



Building and Using ArcGIS Server Map Caches (Best Practices)

Jeremy Bartley

jbartley@esri.com

ESRI Server Development

Map Caches

- **2D Map cache overview**
- **Deconstructing the tiling scheme**
 - **Programming against the map cache**
 - **Example: Querying tile extent**
 - **Example: Computing number of tiles per scale level**
- **Designing and authoring a map cache**
- **Generating the map cache**
- **Using ArcGIS Server Map Services with map caches**
 - **From different clients**
 - **ArcGIS Desktop**
 - **ArcGIS Server Web ADF**

Building and Using ArcGIS Server Map Caches

- **Technologies covered:**
 - ArcGIS Server, ArcGIS Desktop
 - Web ADF Applications built using the Microsoft .NET Framework and Java Server 9.2 sp2 (coming soon!)
- **Assuming familiarity with:**
 - ArcGIS Desktop
 - ArcGIS Server deployment
 - ArcGIS Server Web ADF usage
- **Code samples will be shown using in C# and Python**

2D Map cache overview

2D Map Cache overview

- Why do we want to cache services?
- The best answer to that is illustrated with a demo...
- Demo
 - <http://jbartley/agoblend/>
 - <http://serverz.esri.com/WorldImageryBasemap/>

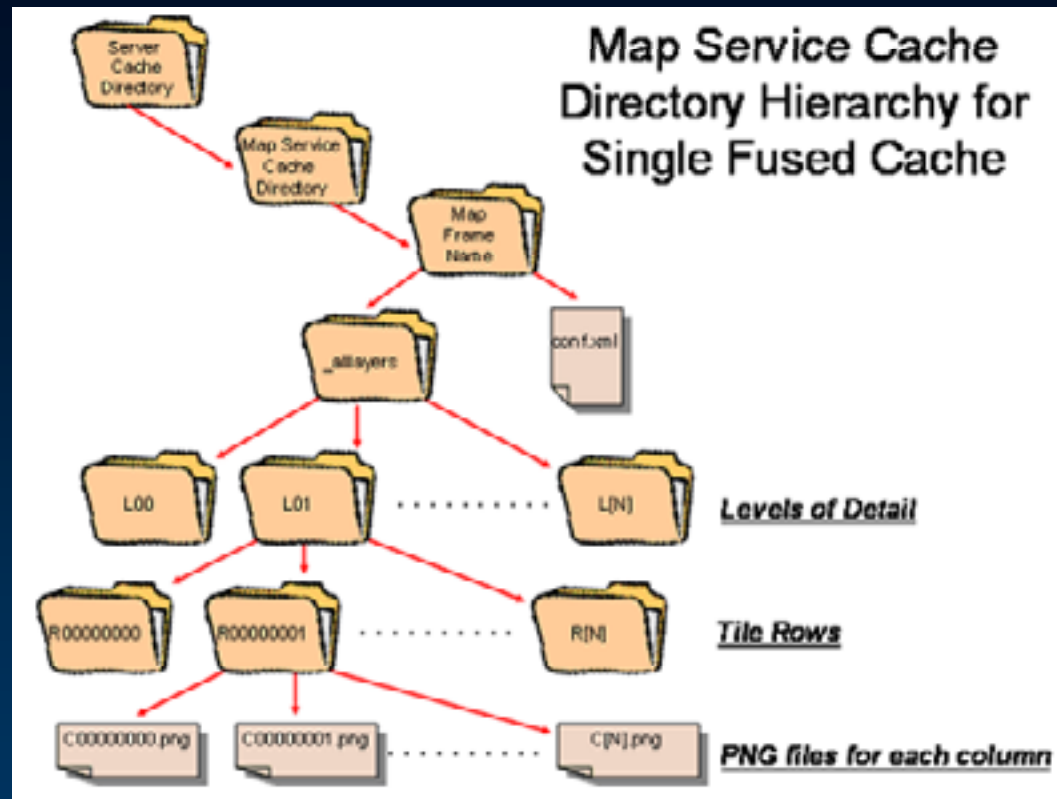
2D Map Cache overview

- **ArcGIS Server map services can now use an image cache to significantly improve performance while delivering maps.**
- **A map service that fulfills requests with pre-created tiles from a cache instead of dynamically rendering part of the map is called a cached map service.**
- **Clients can still access the underlying data of the cached map service**
 - **Identify, Query, Highlight feature geometry**

