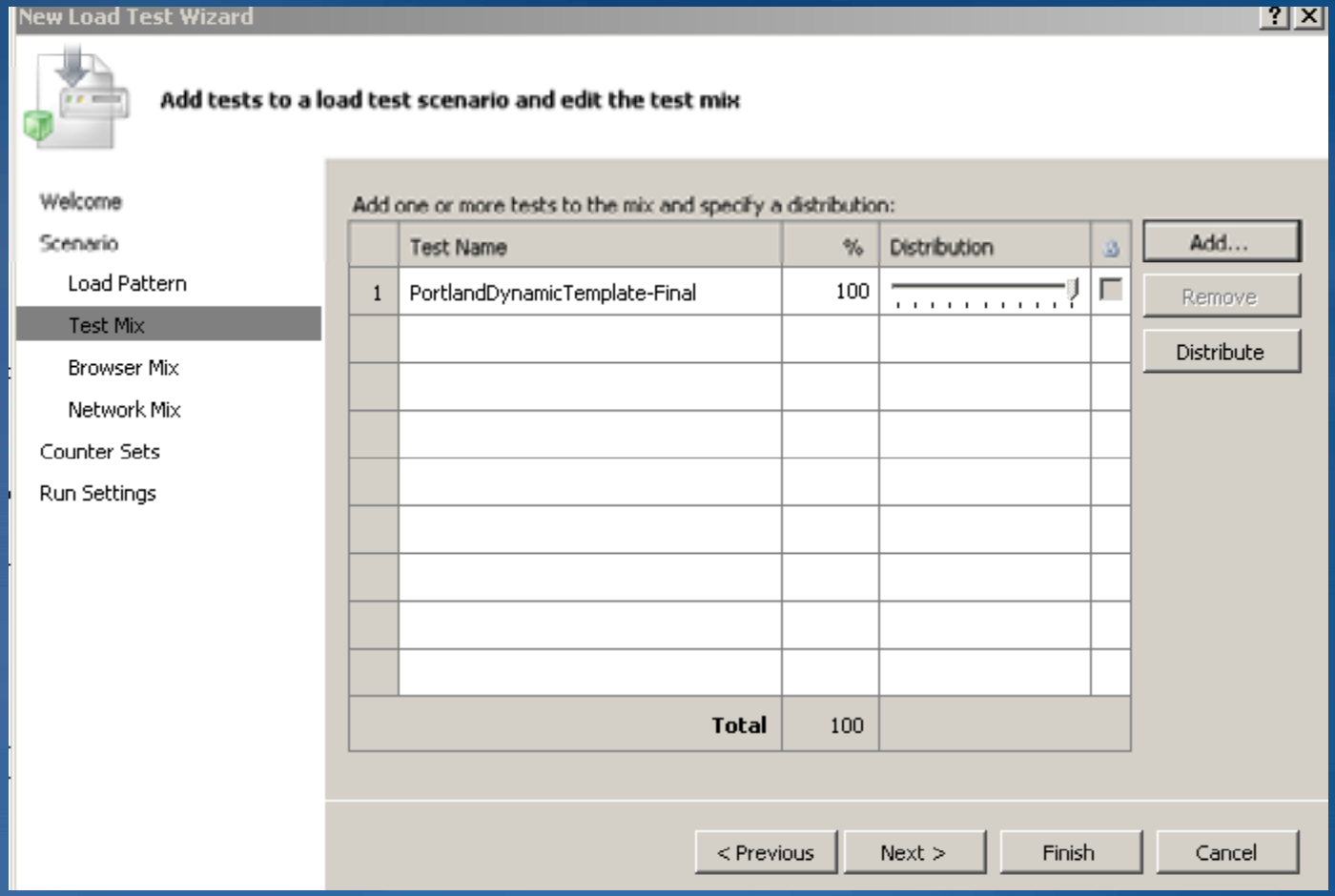




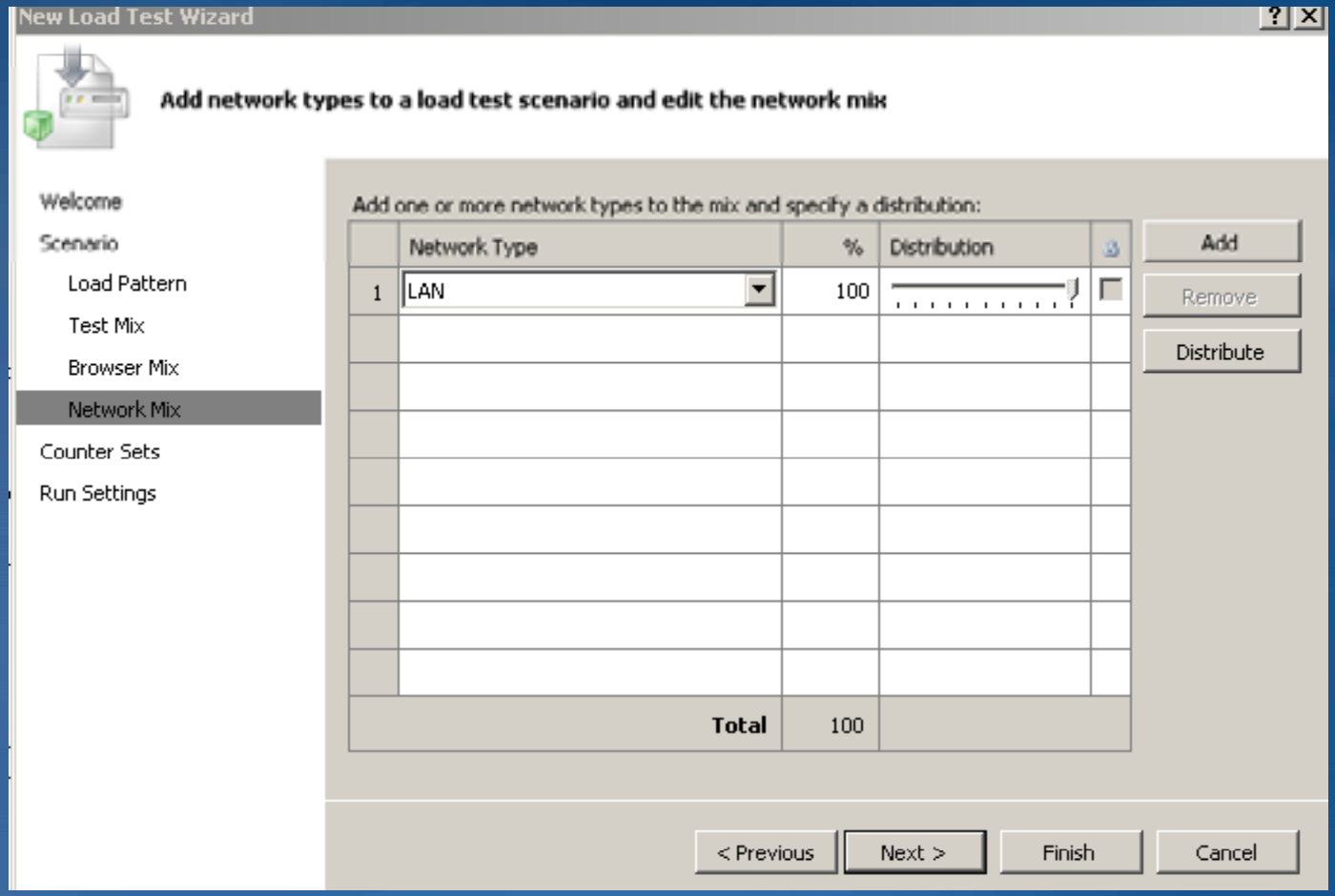
# Creating load test – Visual Studio



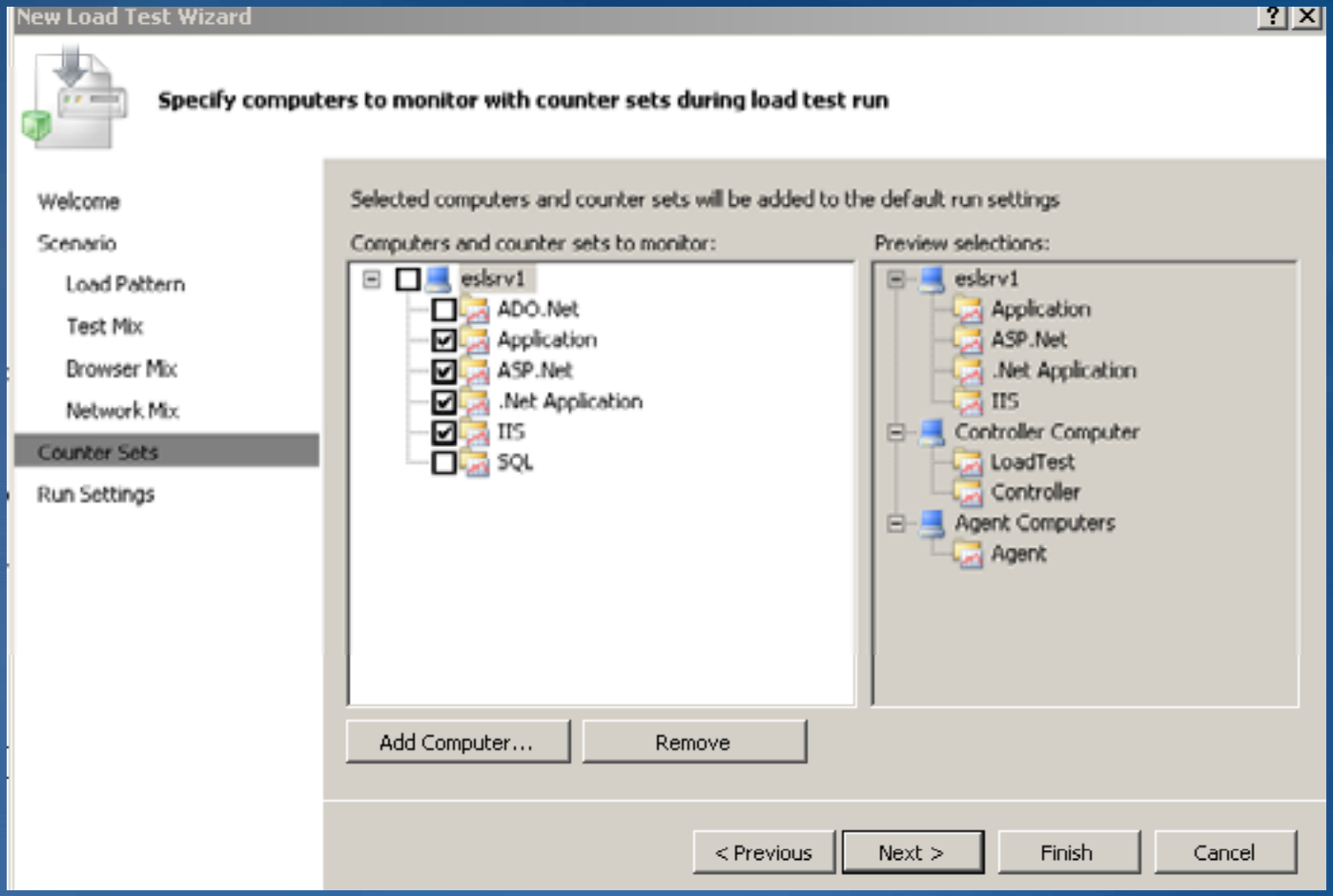
