



ArcGIS for Server Performance and Scalability: Optimizing GIS Services

Craig Williams, Peter Becker,
Andrew Sakowicz, Josh Jones

Technical Workshop

Introduction

Josh Jones

3 Presenters



Craig Williams

Product Engineer, Map Service

Optimizing Map Services



Peter Becker

Product Engineer, Image Service

Optimizing Image Services



Andrew Sakowicz

Enterprise Implementation Services

Performance Factors

Optimizing map services

Craig Williams

Overview

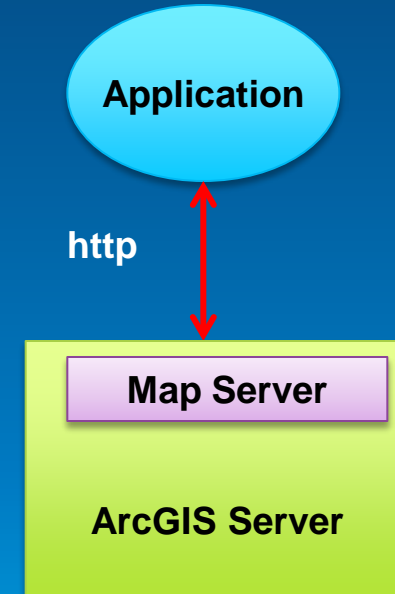
- **Types of map services**
 - **What's new in the last few releases**
- **Factors of map service performance**
 - **Data access**
 - **Rendering speed**
 - **Image size/compression**

Map services

- **9.3.1 – 10.0**
 - **MXD based map services**
 - **MSD based map services (optimized map service)**
- **Use the optimized map service for best quality and performance**
 - **Analyzer workflow guides you through potential problems**

Map services at 10.1 and beyond

- **One unified map service**
 - An updated optimized map service
 - Supports additional capabilities, data types, layers, renderers
- **Includes extension capabilities optimized map service lacked:**
 - Network Analysis
 - Geoprocessing*



Mapping capabilities

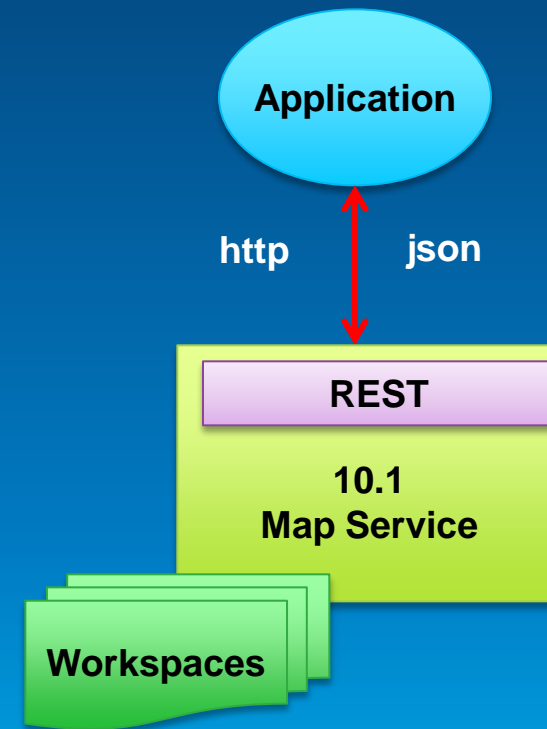
- **Added data source support**
 - XY events
 - Linear referencing events
- **Added feature layer renderer support**
 - Dot density
 - Charts
 - Geostats
- **Added support for layers**
 - Dimensions
 - Schematics
 - Network
 - Network Analysis
 - Tin
 - Terrain
 - Tracking

Dynamic Layers

- **New behavior with the map service that allows for per-request changes to the map**
 - **Optional capability of map services**
- **May allow you to reduce the total number of services you need**
- **Allows for:**
 - **Updating renderers and symbols**
 - **Removing and reordering layers**
 - **Changing layer data sources**
 - **Adding new layers from registered data sources**

Dynamic Layers: How they work

- **Simple updates to the map service**
 - Remove layers or reorder layers
- **Thematic mapping**
 - Updates to renderers
- **Adding content from your datasources**
 - Find data from registered workspaces
 - Including query layers
 - Add to the map on a per-request basis



Factors of map service performance

- **Data access**
- **Rendering**
- **Image compression / size**

- **Consider all of these when creating a map service**



Data access

- **Local data will draw faster than remote data**
- **Spatial index**
 - Do you have one? (e.g. XY Events)
 - Is it sized correctly?
 - Universal features, slow all draws
- **Attribute indexing**
 - Not always needed



Data access case study : X Y Events

- **Often used in cases where data comes from external systems**
 - **As database table or CSV**
- **A draw typically requires a complete row scan**
- **Alternative**
 - **Use a native spatial type in your database**
 - **-Query layers**
 - **Insert features via SQL in external systems**
 - **Features will be indexed and draw much faster**



Data access troubleshooting

- **Publishing analyzers indicate lack of a spatial index etc.**
- **Evaluate index efficiency**
 - The number of features returned for each draw query
 - Large index grid sizes lead to too many features being drawn
- **Evaluate I/O performance if using remote data**
- **Unnecessary attribute indexes**
 - May confuse query plans in some databases
 - Don't index fields just because they exist in a def query

Rendering speed

- **Optimized map services were introduced at 9.3.1 to resolve performance bottlenecks at this stage**
- **Remaining areas to be concerned with:**
- **Complex effects (e.g. geometric effects in representations)**
- **Inline annotation (aka “Bloated” annotation)**
- **Anti-aliasing performance**
 - Higher levels use more RAM and are slower
 - Text anti-aliasing has a negligible effect in most cases
- **Layer transparency**

