MistServer

A Lightweight, Highly Scalable Media Server

Prepared By

Timothy Siglin

Transitions, Inc.

7 September 2013

Transitions, Inc. 101 Neth Drive #1088 Milligan College, TN 37682-3062 USA **T** +1-202-670-2027 **E** projects@transitioned.com

Introduction and Executive Summary

This white paper explores MistServer, a greenfield initiative by DDVTech BV, The Netherlands-based startup who recently created a lightweight, scalable media server. With a goal of solving significant media delivery problems, MistServer's design offers benefits over current solutions.

Brains and brawn, just not in the same place. DDVTech's MistServer is based on a philosophy that a media server should do what it does best: serve audio and video streams. They split the brawn (MistServer) from the brain (MistSteward, MistCenter, or a web control interface).

In single-instance MistServer installations, a web controller interface works well, but MistServer is designed to scale to hundreds or thousands of servers per network. To effectively scale all media servers need to be treated equally; yet if all media servers simultaneously try to control the network, they will base their use (or abuse) of the network on their own observations. Splitting off the decision-making process into a separate robust tool solves this problem.

Scalability one hundred servers at a time. To scale MistServer to multiple servers, DDVTech claims it is as simple as just replicating an existing server. The brain recognizes new servers and applies resources accordingly. Even within a single hardware server, though, MistServer launches a single server instance for every end user request, yielding incredible scalability.

Lightweight. MistServer also treats non-essential tools to non-core modules, yielding a core server that runs in less than 500 kilobytes of memory. That's not a typo: when serving video, MistServer instances occupy less than one-half of one megabyte. Audio-only streams have been served at a rate of ~150,000 instances per commodity hardware server platform.

Specialized HTTP servers. Any media or script content served via MistServer's integrated HTTP server is generated on the fly, including JavaScript embed codes used to embed streams in a webpage. All content is tweaked just prior to delivery to guaranteed best performance.

Statistics in spades. MistServer's community and paid Long Term Support (LTS) versions offer metrics for all streaming content types, from RTMP to HTTP. The LTS version retains history on 2500 sessions per stream, and the community version has a 1000 session per stream history.

Rolling updates, fail-safe protection. If MistServer's main control function fails or is shut down, MistServer maintains current sessions until all content is successfully delivered to end users.

Limiting disk access. To accelerate delivery and guarantee consistency, MistServer uses buffers for live content and server-side players for on demand (VoD) content. DDVTech refers collectively to both as media source processes. Live content buffers all reside in memory, so the only reason for the server to generate disk access would be to record live content into MistServer's intra-server storage format. DTSC allows content to be played back prior to the completion of a live stream, offering catch-up, DVR, and live-pause functionality.

Buy or Try. DDVTech offers two versions of MistServer, a community and a paid LTS version. The LTS business-license version, with three years support and ongoing patches, is affordable at €299 or approximately $400 USD. Try either or download source code at mistserver.org.

Understanding MistServer

Before diving into the benefits of MistServer’s highly scalable yet lightweight approach, let’s first look at the server’s three key elements: Connectors, Buffers/Players, and a Controller.

Connectors are format-specific listening sockets that await requests in one of several discrete streaming delivery formats, like Adobe's traditional RTMP, Apple's adaptive bitrate (ABR) HTTP Live Streaming (HLS), or a number of other industry formats.

Buffers are publishing points where live content is stored in a sophisticated, transparent interim format (DTSC). For on-demand (VoD) files, a virtual server-side *Player* assembles content on the fly and then delivers it to the end user via a Connector in their preferred format.

A *Controller* acts as the streaming administrator's access into MistServer. For single-instance community and LTS versions of MistServer, an integrated web interface overlays the underlying Applications Programming Interface (API). This API offers easy access to all MistServer details and modules, regardless of the underlying native operating system. In multi-server setups, robust control functionality can be found in MistSteward, covered in a separate white paper.