# Creating load test – Visual Studio



# Creating load test – Visual Studio



# Creating load test – Visual Studio

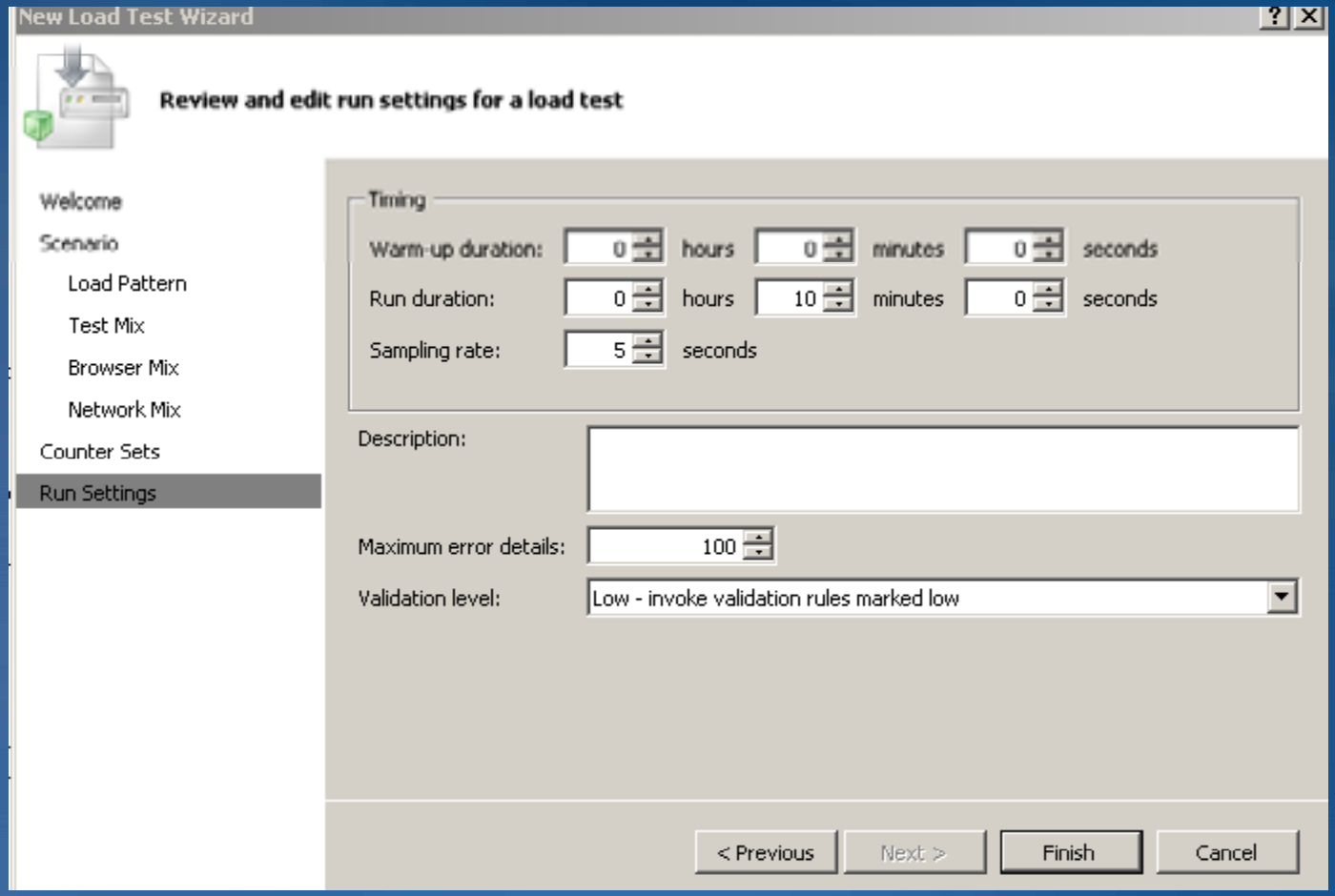


