Esri International User Conference | San Diego, CA
Technical Workshops | 07/12/2011

ArcGIS Server Performance and Scalability—Testing Methodologies

Andrew Sakowicz

Frank Pizzi

Introductions

- Who are we?
  - Enterprise implementation
- Target audience
  - GIS administrators
  - DBAs
  - Architects
  - Developers
  - Project managers
- Level
  - Intermediate

Objectives

Performance engineering—concepts and best practices

- **Technical**
  - **Solution performance factors**
  - **Tuning techniques**
  - **Performance testing**
  - **Capacity planning**
- **Managerial**
  - **Skills**
  - **Level of effort**
  - **Risks**
  - **ROI**

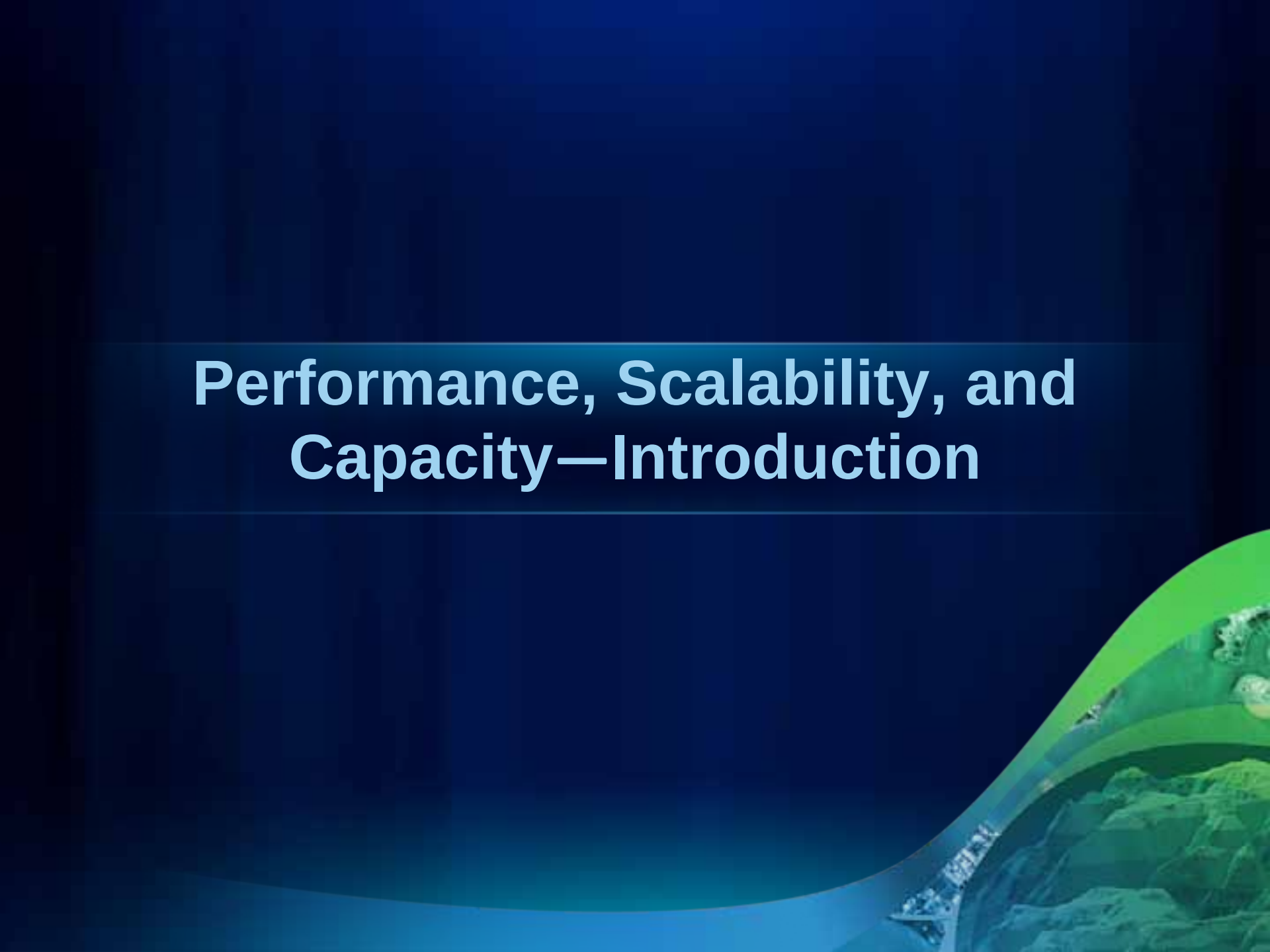
Agenda

Solution performance engineering

- Introduction
- Performance engineering in project phases
  - Requirements
  - Design

  Lunch
  - Development
  - Deployment
  - Operation and maintenance

Performance, Scalability, and Capacity—Introduction



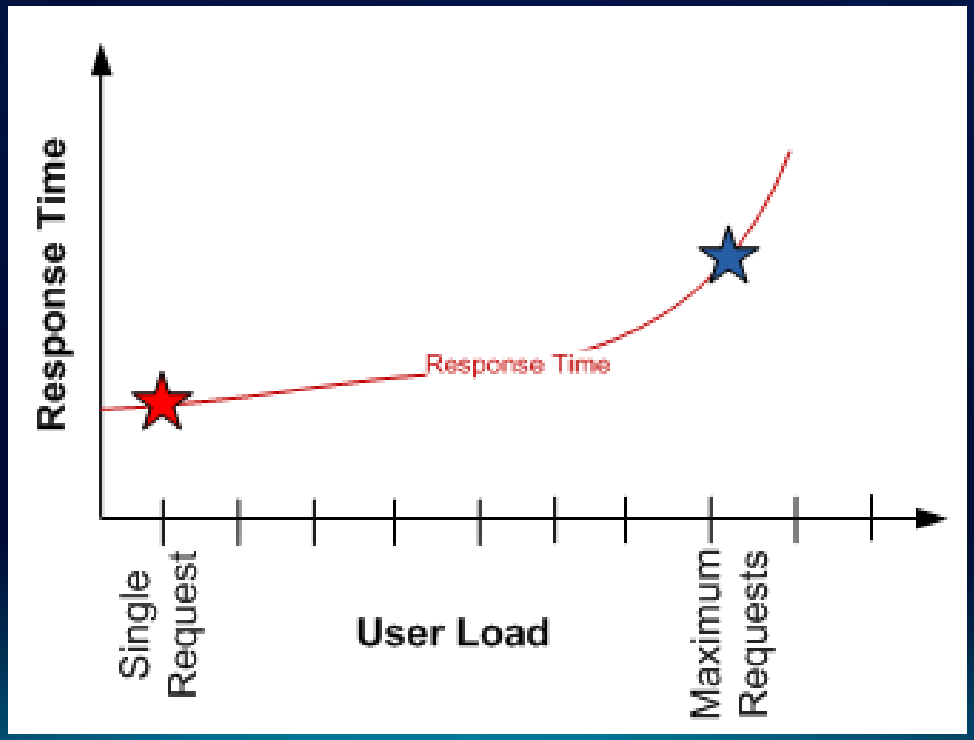
Performance Engineering

Benefits

- Lower costs
  - Optimal resource utilization
  - Less hardware and licenses
  - Higher scalability
- Higher user productivity
  - Better performance
- Reputation
  - User satisfaction

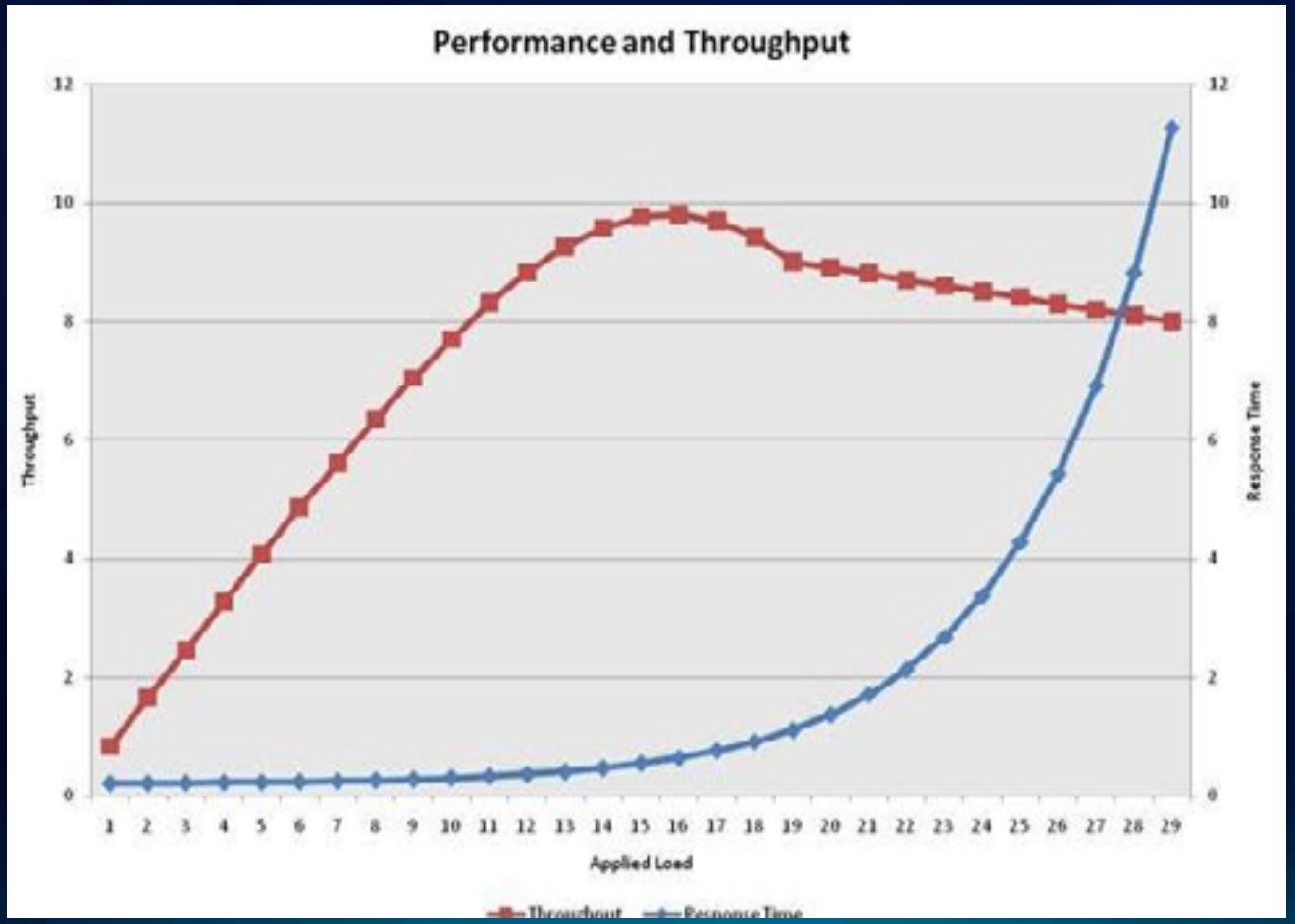
Performance and Scalability Definitions

- **Performance: The speed at which a given operation occurs**
- **Scalability: The ability to maintain performance as load increases**



Performance and Scalability Definitions

Throughput: The amount of work accomplished by the system in a given period of time



Performance and Scalability Definitions

Defining system capacity

- System capacity can be defined as a user load corresponding to
  - Maximum throughput
  - Threshold utilization, e.g., 80
  - SLA response time

Project Life Cycle Phase

Performance engineering applied at each step

Project Life Cycle Phase

Performance engineering applied at each step

- Requirements
  - Quality attributes, e.g., SLA
- Design
  - Performance factors, best practices, capacity planning
- Development
  - Performance and load testing
  - Tuning
- Deployment
  - Configuration, tuning, performance, and load testing
- Operation and maintenance
  - Tuning
  - Capacity validation

Performance Engineering —Solution Requirements

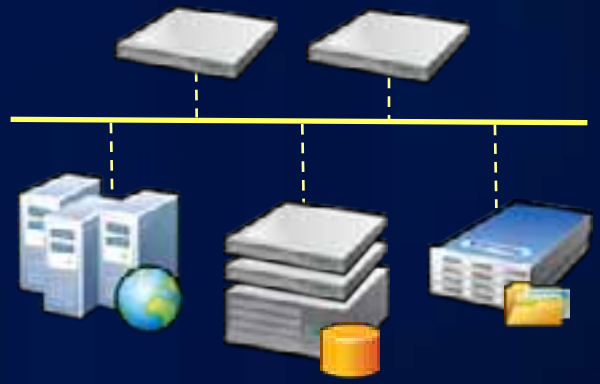
Requirements Phase

Performance engineering addresses quality attributes.

Functional Requirements

- Visualization
- Analysis
- Workflow Integration

Quality Attribute Requirements



- Availability
- Performance & Scalability
- Security

Requirements Phase

- Define System Functions
  - What are the functions that must be provided?
- Define System Attributes
  - Nonfunctional requirements should be explicitly defined.
- Risk Analysis
  - An assessment of requirements
  - Intervention step designed to prevent project failure
- Analyze/Profile Similar Systems
  - Design patterns
  - Performance ranges

Performance Engineering —Solution Design Phase

Design Phase

Selection of optimal technologies

- Meet functional and quality attributes.
- Consider costs and risks.
- Understand technology tradeoffs, e.g.:
  - Application patterns
  - Infrastructure constraints
  - Virtualization
  - Centralized vs. federated architecture

Design Phase

Performance Factors

Design Phase—Performance Factors

Design, Configuration, Tuning, Testing

- **Application**
- **GIS Services**
- **Hardware Resources**

Design Phase—Performance Factors

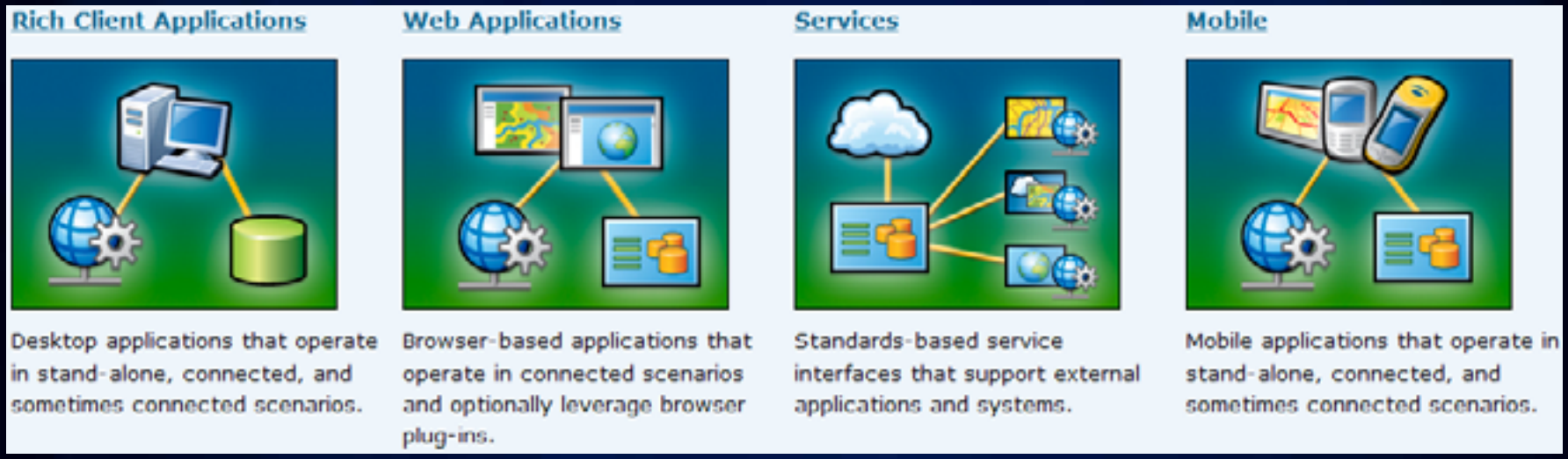
Application

- Type, e.g., mobile, web, desktop
- Stateless vs. state full (ADF)
- Design
  - Chattiness
  - Data access (feature service vs. map service)
- Output image format

Design Phase—Performance Factors

Application Types

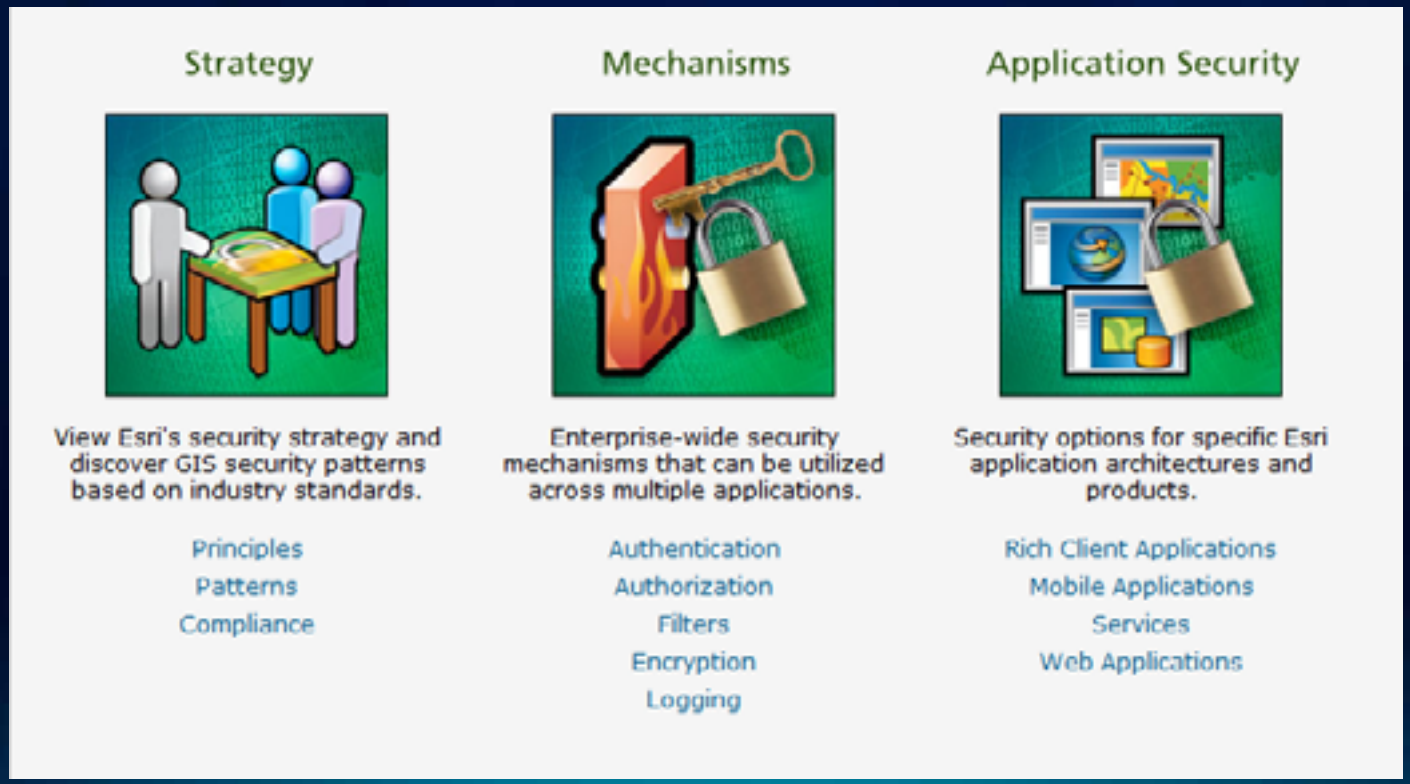
- **Architecture**
  - resources.arcgis.com/content/enterprisegis/10.0/architecture



Design Phase—Performance Factors

Application Security

- **Security**
  - resources.arcgis.com/content/enterprisegis/10.0/application_security



Design Phase—Performance Services

Application—Output image format

- PNG8/24/32
  - Transparency support
  - 24/32 good for antialiasing, rasters with many colors
  - Lossless: Larger files (> disk space/bandwidth, longer downloads)
- JPEG
  - Basemap layers (no transparency support)
  - Much smaller files

Design Phase—Performance Factors

GIS Services

Design Phase—Performance Factors

GIS Services—Map Service

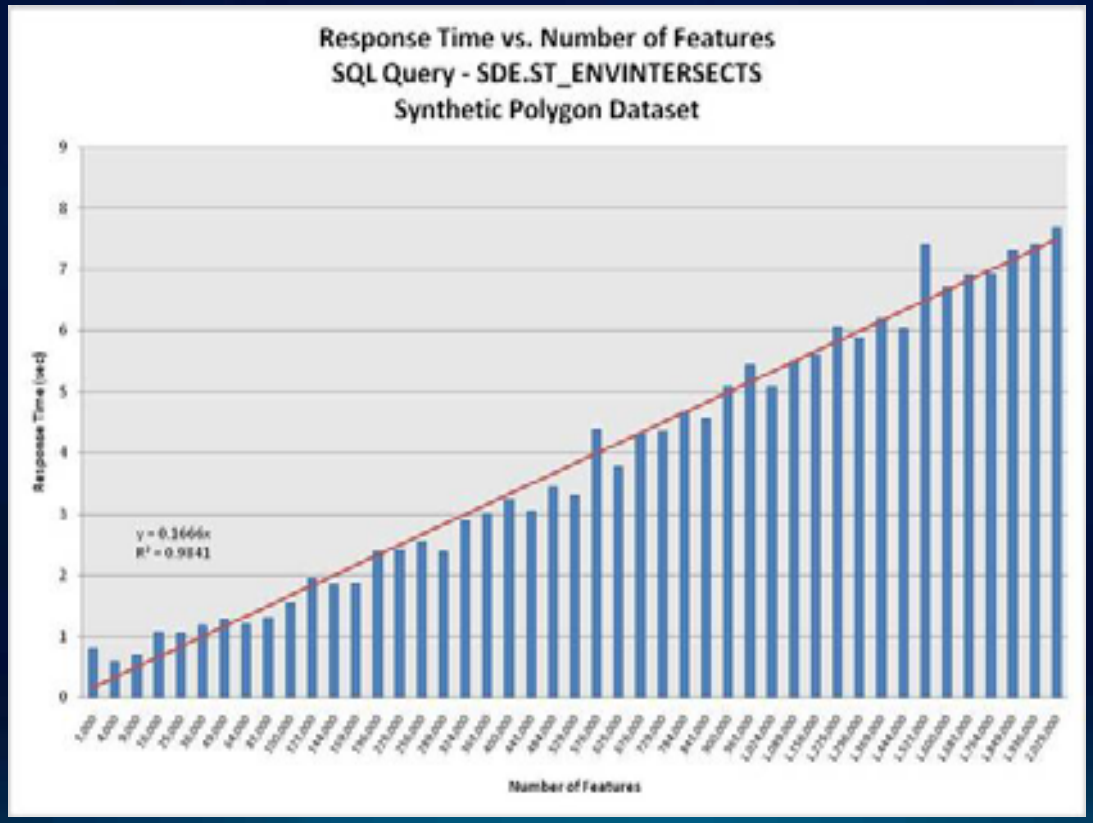
Source document (MXD) optimizations

- Keep map symbols simple.
  - Avoid multilayer, calculation-dependent symbols.
  - Spatial index.
  - Avoid reprojections on the fly.
  - Optimize map text and labels for performance.
    - Use annotations.
    - Cost for Maplex and antialiasing.
  - Use fast joins (no cross db joins).
  - Avoid wavelet compression-based raster types (MrSid, JPEG2000).

