WOWZA®
MEDIA SYSTEMS

Wowza Media Server®

MediaCache

Wowza Media Server: MediaCache

Version 2.0

Table of Contents

Introduction

The Wowza Media Server MediaCache system is a read through caching mechanism for video on demand streaming. Similar to the live repeater technology aimed at increasing the scalability of live streaming, the MediaCache technology is a means of scaling video on demand streaming. The system is very flexible and extendable. The first release of the MediaCache technology can retrieve content from either a web server or http (must support HTTP/1.1 byte range requests) or network attached file system (network attached storage, network file system or any similar device that is recognized as a disk to the operating system).

Installation

To install, follow these steps. The default install will setup an application named **mediacache** from which content from any web server can be re-streamed:

1. Copy **lib/wms-plugin-mediacache.jar** from this package into the Wowza Server folder **[install-dir]/lib** folder

2. Copy **conf/MediaCache.xml** from this package into the Wowza Server folder **[install-dir]/conf** folder

3. Create the folder **[install-dir]/applications/mediacache**

4. Create the folder **[install-dir]/conf/mediacache** and copy the **[install-dir]/conf/Application.xml** into this new folder.

5. Edit the newly copied **Application.xml** file and make the following changes:

    a. Set the **RTP/Authentication/PlayMethod** to: **none**

    b. Add the following property to the **MediaReader/Properties** so that the **<MediaReader>** section looks like this:

```
<MediaReader>
   <Properties>
      <Property>
         <Name>randomAccessReaderClass</Name>
         <Value>com.wowza.wms.plugin.mediacache.impl.MediaCacheRandomAccessReader</Value>
      </Property>
      <Property>
         <Name>bufferSeekIO</Name>
         <Value>true</Value>
         <Type>Boolean</Type>
      </Property>
   </Properties>
</MediaReader>
```

6. Edit **[install-dir]/conf/Server.xml** and add a reference to the **MediaCacheServerListener** class in the **<ServerListeners>** section:

```
<ServerListener>
   <BaseClass>com.wowza.wms.plugin.mediacache.impl.MediaCacheServerListener</BaseClass>
</ServerListener>
```

Configuration

The default **MediaCache.xml** file is configured for re-streaming content from an HTTP source. The default configuration file is tuned assuming Wowza Media Server is running on a 32-bit server with a 1200M (1.2 GB) Java heap size with a single mechanical hard disk used for caching. The file includes additional commented out tuning suggestions based on heap size, disk speed and disk technology.

Below we will explore the different configuration options. Let' start off by taking a look at the **[install-dir]/conf/MediaCache.xml** file.

MediaCache Configuration

The first section is the **<MediaCache>** section which is for configuring the overall MediaCache settings. Below is an explanation of each of the settings.

WriterThreadPool/PoolSize

The number of threads in the writer pool that is used to write media blocks to the caching system. This value should be set to twice the number of cpu cores on the machine.

ReadAheadPool/PoolSize

The number of threads in the read ahead pool that is used to read blocks from the cache source before they are requested. The read ahead system keeps a steady flow of bytes from the source to the cache to avoid stuttering during playback. This value should be set to the number of cpu cores on the machine

MaxPendingWriteRequestSize

The number of bytes of memory that can be occupied by blocks waiting to be written to storage. Think of it as a temporary memory based cache. Values are expressed in bytes and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

MaxPendingReadAheadRequestSize

The number of bytes of memory that can be occupied by blocks waiting to be written to storage for read ahead. Values are expressed in bytes and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

GCFrequency

The number of milliseconds between cache purging/pruning sessions. Based on time to live settings, items stored in the cache will be purged when they have not been used in a given period of time (maximum time to live) or if there is content waiting to enter the cache (minimum time to live).

ContextMapperClass

Not used at this time (leave blank).

Properties

Properties stored in the MediaCache.getProperties() container. Below are a few of the more interesting properties that are available:

addFileExtensionIfNeeded

If true and the stream name does not include a file extension a file extension will be added based on the stream name prefix.

urlEscapeStreamNameSpaces

If true, spaces in stream names will be URL escaped before being sent to the source HTTP server.

urlEscapeStreamNameAll

If true, the entire stream name will be URL escaped before being sent to the source HTTP server.

MediaCacheStores Configuration

The <MediaCacheStores> section defines where the cached files will live. You can configure multiple cache stores to store cached content on multiple volumes on a single machine. You can also use RAID technology and create a single store from multiples disks.

Path

Path to storage location. Always use forward slashes when defining paths. Cached files will be stored in a two tier directory structure.

MaxSize

The maximum size in bytes of data that will be stored in this cache store. Values are expressed in bytes and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

Level1FolderCount

Number of folders created at the first level to store cached items.