2D Map Cache overview

- **A cached map service is composed of a hierarchical collection of cache image tiles and a configuration file which describes the tiling scheme.**
- **Single Fused Cache**
 - All layers in a map service are burned to a single cached image tile (efficient in viewing performance and tile creation)
- **Multi-layer Cache**
 - Each layer in a map service is treated as a separate cache within a map service hierarchy.
 - Layers can be turned on/off.
 - ADF access to multi-layer cache is done via `ExportMapImage`

2D Map Cache Overview – Image hierarchy



- Cache image directory structure
- Association between the map service and the map service cache directory is by name

2D Map Cache Overview – Image hierarchy

http://services.arcgisonline.com/cache_im/I3_Imagery_Prime_World/2D/Layers/_alllayers/L15/R000027f9/C00002d2c.jpg

- Virtual Cache Directory
 - cache_im
- Map Service Cache Directory
 - I3_Imagery_Prime_World
- Map Data Frame name
 - Layers
- Layer name (fused=all layers)
 - _alllayers
- Level
 - L15
- Row (Hexadecimal)
 - R000027f9
- Column (actual tile name)
 - C00002d2c.jpg



Deconstructing the tiling scheme

Deconstructing the tiling scheme

- Cache structure is defined by the cache configuration (conf.xml)
- Conf.xml contains:
 - CacheInfo
 - TileCacheInfo
 - Properties pertaining to the spatial reference, tile origin, tile cols (tile pixel size in X), tile rows (tile pixel size in Y), DPI, levels of detail
 - TileImageInfo
 - Properties pertaining to the cache tile image format and the image compression value (for JPEG only)

http://services.arcgisonline.com/cache_im/l3/Imagery_Prime_World_2D/Layers/conf.xml

Deconstructing the tiling scheme

- **Programming methods for accessing information about the tiling scheme**
 - **Work with an existing MapServer service and use ArcObjects or SOAP to access the tile configuration information**
 - **ITiledMapServer Interface**
 - **GetTileCacheInfo**
 - **GetTileImageInfo**
 - **Direct reading and parsing of an individual cache configuration XML file.**
 - **Useful when programming in Python**

Deconstructing the tiling scheme

- **SpatialReference**

- Defines the spatial reference of the map cache. Used in determining the size in map units of an individual pixel.
- All map caches must have a defined spatial reference
- Derived from the source map service data frame coordinate system.

Deconstructing the tiling scheme

- **TileOrigin**

- Each tile is assigned a row and column number (TROW, TCOL) in a tile coordinate system. TROW, which starts from 0, increases from 'top to bottom' (i.e tile row number increases as map y coordinate decreases). TCOL, also starting from 0, increases from 'left to right' (i.e. tile column number increases as map x coordinate increases).
- TileOrigin is specified in map units
- Can be located inside or outside of your data frame extent
 - Tiles will not be created outside the tile origin.

Deconstructing the tiling scheme

- **TileCols**
 - Number of pixel columns in an individual tile
- **TileRows**
 - Number of pixel rows in an individual tile
- **Both are used to define the tile size in pixel coordinates**

Deconstructing the tiling scheme

- **DPI**

- Dots per inch is used to compute the resolution of an individual pixel for the cache tile that the server generates.

$$\frac{1}{10,000} = \frac{1}{96 \times \text{Pixel_size_in_inches}}$$

- **Pixel_Size_In_Inches = 104.166667 inches**
 - This value is then used to compute the resolution of a pixel in map units
- If printing map caches you may find it useful to increase the DPI beyond the default (96) to match the output devices DPI setting. This will increase the number of the files in the cache.