Rendering speed: transparency

- Layer transparency is applied to a layer as a whole
 - Involves a full layer blend
- Alternative: Use color transparency
 - Capability of optimized map services
 - Enabled via an option from analyzer warning 10009

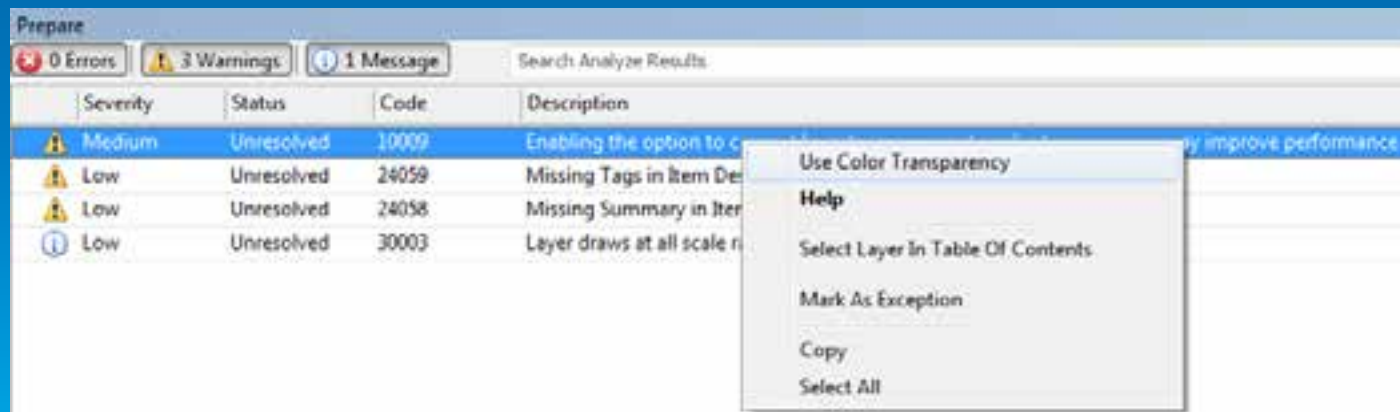


Image compression / size

- **Smaller images are faster to download**
- **Image formats have limitations**
 - e.g. limited color palettes, lossy compression
- **Image compression itself has a performance penalty**
 - Use the preview window to evaluate performance of image type
- **Evaluate size and performance in a test service in network conditions**
 - Balance size vs. quality when choosing the image type based on your needs

Image compression / size (con't)

- **Image type used for cached services affects:**
 - **Download size**
 - **Storage size of the cache**
 - **Portability**
- **For caching vector data**
 - **Consider new PNG image type (introduced at 10.1)**
 - **Chooses the correct PNG type (8, 24, 32) for each tile based on content**
 - **Low content areas use less storage**

Optimizing Image Services

Peter Becker

Ways of Making Imagery Accessible:

- **Download**
 - Clip , Zip, Ship
- **Tile Cache Service**
 - Background image
- **Image Service of Single Raster**
 - OnTheFly processing
- **Image Services of Mosaic Dataset**
 - Dynamic Mosaicking and OnTheFly processing of Collections of Rasters
- **Geoprocessing Task**
 - Access to all ArcGIS tools