JSON / JavaScript not Java. DDVTech believes a media server application should run natively on its host operating system. As such, the company wanted to avoid Java, as it requires extra dependencies beyond the native operating system and its security concerns and bloat didn't fit the MistServer lean approach to multiple server instances.

On the other hand, control interfaces—from the integrated web interface to what goes on under the hood—should be consistent and streamlined. DDVTech wrote the entire Controller interface in JavaScript and uses industry-standard JavaScript Object Notation (JSON) for its API.

The JSON API sends commands to the Controller and receives back configuration information, statistics, and a roster of installed and available capabilities. Stored configurations are pure JSON, too, saved as a snapshot of the configuration in the same format. The API integrates into third-party solutions to fully automate every MistServer aspect.

Storage management. One other feature that keeps MistServer very lean is the offloading of file storage distribution outside the server. In the MistServer community version, a file is manually placed and then its location noted via the Controller. It is possible to automate the process via MistSteward, covered in a later white paper, or with a third-party application via the JSON API.

Lean HTTP Servers

Not all HTTP servers are created equal. DDVTech’s lean approach even trickles down into their approach to the integrated HTTP server. The company says MistServer is specialized software, so they only include a limited HTTP server, one which is optimized to deliver media.

“Users who want to serve static files over HTTP ought to use a real HTTP server,” said Jaron Vietor, DDVTech’s CTO. “Our internal HTTP server handles one task incredibly well: serving video and associated metadata or scripts.”

MistServer's HTTP server runs on port 8080 by default, leaving port 80 open for other HTTP delivery jobs. The port 80 HTTP server forwards stream-only delivery to MistServer, based on the "one tool for one task" mentality pervasive throughout the MistServer delivery philosophy.

From a statistical standpoint—another strong point of MistServer, which we will cover later in this paper—HTTP sessions are kept active for a period of time after the connection disappears, to accommodate the fact that an end-user device may support HTTP 1.0 but not HTTP 1.1. In this way, MistServer addresses the fact one session may involve multiple setups/teardowns.

DDVTech notes that HTTP requests are a small part of larger delivery requests, and feels all served media needs to be tracked, so MistServer provides strong statistics/metadata reporting.

Scaling With Commodity Hardware

What's the impact of scaling thousands of server instances on a single machine? DDVTech needed to find a proper balance in the ongoing debate around specialized video server hardware versus commodity hardware. The company leans heavily towards lighter, commodity server hardware for two reasons: light duty machines cost less and fit well within a lightweight, modular server model. In addition, the company feels that using MistServer on almost any modern system will result in the server's available bandwidth being saturated long before the machine runs out of memory.

What about scalability impact? DDVTech agrees its server-instance-per-session approach will eventually use more resources than competing servers, for a certain number of simultaneous connections. Assessing actual performance, though, they note the point at which performance suffers is often beyond the bounds of available network bandwidth of a single machine.

MistServer instances take up approximately 500 kilobytes of RAM per instance, or less than one-half of one megabyte per server instance, allowing the use of multiple lighter machines versus one heavy machine. Multiple lighter machines provides hardware and software failover.

The 500 kilobyte server instance example is for serving video. An existing MistServer customer runs audio-only streams as part of an internet radio station and finds that audio-only streams require significantly less resources—more on the scale of ~100 kilobytes than ~500 kilobytes per instance—meaning 16GB of RAM provides approximately 150,000 sessions per machine.

Failsafe Streaming and Rolling Restarts

In traditional media server solutions, a server failure either abruptly ends all streaming sessions or drastically reduces the current viewers' quality of experience. DDVTech designed a way to solve this problem since MistServer runs thousands of server instances on a single machine.

Because MistServer separates the brains and the brawn, a carrier-centric approach that offers an elegant safe-guard of all the current sessions, MistServer protects all current sessions a Controller has previously started if the Controller fails or must be shut down. One byproduct of this protection scheme is that MistServer is also designed to allow staggered binary updates on a session-by-session basis.