Level2FolderCount
Number of folders created at the second level to store cached items.

FileCount
Number of items that can be stored in each level2 folder. For example if the level1 folder count is set to 16 and level2 folder count is set to 16 and file count is set to 1000 then 16x16x1000 or 256000 files can be stored in this cache store.

WriteRate
The maximum rate in bytes-per-second that content can be written to this store. Throttling the write rate helps to control the flow of content entering the cache so that it does not overwhelm the file system for content that is being served out of the cache. Values are expressed in bytes per second and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

WriteRateMaxBucketSize
This value works in concert with the WriteRate to control how the write rate is throttled. This value should be set to 6 times the WriteRate. Values are expressed in bytes and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

WriteRateFillFrequency
How often the write rate control mechanism in milliseconds is refreshed. This value works in concert with the WriteRate to control how the write rate is throttled. This value should be set to 100.

Properties
Properties stored in the MediaCacheStore.getProperties() container.

MediaCacheSources Configuration
The <MediaCacheSection> section defines the sources of content that are made available to Wowza Server. This section will just enumerate and explain the settings. The next section will include more examples.

Name
Simple name given to the source for logging purposes. This name is not used to control streaming or the addressing of the content.

BasePath and Prefix
The BasePath and Prefix settings work together to control how content is mapped back to a source configuration. The Prefix is used to map the stream name to the source. To restream the

content the Prefix portion is replaced by the BasePath value. For example if you set Prefix to content1/ and BasePath to http:// then the stream name:

content1/mycoolvideo.flv

Will be retrieved from the url:

http://mycoolvideo.flv

The default configuration sets the Prefix to http/ and the BasePath to http://. This effectively enables streaming of any http:// based content through the cache.

BaseClass

The full class path to the MediaCacheSource implementation that is used to deliver the content. There are currently two MediaCacheSource implementations to choose from:

```
com.wowza.wms.plugin.mediacache.impl.MediaCacheItemHTTPImpl - HTTP content
com.wowza.wms.plugin.mediacache.impl.MediaCacheItemFileImpl - File base (NFS, NAS…)
```

This is explained in the next section.

ReaderClass

Not used at this time.

DefaultBlockSize

Size of the blocks in bytes of data that are read from the source to populate the cache. Values are expressed in bytes and the following units are supported K (kilobytes), M (megabytes), G (gigabytes) or T (terabytes).

MaxTimeToLive

Maximum time in milliseconds that an item will remain in the cache unused. The time to live counter starts when the last client to view the content stops viewing the content. If no other viewers view this content within the time to live window the content will be purged from the cache.

MinTimeToLive

Minimum time in milliseconds that an item will remain in the cache unused if there are items waiting to enter the cache and the cache is full.

ReadAhead and ReadAheadThreshold

If ReadAhead is set to true, content blocks will be pre-populated in the cache based on the ReadAheadThreshold percentage. If set to false, no read ahead occurs. The ReadAheadThreshold is a percentage that controls when the request is made to read the next block. For example if ReadAheadThreshold is set to 50% then when any content is read beyond the half way mark of the current block of content, the next block will queued to be read.

IsPassThru

Not used at this time but will be used to control content that is managed by the cache but does not live in the cache.

Properties

Properties stored in the MediaCacheSource.getProperties() container. Below are a few of the more interesting properties that are available (these properties are commented out in the default conf/MediaCache.xml file – to use them uncomment them):

maxPoolSize

The MediaCache system creates a pool of HTTP TCP connections per-source. Requests are sent to the source using this HTTP TCP pool. This property controls the number of HTTP TCP connections in the pool.

httpReadTimeout, httpConnectionTimeout, httpReadRetries

These settings control the how HTTP TCP connections are handled for this source. httpConnectionTimeout is the number of milliseconds the server will wait for a TCP connection. httpReadTimeout is the number of milliseconds the server will wait for a read request to be fulfilled. httpReadRetries is the number of times an HTTP request will be retried if it fails.

httpSendBufferSize and httpReceiveBufferSize

These settings control the low level TCP buffer settings for this source (send and receive buffer sizes in bytes).

isAmazonS3 and s3BucketNameInDomain

This enables re-streaming from an Amazon S3 bucket. This property when set to true, instructs MediaCache to do extra URL formatting to properly address content on S3. The s3BucketNameInDomain property controls how the HTTP requests are formatted. If true, the domain name will be include the bucket name. If false, the bucket name will included as part of the HTTP path.