Tile Cache Data Flow

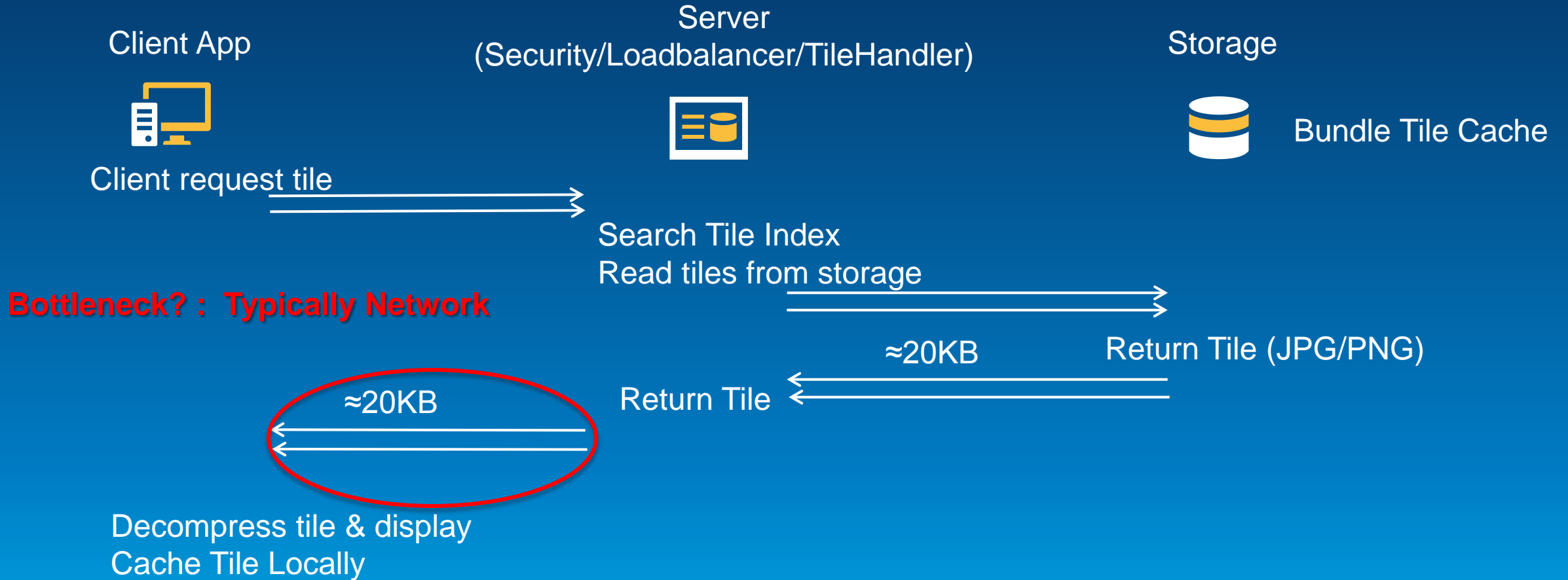


Image Service – Single Raster – Data Flow

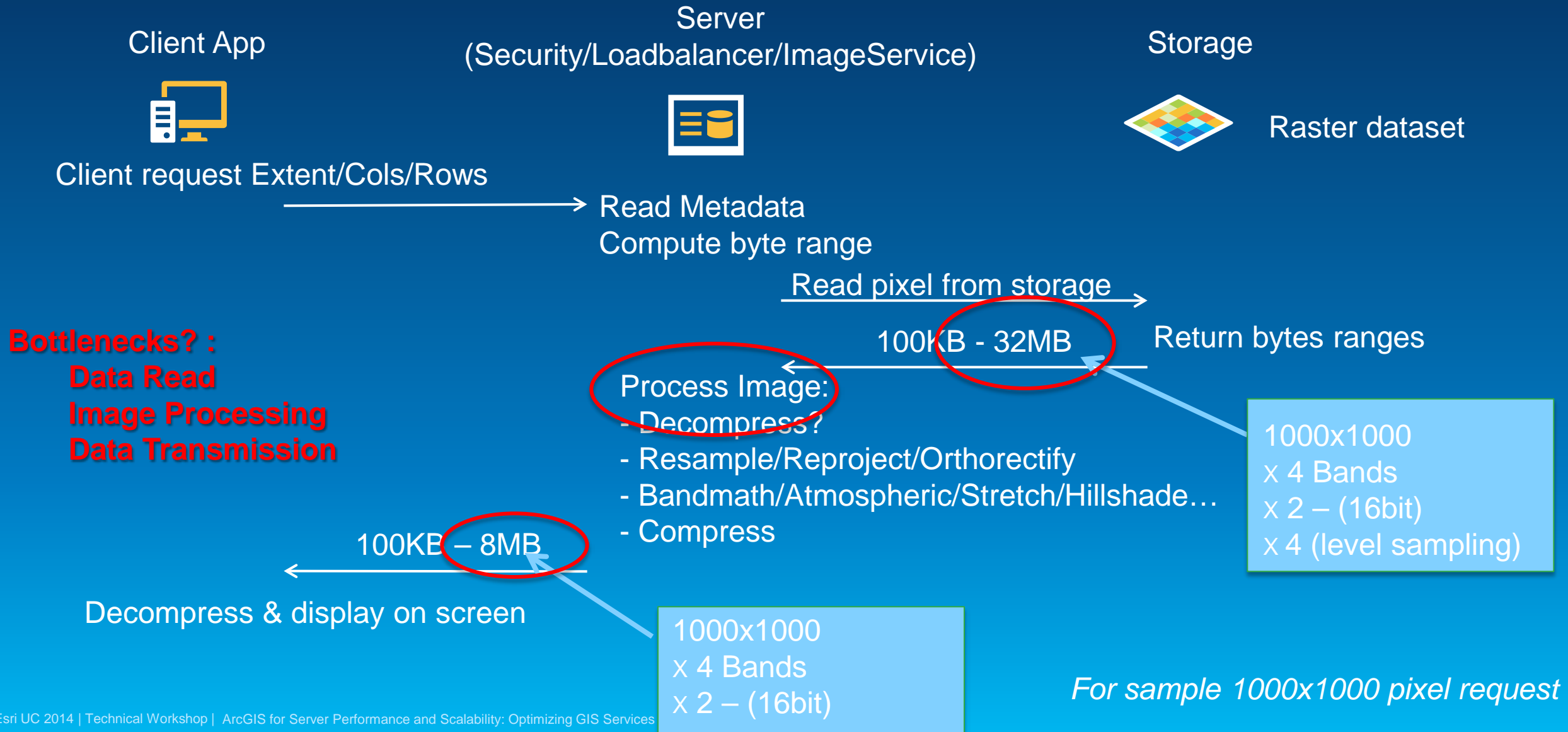


Image Service – Mosaic Dataset – Data Flow

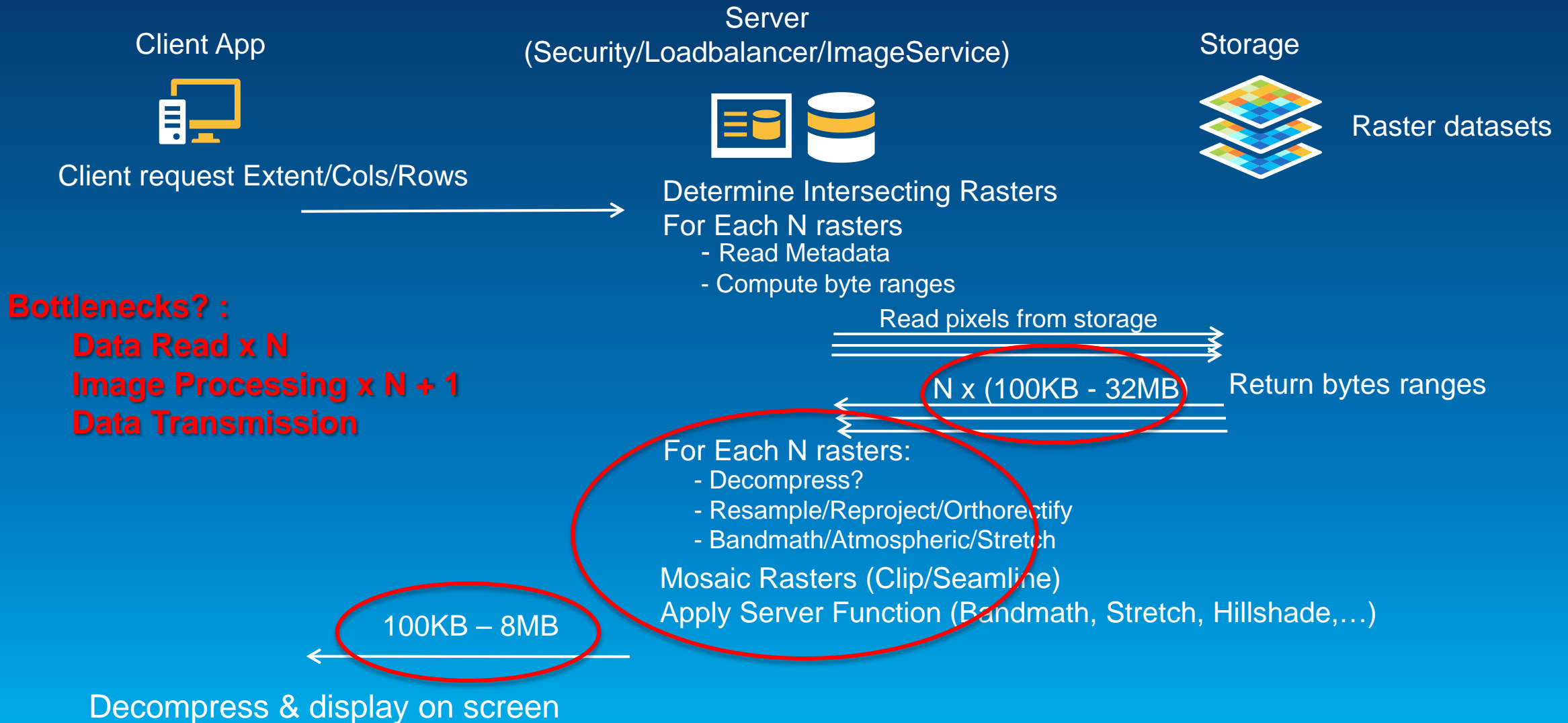


Image Service or Tile Cache ?

- **What is faster ?**
 - **Tile Cache** – If structured requests – Due to caching
 - **Image Service** – If random request and optimized – Due to single request
- **What is more scalable on a specific set of HW ?**
 - **Tile Cache is more Scalable**
 - Uses Preprocessed data and caching at Server, Client and Internet (edge cache)
- **What is more flexible ?**
 - **Image Services** - Provide access to full information content
- **On what can I do Image analysis & Interpretation**
 - **Image Services** – Provide wide range of server based image processing
- **What performance is typical for Image Services**
 - **1-8MP/s/Core**

Recommendations:

- **General**
 - If just want static background Use Tile Cache
 - For dynamic imagery use image services
 - Do not add imagery to a Map and Publish as Map. (Best practice is to keep image as separate service)
- **Will split recommendations into:**
 - Data – How to structure
 - Storage Infrastructure
 - Mosaic Dataset Design
 - Server
 - App Requests

Data Recommendations