The traditional way of updating the server binary—restarting a server after kicking every viewer off the system—just isn't practical in today's high-volume media environment, so the MistServer LTS version includes an "auto updater" tool that starts a rolling update process as soon as the server administrator clicks a button on the configuration page.

MistServer closes the listening socket of any idle Connector, updates the binary, and re-opens the socket. Since MistServer binaries are small, updates take milliseconds per Connector.

It is also possible to enforce a maximum play time or duration constraint, forcing the last few viewers off the server after a set period of time. Once all sessions using the old binary have been closed, the operating system kernel determines that the old binary is not in use and removes it from memory.

The use of a separate media process for each viewer makes this possible, either automatically or by manually killing the old processes. Remember that shutting the Controller down leaves a server with no way to modify configurations or receive statistics until the Controller is restarted.

Tracking Statistics

In this brave new world of ABR, HTTP delivery, and stateless environments, how to best track sessions for monetization and overall performance is a very valid concern for content owners.

HTTP delivery with no defined session start/end creates an analytics nightmare for our industry. Many companies are attempting to address this issue, but few understand the intricacies of a media server. MistServer's community and paid LTS versions gather metrics for HTTP content. The LTS version holds detail on 2500 sessions per stream and the community version holds history for 1000 sessions per stream. Maintaining longer statistics requires pulling stored data, as the oldest session history is automatically culled to make way for newer session histories.

Each request is automatically merged into a single set of statistics, no matter how many HTTP connections or HTTP requests are part of a single session. Content from an entire session is recorded from beginning to end, providing accurate and highly granular detail, such as the exact number of bytes transferred in each direction.

In addition, the server injects a session identifier in to all HTTP response headers. This feature helps other analytics system easily parse statistics if MistServer's internal reporting is not used.

All statistics are immediately accessible via the API, meaning there are no log files to parse. In addition, instant accessibility means performance can be judged before a session finishes: all current ongoing sessions are also included in the statistics, updated at least once per second.

In the MistServer community version, the current limitations apply to ongoing statistics:

- Summarized totals of data per second per session are held for up to 600 entries. At one entry per second, the details of user count and up/down bitrate are available for a period of 10 minutes. This is repeated for every available stream.

- VoD stream statistics are erased if there are no connections for more than sixty seconds.

- Completed session information is held for up to 1000 entries. MistServer refers to these as internal logs, but they are more robust, tracking five key areas: total up/down data transfer, total session length, session start time, protocol usage and user's IP address.

- Current session information is held for all current sessions, with no limit. Current sessions are updated once per second to reflect the current state. Once a session completes, the last state is inserted into a completed session log and removed from current session list.

MistServer statistics can be pulled via third-party systems using the API, MistSteward (the multi-server controller), or any custom-designed software. Once session data is pulled it is erased from the server. Since each Connector reports its current state once per second, however, and remains active until a current session is complete, no statistical data is lost.

In-Memory Storage Aggregation

Disk seek and read/write times are a key hurdle in efficient media server design, wasting time for even simple processes. To accelerate data delivery in multiple VoD sessions, content must be stored in memory and a virtual server-side player used to publish content to an end-user.

MistServer uses in-memory storage to serve thousands of VoD sessions with very limited disk access. It also aggregates all versions of a VoD asset into one file, housing all resolutions, formats, audio bitrates, and even disparate codec versions in one container format.

To store all versions of a single media asset together, DDVTech created a new storage format. Before you stop reading at the mention of a new file format, understand that DTSC is for intra-server storage not end-user delivery. One needn't even know DTSC exists to deliver content.

According to DDVTech, the DTSC format internally guarantees all media data is treated equally. While it is theoretically possible to do this in existing container formats, in practice the existing container formats require tradeoffs, especially when storing disparate codec content in one file.

DTSC also allows additional content—like new audio tracks or alternate camera angles—to be appended to the end of a VoD file, even during playback. We'll explore this feature below, but let's first consider how MistServer handles conversion between FLV, ISMV, OGG, and DTSC.