Deconstructing the tiling scheme

- **LODInfos**
 - **Levels of Detail**
 - **Defines the set of scales that the map cache was generated at**
 - **At each level the cache knows about:**
 - **LevelID**
 - The level index
 - **Scale**
 - The user defined scale
 - **Resolution**
 - The derived size in map units of an individual pixel (derived from DPI and map projection information)

Deconstructing the tiling scheme

- **To define the size of an individual tile at a particular scale in map units you need the following information**
 - **TileRows = number of rows in an individual tile**
 - **TileCols = number of cols in an individual tile**
 - **TileWidth = TileCols * Resolution (for a particular scale)**
 - **TileHeight = TileCols * Resolution (for a particular scale)**

Deconstructing the tiling scheme

- **Given a map coordinate (x,y) the tile coordinates (Row and Column) of the tile containing it for a given LOD are given by:**
 - **Column = floor((x- xorigin) / TileWidth)**
 - **Row = floor ((yorigin – y) / TileHeight)**
- **The corner points in map coordinates of a tile with tile coordinates (row and column) are:**
 - **Xmin = xorigin + (Column * TileWidth)**
 - **Xmax = xmin + TileWidth**
 - **Ymax = yorigin – (Row * TileHeight)**
 - **Ymin = ymax - TileHeight**

Deconstructing the tiling scheme

- Python example of extracting the bounding box of an individual tile.
- `getExtent.py`
- View results in ArcMap



L15/R000027f9/C00002d2c.jpg

Deconstructing the tiling scheme

- Lets use the information we just learned to compute the number of tiles that need to be generated for each scale...
- C# example
 - <http://jbartley/TileCacheInfo>

Deconstructing the tiling scheme

- **TileImageInfo – Contains information about the image type of the map cache**
 - **PNG8 — a lossless, 8-bit color, image format that uses an indexed color palette**
 - Transparency is stored in the color index palette, excellent browser support
 - **PNG24 — a lossless, three-channel image format that supports large color variations (16 million colors) and has limited support for transparency.**
 - Transparency value is stored in the image header. Versions of Internet Explorer less than version 7 do not support this type of transparency.

Deconstructing the tiling scheme

- **TileImageInfo** – Contains information about the image type of the map cache
 - **PNG32** — a lossless, four-channel image format that supports large color variations (16 million colors) and transparency.
 - **JPEG** — a lossy, three-channel image format that supports large color variations (16 million colors) but does not support transparency.

Designing and authoring a map cache

Designing and authoring a map cache

- **Before you build a map cache, it's important to think about the tiling scheme you will use and the resources that will be needed to build the cache.**
- **When possible, it's a good idea to select one tiling scheme and use it for all of the maps that you will cache for your organization.**
 - **When you do this, you cache the same map area at the same set of scale levels for all caches.**
 - **This will ensure that all your fused map caches can be efficiently combined within your end client (ArcGIS Desktop, Web ADF)**

Designing and authoring a map cache

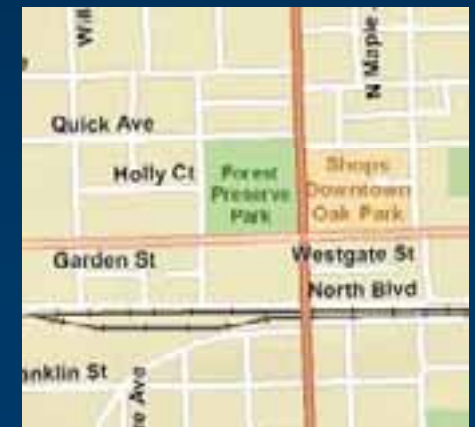
- **For one organization**
 - One tiling schema
 - Identify a set of basemaps
 - Try not to duplicate basemaps across multiple tiling schemes
 - Group layers into logical thematic maps
 - Aerial Imagery
 - Hydrography (lakes, streams, ponds, rivers, etc.)
 - Transportation (local roads, main thoroughfares, highways, etc.)
 - Landbase (parcels, building footprints)
 - Elevation
 - Best performance will be achieved by blending multiple fused cached map services

Designing and authoring a map cache

- **Example...**

- **Your are the GIS Analyst at a county government. Your boss would like you to build a cached map service of your county. She would like you to build this map so that it can be viewable from 1:1,000,000 down to 1:10,000.**