Minimize the amount of data the needs to be read

You may or may not have control of the following:

- **Format – Tiled: Eg GeoTIF with internal tiles** (Max 1000%)
- **Compression** (Max 500%)
 - Helps reduce volume of data read from disk
 - Can add CPU Load to decompress. (JPEG good, Wavelet (JPEG2000) can be expensive)
- **Include Pyramids**
 - Reduced data read at smaller scales
- **Projection**
 - If possible create in same as majority output. (Suggest that do not pre-reproject) (Max 40%)
- **PixelSize - If using standard WebMap chose – 0.25,0.5,1,2,4,8,16,32.5,75,150,300,600** (Max 100%)

For More info see:

“Image Management Guidebook” <http://esriurl.com/6007>

“Designing and Optimizing Image Services for High-Performance Analysis Blog” <http://esriurl.com/OptimizingISBlog>

Storage Infrastructure

Ensure fast transfer to Servers

- **Disk Storage**

- Needs to be Fast
- Stripe Disks
- Tune NAS. Check turning off Read Ahead Cache?
- Storage performance can vary significantly

On Amazon: Use Striped Ephemeral else Striped EBS

- **Internal Network**

- Min 1GB between server and storage
- If necessary use dedicated network for imagery

- **For Smaller implementations**

- Consider using DAS (Direct Access Storage)
- Use DAS for file geodatabase (see later)