For example, if you set the **BasePath** value to **http://s3.amazonaws.com/** and the **Prefix** to **amazons3/** then MediaCache will treat stream names that start with the **amazons3/** prefix as Amazon S3 content. The first path element of the stream name after the prefix is the bucket name and the rest of the elements are the path to the content. For example, if the stream name is:

mp4:amazons3/mybucket/mypath/myfile.mp4

The bucket name is **mybucket** and the path to the content is **mypath/myfile.mp4**. Based on the s3BucketNameInDomain setting MediaCache will fetch the content from one of the following URLs:

If s3BucketNameInDomain is true:

http://mybucket.s3.amazonaws.com/mypath/myfile.mp4

If s3BucketNameInDomain is false:

http://s3.amazonaws.com/mybucket/mypath/myfile.mp4

awsSecretAccessKey and awsAccessKeyId

This enables re-streaming from an Amazon S3 bucket that is not publicly available. Set these properties to your Amazon S3 secret access key and key id to instruct MediaCache to properly sign the HTTP requests so that they are authenticated for access to protected content.

maxPoolSize

The MediaCache system creates a pool of HTTP TCP connections per-source. Requests are sent to the source using this HTTP TCP pool. This property controls the number of HTTP TCP connections in the pool.

proxyHost, proxyPort and requestFullURL

If these properties are set, the MediaCache system will send all HTTP requests to the defined proxy.

Example Configurations

Below are a few example source configurations. These examples reference the basic video on demand tutorial that can be found here:

How to play a video on demand file

Example1

```
<MediaCacheSource>
   <Name>http</Name>
   <BasePath>http://</BasePath>
   <Prefix>http/</Prefix>
   <BaseClass>com.wowza.wms.plugin.mediacache.impl.MediaCacheItemHTTPImpl</BaseClass>
   <ReaderClass></ReaderClass>
   <DefaultBlockSize>262144</DefaultBlockSize>
   <MaxTimeToLive>1200000</MaxTimeToLive>
   <MinTimeToLive>600000</MinTimeToLive>
   <ReadAhead>true</ReadAhead>
   <ReadAheadThreshold>50</ReadAheadThreshold>
   <IsPassThru>false</IsPassThru>
   <Properties>
   </Properties>
</MediaCacheSource>
```

This is a simple HTTP configuration that enables streaming of any HTTP based content through the MediaCache system.

For example, say you have a media file at the URL:

```
http://mycompany.com/media/sample.mp4
```

To play this stream, use the following URLs and stream names (see the How to play a video on demand file tutorial for more detailed information and example players):

To play using Adobe Flash player (RTMP)

```
Server: rtmp://[wowza-address]/mediacache
Stream: mp4:http/mycompany.com/media/sample.mp4
```

To play using Adobe Flash player (San Jose/Flash HTTP)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http/
mycompany.com/media/sample.mp4/manifest.f4m
```

To play using an Apple iOS device (Cupertino/Apple HTTP Live Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http/
mycompany.com/media/sample.mp4/playlist.m3u8
```

To play using Microsoft Silverlight (Smooth Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http/
mycompany.com/media/sample.mp4/Manifest
```

To play using RTSP/RTP player or device

```
rtsp://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http/
mycompany.com/media/sample.mp4
```

Example2

```
<MediaCacheSource>
   <Name>http1</Name>
   <BasePath>http://mycompany.com/</BasePath>
   <Prefix>http1/</Prefix>
   <BaseClass>com.wowza.wms.plugin.mediacache.impl.MediaCacheItemHTTPImpl</BaseClass>
   <ReaderClass></ReaderClass>
   <DefaultBlockSize>262144</DefaultBlockSize>
   <MaxTimeToLive>1200000</MaxTimeToLive>
   <MinTimeToLive>600000</MinTimeToLive>
   <ReadAhead>true</ReadAhead>
   <ReadAheadThreshold>50</ReadAheadThreshold>
   <IsPassThru>false</IsPassThru>
   <Properties>
   </Properties>
</MediaCacheSource>
```

This is an HTTP configuration that limits streaming to a particular domain. In this case we have mapped the prefix **http1/** (Prefix) to the base path **http://mycompany.com** (BasePath).

For example, say you have a media file at the URL:

```
http://mycompany.com/media/sample.mp4
```

To play this stream, use the following URLs and stream names (see the How to play a video on demand file tutorial for more detailed information and example players):

To play using Adobe Flash player (RTMP)

```
Server: rtmp://[wowza-address]/mediacache
Stream: mp4:http1/media/sample.mp4
```

To play using Adobe Flash player (San Jose/Flash HTTP)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http1/media/sample.mp4/manifest.f4m
```

To play using an Apple iOS device (Cupertino/Apple HTTP Live Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http1/media/sample.mp4/playlist.m3u8
```

To play using Microsoft Silverlight (Smooth Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http1/media/sample.mp4/Manifest
```

To play using RTSP/RTP player or device

```
rtsp://[wowza-ip-address]:1935/mediacache/_definst_/mp4:http1/media/sample.mp4
```

