Engine Scalability

Paper describing how to scale an Engine installation, and the implications for Control Mechanisms and Redundancy

Engine Architecture - Background

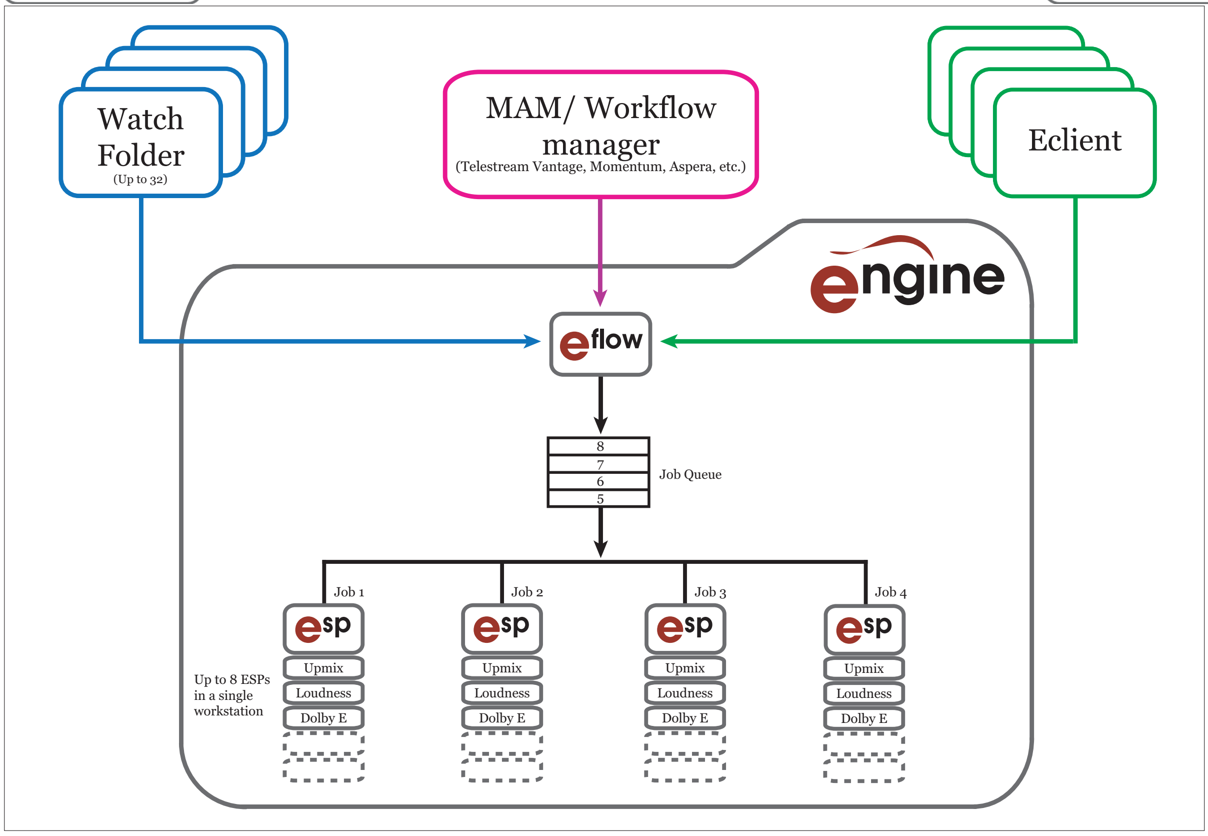
The most visible component in an Engine system is the Engine-Configurator. This tool lets the operator configure nearly every aspect of an Engine system. It is also the place to create workflow profiles, and to process files under manual control using these profiles.

All settings and workflow profiles configured using Engine-Configurator are stored in a range of files.

Completely separately to Engine-Configurator we have a group of components which together constitute the automation portion of Engine. The automation consists of the following –
Eflow
Up to eight ESPs, depending on licence
Eclient

In addition, Engine includes a MessageBus system, which automatically broadcasts status during processing.

The automation components access the settings and workflow profile files that are created by Engine-Configurator. These common files are the only link between the manually operated, configuration tools, and the automation system.



Eflow

Eflow is the central management utility that everything talks to. It manages the watchfolders and reports on their status. It connects, using the published API, to Eclient, and also connects to any third party tools such as Telestream Vantage or Aspera Orchestrator.

Eflow launches ESPs whenever a file is available for processing, and it maintains communications with each ESP, whilst processing is taking place.