Mosaic Dataset Design – See Guidebook

Minimize Amount of Processing

See Image Management Guidebook - <http://esriurl.com/6007>

- **Create Overviews on Mosaic Dataset** – Reduce number of images accessed at smaller scales
- **Location of Mosaic Dataset** – MDs are chatty. Best to keep on DAS
- **Processing Functions** – Review. Especially regional functions
- **Check NoData** – Use Footprints to constrain extents. Turn off ‘Footprint contains no data’ if possible
- **Sampling method** – Nearest ,Bilinear -10% Cubic -18%
- **Footprints** - Balance complexity and approximation. Possibly shrink & generalize
- **Split Mosaic Dataset by data type** - Use Suitable # Bands / BitDepth
- **Size of Allowed Table Attributes** – Too many fields slow down table display
- **Index** – Add Indices to fields that will be queried
- **Projection of Mosaic Dataset** – Hardly any effect

Server

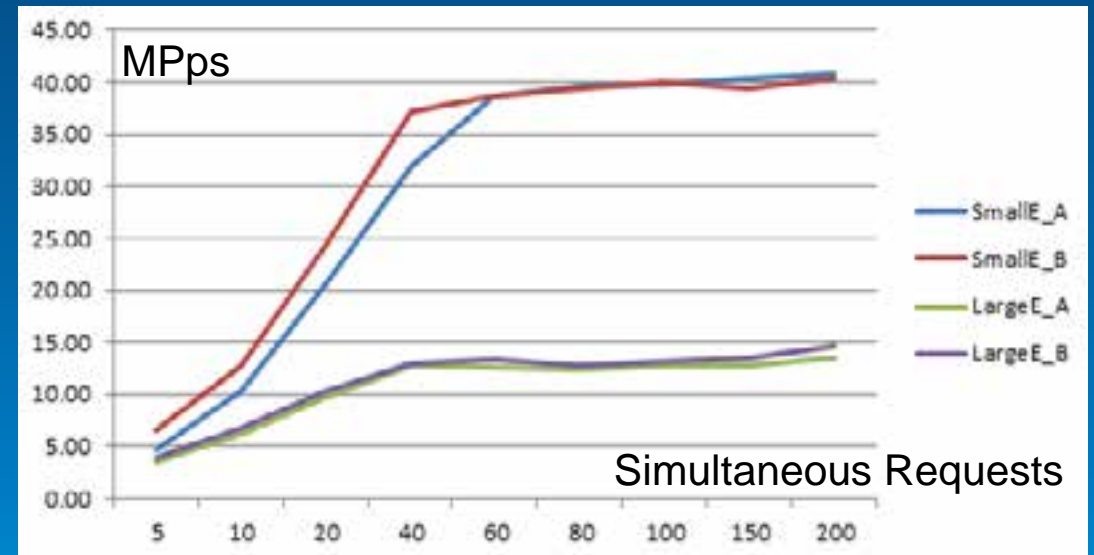
- • **CPU – Faster Better**
- • **Memory – 2GB/Core**
- • **Internet Access**
- • **Virus Checker – Can be a real hog!**

App

- **Reduce Client Bandwidth - Compression for transmission**
 - **Set to JPEG (Q75-80) for continuous**
 - **Set to PNG for discrete**
 - **Use suitable defaults**
 - **Use Layers to set appropriate compression**
 - **On Web Reduce PNG request (caused by NoData) when using JPG/PNG**
- **User Server Functions – Allow server to do processing**
- **Size Cols/Rows of request**

How to Test

- **Run Fiddler**
 - Review size of response and time of response
- **Set Number of Service Instances = Cores**
- **Put Server Under Load**
 - **Use JMeter / Visual Studio Load Test /....**
 - Simultaneous requests say 1000x1000 cols/rows
 - Possibly run on server to ignore network?
 - Covering Small Extent – (Data cached by OS)
 - Covering Large Extents – (Can not be Cached)
 - Measure Pixels/Second
- **Use Perfmon – Under Load**
 - CPU Total : Should reach 80-90%
 - Available memory > 10% and Constant
 - QueLength : <5 or IdleTime > 5%
 - Local Network Utilization : <90%
- **Disk Benchmark tool – Tests Random Access 8KB, 512KB**

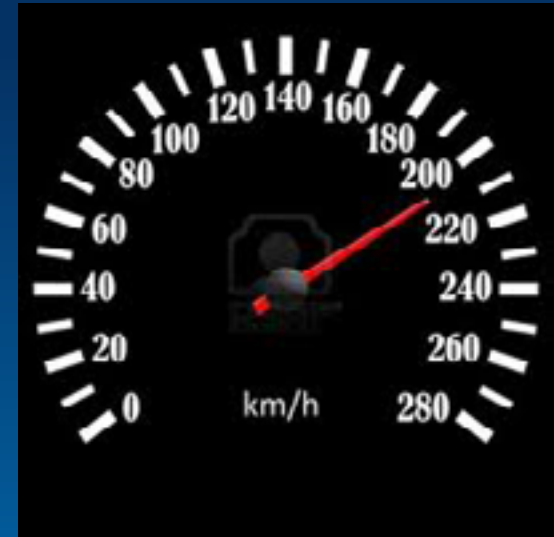


Performance Factors

Andrew Sakowicz

Top 5 tips for good performance and scalability

1. **Tune map (each scale)**
2. **Provide sufficient hardware resources**
3. **Set appropriate min and max number of instances**
4. **Test**
5. **Monitor**