Example3

```
<MediaCacheSource>
   <Name>http1</Name>
   <BasePath>http://www.mycompany.com/</BasePath>
   <Prefix>http1/</Prefix>
   <BaseClass>com.wowza.wms.plugin.mediacache.impl.MediaCacheItemHTTPImpl</BaseClass>
   <ReaderClass></ReaderClass>
   <DefaultBlockSize>262144</DefaultBlockSize>
   <MaxTimeToLive>1200000</MaxTimeToLive>
   <MinTimeToLive>600000</MinTimeToLive>
   <ReadAhead>true</ReadAhead>
   <ReadAheadThreshold>50</ReadAheadThreshold>
   <IsPassThru>false</IsPassThru>
   <Properties>
      <Property>
         <Name>proxyHost</Name>
         <Value>myproxy.mycompany.com</Value>
      </Property>
      <Property>
         <Name>proxyPort</Name>
         <Value>8080</Value>
         <Type>Integer</Type>
      </Property>
      <Property>
         <Name>requestFullURL</Name>
         <Value>true</Value>
         <Type>Boolean</Type>
      </Property>
   </Properties>
</MediaCacheSource>
```

This is an HTTP configuration that limits streaming to a particular domain and sends all requests through a forward proxy server. The proxy server details are specified through the **proxyHost**,

proxyPort and **requestFullURL** properties. The **requestFullURL** property instructs the MediaCache system to use full URLs when making requests to the proxy server. This is a great way to create a multiple tier caching system. For example if you would like to server the same set of files out of multiple data centers you would setup a forward proxy server in each data center. You then configure the edge servers in a particular data center to use that data center's forward proxy server. All requests would be sent to the data center proxy server. If this proxy server could not fulfill the request then it would be sent to the origin for fulfillment.

Example4

```
<MediaCacheSource>
   <Name>content1</Name>
   <BasePath>C:/content/</BasePath>
   <Prefix>path1/</Prefix>
   <BaseClass>com.wowza.wms.plugin.mediacache.impl.MediaCacheItemFileImpl</BaseClass>
   <ReaderClass></ReaderClass>
   <DefaultBlockSize>262144</DefaultBlockSize>
   <MaxTimeToLive>1200000</MaxTimeToLive>
   <MinTimeToLive>600000</MinTimeToLive>
   <ReadAhead>true</ReadAhead>
   <ReadAheadThreshold>50</ReadAheadThreshold>
   <IsPassThru>false</IsPassThru>
   <Properties>
   </Properties>
</MediaCacheSource>
```

This is an network attached storage example. You can see this based on the fact that it uses the **BaseClass com.wowza.wms.plugin.mediacache.impl.MediaCacheItemFileImpl**. In this case the prefix is **path1/** (Prefix) and the base path is **C:/content/** (BasePath).

For example, say you have a media file at the URL:

```
C:/content/media/sample.mp4
```

To play this stream, use the following URLs and stream names (see the How to play a video on demand file tutorial for more detailed information and example players):

To play using Adobe Flash player (RTMP)

```
Server: rtmp://[wowza-address]/mediacache
Stream: mp4:path1/media/sample.mp4
```

To play using Adobe Flash player (San Jose/Flash HTTP)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:path1/media/sample.mp4/manifest.f4m
```

To play using an Apple iOS device (Cupertino/Apple HTTP Live Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:path1/media/sample.mp4/playlist.m3u8
```

To play using Microsoft Silverlight (Smooth Streaming)

```
http://[wowza-ip-address]:1935/mediacache/_definst_/mp4:path1/media/sample.mp4/Manifest
```

To play using RTSP/RTP player or device

```
rtsp://[wowza-ip-address]:1935/mediacache/_definst_/mp4:path1/media/sample.mp4
```

Notes and Observations

Below are a few notes and observations:

MediaCache Disk Allocation

On the Linux operating system you may see a discrepancy between what the MediaCache reports as the disk utilization/fullness of the cache disks and what is reported at the operating system level. MediaCache pre-allocates the entire file size when an item enters the cache and fills it in chunks. So if you add up all the file sizes of the individual files on the disk you will see it matches with the value reported by MediaCache. We fill the file in chunks based on what was viewed. Linux will only allocate a disk block at the OS level when it is used. So even though the file size is say 100MB if only the first 10MB are filled then the OS reports 10MB used. So if you have a high level of abandonment (viewers only watch the start of most files) then you will see this discrepancy.

We may at some point change the way we manage the cache to work around this issue. We have no short term plans to change this behavior.

MediaCache Performance on Windows

Depending on disk setup and disk drivers (we are not sure of the exact details) there may be a performance issue with MediaCache when running on the Windows operating system. It is suggested to run MediaCache on the Linux operating system using the most recent version of Java for your platform.