Simplified Media Conversion

To best optimize delivery, DDVTech recommends converting VoD files ahead of time into DTSC storage format, using a supplied converter binary. In fact, the community version of MistServer requires advance conversion to the scalable and processor efficient DTSC format.

The LTS version of MistServer adds a special automated conversion feature: industry-standard VoD file formats recognized by MistServer will automatically convert to DTSC on the fly with no required administrator intervention. This on-demand conversion only converts the requested portion of a VoD file, meaning that a long-form 30-minute VoD file for which a user requests playback partway into the program won't first require conversion of the entire 30 minute file before beginning streaming playback.

Regardless of whether a request is for RTMP or HTTP playback, end users won't be aware that content is being served from the DTSC container format. For instance, if a request for a VoD asset arrives via an HLS Connector, the resulting stream is constructed from all the possible permutations stored in the DTSC file and served from an in-memory server-side HLS player.

Rapid DTSC Conversions

Converting content ahead of time for VoD use is both simple and very fast. Converting from a recognized VoD format like FLV to DTSC takes no more than 8-9 seconds to convert a 2 GB file, including two disk access times since two read/writes occur.

How simple? Using the FLV to DTSC binary (FLV2DTSC) with an FLV file named test.flv, one need only type a single line: *MistFLV2DTSC < test.flv > test.dtsc*

DTSC Merging Increases Throughput

Converting the VoD file into DTSC allows several new format-specific features: append new content, mix existing content, or "defrag" existing content for better disk optimization. All these features are accomplished using the MistDTSCMerge binary, a simple but very powerful mixing tool that optimizes DTSC files for maximum playback flexibility.

MistDTSCMerge can append content to a VoD file while it is still in server memory, and even while it is playing on an end user’s player. Appending is only possible with two DTSC files, as other existing file container formats have no way to append content without completely rewriting the entire file. As such, both the primary DTSC file and the file being appended to the primary file must be converted to the DTSC format prior to appending.

Consider the following example: A Tollywood (south Indian cinema) movie with a Telegu audio track is stored as an FLV and then converted to DTSC format using FLV2DTSC. At time of the first request for the VoD file, the movie is immediately added to in-memory server storage and transferred to an appropriate Connector. This holds true for both converted and unconverted VoD files, with unconverted files being converted while they are forwarded to the Connector.

The immediate transfer helps keep memory and CPU usage low. The connector may itself decide to buffer a certain amount, if needed, but MistServer avoids this as much as possible.

Due to the increasing popularity of the movie in northern India, producers decide to add a Hindi overdub. A separate audio file, containing multiple bitrates of the Hindi audio overdub, is converted into the DTSC storage format and is now ready to be appended or mixed into the primary DTSC file that contains multiple video resolutions and multiple Telegu audio bitrates.

If the primary DTSC file is not being viewed by end users, MistDTSCMerge can mix the Hindi audio content with the primary DTSC content. Mixing the files together to a brand new (third) DTSC file results in the Hindi audio track being interleaved at the proper time points. Mixing eliminates a possibility that the primary DTSC file or overdub DTSC file are modified in any way.

Interleaving reduces hard disk seek times by placing time-aligned content together in storage, as most filesystems attempt to place all of a single file’s content in the same area of the disk.

If the DTSC file is still being played by a number of viewers, however, is it possible to add the Hindi audio overdubs without stopping all current view of the film?

Yes, MistDTSCMerge can append the additional overdub Hindi tracks to the end of the primary DTSC file. The beginning of the original file is never changed, so playback need not stop while the appending occurs. Once content is appended, it is immediately ready for use, and the Connector sends a new manifest file alerting the end-user player of the existence of this new audio option.

One drawback to appending, rather than mixing, is lower disk optimization. Appending content to an existing DTSC file means appended content is not stored as close to its corresponding time-stamped content, so throughput degradation may occur in high-volume environments.