Eflow runs as a service on Windows computers, and therefore has an independent user name/login to the desktop user. On Apple Mac OS and Linux it runs as a daemon.

ESPs

The ESP is an application that processes audio found inside media files. The ESP comprises a set of modules chosen at time of purchase from the full range of possibilities, including Loudness, track mapping, Dolby transcode, upmix, and so on.

Each ESP can process a single file at a time. Depending on licence, an Engine system can have between one and eight ESPs. An ESP processes a file under the control of Eflow. When the processing starts, the ESP reads the various settings files, ensuring that it always has up to date information.

The status from each ESP processing is passed to a MessageBus service, and this broadcasts details to any configured listeners.

Eclient

Eclient is our status reporting tool, and also has the ability to submit files for processing. Eclient uses our API, and so third party tools that are used to control Engine have the same potential capabilities as Eclient. EClient listens to the MessageBus broadcasts, so can update its monitoring display in real time.

Engine Control Mechanisms

Having explained the what an Engine system is, we now cover the available control mechanisms.

- Manual control using Engine-Configurator
- Watch Folder automation (which uses ESPs for processing)
- API control from third party or from Eclient (which uses ESPs for processing)

Manual Control

Manual control using Engine-Configurator is very straight forwards. You select a workflow, and choose the source file, and a folder for the destination file, and start processing. There are two important factors to note.

1. Processing is done within Engine-Configurator, so this does not use any of your available ESP resource.
2. File access is done using the user credentials of the user logged in to the desktop, so this user must have access to any network storage that is being used.

Watch Folders

Watch folders are configured by Engine-Configurator, but run entirely under the control of the automation system that is built in to Eflow. Depending on licence, up to 32 watch folders can be used.

1. Processing is done using ESPs. Each ESP runs independently. You can purchase additional ESPs to speed up processing. If one ESP processes a given file in one hour, then eight ESPS could process eight versions of that file in one hour ASSUMING sufficient resources (see later)
2. File access inherits the user credentials of the permissions assigned to the Eflow service, so consideration needs to be given to ensuring this will match the requirements (Windows only)

API Control from Eclient

Eclient connects to the Eflow server using our API. It has the following features –

- ability to display details and status of all watchfolders
- provide summary information on workflows, including displaying the workflow drawing
- open PDF reports, assuming you have chosen to create these, and if you have a PDF reader tool installed.
- Display status on all files that have been processed, are currently being processed, or are queued for processing
- Option to change the priority (from ‘normal’ to ‘do next’ or vice versa) for all files currently queued for processing
- Display status relating to the overall usage of Engine
- Submit new files for processing, using any of the existing workflows.

Every Engine licence includes one Eclient that is installed on to the main Engine server. It is possible to purchase additional Eclient licences that can be installed anywhere on your connected network. The user logged in to Windows needs to have read and write access to all of the network storage being used for your files. The Engine server also needs to have read and write access to the files you wish to process. Normally this implies that it is not possible to have the central Engine server process the files stored on each users’s individual workstation, so instead all files to be processed need to be placed on to the central storage.

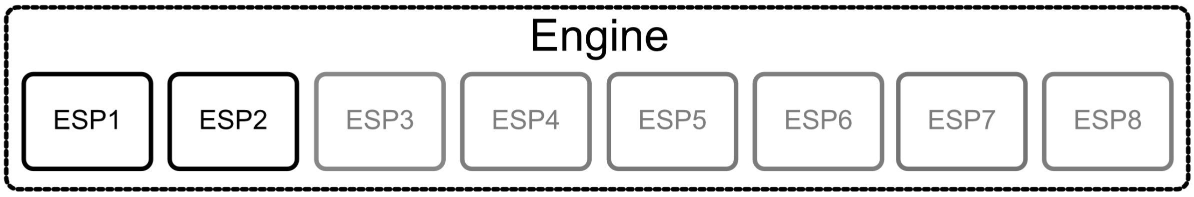
API Control from Third Party Tools

Engine can be controlled from products such as Telestream Vantage, Aspera Orchestrator, and others. Various MAM providers have written integrations for Engine, and they can customise these for particular requirements. You can create an integration to your own workflow tools.

All of the features available over the API and listed in the Eclient section are available to other tools, but different manufacturers may choose to only implement a subset.

Scaling Processing in Single Engine

As noted elsewhere, each Engine licence has between one and eight ESPs, and assuming the overall system has sufficient resources, eight ESPS can achieve eight times as much processing as a single ESP, in the same time. You can purchase Engine with between one and eight ESPS, and you can add more at any time, until the full eight are installed.