Design Phase—Performance Factors

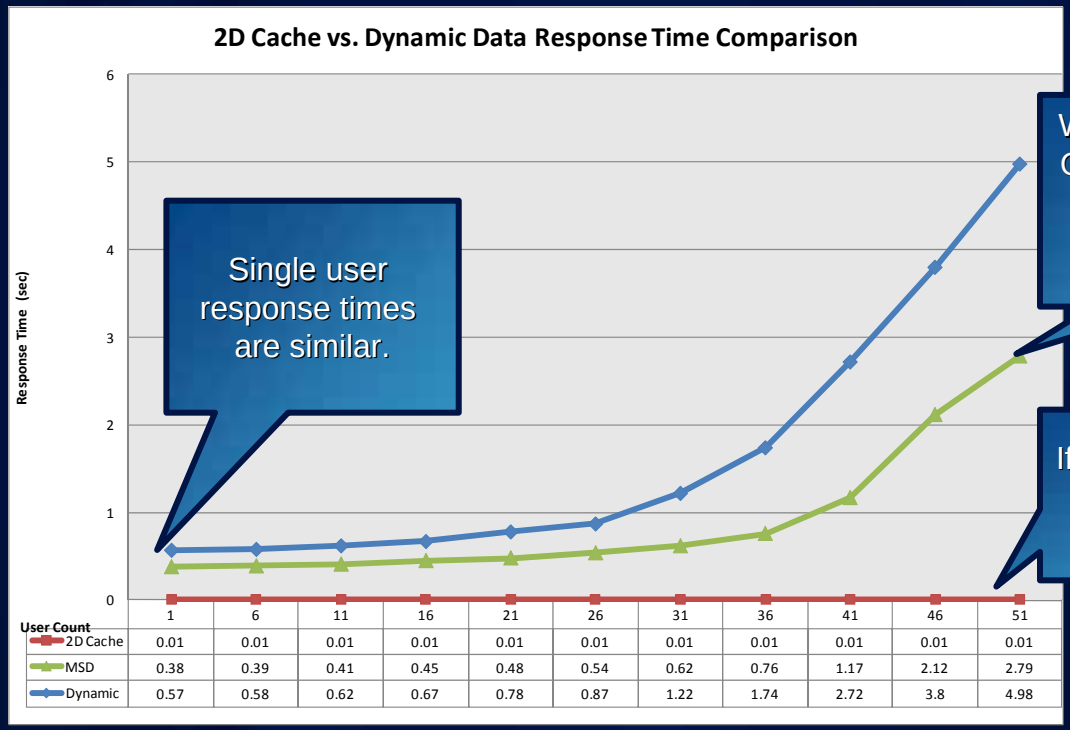
GIS Services—Map service

- **Performance linearly related to number of features**



Design Phase—Performance Factors

Performance Test Cache vs. MSD vs. MXD



User Count	1	6	11	16	21	26	31	36	41	46	51
2D Cache	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
MSD	0.38	0.39	0.41	0.45	0.48	0.54	0.62	0.76	1.17	2.12	2.79
Dynamic	0.57	0.58	0.62	0.67	0.78	0.87	1.22	1.74	2.72	3.8	4.98

Cache map services use the least of hardware resources.

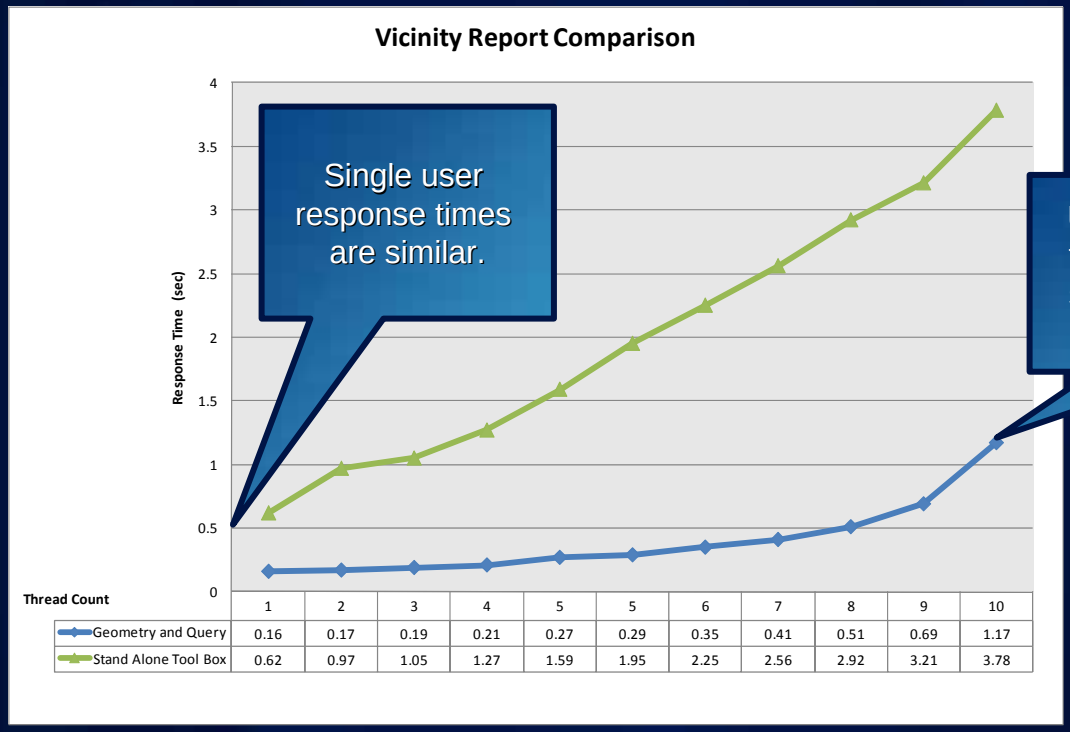
Design Phase—Performance Factors

GIS Services—Geoprocessing

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

Design Phase—Performance Factors

GIS Services—GP vs. Geometry



Thread Count	1	2	3	4	5	5	6	7	8	9	10
Geometry and Query	0.16	0.17	0.19	0.21	0.27	0.29	0.35	0.41	0.51	0.69	1.17
Stand Alone Tool Box	0.62	0.97	1.05	1.27	1.59	1.95	2.25	2.56	2.92	3.21	3.78

Design Phase—Performance Factors

GIS Services—Image

- Tiled, JPEG compressed TIFF is the best (10–400% faster).
- Build pyramids for raster datasets and overviews for mosaic datasets.
- 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).

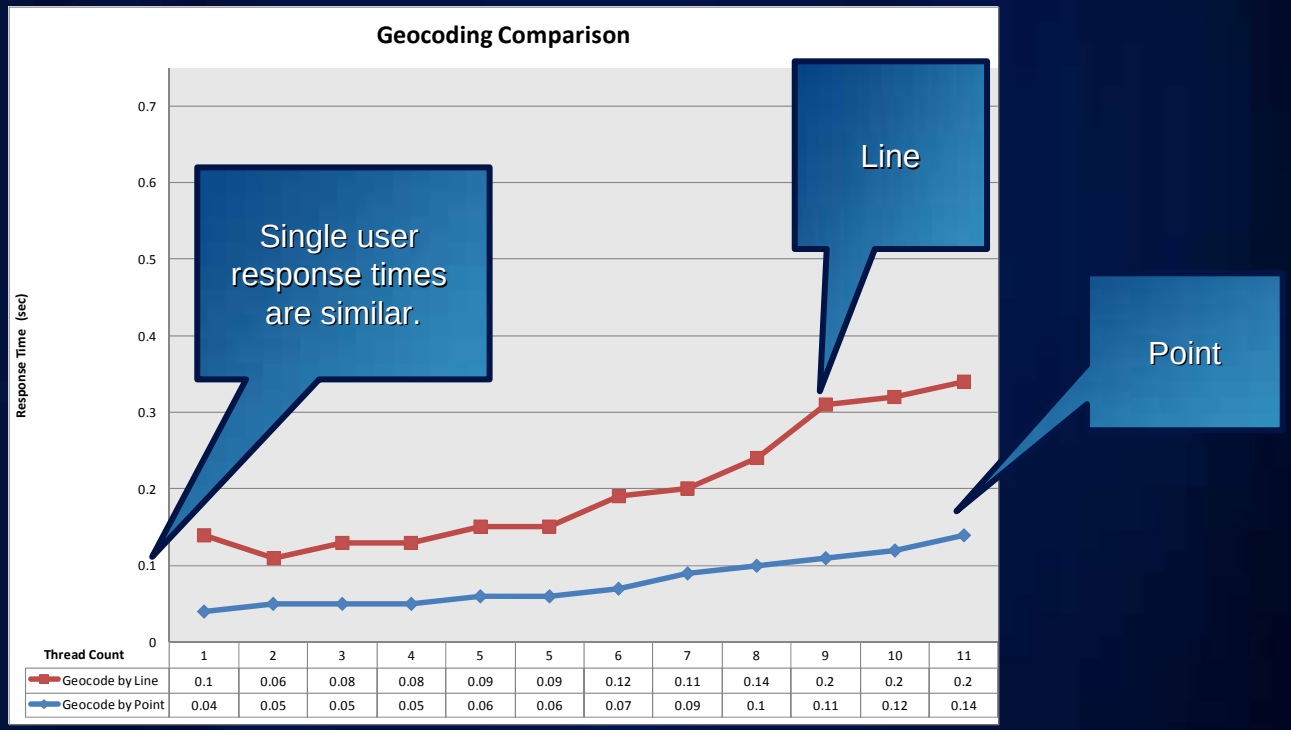
Design Phase—Performance Factors

GIS Services—Geocode

- Use local instead of UNC locator files.
- Services with large locators take a few minutes to warm up.
- New 10 Single Line Locators offer simplicity in address queries but might be slower than traditional point locators.

Design Phase—Performance Factors

GIS Services—Geocode



Thread Count	1	2	3	4	5	5	6	7	8	9	10	11
Geocode by Line	0.1	0.06	0.08	0.08	0.09	0.09	0.12	0.11	0.14	0.2	0.2	0.2
Geocode by Point	0.04	0.05	0.05	0.05	0.06	0.06	0.07	0.09	0.1	0.11	0.12	0.14

Design Phase—Performance Factors

GIS Services—Geodata

- **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 (200 K upload/ 4 MB download)**

Design Phase—Performance Factors

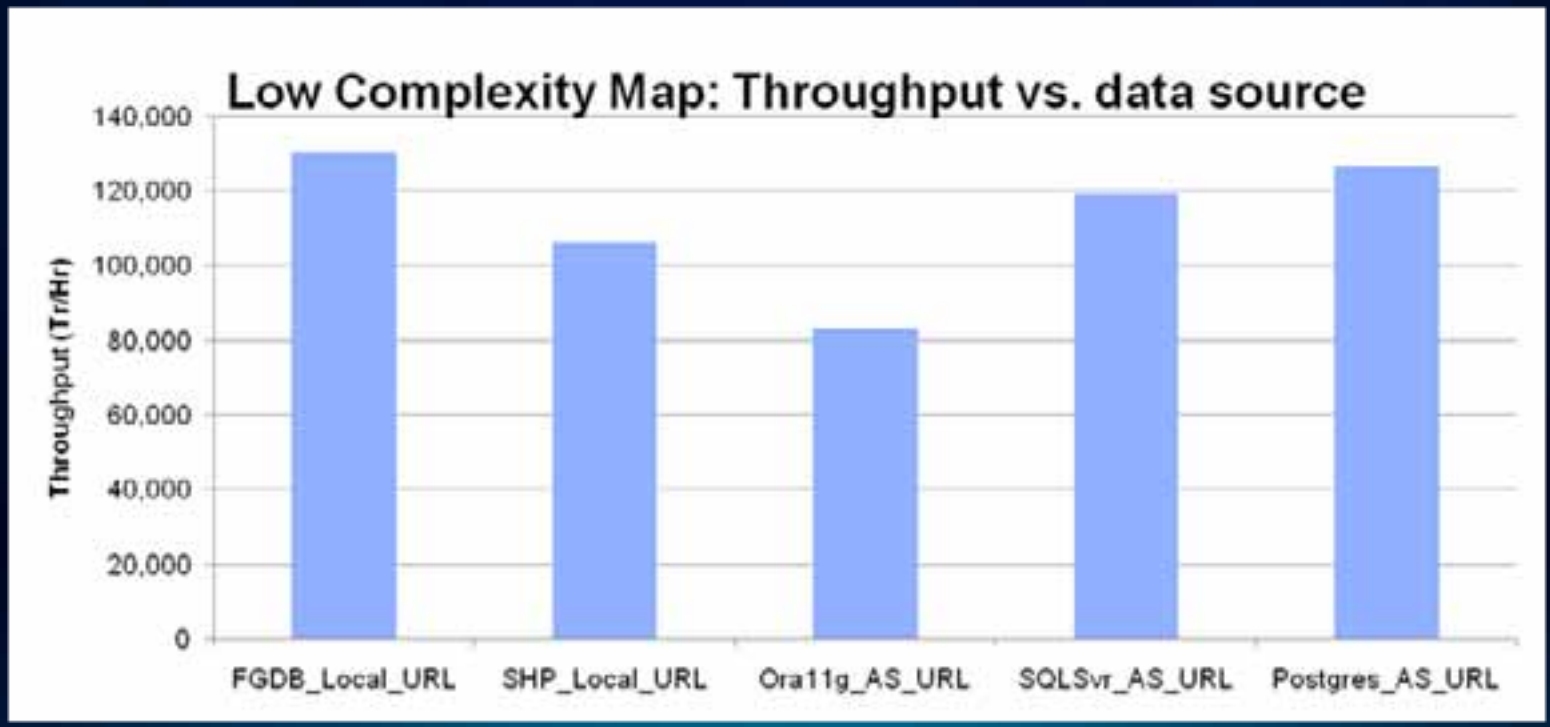
GIS Services—Feature

- Trade-off between client-side rendering and sending large amounts of data over the wire

Design Phase—Performance Factors

GIS Services—Data storage

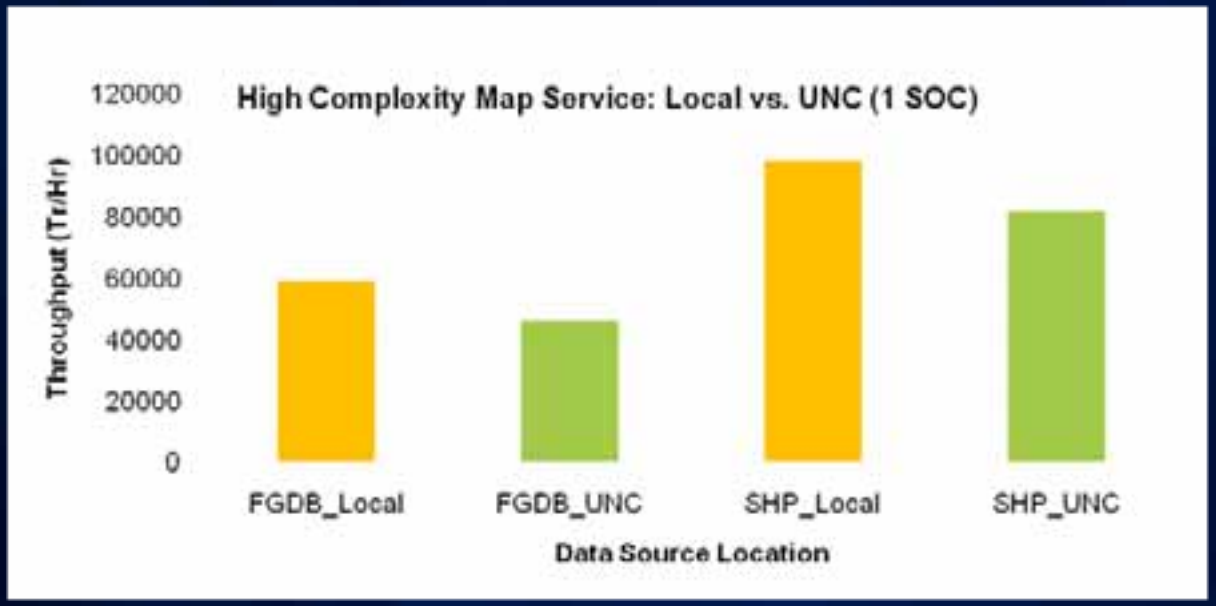
- Typically a low impact
- Small fraction (< 20%) of total response time



Design Phase—Performance Factors

GIS Services—Data source location

- Local to SOC machine
- UNC (protocol + network latency overhead)



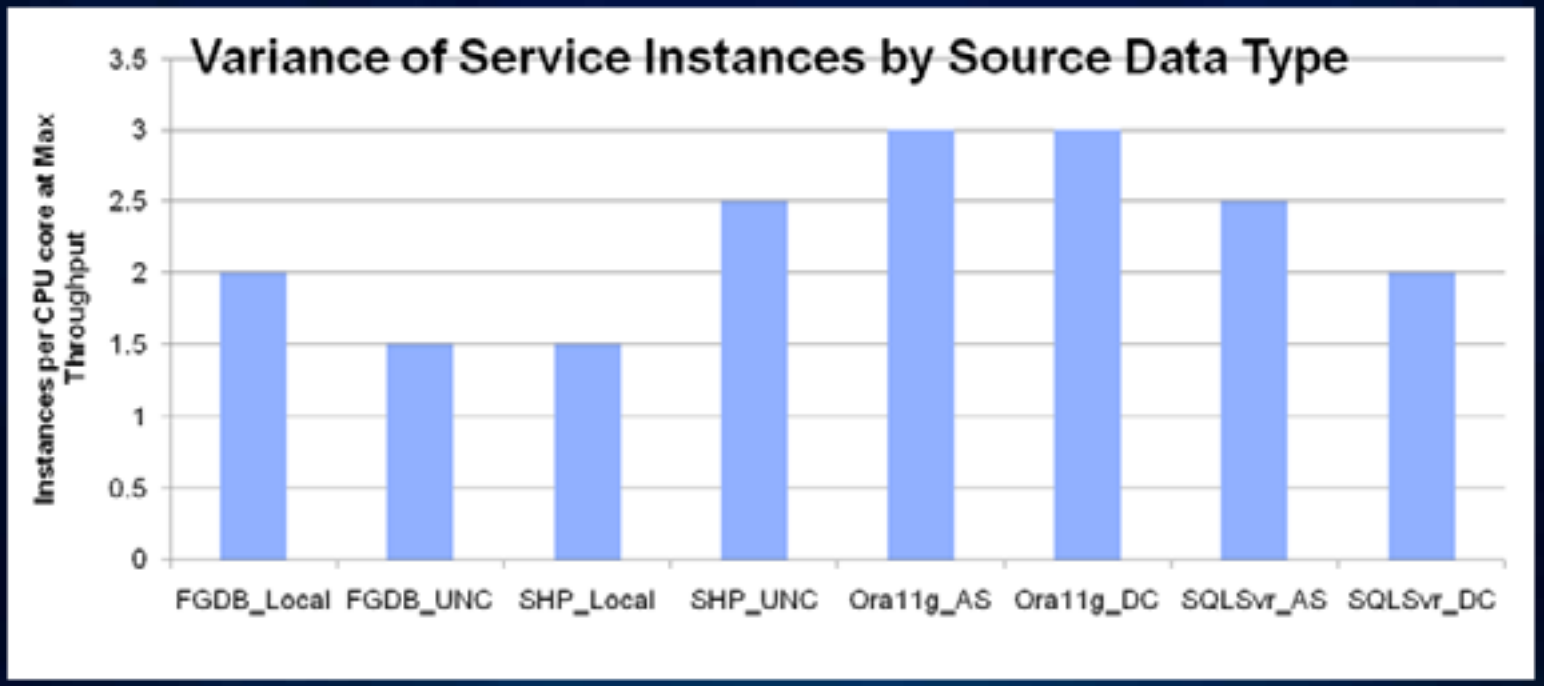
All disks being equal, locally sourced data results in better throughput.

Design Phase—Performance Factors

GIS Services—ArcSOC instances

ArcSOC Instances max = n * #CPU Cores

n = 1 … 4



If max SOC instances are underconfigured, system will not scale.