Once the content is appended to the primary DTSC file, however, a "defrag" can be run on the DTSC file, mixing the primary DTSC file into a new file, resulting in a re-sorting of content into proper mixed form with all time-stamped content close at hand. Just like running a defrag on a hard drive optimizes performance, so too a properly mixed DTSC file optimizes performance.

What About Live?

Live content resides temporarily in a Buffer (similar to the binaries called *Players* for VoD content) and does not require disk access at all, unless MistServer has been programmed to save to disk at the same time using DTSC's on-the-fly stream recorder and converter.

A live stream doesn't have an identifiable beginning or end. DDVTech can capture live output to a DTSC file, though, so content can be saved for future VoD use. In addition, DTSC content can play back before the entire file is complete, allowing DVR, catch-up or live pause features.

Here's an example of the simple syntax used to capture the pipe output from an FLV stream via ffmpeg directly to a DTSC file: *ffmpeg options -f flv - | /path/to/MistFLV2DTSC > test.dtsc*

Appending one live content stream to a second is possible with DTSC. Live content captured in to a DTSC file can be merged with a second stream, whether an additional camera angle or an extra real-time audio translation stream.

This live multi-bitrate option is an LTS-only feature. By encoding and publishing multiple streams to the same streamname, MistServer associates the streams together, automatically "merging" the disparate streams together into a single stream that's the equivalent to the mixed VoD files. The multiple stream names can either be multiple resolutions / bitrates of the same stream, additional audio languages, live translated subtitles, a metadata track, or even alternate camera angles.

Finally, as the live DTSC-formatted data is pushed out of the buffer at the end of a session, it can be written to disk, which is then immediately usable as a VoD file. This is possible since all live content is captured, yielding a DTSC file pre-optimized and pre-interleaved, if multiple tracks are involved, for immediate VoD playback.

Conclusion & Next Steps

Almost all of the benefits discussed in this white paper are easy to confirm. The mistserver.org website contains a number of key documents. One compares the full MistServer feature set, another offers a quick overview of how to set up a MistServer, and the site contains even more in-depth information on how to set up the first stream for Linux, Mac and Windows servers.

DDVTech offer three choices of how to obtain the MistServer software: source code access and both paid and community binaries. Both binary options are pre-compiled with dependencies included, as well as a complete set of instructions: community version and paid binaries differ slightly, with the latter labeled as the Long Term Support (LTS) version of MistServer.

The LTS business-license version, with three years support and ongoing patches, is affordable at €299 or approximately $400 USD. LTS customers receive bug fixes for three years for the release they purchased. After a pre-compiled, OS specific binary is downloaded, the user decides whether to install all or just some of the included modules. The only required binary is the main executable, MistController, which then auto-detects what modules have been installed and updates the web user interface, displaying only currently available/installed options.

The third option, source code access, is part of DDVTech's ongoing commitment to the open-source community. Source code can be obtained from the site and contains both libmist and mistserver, as well as details on dependencies.

Visit mistserver.org to try it out, or email DDVTech at info@ddvtech.com or info@mistserver.org.

About the Author

Tim Siglin has been involved in visual communications for over eighteen years, from market analysis and implementation to product launches. He provides consulting services to numerous application providers and network operators, including design services for carrier-grade media asset management systems.

Siglin provides research and consulting services to several Big 5 consulting firms and internal skunk works projects for several Fortune 10 and numerous Fortune 500 clients. He had an MBA with an entrepreneurship emphasis, is a co-founder of Transitions, Inc. and serves as a contributing editor for several tech publications.

About the Company

Transitions, Inc., is a technology and business development firm with extensive experience in design and go-to-market strategy consulting. Transitions specializing in assistance to businesses seeking to identify "transition points" that hinder growth or a return to profitability. Repeat customers account for more than 80% of all on-going business, but Transitions also takes on select project challenges assisting startups, distressed and expanding small businesses. Based in Tennessee, Transitions' business strategy and marketing consulting clients include companies in Silicon Valley, Boston, London, Milan, Mumbai, New York and Switzerland. Current Transitions' projects include established businesses and startups in digital media and global marketing.