Each ESP has the same functionality as every other ESP. So if you have two ESPs with Loudness Compliance and track mapping, then if you need extra ESPs, the new ones would be purchased with the same modules.

We have mentioned that overall processing does scale well as new ESPs are added. However there are a number of factors that impact whether the overall system is limited by insufficient resources.

Processor requirements

Engine products use Intel hardware. Although our applications are multi-threaded, and processor cores are utilised in various ways, the bulk of each ESPs processing will take place on a single CPU core. Therefore it is necessary to always have at least one CPU core available for each ESP that is licensed, plus some additional cores so that normal computer activities can continue running as normal even when the maximum number of ESPs are in use.

We recommend that you should have two more processor cores than the number of ESPs. So, you should have at least a quad core system to run one or two ESPs. To run up to four ESPS you should have a minimum six core system. To run the maximum eight ESPs, you should have a minimum of 10 processor cores in your system.

Memory Requirements
Memory usage is affected by the types of workflows you run, and the duration of the media files being processed. Engine keeps as much data in memory as possible, to maximise processing speeds. When doing loudness correction, this requires keeping every sample of source audio in memory, whilst also creating a memory buffer for all the destination audio that is being created, plus building up the detailed analysis of the audio that is required to provide for high quality processing.

For long form media, depending on workflow, this can use in excess of 24 GB per file being processed. If you are primarily doing long form processing, and doing Loudness compliance on each workflow, we recommend 32 GB of memory PER ESP. The performance will be significantly reduced if there is insufficient memory.

For short form media, 16 GB per ESP is sufficient. If you processing both long and short form, then adjust the total available memory depending on the balance of file types being processed.

Network and Storage Requirements
In all cases, it is necessary to move the entire media from a source location, to the Engine server, and then to the required output location as the destination file is being created. Moving the video data is an important consideration, as for HD, we will need to read and transfer 100Mbits/s of video, whilst at the same time extract only 48 KHz of audio data per channel. For UHD the video is typically 500 Mbits/s but still with the same audio data rate.

Consequently processing UHD files will take significantly longer than HD, even when the audio is identical.

The entire data for a source file has to be transferred across your network to be processed, and then again transferred to the destination folder so as a minimum we have to move double the file size. This volume of data will have to be moved at the same time for each ESP on the system.

You can see that systems with eight ESPS will therefore be move considerable amounts of data across your network, to and from your different storage systems. Generally the limiting factor on overall performance is the speed of the network and the storage. When considering how to scale an Engine system, this is the most important consideration – a faster network leads to faster processing.

Local Workstation Caching

We have already explained that the entire source data must exist on the Engine server at some point, in order to create a new file. Engine can work directly with files stored on the network, but Engine will be moving the audio to the Engine server for processing, plus moving the video data to itself, to as part of creating the destination file.

Measuring Program Loudness requires analysis of the full duration of the audio, then calculating the changes required and applying them. This means that any Program Loudness compliance tool has to make two passes through the audio. If the audio source is on a network location, the default processing mode will therefore require obtaining that data twice.

For these reasons, Engine has a cache mode that can be enabled. This caches the entire source file to a chosen location, then does all analysis and correction locally, then creates the new file before moving it to the configured destination path. Lastly, once Engine verifies that the new file is in place, any required actions are carried out on the source file, such as deleting it, or renaming/moving it.

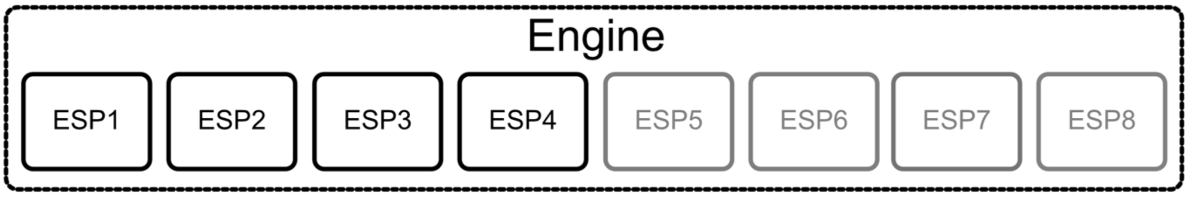
Engine's performance can frequently be maximised by installing a fast disk on to the server, and we would typically recommend something like a 2 TB SSD drive. Alternative methods of providing very fast local storage do exist and can be equally successful. However if the Engine caches to a 5400 RPM spinning disk as an extreme comparison, then processing will be substantially slower.