# Running a Load Test with System Monitoring

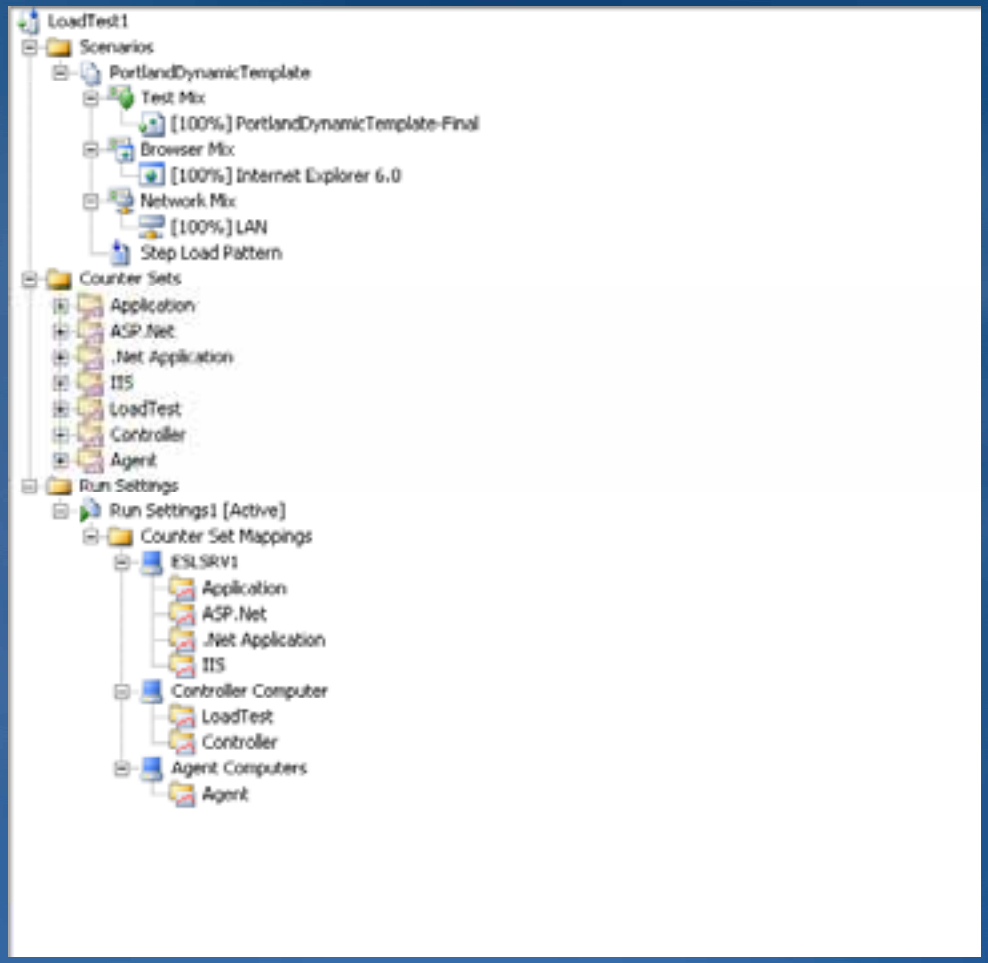
While the load test is running, various system components should be monitored including the CPUs, disk subsystem, memory, and network. System monitoring applies to all the systems that play a part in the test configuration, including the database server. At a minimum, the following core components should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