Design Phase—Capacity Planning

Design Phase—Capacity Planning

CPU Factors

1. **User load: Concurrent users or throughput**
2. **Operation CPU service time (model)—performance**
3. **CPU SpecRate**

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

subscript t = target
subscript b = benchmark
ST = CPU service time
TH = throughput
%CPU = percent CPU

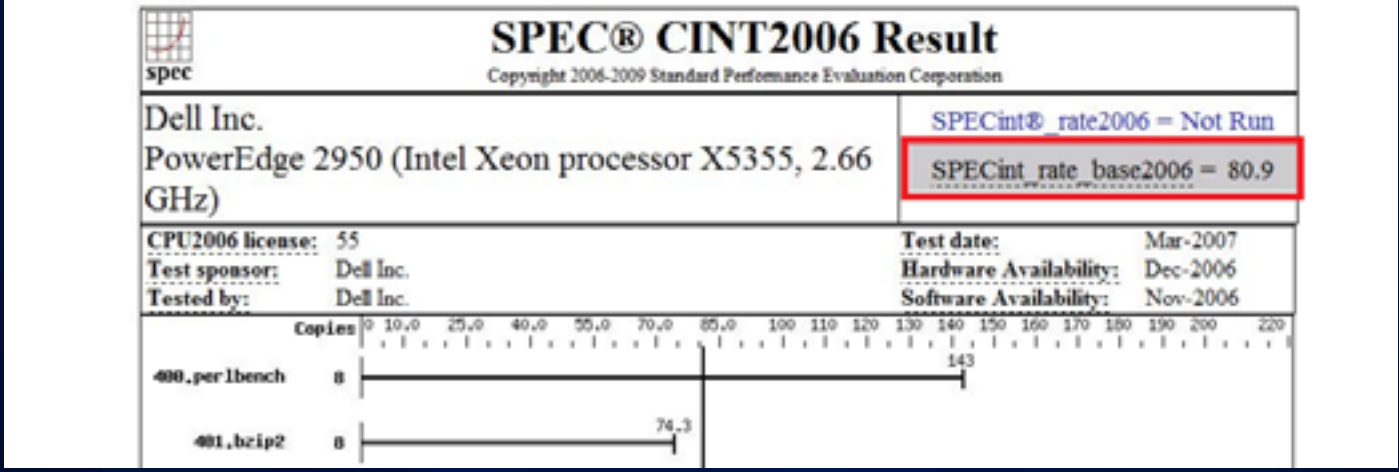
Design Phase—Capacity Planning

- Service time determined using a load test
- Capacity model expressed as Service Time

$$ST = \frac{\#CPU \times 3600 \times \%CPU}{TH \times 100}$$

Design Phase—Capacity Planning

CPU



SPEC® CINT2006 Result

Copyright 2006-2009 Standard Performance Evaluation Corporation

Dell Inc.
PowerEdge 2950 (Intel Xeon processor X5355, 2.66 GHz)

SPECint®_rate2006 = Not Run

SPECint_rate_base2006 = 80.9

CPU2006 license:	55	Test date:	Mar-2007
Test sponsor:	Dell Inc.	Hardware Availability:	Dec-2006
Tested by:	Dell Inc.	Software Availability:	Nov-2006

Design Phase—Capacity Planning

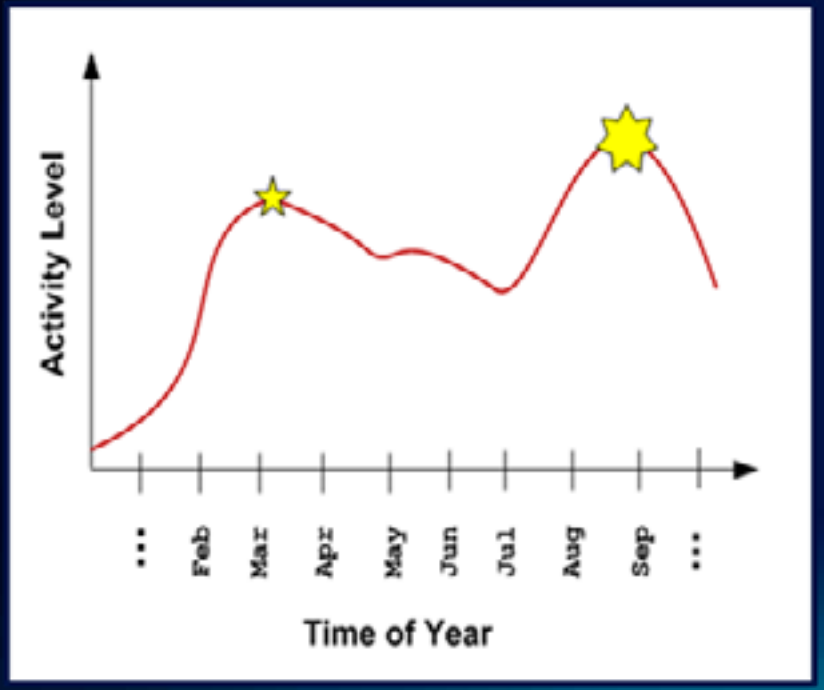
Additional Resources

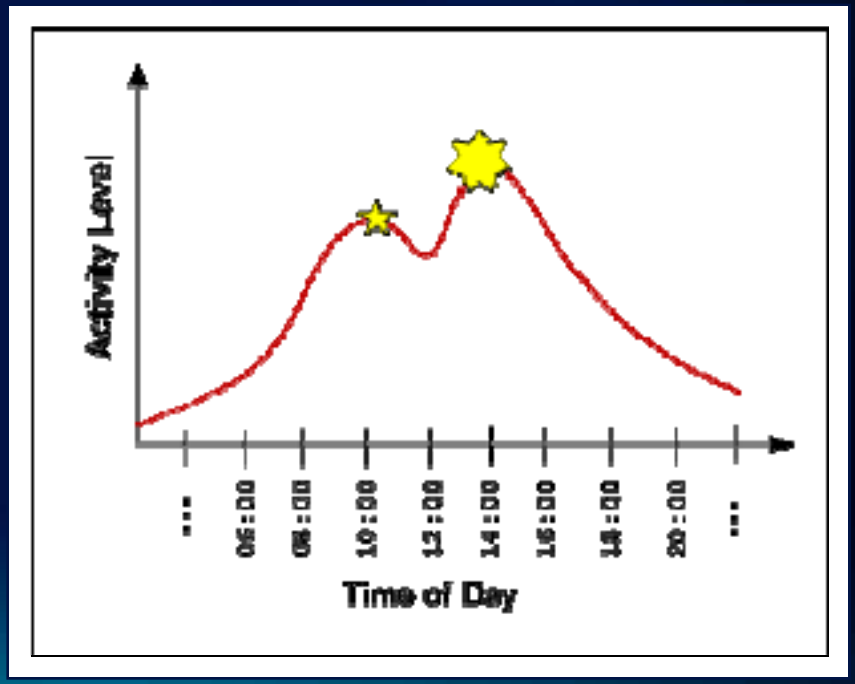
- System Designer
- Guide: Capacity Planning and Performance Benchmarks
  - resources.arcgis.com/gallery/file/enterprise-gis/details?entryID=6367F821-1422-2418-886F-FCC43C8C8E22
- CPT
  - http://www.wiki.gis.com/wiki/index.php/Capacity_Planning_Tool

Design Phase—Capacity Planning

Uncertainty of input information—Planning hour

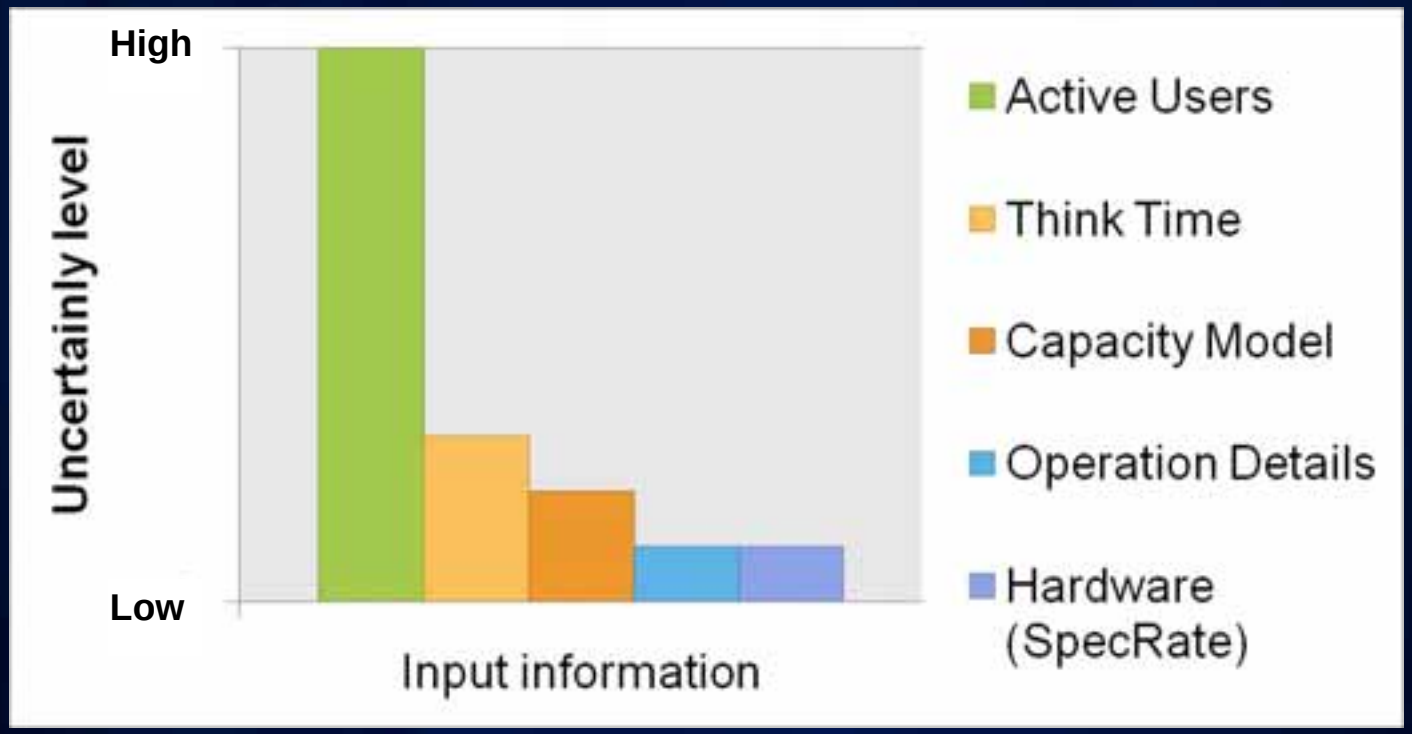
- Identify the Peak Planning Hour (most cases)





Design Phase—Capacity Planning

Uncertainty of input information

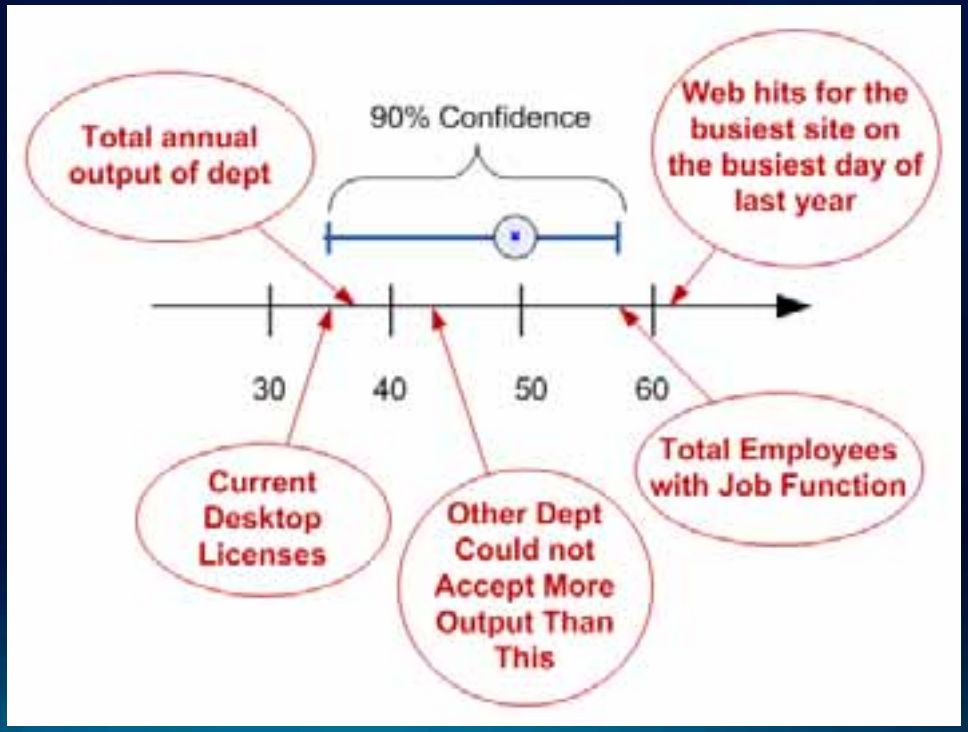


Define user load first

Design Phase—Capacity Planning

Uncertainty of input information

- License
- Total employees
- Usage logs



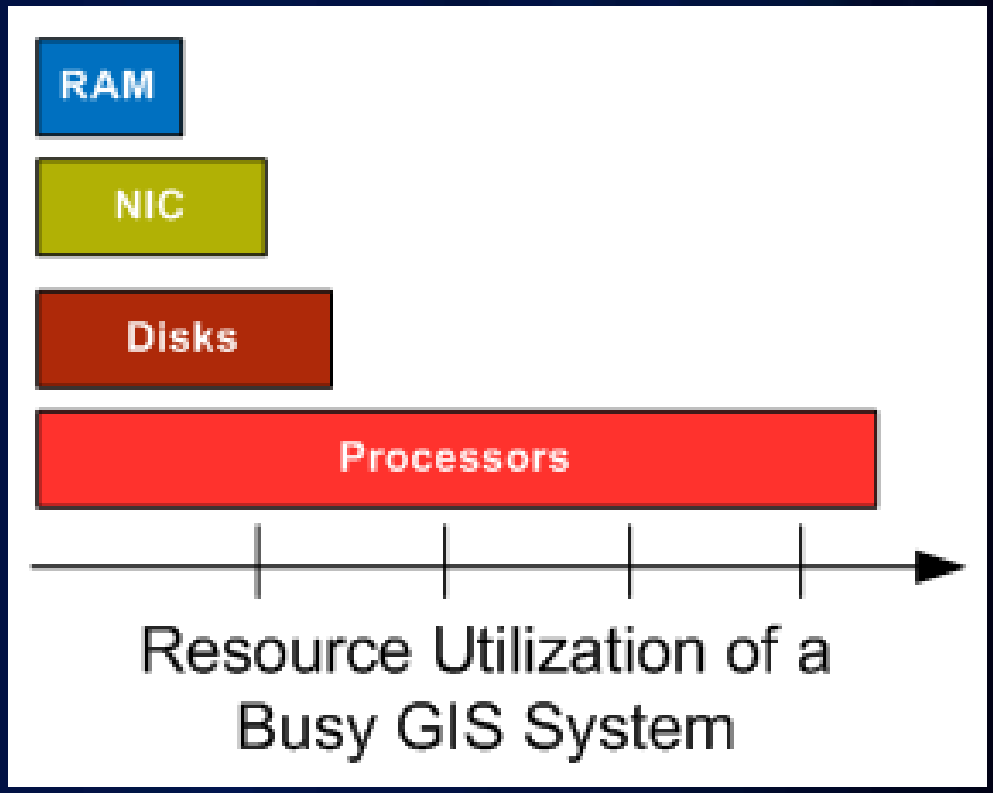
Design Phase—Performance Factors

Hardware Resources

Design Phase—Performance Factors

Hardware Resources

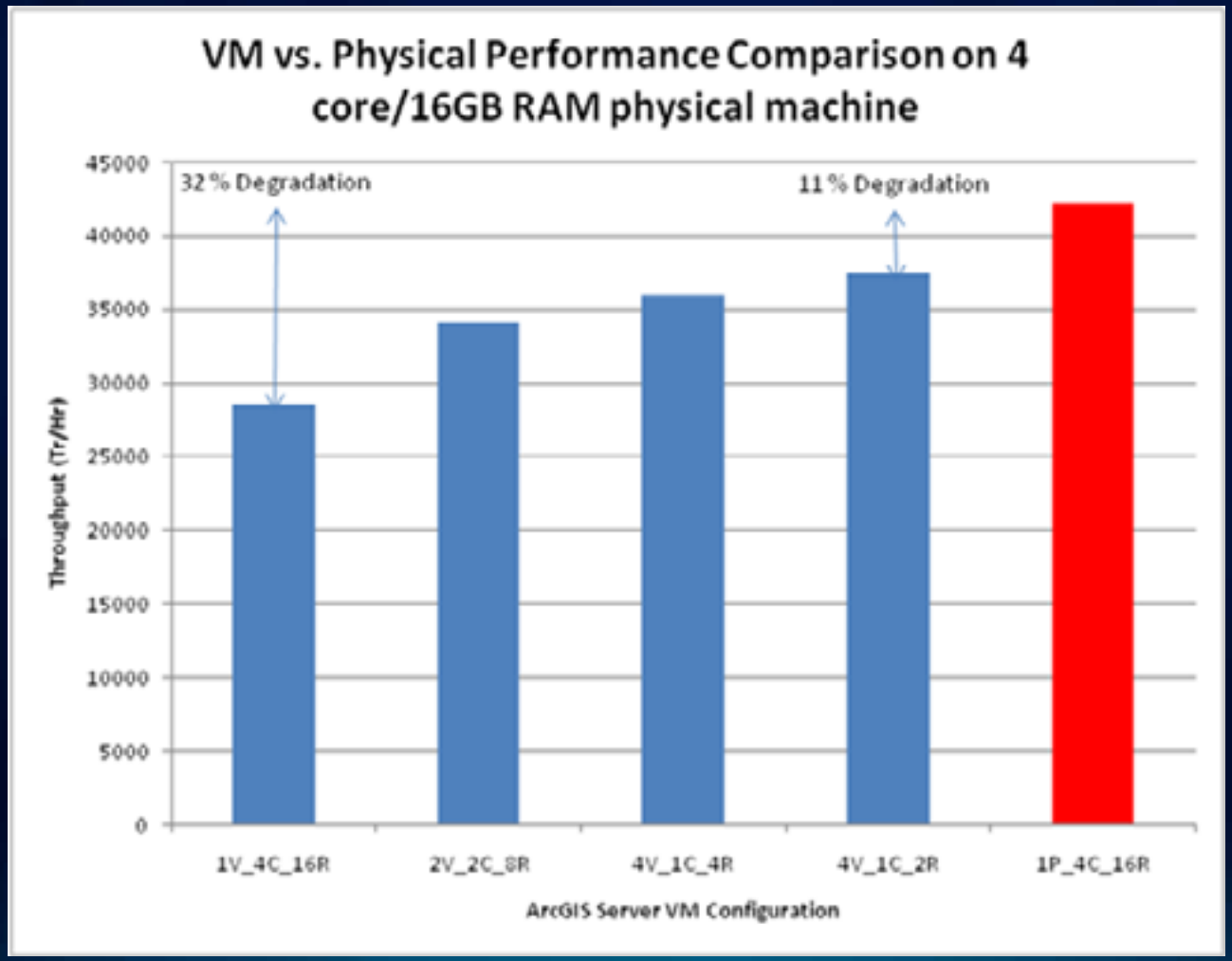
- CPU
- Network bandwidth and latency
- Memory
- Disk



Most well-configured and tuned GIS systems are processor-bound.

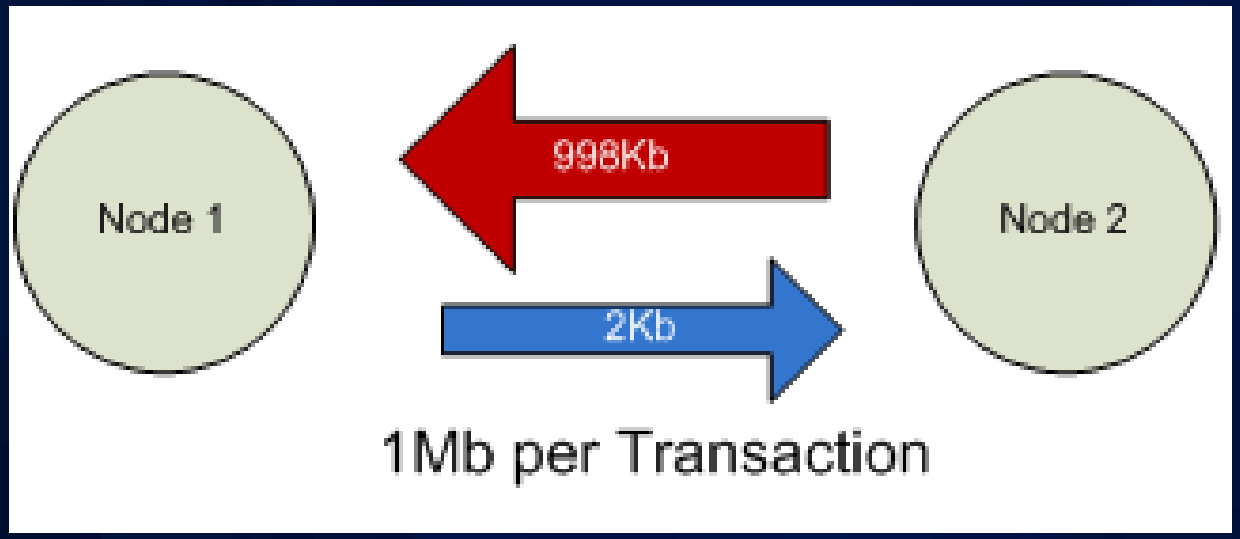
Design Phase—Performance Factors

Hardware Resources—Virtualization overhead



Design Phase—Performance Factors

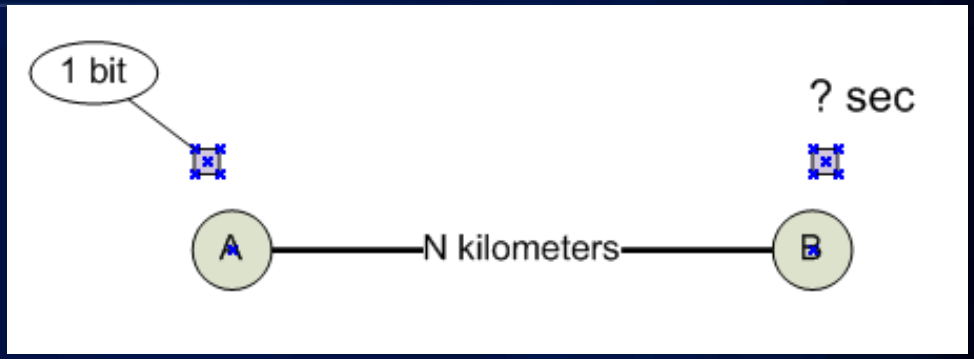
Hardware Resources—Network bandwidth directionality