Provide sufficient hardware resources

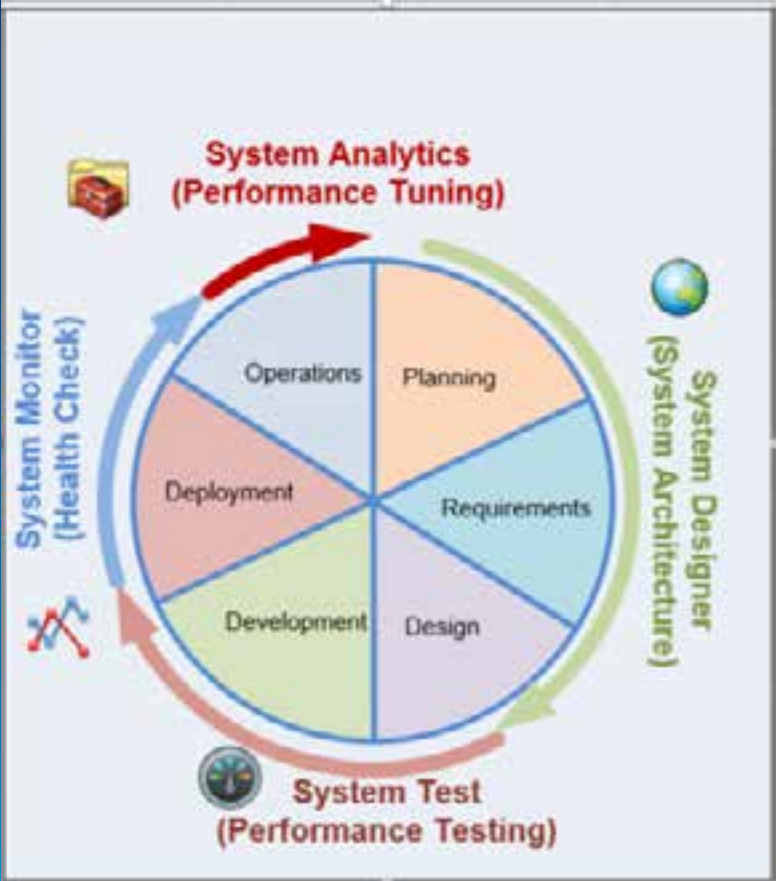
Most systems are CPU bound

GIS Systems are bound by:

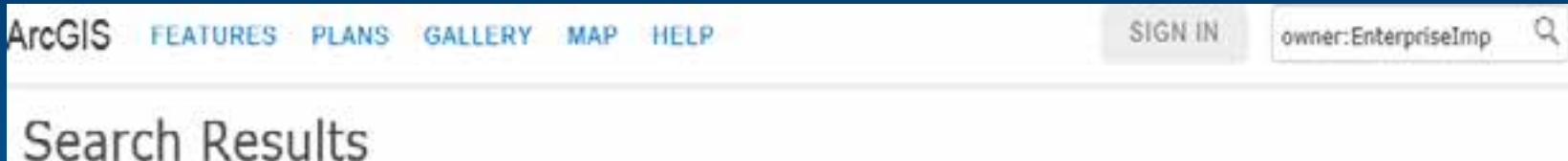
- 1. CPU - typically**
- 2. Memory – when large number of services**
- 3. Disk – Image Service, Synchronization**
- 4. Network – low bandwidth deployment**
- 5. Poorly configured virtualization can result in 30% or higher performance degradation**