Note on Temporary Files

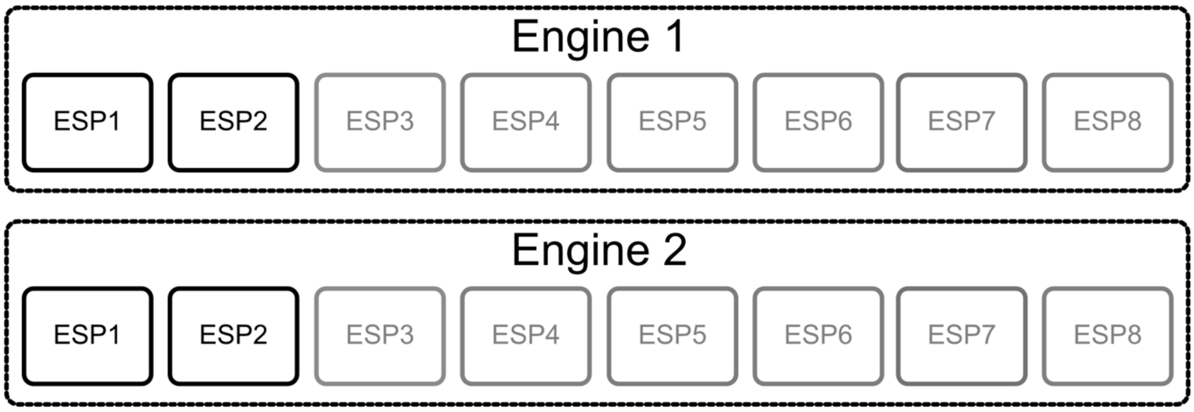
Engine's configurations include mention of Temporary Files. These are audio only files that Engine produces when required during processing. There is no control over whether these files are produced, but it is possible to determine where these are stored. Like caching, we recommend storing these temporary files on fast local storage such as an SSD. For systems with more than four ESPs, reserving one SSD for temp file and one SSD for cache files is likely to be beneficial. For up to four ESPs, the temp files and cache files can share the same SSD.

Scaling Processing with Multiple Engines

Whilst it is possible to have between one and eight ESPs on a single Engine, an alternative scaling strategy is to have multiple Engines, and spread the ESPs amongst these Engines. For example, for a four ESP solution, you could use a single Engine as shown here –



Or you could spread those four ESPs between two Engines, each with two ESPs.



What are the advantages and disadvantages of the two approaches?

Control Implications

Engine contains a job queuing and management system, as part of Eflow. When all of the ESPs are connected to a single Engine, this is a very effective way of spreading jobs between the available ESPs and will ensure no ESP is ever idle, apart from when no files are waiting to be processed.

If the ESPs are spread amongst multiple Engines, then the job queuing and management has to be done by an external tool. If this is not done efficiently, you could have several jobs queued on one Engine, whilst another Engine is idle, with its ESPs not being used.

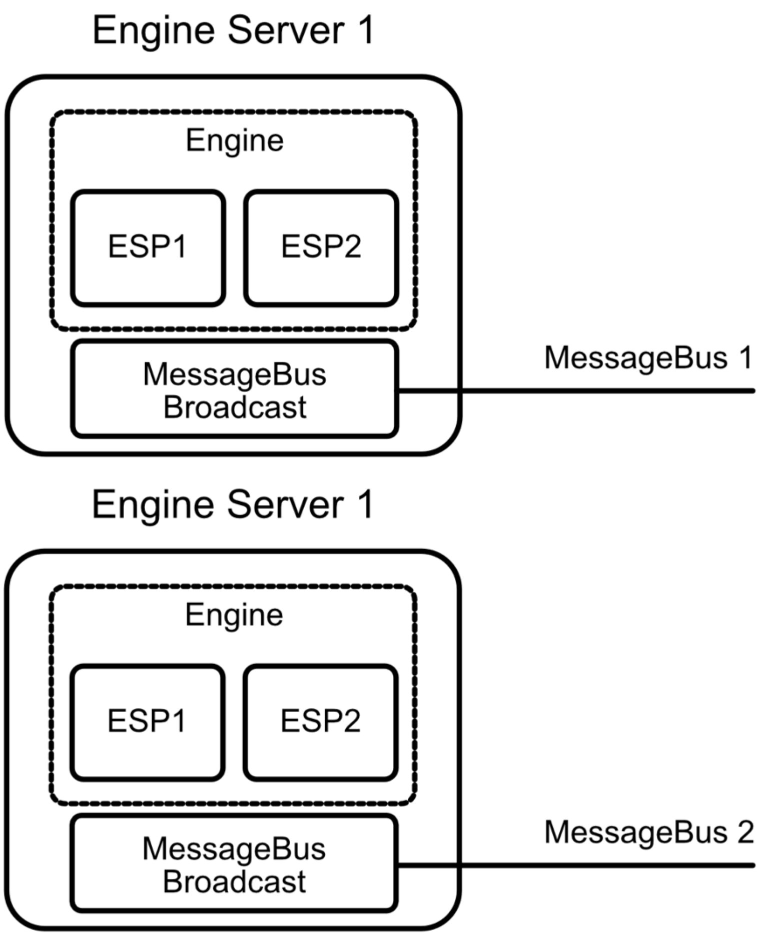
You might also need a separate monitoring system, in place of the supplied Eclient. With all ESPs on a single Engine, you can use Eclient to monitor progress of file processing, to check the status of your watchfolders, to change the priority on any queued job, or to delete a job that is queued or currently processing. If you continue to want that level of control in an system where the ESPs are spread amongst mutiple Engines, then you will need to create a tool yourself.