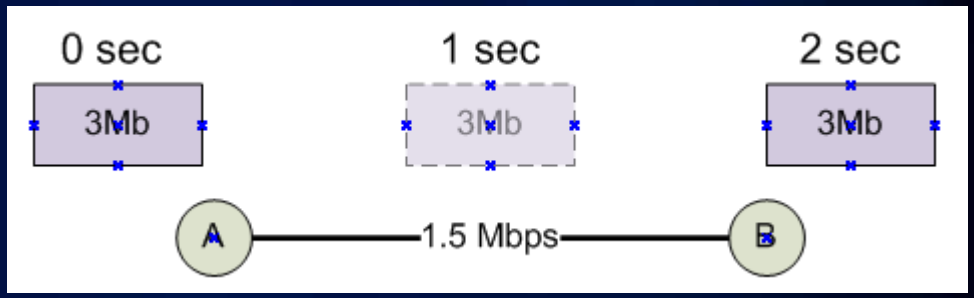
Design Phase—Performance Factors

Hardware Resources—Network

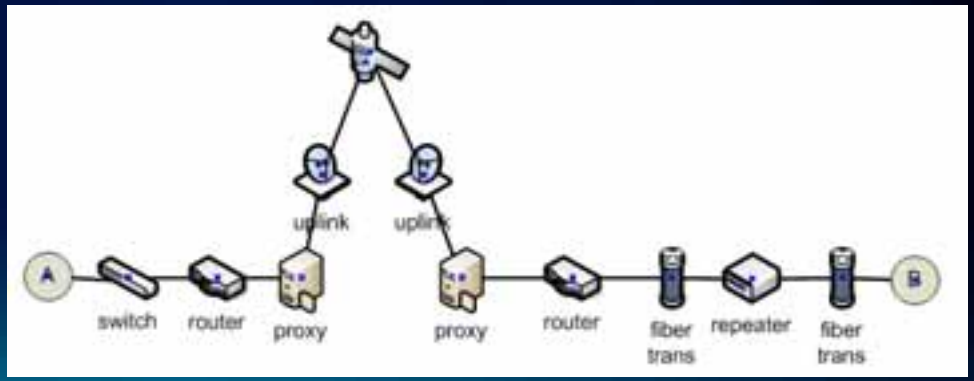
1. Distance



2. Payload

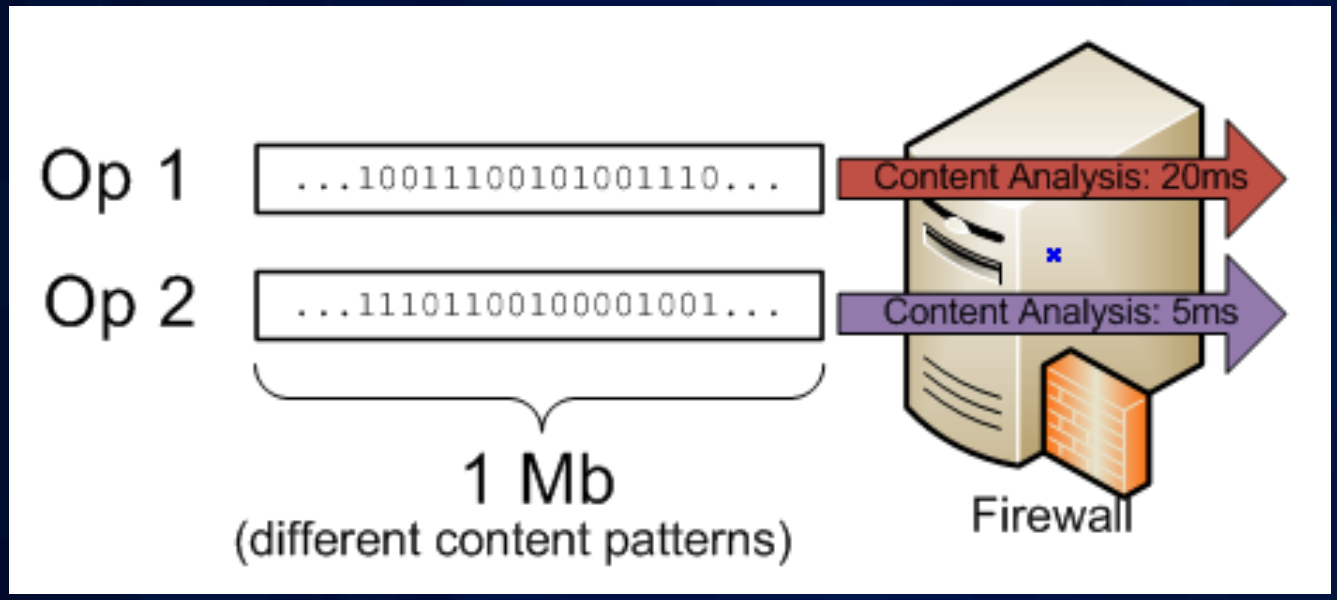


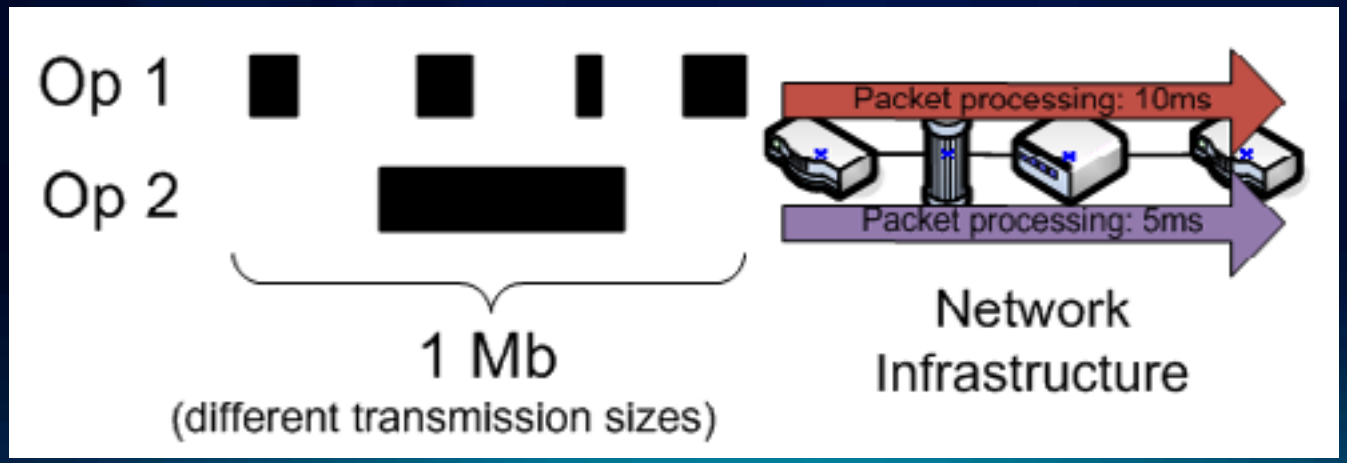
3. Infrastructure



Design Phase—Performance Factors

Hardware Resources—Network

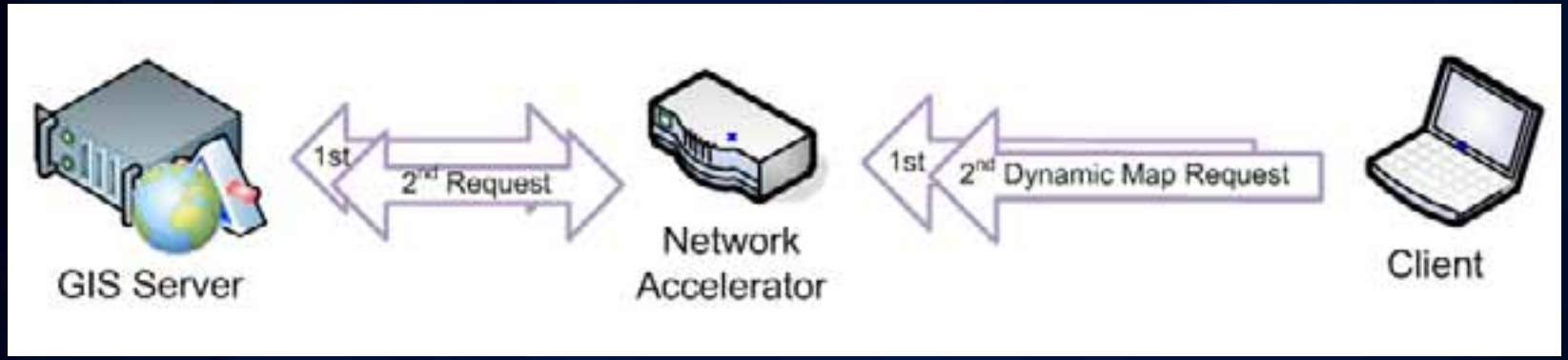


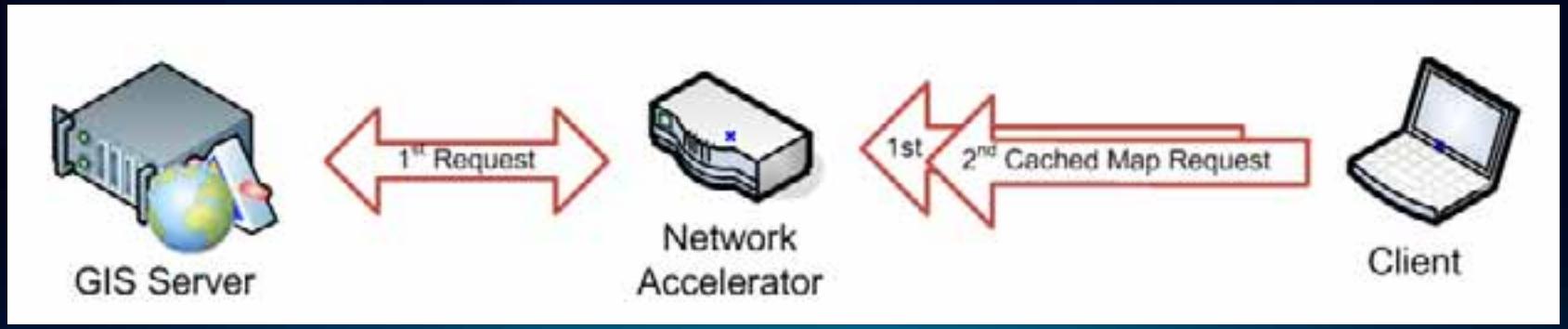


Design Phase—Performance Factors

Hardware Resources—Network

- **Network accelerator improves performance of repeated request**





Design Phase—Performance Factors

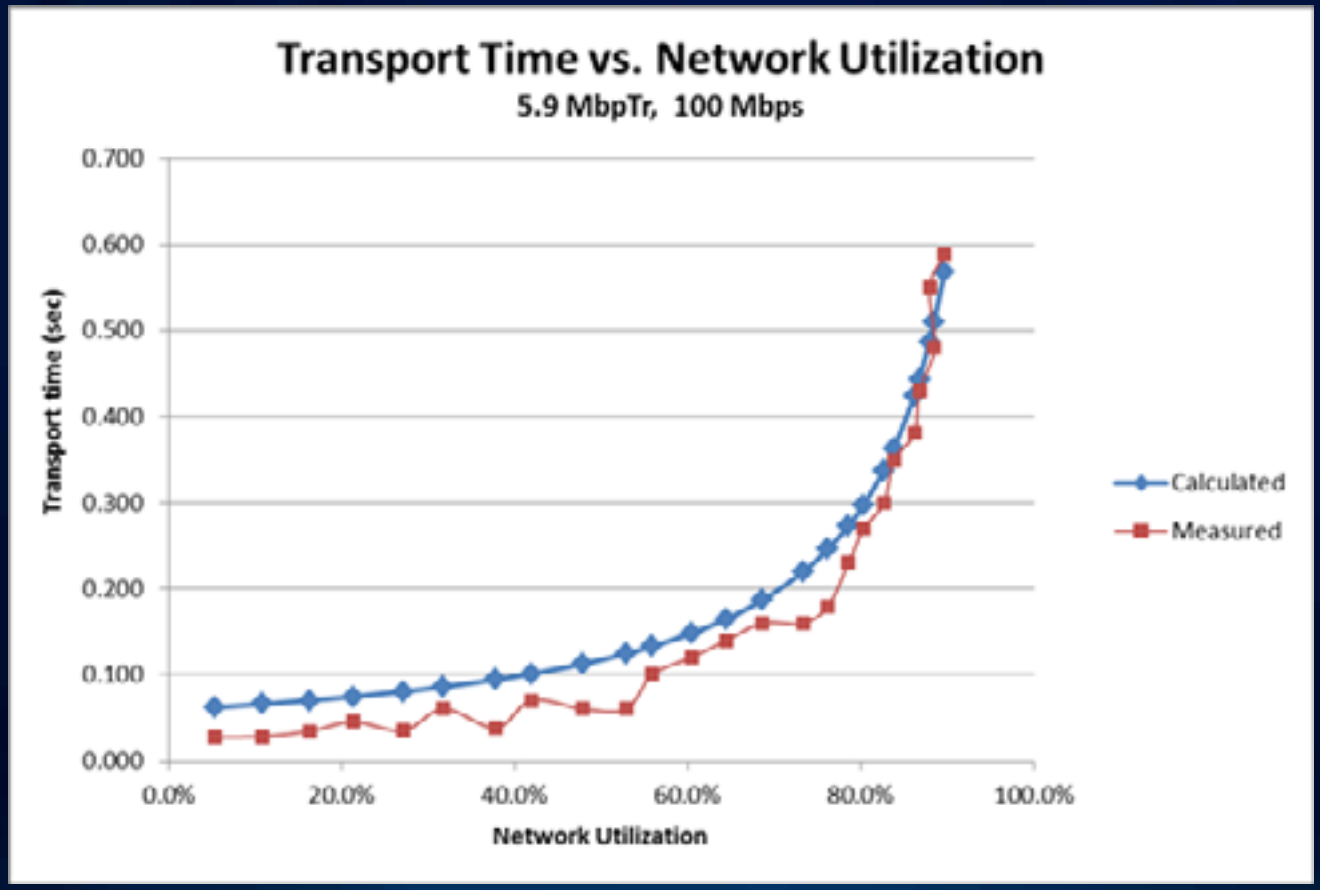
Hardware Resources—Network

- **Impact of service and return type on network transport time**
  - **Compression**
  - **Content, e.g., Vector vs. Raster**
  - **Return type, e.g., JPEG vs. PNG**

					Network Traffic Transport Time (sec)					
					56 kbps	1.54 Mbps	10 Mbps	45 Mbps	100 Mbps	1 Gbps
Application Type	Service/Op	Content	Return Type	Mb/Tr	0.056	1.540	10.000	45.000	100.000	1000.000
ArcGIS Desktop	Map	Vector		10	178.571	6.494	1.000	0.222	0.100	0.010
Citrix/ArcGIS	Map	Vectror+Image	ICA Comp	1	17.857	0.649	0.100	0.022	0.010	0.001
Citrix/ArcGIS	Map	Vector	ICA Comp	0.3	5.357	0.195	0.030	0.007	0.003	0.000
ArcGIS Server	Map	Vector	PNG	1.5	26.786	0.974	0.150	0.033	0.015	0.002
ArcGIS Server	Image		JPG	0.3	5.357	0.195	0.030	0.007	0.003	0.000
ArcGIS Server	Map Cache	Vector	PNG	0.1	1.786	0.065	0.010	0.002	0.001	0.000
ArcGIS Server	Map Cache	Vector+Image	JPG	0.3	5.357	0.195	0.030	0.007	0.003	0.000

Design Phase—Performance Factors

Hardware Resources—Network



Design Phase—Performance Factors

Hardware Resources—Memory

Item	Low	High	Delta
XenApp Session	500 MB	1.2 GB	140%
Database Session	10 MB	75 MB	650%
Database Cache	200 MB	200 GB	99,900%
SOC Process (Dynamic Map Service)	50 MB	500 MB	900%
SOC Process (Image Service)	20 MB	1,024 MB	5,020%
SOC Process (Geoprocessing Service)	100 MB	2,000 MB	1,900%
SOM	30 MB	70 MB	133%

Wide ranges of memory consumptions

Performance Engineering—Solution Development Phase

Development Phase—Testing

- Performance and load test early to validate that nonfunctional requirements can be met.

Development Phase—Testing

Performance Testing—Objectives

- Define Objectives
  - Contractual Service Level Agreement?
- Bottlenecks
- Capacity
- Benchmark

Development Phase—Testing

Performance Testing —Prerequisites

- Functional testing completed
- Performance tuning

Development Phase—Testing

Performance Testing—Test Plan

- Test Plan
  - Workflows
    - Expected User Experience (Pass/Fail Criteria)
    - Single User Performance Evaluation (Baseline)
    - Think Times
    - Active User Load
    - Pacing
    - Valid Test Data and Test Areas
  - Testing Environment
    - Scalability/Stability
    - IT Standards and Constraints
    - Configuration (GIS and Non-GIS)

Development Phase—Testing

Performance Testing—Test tools

Development Phase—Testing

Performance Testing —Test tools

- **Tool selection depends on objective**
  - **Commercial tools all have system metrics and correlation tools.**
  - **Free tools typically provide response times and throughput, but leave system metrics to the tester to gather and report on.**

Development Phase—Testing Tools

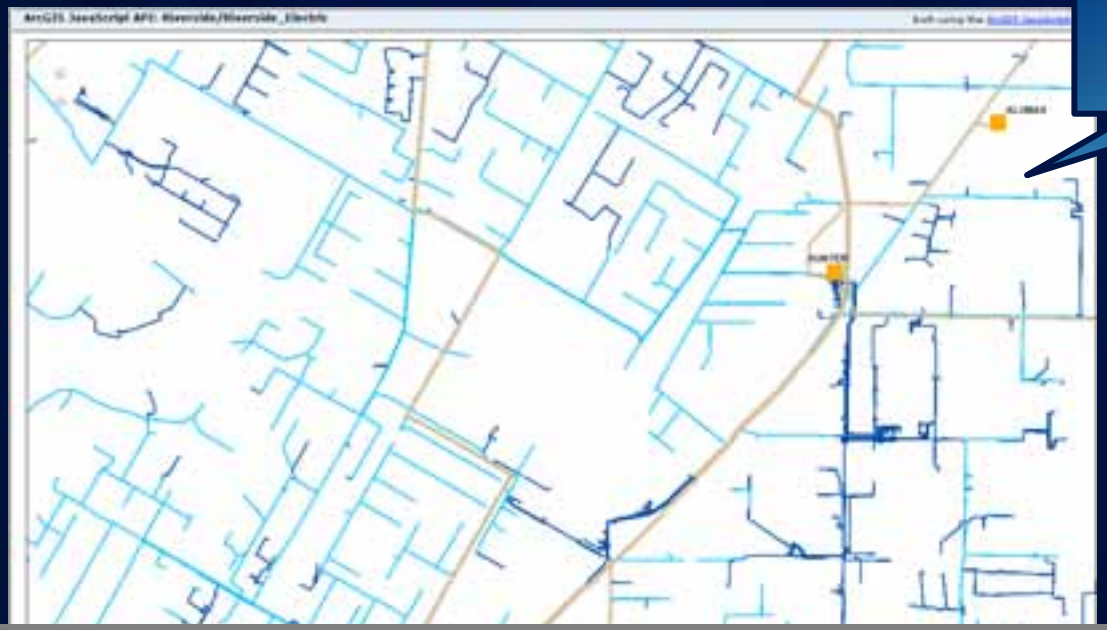
Test Tools	Open Source	Pros	Cons
LoadRunner	No	•Industry Leader •Automatic negative correlations identified with service level agreements •HTTP web testing •Click and script •Very good tools for testing SOA • Test results stored in database •Thick client testing •Can be used for bottleneck analysis	•High cost •Test development in C programming language •Test metrics difficult to manage and correlate •Poor user community with few available examples
Silk Performer	No	•Good solution for testing Citrix •Wizard driven interface guides the user •Can be used for bottleneck analysis	•Moderate to high cost •Test metrics are poor •Test development uses proprietary language •Test metrics difficult to manage and correlate •Poor user community with few available examples
Visual Studio Test Team	No	•Low to moderate cost •Excellent test metric reporting •Test scripting in C# or VB .NET •Unit and web testing available •Blog support with good examples •Very good for bottleneck analysis	•No built in support for AMF •No thick client options •Moderate user community
JMeter	Yes	•Free •Tool	•Provides only response times •Poor user community with few available examples

Development Phase—Testing

Test Data

Development Phase—Testing

Test Data



Area of Interest

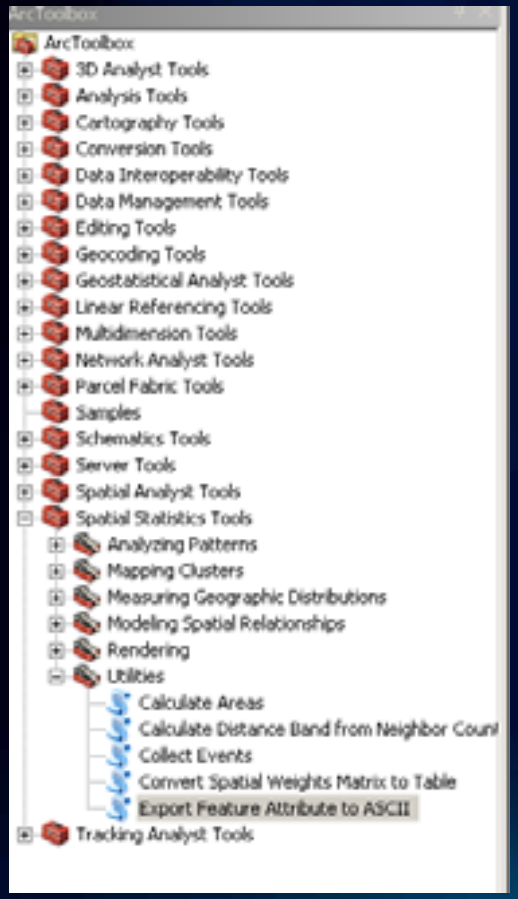
QueryString

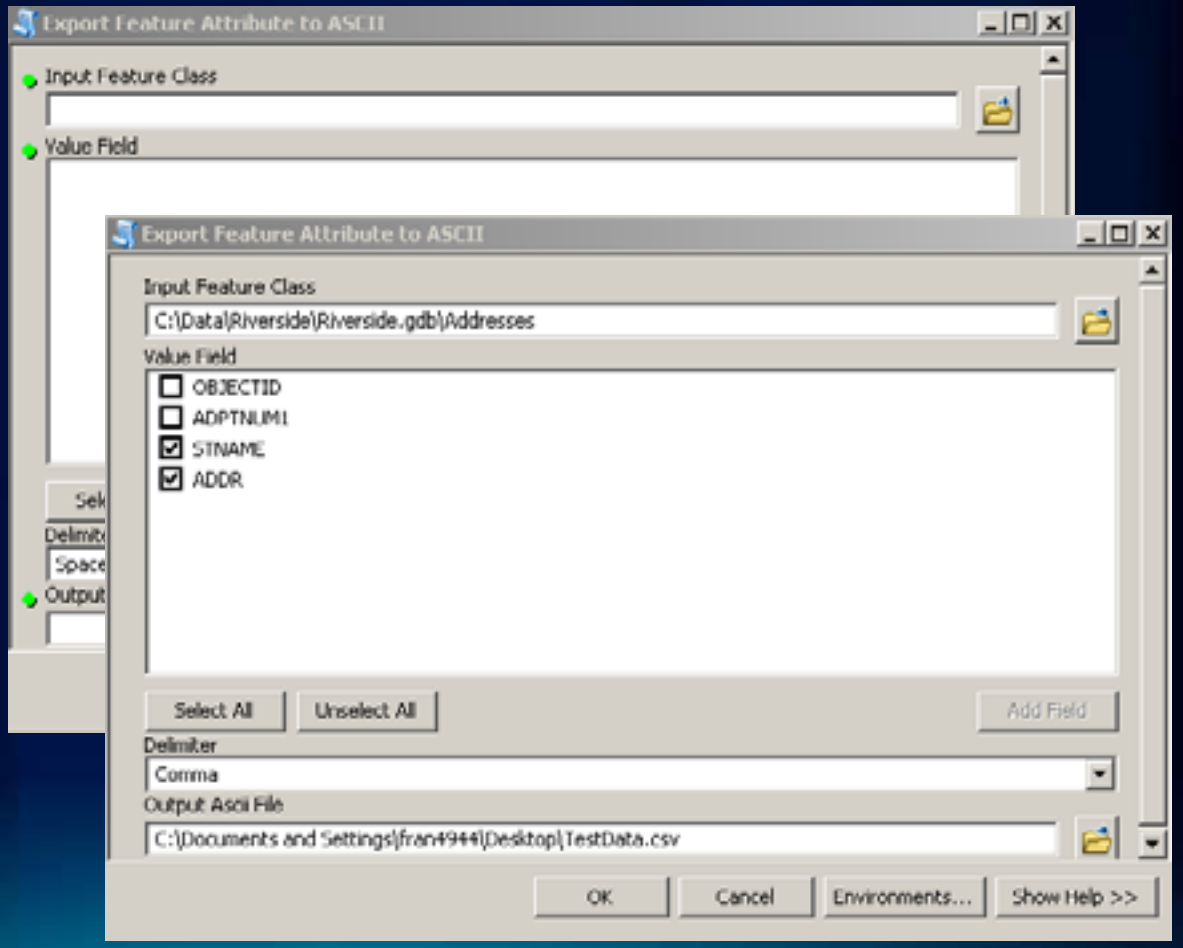
Name	Value
f	image
dpi	96
transparent	true
format	png8
bbox	{"xmin":6219593.737018972,"ymin":2303862.765275147,"xmax":6231478.566248391,"ymax":2311468.277924415,"spatialReference":{"wkid":2230}}
bboxSR	2230
imageSR	2230
size	1222,782