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

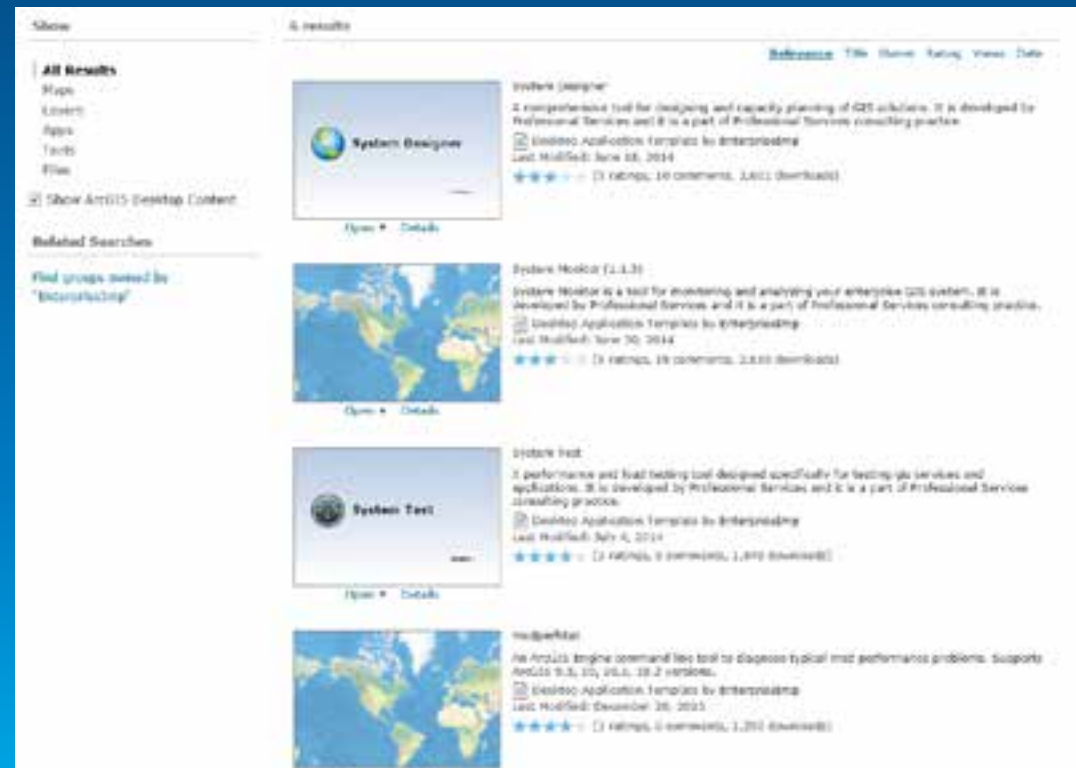
Process and Tools



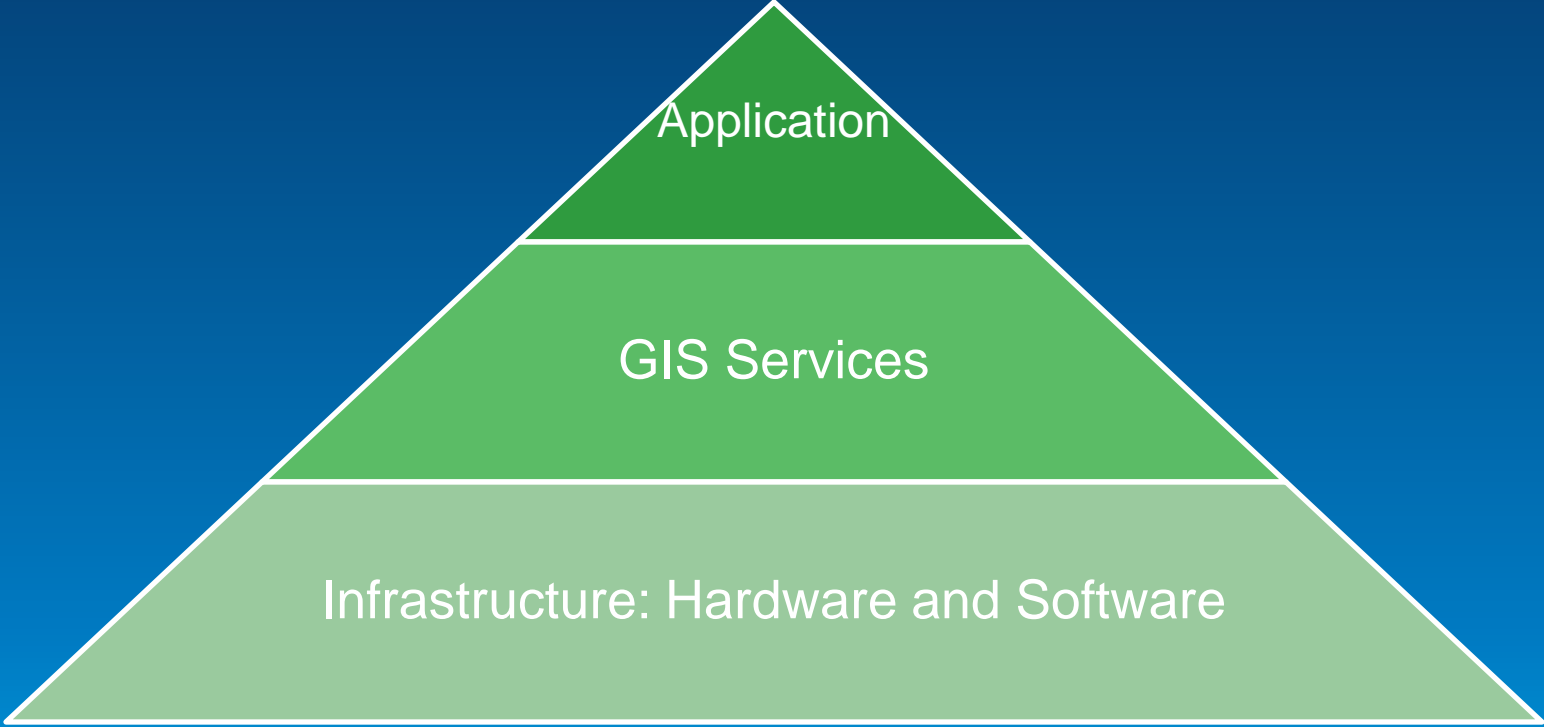
System Tools overview



- <http://www.arcgis.com>
- owner:EnterpriseImp
- Show ArcGIS Desktop Content



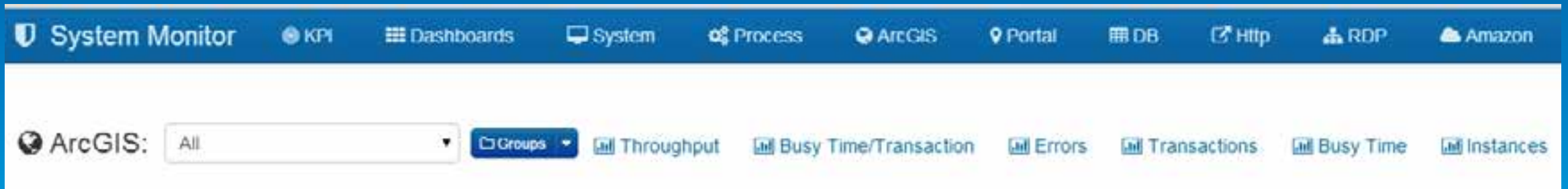
Testing process



Enterprise GIS effective monitoring

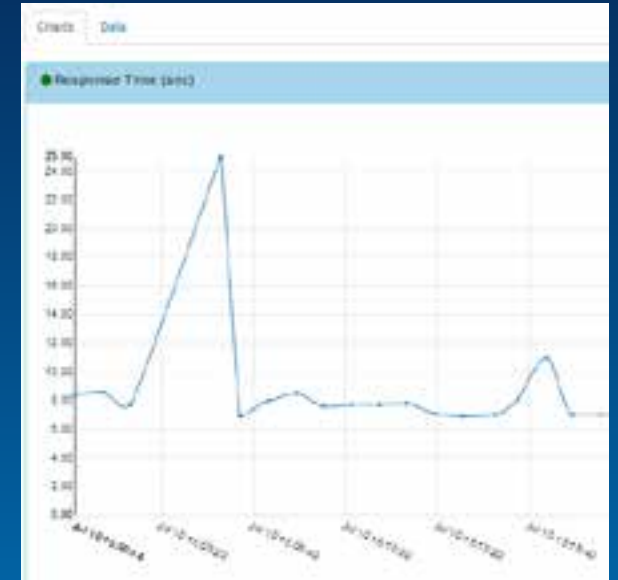
End-to-End monitoring using System Monitor