Consistency Implications

If you choose to spread the ESPs amongst multiple Engines, you need to consider how you will manage to achieve consistent settings across each Engine. Every Engine has an individual set of database files and settings. For a distributed model, you will have to ensure that whenever changes are made, that these changes are replicated to all other systems. There is a risk that processing could continue, but using an old version of a workflow, leading to problematical results. Engine does contain tools to help achieve with this, but it also requires rigour on the part of the operators.

Status Implications

Engine includes a Message Bus system to efficiently broadcast status in real time, with the Eflow API able to be used to get more information on request. When the ESPs are distributed across multiple Engines, each Engine will broadcast using an independent MessageBus.



You could design your own tools to listen to multiple MessageBus broadcasts, but you might need to create a mechanism that aggregates these broadcasts into a single table of information. Alternatively, you could choose to ignore the MessageBus broadcasts and instead collect the required information directly from both Engines, and combine that in to a single result. Either way does require a step up in the level of complexity as compared to using a single Engine.

Advantages of distributing ESPs across Multiple Engines

Whilst there are clear disadvantages to using a distributed model, there are a few advantages, and a more few potential advantages, that are dependent on various factors.

Partial Redundancy and Security Advantages

If the Engine processing resources are spread across multiple licences, these can be hosted on separate hardware. This provides the advantage that a single server hardware failure does not kill your entire processing facility. Even if a single server failed, the remaining Engines will continue processing as normal. This can be a major advantage.

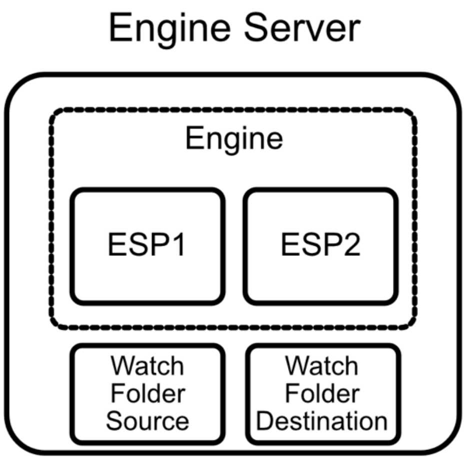
Full Redundancy

This can be turned in to a full redundancy solution by building in excess capacity when everything is running fully, and still leaving sufficient capacity if one server failed. For example, if your throughput required four ESPs, you could consider have three Engines, each with two ESPs. If any one of the three Engines were to fail, then you would still maintain the required capacity, and this would be a cheaper option than having a single Engine with four ESPs in conjunction with a redundant Engine also with four ESPs. Additionally the extra capacity when everything is working normally will help smooth periods of peak demand very easily.

Redundancy Considerations with Watch Folders

Creating redundant configurations where Engine is whole controlled by tools such as a MAM, or Telestream Vantage are quite easy, as third party workflow tools will maintain a complete history of all processing, including what is queued waiting for processing. This means that these external tools can redirect files to different systems in the event of failure of an Engine server.