Selected Extent From HTTP Debugging Proxy

Development Phase—Testing

Attribute Data





Development Phase—Testing

Test Data

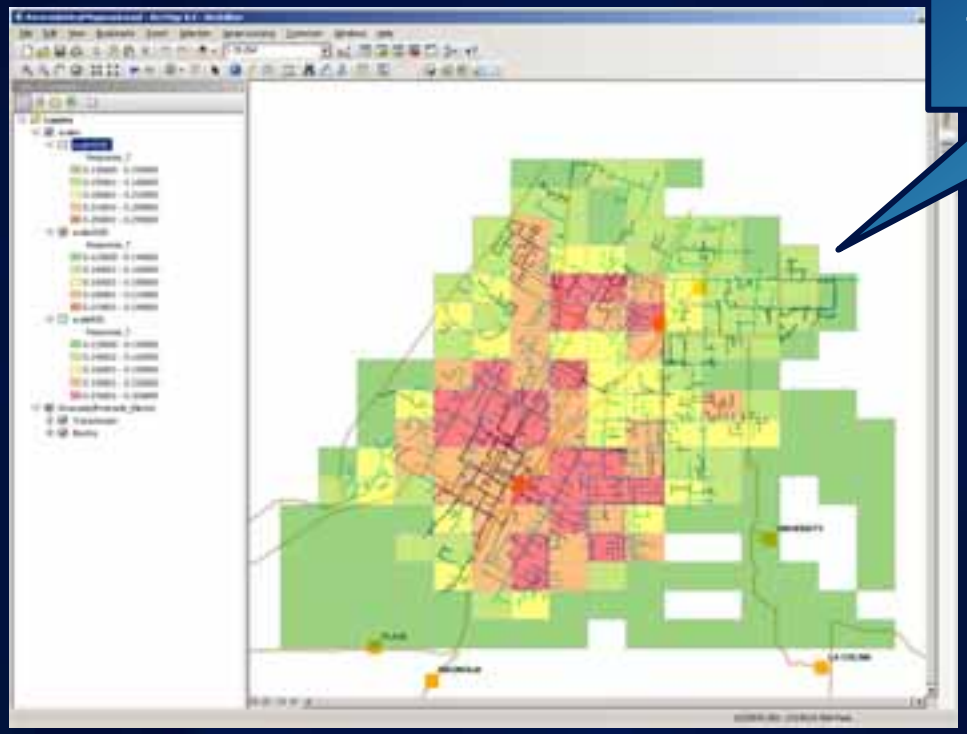
Generate Bboxes

One simple example of Python script to generate Bboxes

```
"""
Generate Bboxes from selected fiveride extent from ...
Note* Image size=1222,762
"""
__author__ = ""
__version__ = "0.1"

import random

def generateBoxs(fullExtent, gridControl):
    bBoxs = []
    bBoxs.append(fullExtent)
    width  = fullExtent[2]-fullExtent[0]
    height = fullExtent[3]-fullExtent[1]
    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
                bBoxs.append((minX, minY, maxX, maxY))
    return bBoxs

def writeTuple(path, arr):
    try:
        f = open(path, 'w')
        for item in arr:
            f.write(",".join([str(x) for x in item])+"\n")
        f.close()
    except IOError, (errno, strerror):
        print path
        print "writeTuple I/O error(%s): %s" % (errno, strerror)
    except:
        print "writeTuple Unexpected error:", sys.exc_info()[0]
        raise

if __name__=="__main__":

    extent =(6219593.737018972,2303862.765275147,6231478.566248391,2311468.277924415)
    grid = [2, 8, 16]
    bBoxs = generateBoxs(extent, grid)
    for item in bBoxs:
        print item
    writeTuple("C:\\test.csv", bBoxs)
```

Development Phase—Testing

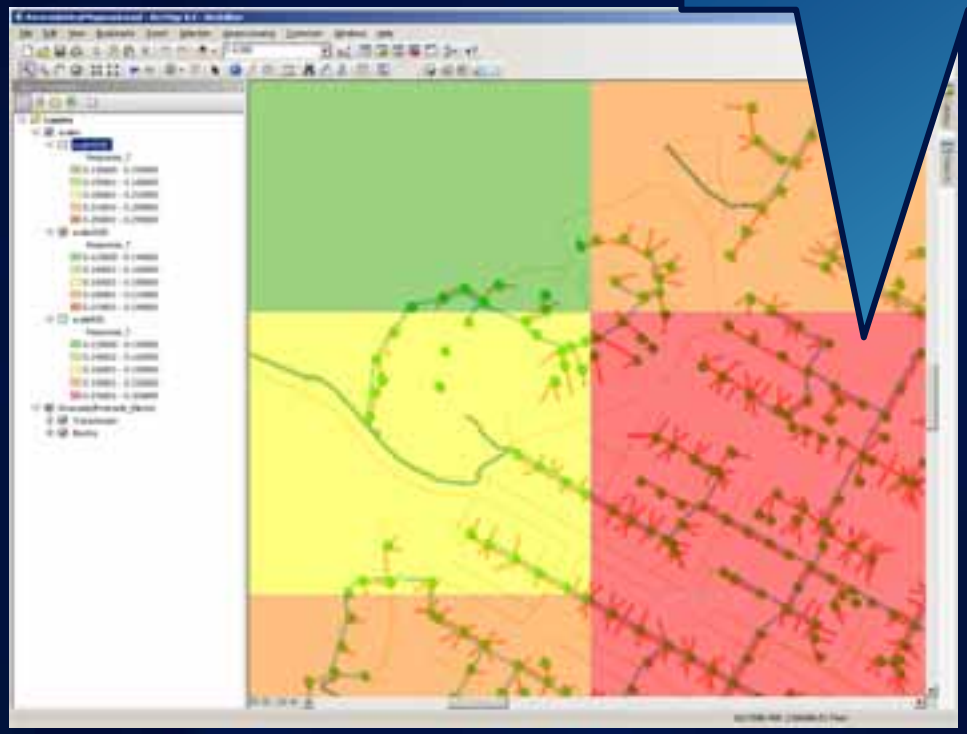
Test Data



Heat map based on response times from ArcGIS Server

Development Phase—Testing

Test Data



Observe correlation between feature density and performance.

Demo

Discovering Capabilities using ArcGIS REST

Discovering Test Data with System Test Tool

Add STT intro slide

Development Phase—Testing

Test Scripts—Request Profiling

- Sample selected functions
  - Observe response times with variable data.
  - Observe system resources while sampling.
  - Track changes over time if system is changing.
  - Example
    - Use Fiddler to record user workflows and find the expected single user response times for transactions.
  - Benefits
    - Low cost method for evaluating systems

Development Phase—Testing

Test Scripts

- 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.

Development Phase—Testing

Test Scripts—Visual Studio Quick Introduction

Transaction

HTTP Request

Query String parameter referencing data source

Data source

Development Phase—Testing

Load Test

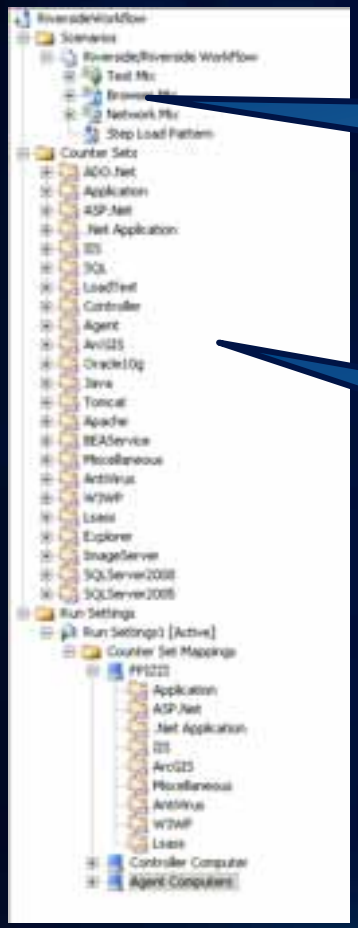
Development Phase—Testing

Load Test

- Create load test.
  - Define user load.
    - Max users
    - Step interval and duration
  - Create machine counters to gather raw data for analysis.
- Execute.

Development Phase—Testing

Load Test—Visual Studio Quick Introduction



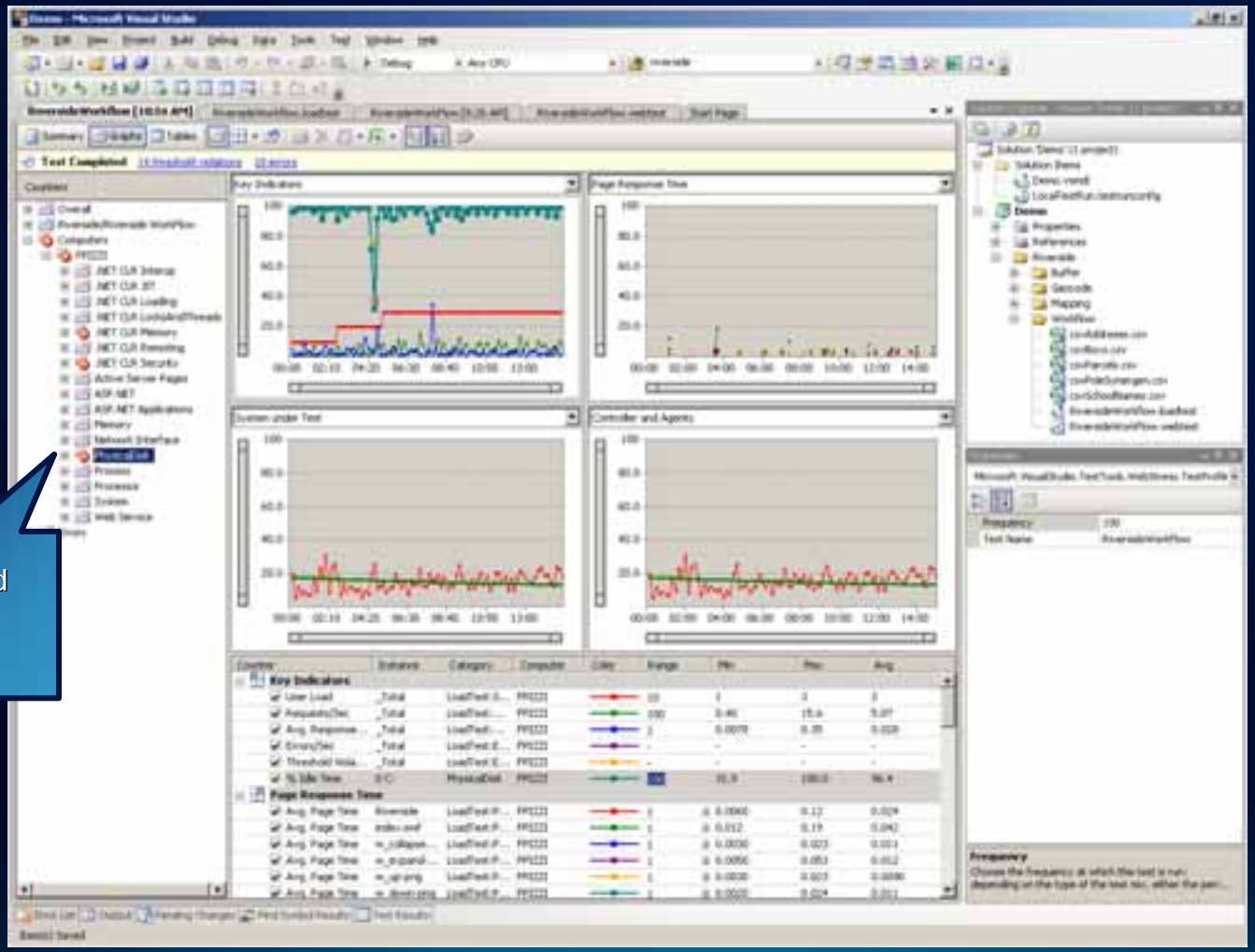
Scenarios:
Test Mix (WebTest or Unit Test),
Browser Mix,
Network Mix,
Step Loads

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

Run Settings:
Counter Set Mappings – Machine metrics
Test duration

Development Phase—Testing

Load Test—Visual Studio



Development Phase—Testing

Execute

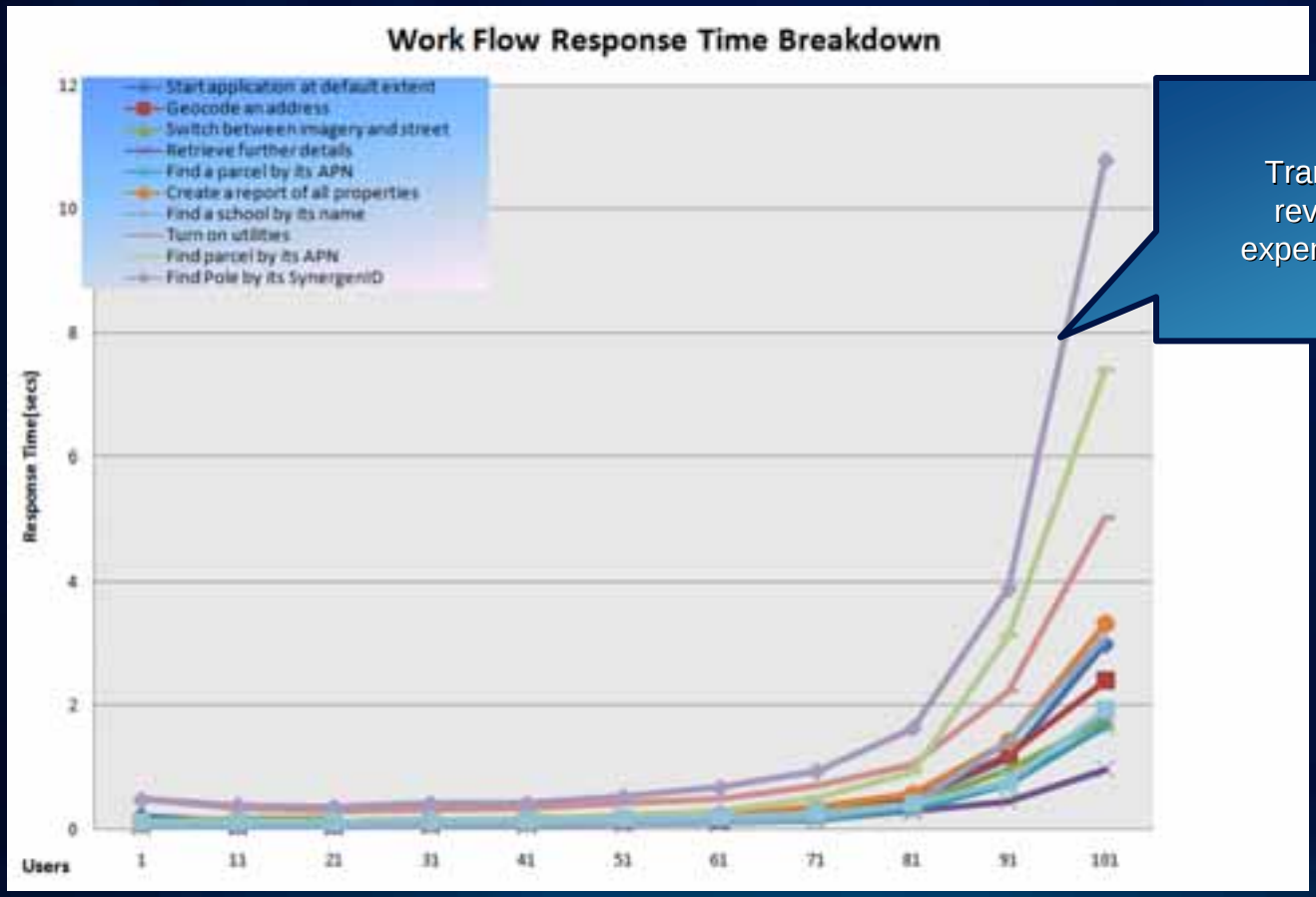
- **Ensure**
  - **Virus scan is off.**
  - **Only target applications are running.**
  - **Application data is in the same state for every test.**
  - **Good configuration management is critical to getting consistent load test results.**

Development Phase—Testing

Analysis

Development Phase—Testing

Analysis—Workflow response time breakdown



Transaction profile reveals the most expensive operations.

Development Phase—Testing

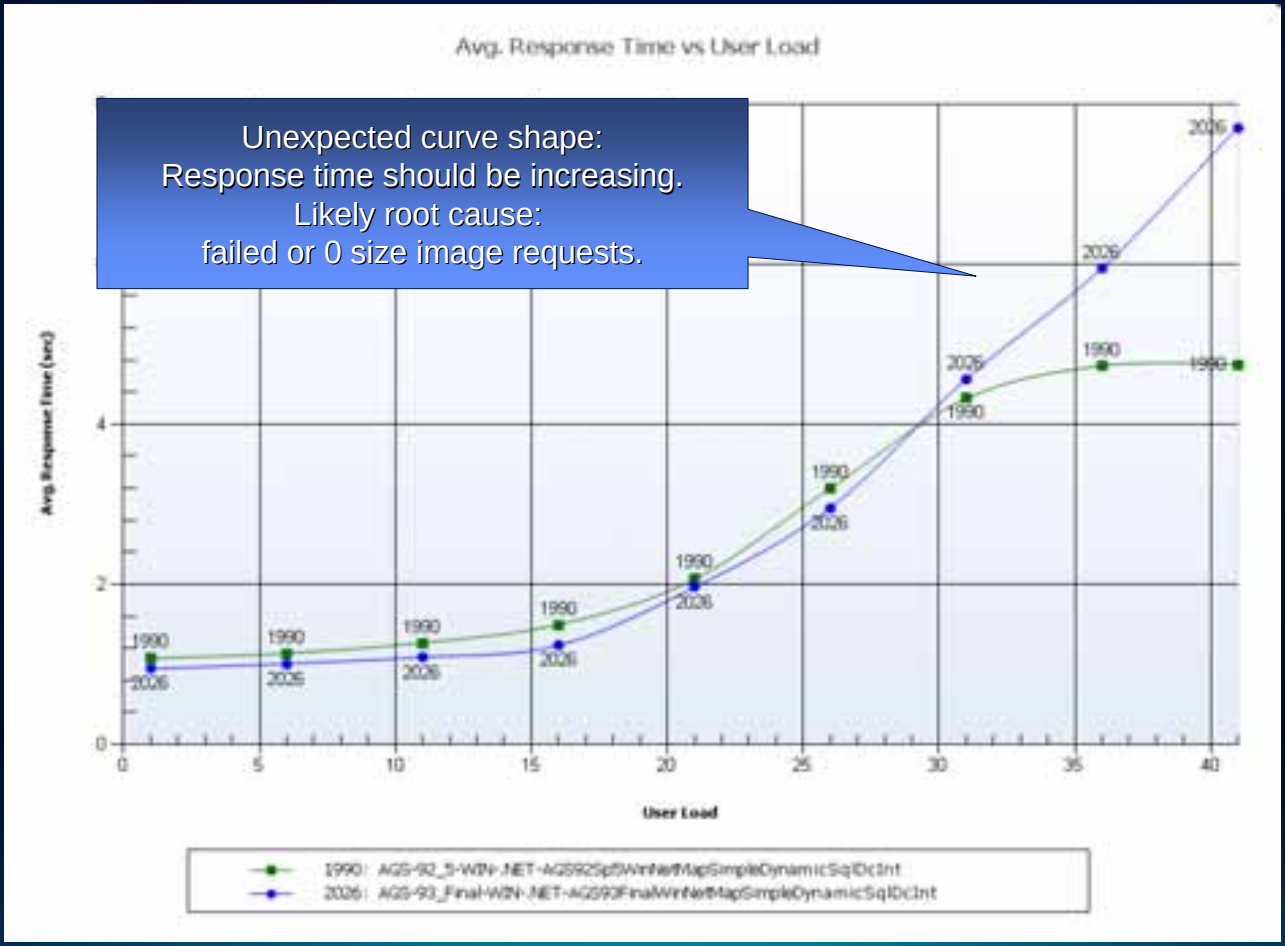
Analysis—Compare and correlate key measurements

- **Most counters and utilization should be increasing with increased load:**
  - **Throughput**
  - **Response time**
  - **Metrics**
    - **CPU**
    - **Network**
    - **Disk**
    - **Memory**
  - **Errors**