- CPU Utilization
- Memory
- Disk Utilization
- Network Utilization

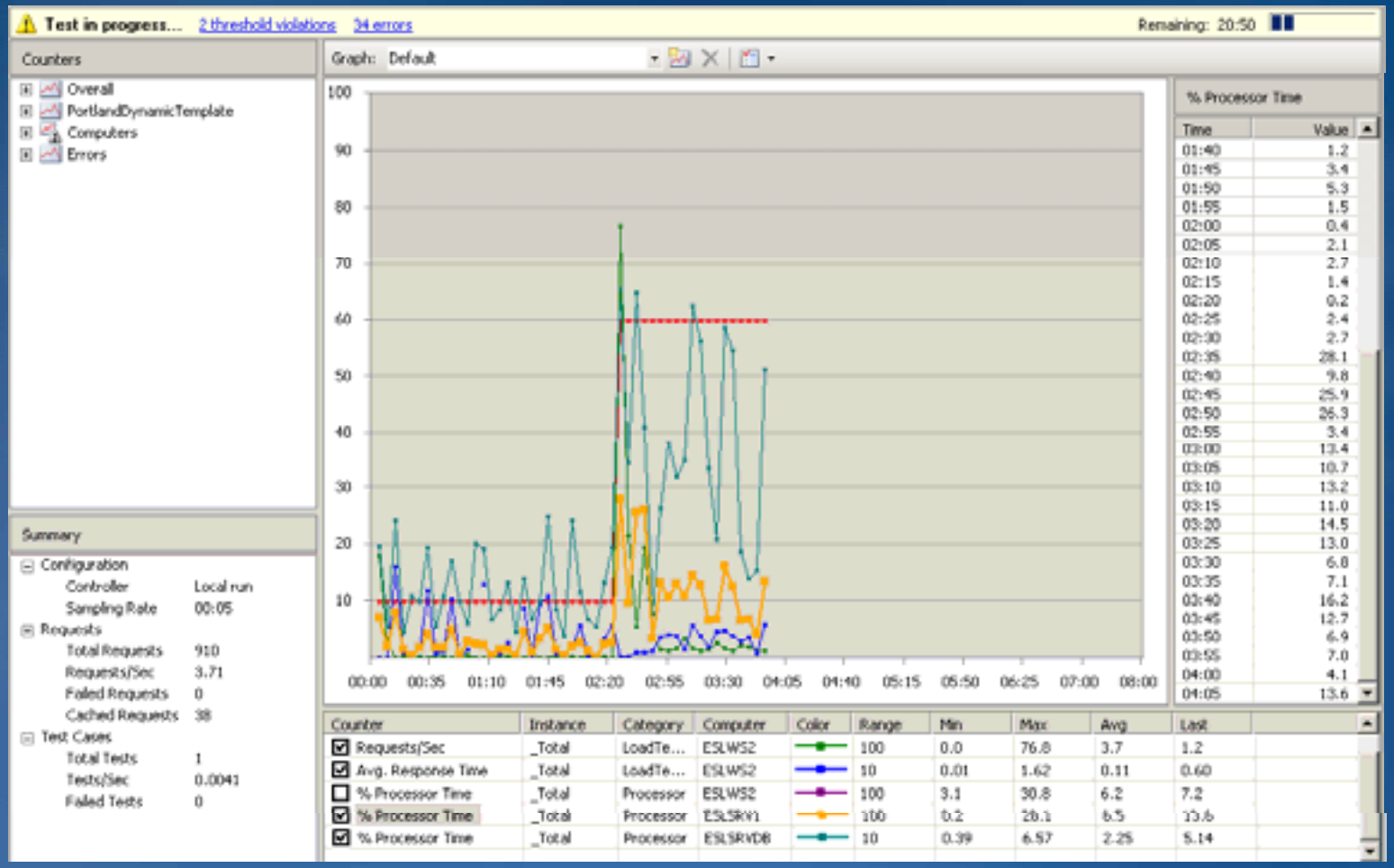
# Creating load test – Visual Studio



# Creating load test – Visual Studio



# Executing load test – Visual Studio



# Glossary

- **Transactions** - A transaction is defined as a Web page request that results in a Web browser output display
- **Response Time** - Response time is the accumulation of time associated with processing requests (service time) and waiting for service (queue time).
- **Queue Time** - Queue time is the amount of time spent waiting to receive service
- **Service Time** - Service time is the time required by a device (or server) to service a request
- **Throughput** - Throughput is the rate at which some amount of work is being performed
- **Utilization** - Utilization measures the fraction of time a device is busy servicing a request and is usually reported as a percentage

# Glossary

- **Scalability** - Scalability is the ability of a computer system to adapt to an increasing load demand while providing minimal performance degradation
- **Performance** - Performance is related to response time and is based on a user's observation of how well a system is performing a given task.
- **Performance Testing** – Used to determine how fast some aspect of a system performs under a particular workload
- **Stress Testing** - Stress testing is a form of testing that is used to determine the stability of a given system or entity
- **Load Testing** - Load testing is the process of putting demand on a system or device and measuring its response