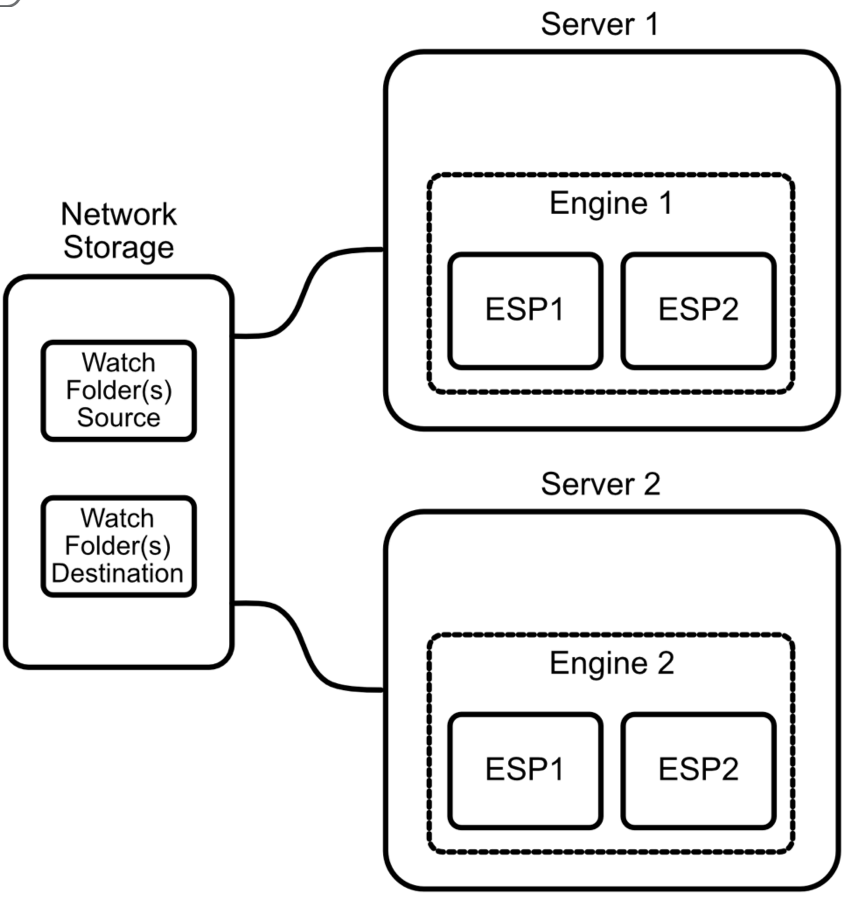
With a pure watch folder system, this requires greater consideration. The drawing below represents a single Engine server, with a couple of ESPs, and running a number of watchfolders, with the physical location of the watchfolders being on the Engine server.



If you only have a single Engine, this configuration will work perfectly successfully. And since all of the data ultimately has to be processed on the Engine server, this will be the most efficient configuration.

However, imagine that this is extended to a fully redundant system, so one with a Main Engine, plus an identical Backup, which is normally idle. A number of watch folders are configured to exist on the disk inside the Main server. If that server should fail, you have lost the details of which files are in that watch folder.

For this scenario you should consider having the watch folder locations on network storage somewhere. In the example drawing below, we have arrange two Engine servers, each with two ESPs. All of the source and destination watchfolders are now on network locations that can be accessed by both servers.



We would configure both Engines to automatically delete files from the watch folders once new files have been created and verified. Then if the main server fails, every file that was being processed at that time, or was queued, still remains in the original watch folders, and can be processed by the backup server with no loss of service or performance. Crucially, no files will be missed from the processing queue.

With the two Engine systems, you would configure both to have an identical list of watch folders, but only have the folders active on the Main Engine. At time of failure, you would need to manually change the settings on the backup Engine to make the watch folders active, and once this is done, you can be certain that no files are lost or missing from your watchfolders.

If you have multiple Engines each with some ESPs, then you will have to split the active watchfolders amongst all of the Engines. So with two Engines, each could have half of the watchfolders. For a fully redundant solution, we would suggest configuring all of the watchfolders on both Engines, but only half of the watchfolders being active on each system. In the event of a failure, you would need to manually activate the previously inactive watchfolders on the server that is still running. Again you need to ensure that the watchfolder locations are on network storage rather than on the Engine servers.

Potential Processing Speed advantages

There is an additional possible advantage of distributing your processing, relating to your stoage speeds and available network bandwidth. When all of the processing takes place on a single node, the entire data of the files is passed through that single point. If this results in the network clogging, you may see some benefit in splitting that amount of network traffic around a few parallel paths.

You could also split you network traffic by careful use of switches and arrangement of your storage, although whether this flexibility exists depends on factors relating to your storage and processing requirements.