Development Phase—Testing

Analysis—Compare and correlate key measurements

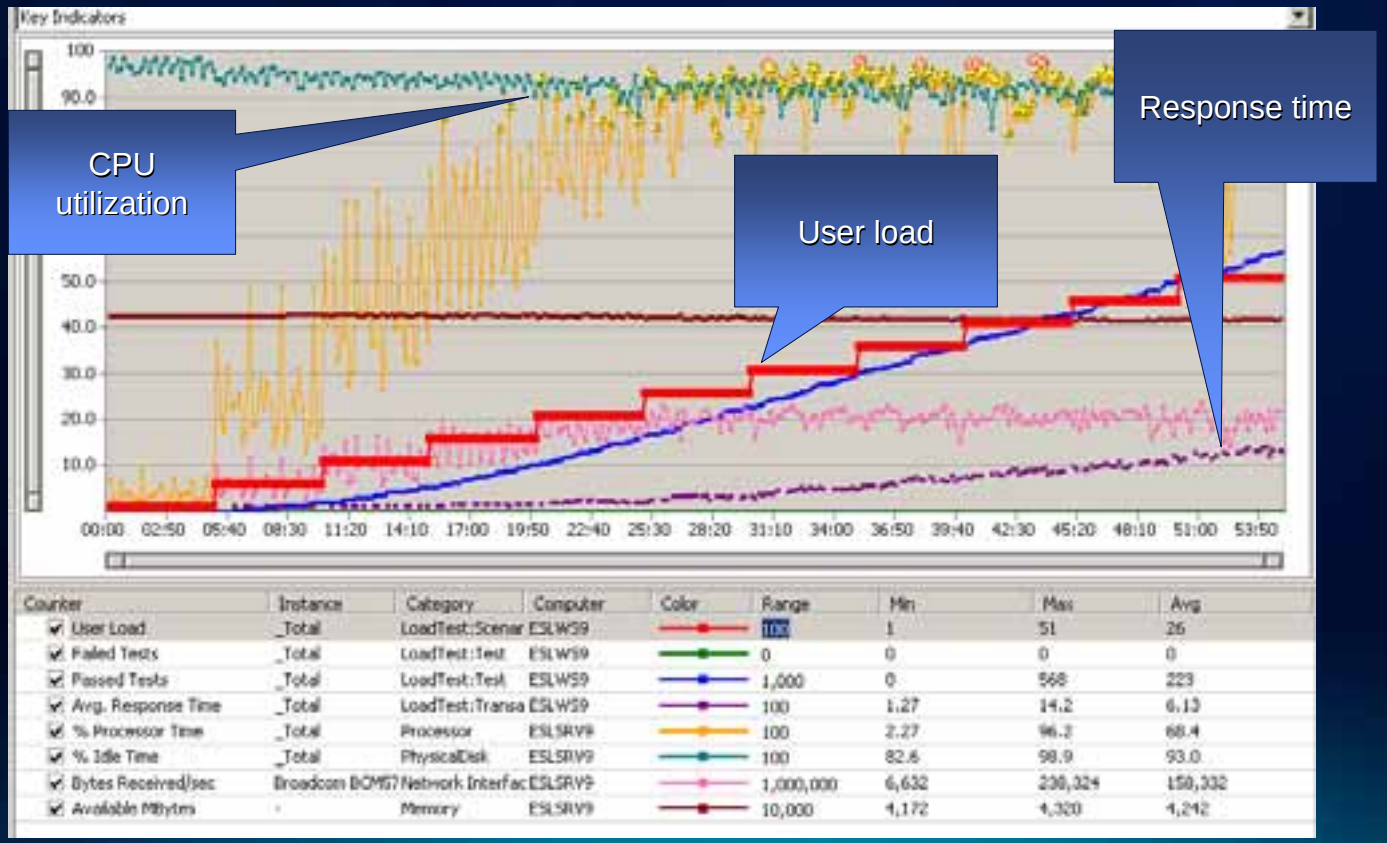
- **Unexpected curve**



Development Phase—Testing

Analysis—Compare and correlate key measurements

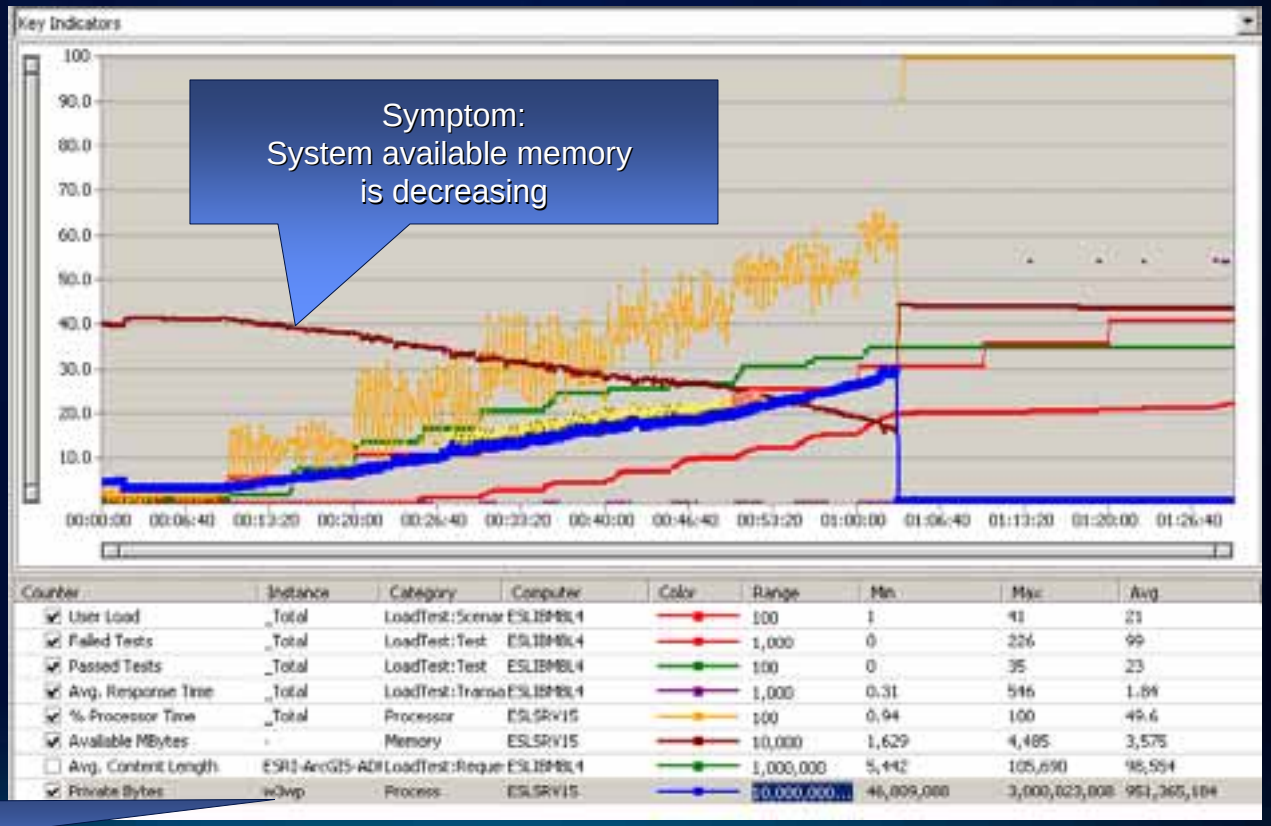
- **Expected counters correlation: increasing user load, CPU utilization, response time**



Development Phase—Testing

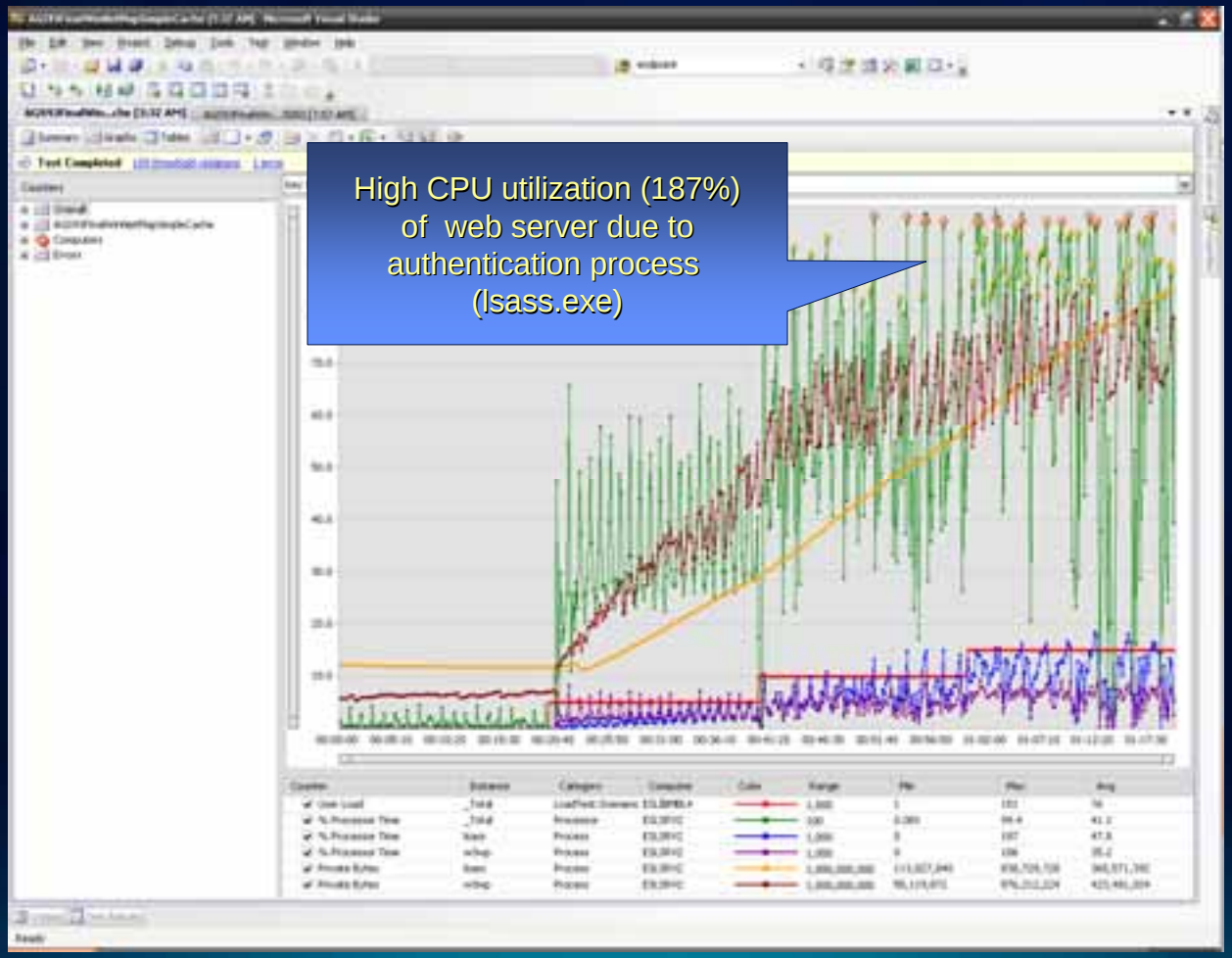
Analysis—Compare and correlate key measurements

- **Memory leak example**



Development Phase—Testing

Analysis—Unexpected CPU utilization for cached service



Development Phase—Testing

Analysis—Validate results

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

Development Phase—Testing

Analysis—Valid range

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

Development Phase—Testing

Analysis—Determining capacity

- Maximum number of concurrent users corresponding to, for example:
  - 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), for example:
  - 95% of requests under 3 sec
  - Max request under 10 sec

Development Phase—Testing

Report

- Executive summary
- Test plan
  - Workflows
  - Work load
- Deployment documentation
- Results and charts
  - Key indicators, e.g., response time, throughput
  - System metrics, e.g., CPU %
  - Errors
- Summary and conclusions
  - Provide management recommendations for improvements.
- Appendix

Demo

Fiddler Request Profiling

Fiddler to Visual Studio Web Test

Visual Studio Load Test

Performance Engineering —Deployment

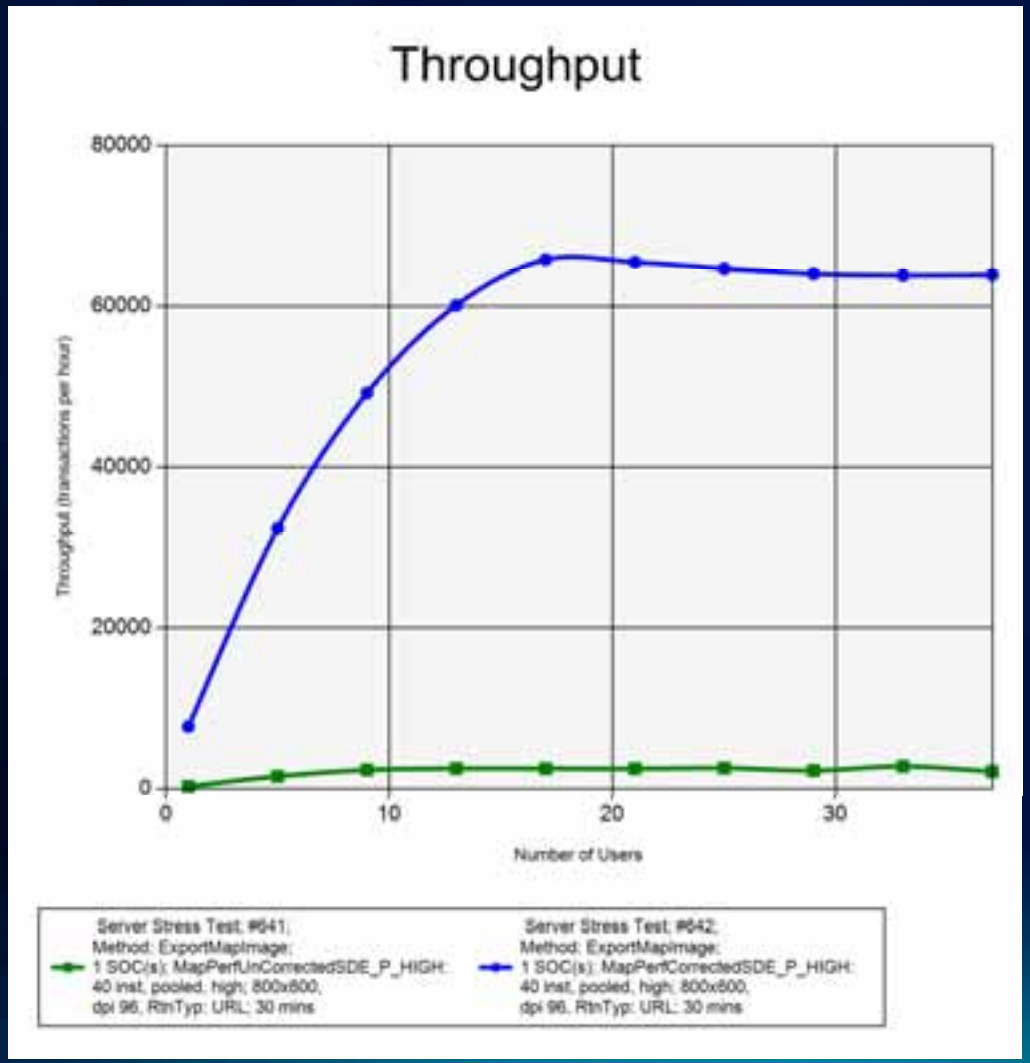
Deployment Phase

Performance engineering tasks

- Configuration management
- Performance tuning
- Performance and load testing

Deployment Phase

Configuration—ArcSOC max instances



Optimal
(Unloaded TT: 0.34 sec)
(2.1 Instances/core)

Nonoptimal
(Unloaded TT: 11.97 sec)
(1.6 Instances/core)

Deployment Phase

Performance tuning

Deployment Phase—Tuning

Performance tuning

- Benefits
  - Improved performance—User experience
  - Optimal resource utilization—Scalability

- Tools
  - Fiddler
  - Mxdperfstat, resources.arcgis.com/gallery/file/enterprise-gis/details?entryID=6391E988-1422-2418-88DE-3E052E78213C
  - Map Service Publishing Toolbar
  - DBMS trace

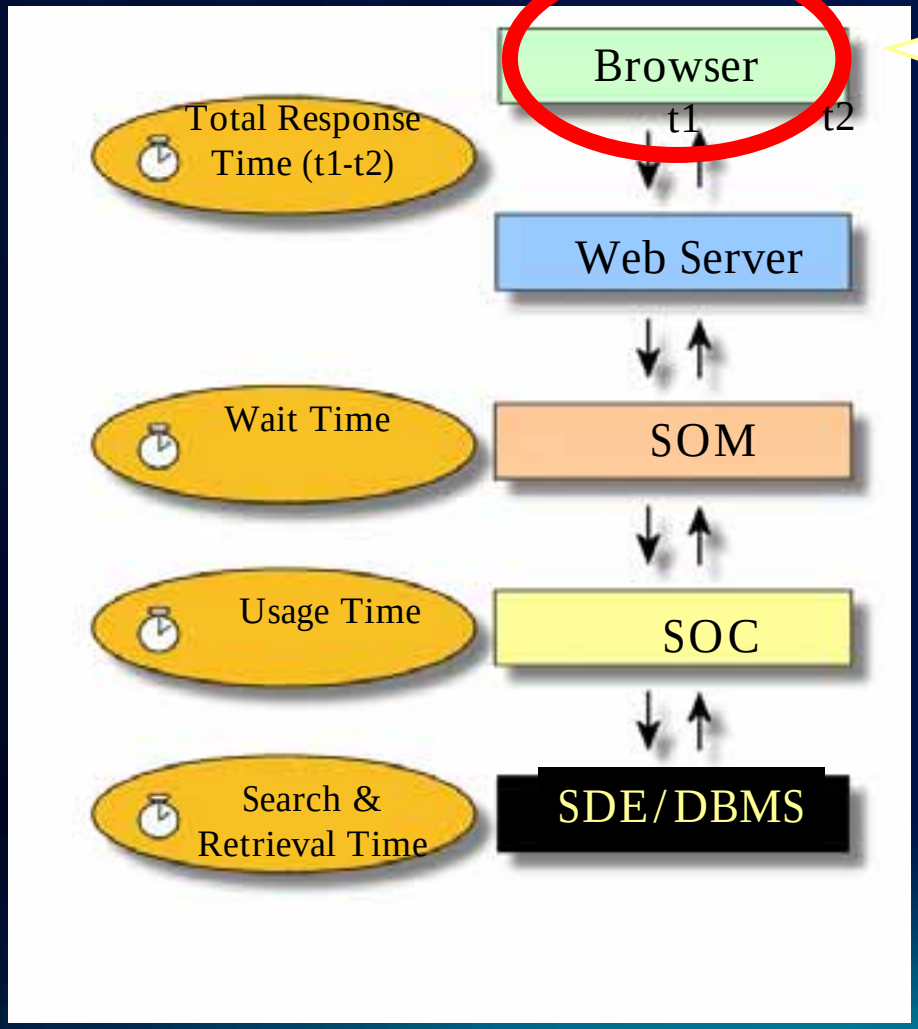
Deployment Phase—Tuning

Process

- Optimize ArcGIS Services.
- Profile individual user operations and tune if needed.
- Drill down through software stack:
  - Application
  - Service
  - MXD
  - Layer
  - DBMS query
- Correlate your findings between tiers.
- Performance and load test.

Deployment Phase—Tuning

Profile user transaction response time

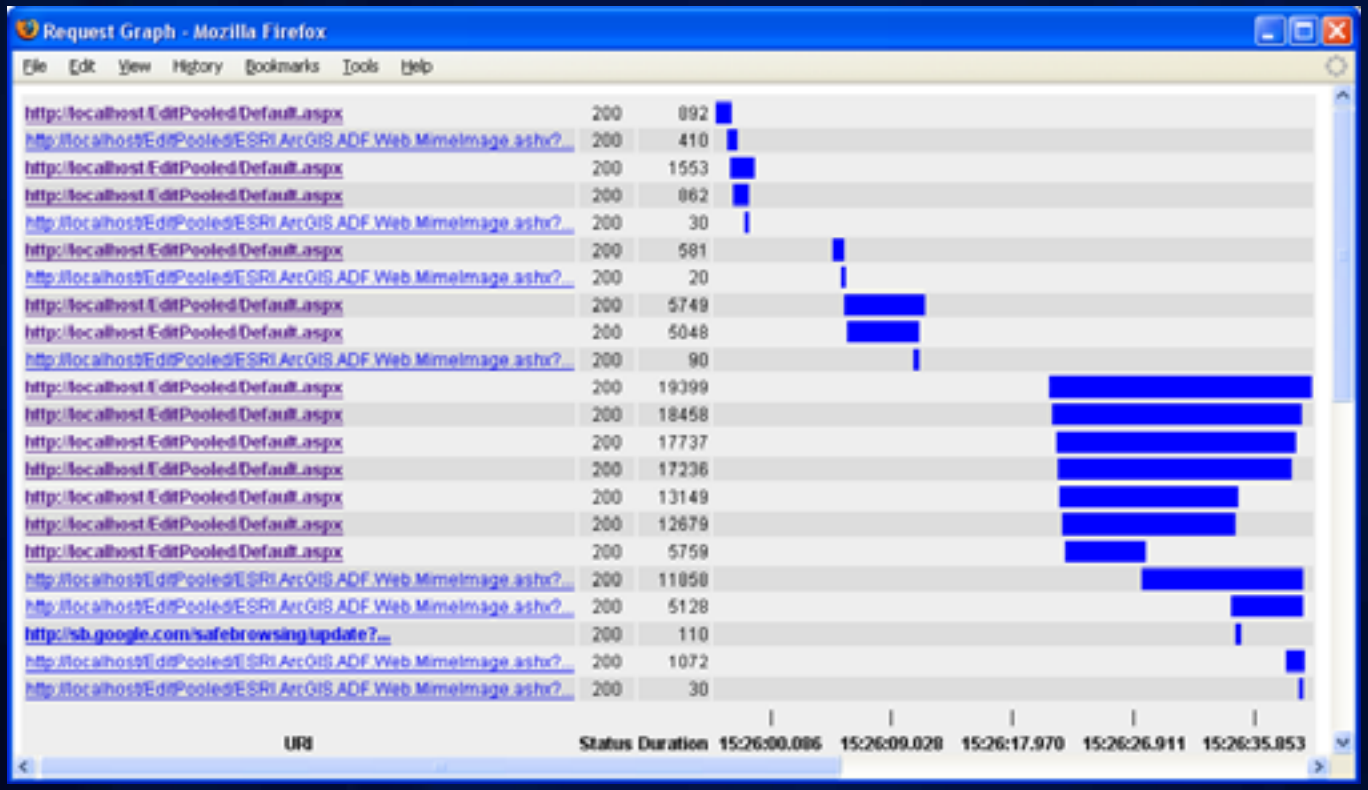


A test is executed at the web browser.