- **Easy right?**



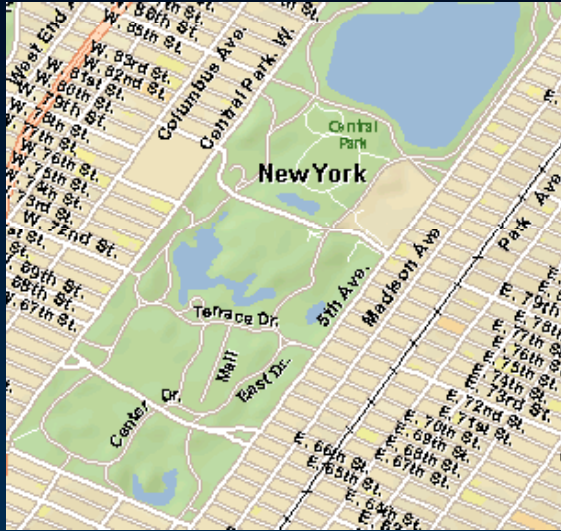


-122.25, 37.75

Designing and authoring a map cache

- **Cached map services must be authored at EACH scale that is to be cached.**
- **You can ensure that your map is well-designed by applying scale-dependencies to labels and features.**
- **These scale-dependent changes help the map convey more information in an organized manner.**
- **The scale dependencies you set for your map layers and the scale levels you choose for your tiling scheme should be carefully coordinated.**

Designing and authoring a map cache – Anti-aliasing



- Smooths the edges of labels and lines by blending them with the background.
- The resulting screen display quality can be better than standard rendering in ArcMap.
- Tiles are rendered at finer resolution by down sampling (takes twice as long to cache an area when using anti-aliasing)
- Be cognizant of the client application (especially when building a service that is meant to be used as an overlay service)
 - <http://serverx.esri.com/antialiasingexamples/>

Designing and authoring a map cache

- **Background color**

- The background color is used to define the transparent part of the image
- Explicitly define the background color
 - If the background color is not defined then the transparent color will be set to 253,253,253
- Use a color not used in the symbology
- If using anti-aliasing with a cached overlay service set the background color close to the average color of your basemap

Generating the map cache

Generating the map cache

- **The caching process can be a very time consuming process.**
- **Before initiating the cache building process for the entire map service, build it for a small area to test the output symbology, labeling, and the performance criteria in your primary map client.**

Generating the map cache

- **Two types of caching scenarios**
 - **Cities, counties**
 - **The Continental US down to 1:10,000**
- **The geographic extent and the levels of detail (scale levels) of the area of your caching job directly influences the amount time you will need to devote to the caching process**

Generating the map cache – current 9.2 GP tools

- **GenerateMapServerTilingScheme**
 - Generates a tiling scheme that can be used to create caches for multiple services
- **GenerateMapServerCache**
 - Generates the cache for a map service
 - Works with either a pre-defined or a newly defined tiling scheme
 - Creates the cache and populates it
- **UpdateMapServerCache**
 - Updates the cache for a map service within a specified extent
 - Creates only missing and empty tiles
 - Or
 - Recreates all tiles
- **DeleteMapServerCache**
 - Deletes the cache for a map service

Generating the map cache

- **Caching jobs are defined by...**
 - Cache tiling scheme
 - Full extent of the map service (derived from the data frame full extent in the source MXD)
- **For example:**
 - Tiling origin is -180, 90
 - Map service extent is the bounding box of Colorado
 - Tiles will only be generated within and around the bounding box of Colorado
 - They tile rows and column will be referenced from the tile origin (-180, 90)

Generating the map cache

- **Impact of tile size selection**
 - **Larger size produces fewer tiles**
 - **Less disk space (block size)**
 - **Faster creation**
 - **Easier to manage**
 - **Smaller size**
 - **Allows partial update of the display**
 - **Takes approximately 5X as long and takes up 1Gb more of space when creating a cache at 128x128 tile size versus 512x512 tile size with the same data (Hawaii)**

Generating the map cache

- **When caching very large geographic areas break up caching job to distinct areas.**
 - Use `UpdateMapServerCache` at specific user defined extents in a script environment (SP2)
- **Areas that don't need to be cached should be built using custom extents**
 - Alaska, Hawaii, Continental US, but not all scale levels of the Pacific Ocean

Generating the map cache

- **Difference between caching GP tools:**
 - **GenerateMapServerCache**
 - Used to generate a map cache from a map service
 - Generates map tiles for all scales defined in the cache schema
 - Partially checks to see if tiles have been created previously
 - Caches everything within map service full extent at all defined levels of detail
 - **UpdateMapServerCache**
 - Used to update individual scales and extents
 - Updates can be for all tiles or for missing tiles in given extent
 - When updating missing tiles, performs a check on cache to find missing tiles and tiles of 0k size