- To verify resources, must monitor end-to-end
- ArcGIS Serve key stats
 - Busy Time/Transaction
 - Free instances
 - Throughput



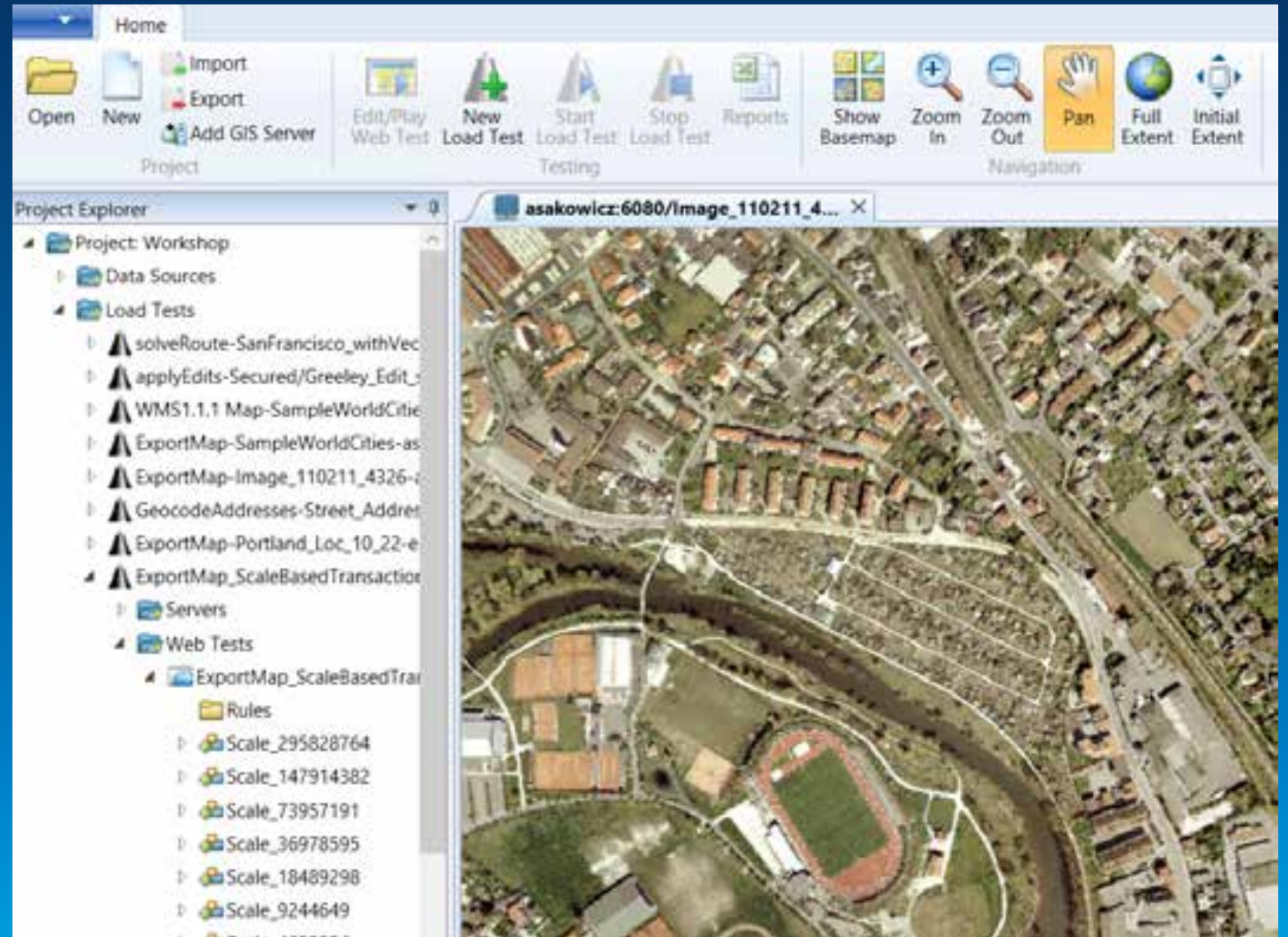
Demo

Intermittent slow performance:
CPU spike



Demo

Scale Based Transactions



Demo

Intermittent slow performance:
ArcGIS Server free instances

Last Updated: Jul 10, 2014 4:34:13 PM

Filter: Limit: Top 10 Folder: All

Busy Time per Tr (sec)	Transactions	Max	Busy	Free
0	21,700	46	0	33
0	21,549	32	0	32
0	125	2	0	1
0	0	3	0	0

Questions

Thank you...

- **Please fill out the session survey (session 609):**

First Offering ID: 1225 (Wednesday)

Second Offering ID: 1356 (Thursday)

Online – www.esri.com/ucsessionsurveys

Paper – pick up and put in drop box



Understanding our world.