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

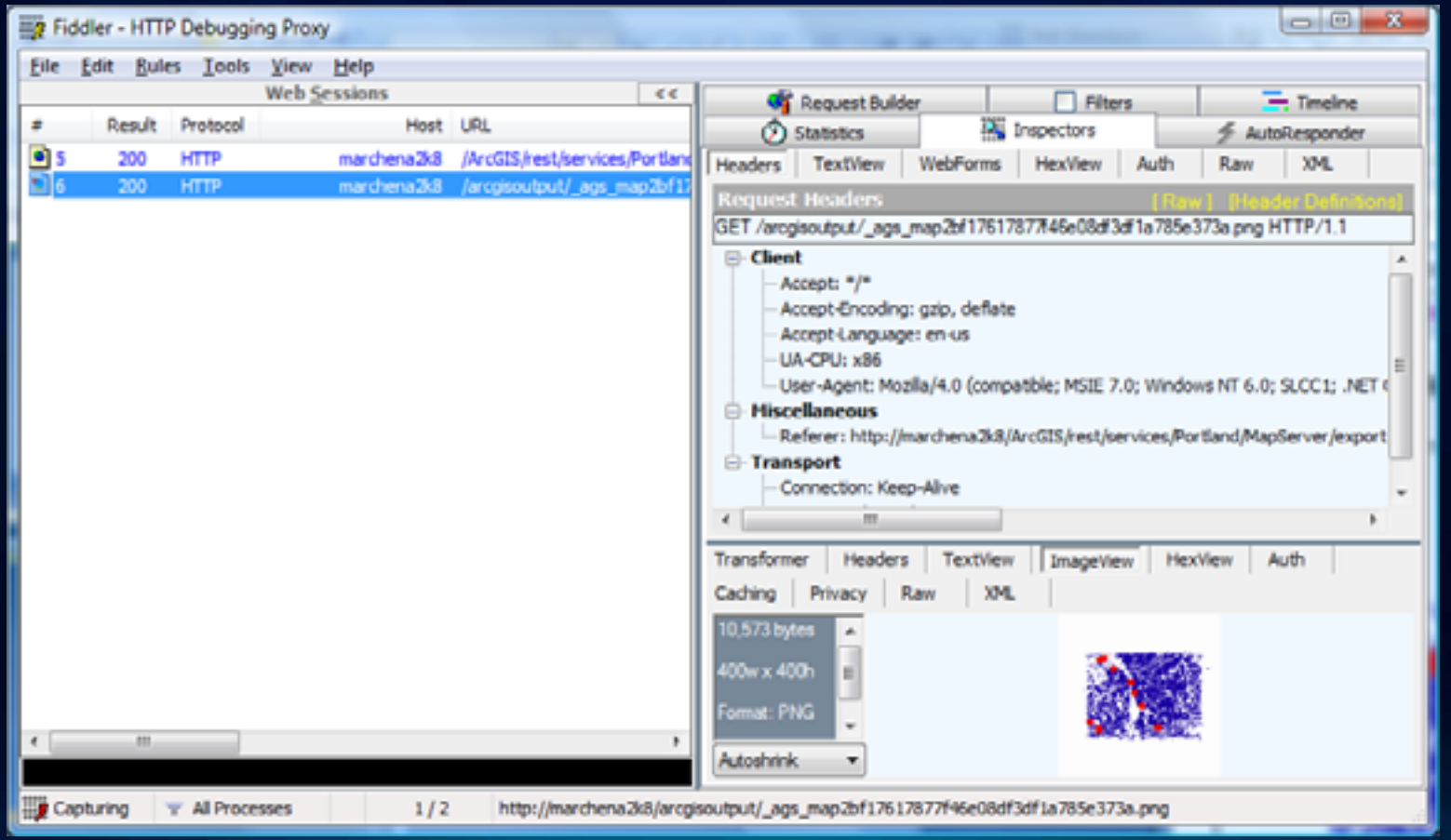
Deployment Phase—Tuning

Web diagnostic tools: Fiddler, Tamperdata, Yslow



Deployment Phase—Tuning

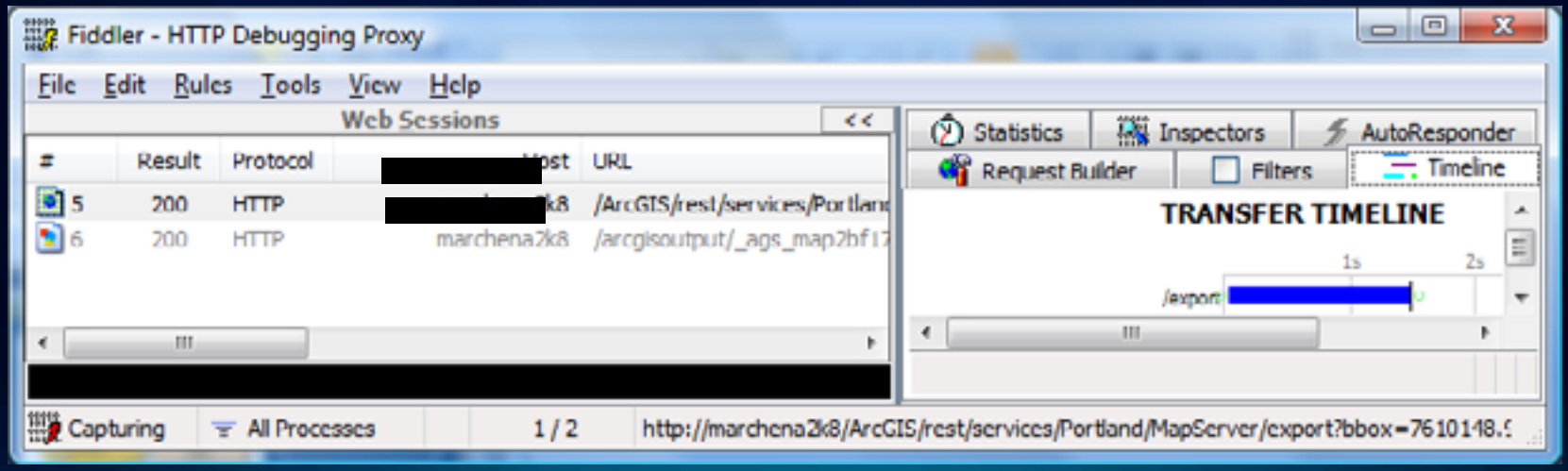
Web diagnostic tools: Fiddler validate image returned



Deployment Phase—Tuning

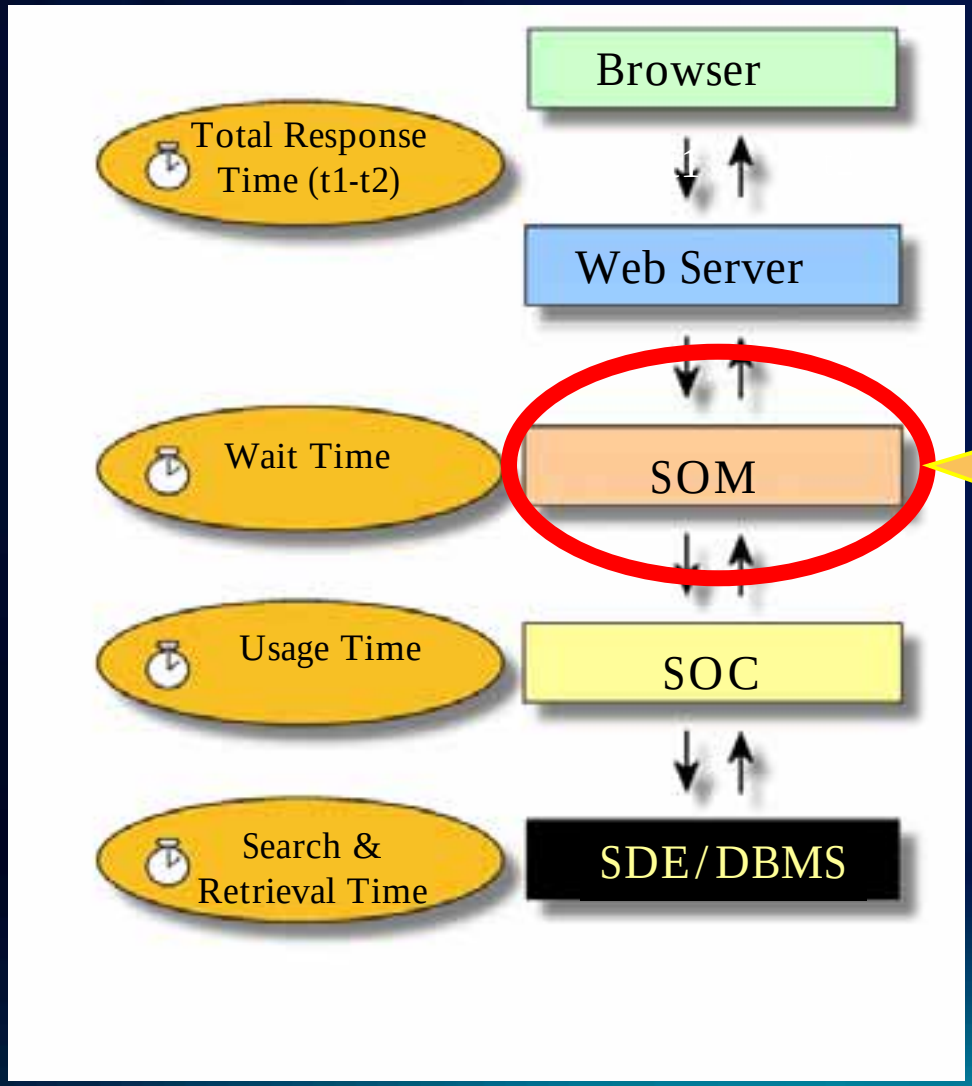
Web diagnostic tools: Fiddler

- 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.



Deployment Phase—Tuning

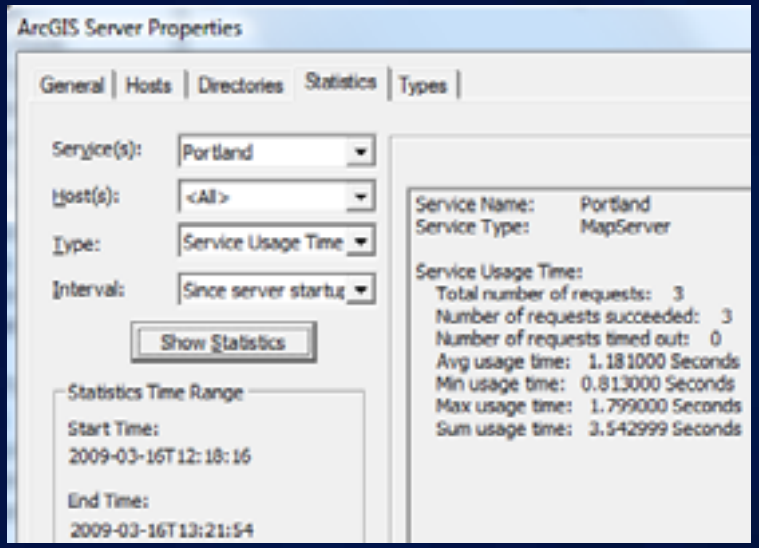
Analyze SOM/SOC statistics



Analyze ArcGIS context server statistics using ArcCatalog, Workflow Manager, or logs. They provide aggregate and detailed information to help reveal the cause of the performance problem.

Deployment Phase—Tuning

Analyze SOM/SOC statistics



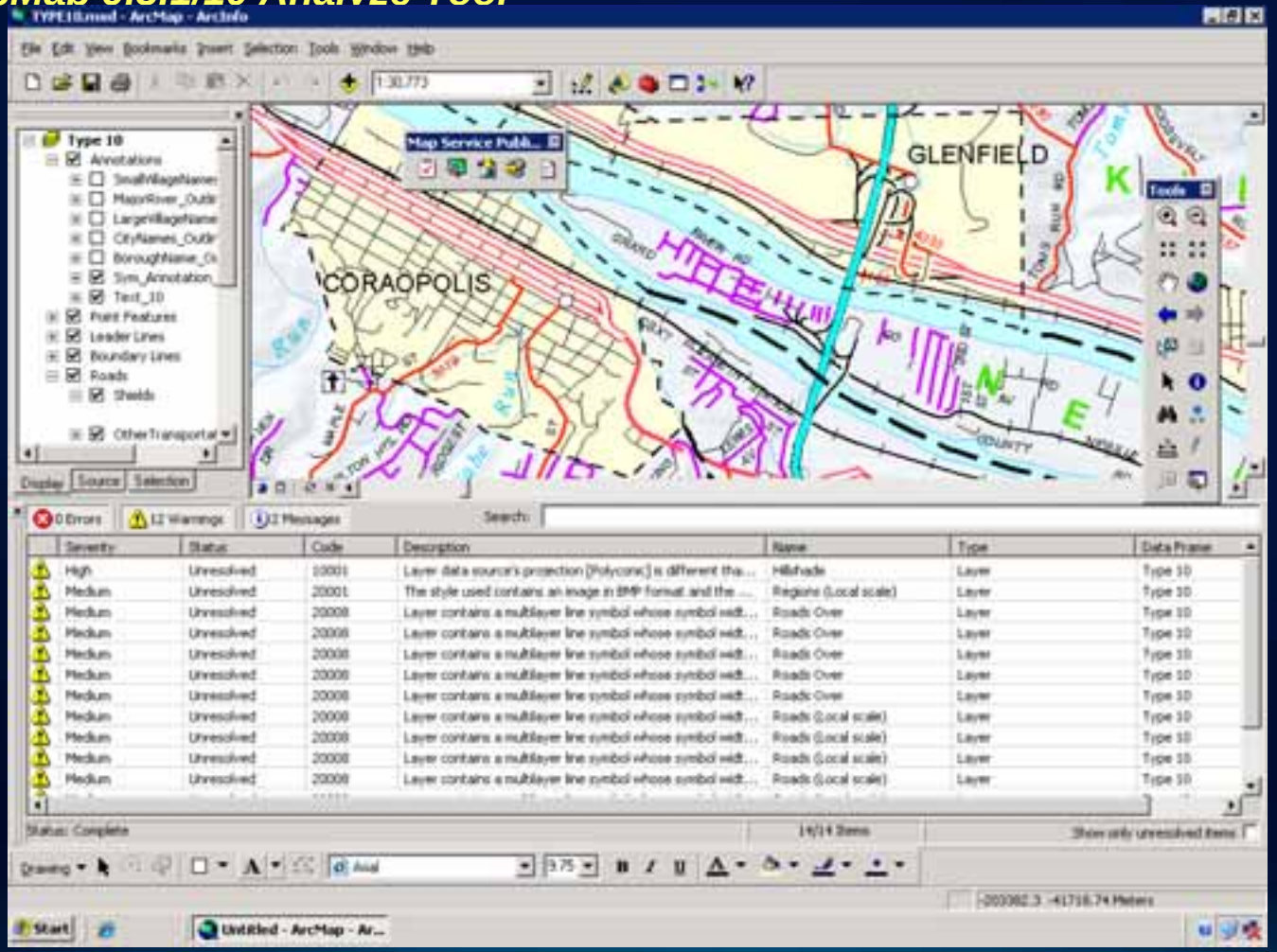
<Msg time="2009-03-16T12:23:22" type="INFO3" code="103021" target="Portland.MapServer" methodName="FeatureLayer.Draw" machine="myWebServer" process="2836" thread="3916" elapsed="0.05221">Executing query.</Msg>

<Msg time="2009-03-16T12:23:23" type="INFO3" code="103019" target="Portland.MapServer" methodName="SimpleRenderer.Draw" machine="myWebServer" process="2836" thread="3916">Feature count: 27590</Msg>

<Msg time="2009-03-16T12:23:23" type="INFO3" code="103001" target="Portland.MapServer" methodName="Map.Draw" machine="myWebServer" process="2836" thread="3916" elapsed="0.67125">End of layer draw: STREETS</Msg>

Deployment Phase—Tuning

ArcMap 9.3.1/10 Analyze Tool



Deployment Phase—Tuning

mxdperfstat

http://resources.arcgis.com/gallery/file/enterprise-gis/details?entryID=6391E988-1422-2418-88DE-3E052E78213C

C:>mxdperfstat -mxd Portland_Dev09_Bad.mxd -xy 7655029;652614 -scale 8000

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
18	8,000	Tax Lots	1.95	Simplify labeling, symbology: GraphicsPhase=.83;	2,226	33,872	True	.14	.83	.20	.08	6,396
19	8,000	Tax Lots Query Def	.13		1	26	False	.03	.02	.06	.03	3,204
20	8,000	TaxlotDenseLabel	1.84	Simplify labeling, symbology: GraphicsPhase=1.03; 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	.73	1.03	.95	.01	266
21	8,000	TaxlotDenseNoLabel	.53	simplify geometry: vertices fetched=200001;	1	200,001	False	.47	.02	.97	.00	140

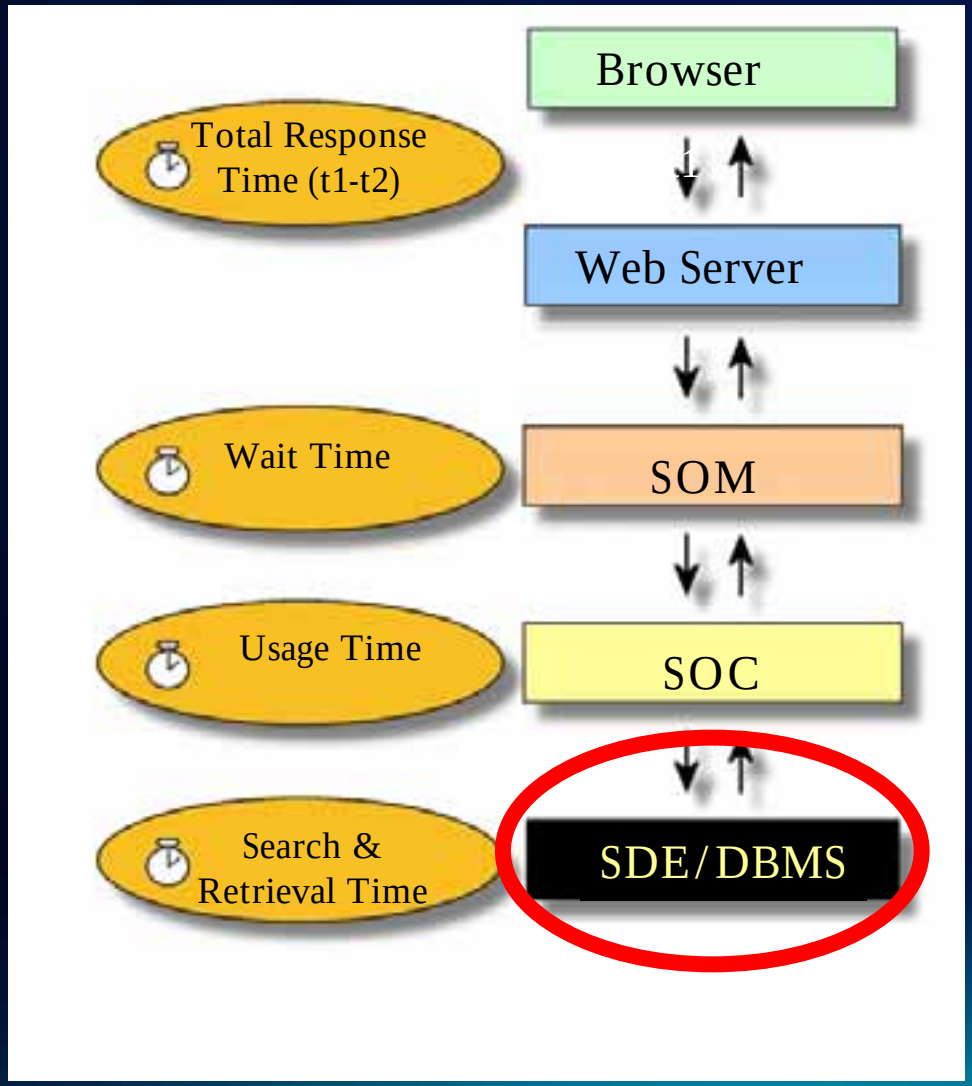
- **Issues discovered**
  - **Large numbers of vertices on features**
  - **Labeling of dense features expensive**

Demo

GIS Test Tool

Deployment Phase—Tuning

Data sources



Deployment Phase—Tuning

Data Sources—Oracle Trace

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

USERNAME     SID    SERIAL# PROGRAM     LOGON_TIM

------------------------------ --------- --------- ------------------------------
  -----STUDENT   132      31835 gsrvr.exe    23-OCT-06

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.