Using ArcGIS Server Map Services with map caches

Using ArcGIS Server Map Services with map caches

- **ArcGIS Desktop consumption**
 - **More control of map cache display**
 - **Continuous zoom against map cache**
 - **Snaps current view to a level of detail in the map cache and then resamples up to match the current view**
 - **Usually snaps to the next lowest scale compared to the current scale**
 - **ArcGIS Desktop locally caches map caches on the local client**
 - **Located: %temp%\esrimapcache**
 - **Can get out of sync with server side cache**
 - **Can easily blend with other services and data**

Using ArcGIS Server Map Services with map caches

- Lets take a look in ArcMap.
 - Examine cache properties
 - Display local cache
 - Remove local cache
 - Change projections of the cache
 - View other datasets with the cache

Using ArcGIS Server Map Services with map caches

- **ArcGIS Server Web ADF**
 - Significant difference between SP1 and SP2
 - Differences between Java and .NET implementations
 - Unfortunately application developer must learn about browser idiosyncrasies

Using ArcGIS Server Map Services with map caches

- **Fused caches and the Web ADF for the Java Platform at 9.2 SP1**
 - **Handles only one fused cache map service by retrieving tiles from the virtual directory**
 - **Any layer above the base layer in the Java ADF is treated as a dynamic layer drawn to the graphics layer as a single image overlay**
 - **Combination of two fused cached services will result in tile retrieval from the virtual cache directory for the base layer with single ExportMapImage requests to the map server for the top layer**

Using ArcGIS Server Map Services with map caches

- **Displaying multiple fused caches services in the Web ADF for the Microsoft .NET Framework at SP1**
 - **Cached service + cached service(s) with browser blending. Bottom layer is treated as the primary map resource unless the resource has been explicitly designated in Visual Studio.**
 - **The non-primary resource will be accessible via the virtual cache directory if the two services share:**
 - **Identical tiling schemes (origin, levels of detail, tile size)**
 - **Identical projection**
 - **Identical full extents in the parent map services**
 - **If resources are not identical then the top or non-primary resource will be requested using ExportMapImage**

Using ArcGIS Server Map Services with map caches

- **9.2 SP2 has a number of enhancements for blending multiple cached services.**
 - **Criteria to efficiently blend multiple cached services in Java**
 - **Projection must be the same**
 - **Scale levels should be similar between multiple services**
 - **Criterion to efficiently blend multiple cached services in .NET**
 - **Projection must be the same**
 - **Scale levels should be similar**
 - **Tile origin and tile size should be identical between services**

Using ArcGIS Server Map Services with map caches

- **IE6 with transparent PNG images.**
 - Transparency in PNG24 is not honored in IE6
 - Best practice is to use PNG8 or PNG32
- **In .NET SP2**
 - The service that is either non-primary or physically located above the base service will not retrieve from tiles via the virtual directory for PNG24 cache services in IE6
 - The resource will be requested via ExportMapImage (can result in slower performance)
- **In Java at 9.2 SP2**
 - PNG24 caches in IE6 will be used via the virtual directory, but they will not be transparent

Map caches

- **2D Map cache overview**
- **Deconstructing the tiling scheme**
 - **Programming against the map cache**
 - **Example: Querying tile extent**
 - **Example: Computing number of tiles per scale level**
- **Designing and authoring a map cache**
- **Generating the map cache**
- **Using ArcGIS Server Map Services with map caches**
 - **From different clients**
 - **ArcGIS Desktop**
 - **ArcGIS Server Web ADF**

Presentation materials

- PowerPoint presentation and code are posted on the conference web site
 - <http://www.esri.com/events/devsummit/index.html>
- EDN – downloads and videos

Further questions?

- **TECH-TALK AREAS**

- **What:** Further opportunity to discuss questions and concerns with presenters and subject matter experts
- **Where:** Tech Talk Area 2 (located in Oasis 3)
- **When:** during the next 30 minutes

- **ESRI Showcase**

- **Meet the teams**

- **ArcGIS Server Development Blog**

- <http://blogs.esri.com>

- **ESRI Developers Network (EDN) website**

- <http://edn.esri.com>

- **ESRI Webhelp**

- <http://webhelp.esri.com>



Map Caches

Thanks!

Jeremy Bartley
jbartley@esri.com