Deployment Phase—Tuning

Starting Oracle trace using a custom ArcMap UIControl

```
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
```

Deployment Phase—Tuning

Data Sources—Oracle Trace (continued)

```
SQL ID : 71py6481sj3xu
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
   call     count       cpu    elapsed       disk      query    current        rows
   ------- ------  -------- ---------- ---------- ---------- ----------  ----------
   Parse        0      0.00       0.00          0          0          0           0
   Execute      1      0.07       0.59        115       1734          0           0
   Fetch      242      0.78      12.42       2291      26820          0       24175
   ------- ------  -------- ---------- ---------- ---------- ----------  ----------
   total      243      0.85      13.02       2406      28554          0       24175
                  Elapsed times include waiting on following events:
      Event waited on                             Times   Max. Wait  Total Waited
      ----------------------------------------   Waited  ----------  ------------
      SQL*Net message to client                      242        0.00          0.00
      db file sequential read                       2291        0.39         11.69
      SQL*Net more data to client                    355        0.00          0.02
      SQL*Net message from client                    242        0.03          0.54
   ********************************************************************************
```

Deployment Phase—Tuning

Data Sources—Oracle Trace (continued)

- Definitions
  - Elapsed time [sec] = (CPU + wait event)
  - CPU [sec]
  - Query (Oracle blocks, e.g., 8 K read from memory)
  - Disk (Oracle blocks read from disk)
  - Wait event [sec], e.g., db file sequential read
  - Rows fetched

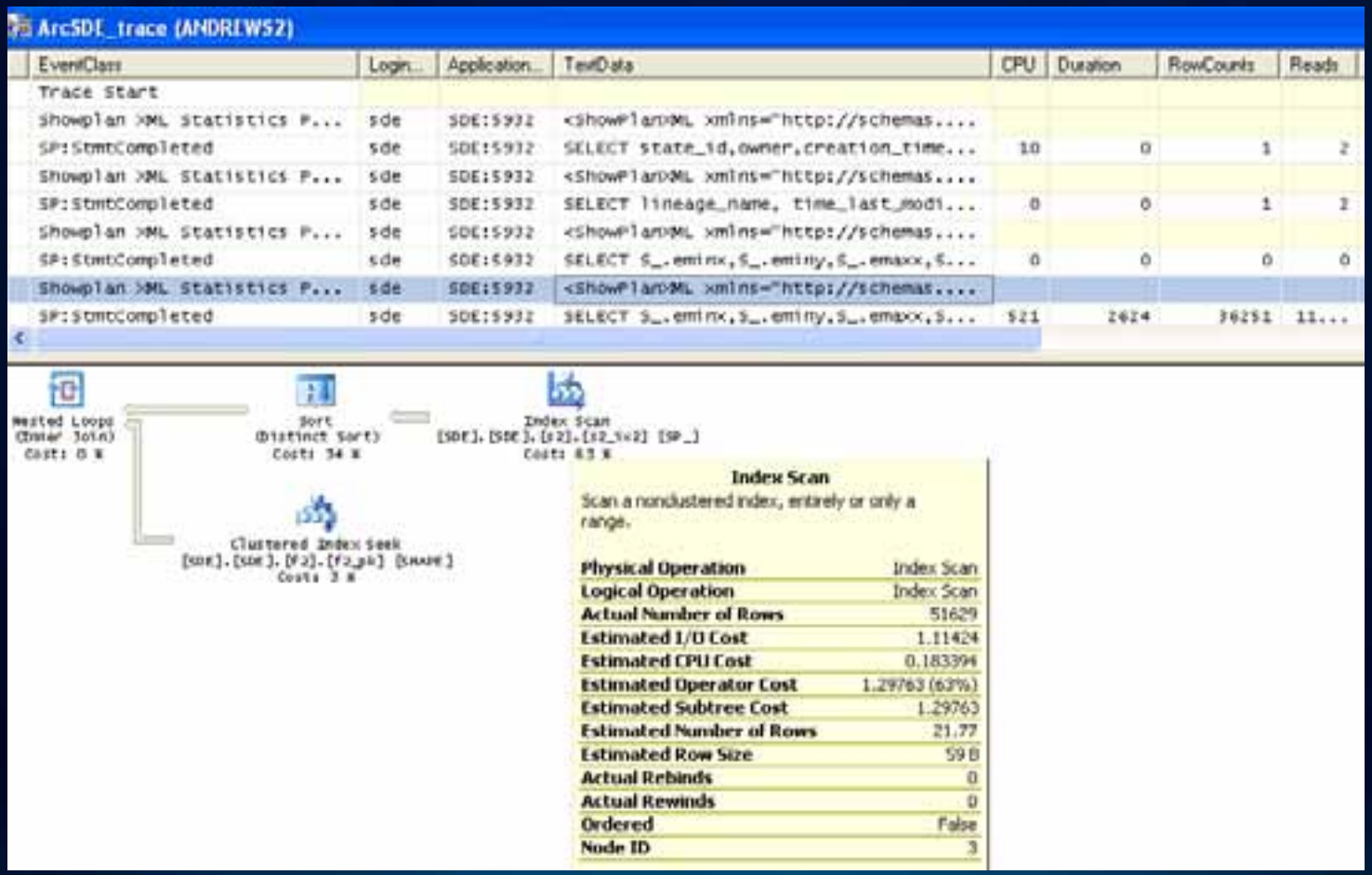
Deployment Phase—Tuning

Data Sources—Oracle Trace (continued)

- 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

Deployment Phase—Tuning

Data Sources—SQL Profiler



Deployment Phase—Tuning

Summary

- Optimize ArcGIS Services.
- Profile individual user operations and tune if needed.
- Drill down through software stack:
  - Application
  - Service
  - MXD
  - Layer
  - DBMS query
- Correlate your findings between tiers.
- Performance and load test.

Operation and Maintenance

Operation and Maintenance—Monitoring

- Baselines
- Trends
  - Used for growth estimation and variance
- Capacity models calibration
- Thresholds alerts

Test Results as Input into Capacity Planning

View Test Result

Calculate Service Time

Project Service Time to Production Hardware

Calculate Capacity

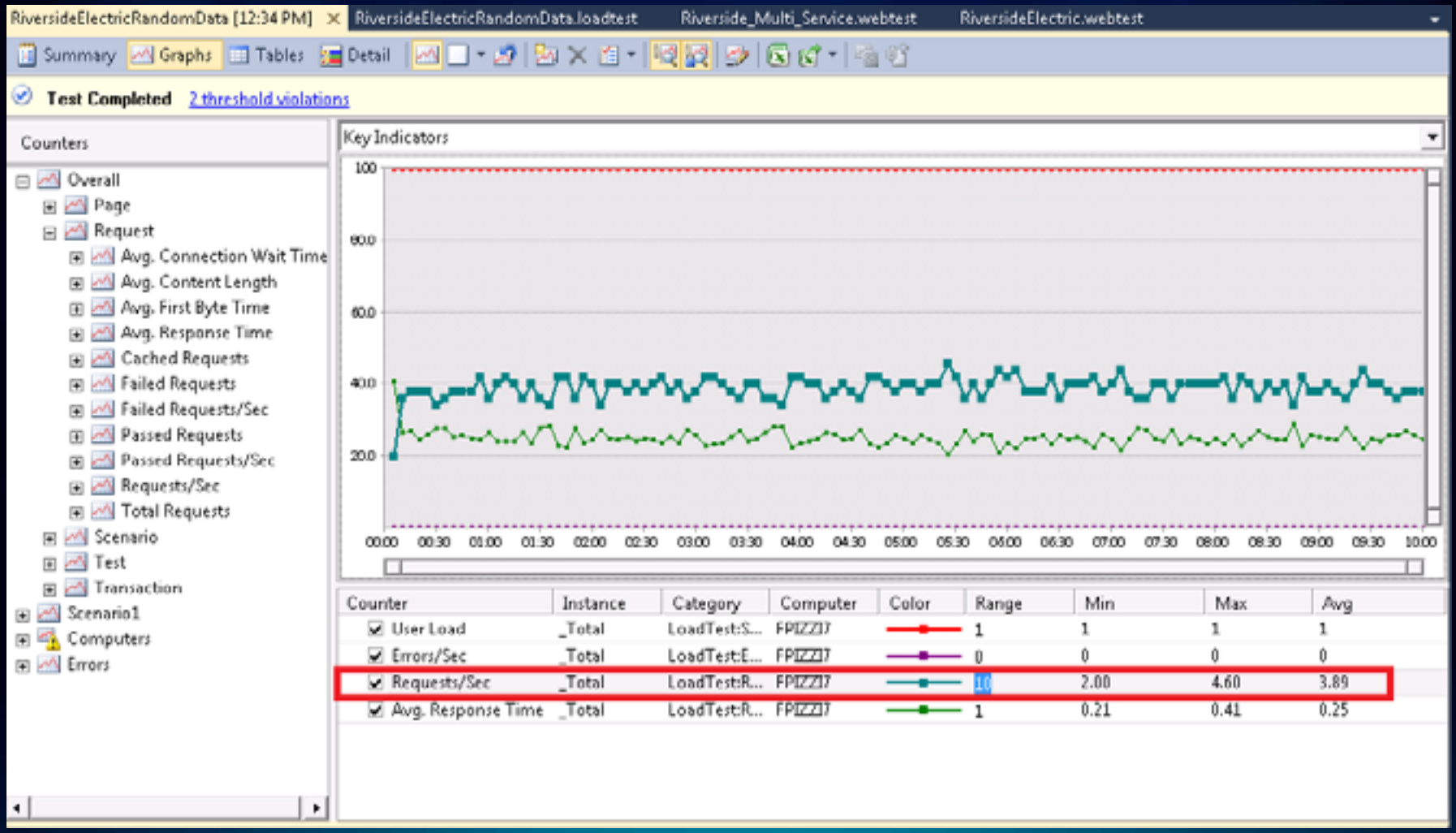
Test Results as Input into Capacity Planning

Load Test Results - Riverside Electric

- Baseline Test with Single Thread
  - Note* Service Time is Load Independent
- Think Time=0
- Evaluate Key Metrics
  - Throughput
  - Response Time
  - QA Check
- Evaluate System Under Test
  - CPU, Network, Memory, and Disk

Test Results as Input into Capacity Planning

Load Test Results - Key Indicators



Test Results as Input into Capacity Planning

Load Test Results - System Metrics

Test Results as Input into Capacity Planning

Load Test Results – input into capacity models

- Average throughput over the test duration
  - 3.89 request/sec ~ 14,004 request/hour
- Average response time over the test duration
  - .25 seconds
- Average CPU Utilization
  - 20.8%
  - Mb/request = 1.25 Mb

Test Results as Input into Capacity Planning

Load Test Results – input into CPU capacity model

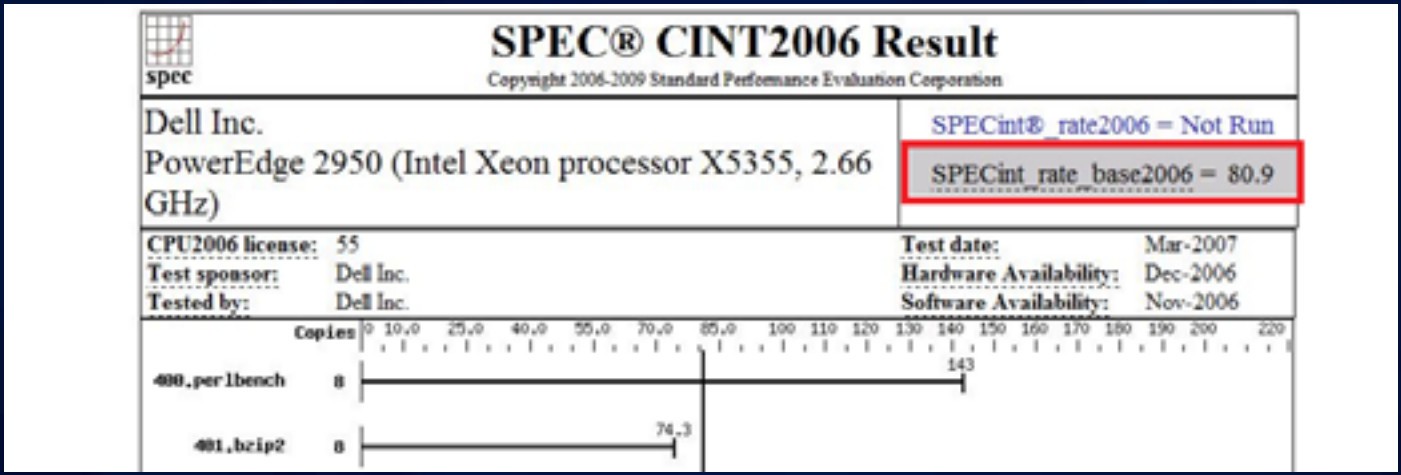
- **Input from testing**
  - **#CPUs = 4 cores**
  - **%CPU = 20.8**
  - **TH = 14,004 requests/hour**
  - **SPEC per Core of machine tested = 35**
- **ST = (4*3600*20.8)/(14,004 *100) = 0.2138 sec**
  - **Note* very close to Average response time of .25**

$$ST = \frac{\#CPU \times 3600 \times \%CPU}{TH \times 100}$$

Test Results as Input into Capacity Planning

Target values

1. **Server SpecRate/core=10.1**



SPEC® CINT2006 Result

Copyright 2006-2009 Standard Performance Evaluation Corporation

Dell Inc.
PowerEdge 2950 (Intel Xeon processor X5355, 2.66 GHz)

SPECint®_rate2006 = Not Run

SPECint_rate_base2006 = 80.9

CPU2006 license:	55	Test date:	Mar-2007
Test sponsor:	Dell Inc.	Hardware Availability:	Dec-2006
Tested by:	Dell Inc.	Software Availability:	Nov-2006

2. **User load=30,000 req/hr**
3. **Network=45 Mbps**

Test Results as Input into Capacity Planning

Target CPU cores calculation

- **Input to Capacity Planning:**
  - **ST = Service Time = .2138 sec**
  - **TH = Throughput desired = 30,000 request/hour**
  - **%CPU = Max CPU Utilization = 80%**
  - **SpecRatePerCpuBase = 35**
  - **SpecRatePerCpuTarget = 10.1**
- **Output**
  - **#CPU required =([.2138*30,000*100]/3600*80]) *[35/10.1]**
  - **#CPU required = 7.7 cores ~ 8 cores**

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

Test Results as Input into Capacity Planning

Target network calculation

- **Input to Capacity Planning:**
  - **Mb/req=1.25**
  - **TH = 30,000 request/hour**

$$Mbps = \frac{TH \times Mbits/req}{3600}$$

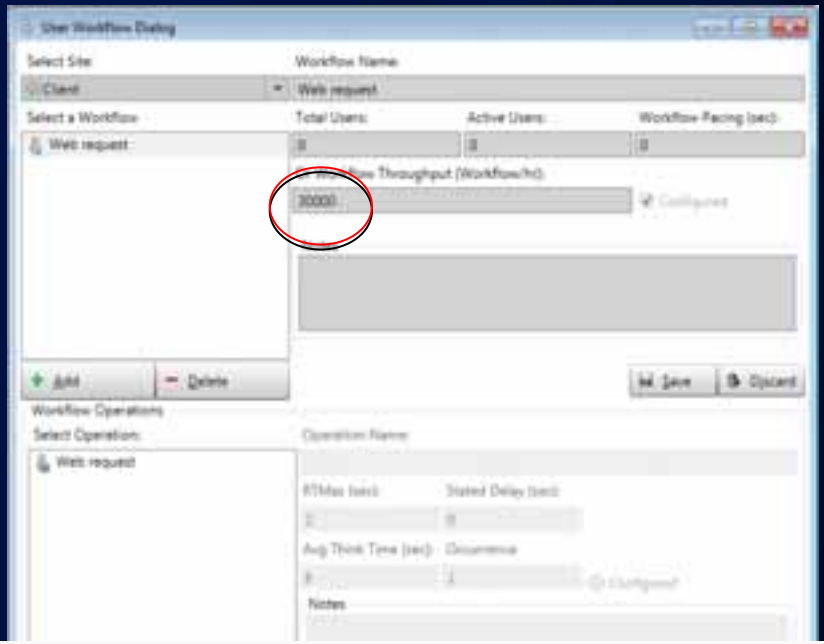
- **Output**
  - **Network bandwidth required = 30000x1.25/3600**
  - **=10.4 Mbps < 45 Mbps available**
  - **Transport=1.25/(45-10.4)=0.036sec**

$$Transport\,(\text{sec}) = \frac{Mbits/req}{Mbps - Mbps_{used}}$$

Test Results as Input into Capacity Planning

System Designer

- **Input:**
  - **Throughput=30000**
  - **ST=0.21**
  - **Mb/tr=1.25**
  - **Hardware=80.9 Spec**



Model Review | Model Assigned

Selected Model

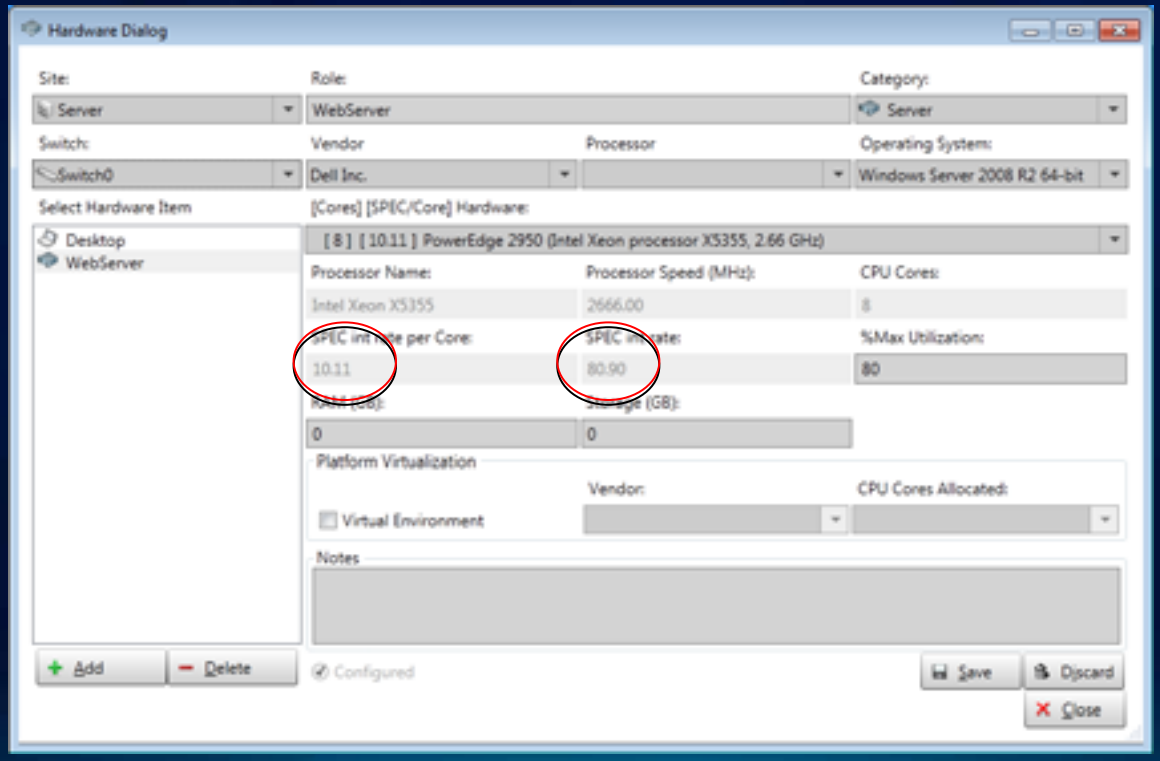
Service Type: Map **Model Name:** ExportMap REST MapService 9.3.1

Model	Function	Tier	Modified	Service Time(sec)	Queue Time(sec)	CPU Cores Calc	Modified	Mb/Tr
✓	Client	Client	○		0.000	0.00	○	1.250
✓	WebService	Web Services	✓	0.210	0.004	1.01	✓	1.250

Test Results as Input into Capacity Planning

System Designer

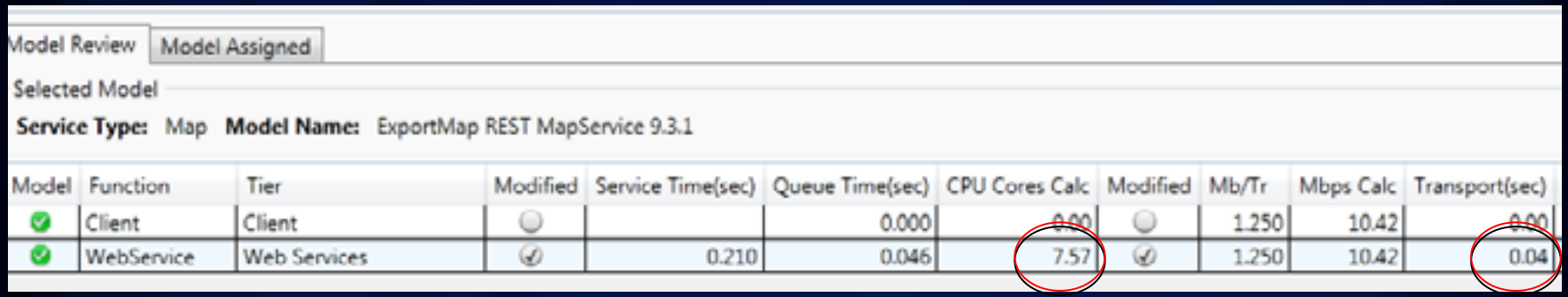
- **Input**
  - **Hardware=80.9 Spec**



Test Results as Input into Capacity Planning

System Designer

- **Review results**


Model Review | Model Assigned

Selected Model

Service Type: Map **Model Name:** ExportMap REST MapService 9.3.1

Model	Function	Tier	Modified	Service Time(sec)	Queue Time(sec)	CPU Cores Calc	Modified	Mb/Tr	Mbps Calc	Transport(sec)
✓	Client	Client	○		0.000	0.00	○	1.250	10.42	0.00
✓	WebService	Web Services	✓	0.210	0.046	7.57	✓	1.250	10.42	0.04

Demo System Designer

Calculate Capacity

System Designer evaluation and training

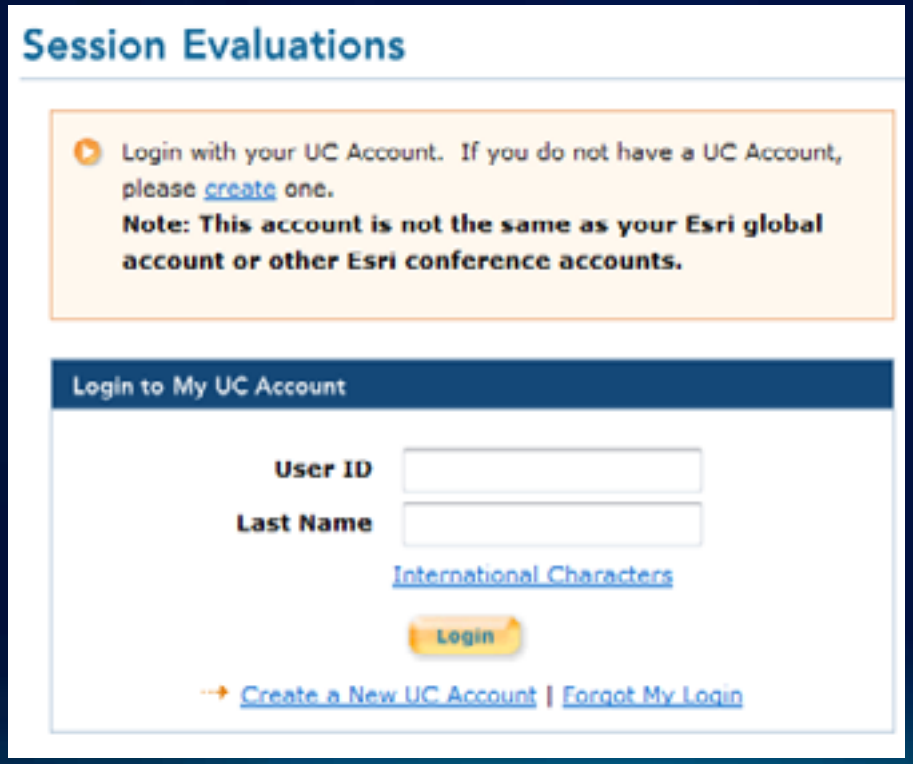
- Contact us
  - Chad Helm, chelm@esri.com
  - Andrew Sakowicz, asakowicz@esri.com
-
- Download free evaluation
  - ftp://ftp.esri.com/
  - Click the File menu and choose Login As
  - username: eist
  - password: eXwJkh9N

Related sessions

- **Enterprise GIS Architecture Deployment Options**
  - **Thu 08:30 AM**

Session Evaluation

- http://events.esri.com/uc/2011/sessionEvals/index.cfm?fa=app_login_form



Questions?

Contact us

Contact us

- **Andrew Sakowicz**
  - asakowicz@esri.com
- **Frank Pizzi**
  - fpizzi@esri.com