Configuration guidelines

Scaling QlikView Publisher

What to take into account when scaling your Publisher

TABLE OF CONTENTS

SUMMARY

- These configuration guidelines are intended for anyone wishing to understand what to factor in when designing a QlikView® Publisher cluster.

INTRODUCTION

QlikView Publisher is the Reload & Distribution engine in a QlikView deployment. Using QlikView Publisher, you can automate the generation of data stores to be consumed, the reload of fresh data and the distribution of QlikView documents via email or to a specific QlikView Server or directory folder. In the QlikView Management Console, the Publisher is referred to as the QlikView Distribution Service, which is also the name of the Windows service managing the role. As a service, it can be clustered over multiple Windows server nodes.

What is a QlikView Publisher Cluster?

A QlikView Publisher cluster can be defined as two or more nodes that all run the QlikView Distribution Service and are set up to share the load of the tasks configured in the same environment. The nodes in the cluster share the same root folder, the “ApplicationData” folder as defined in the QlikView Management Console. Any task that needs to run in the environment is sent to the node with the most resources available to run the task, or to a specific node if Publisher Groups have been configured. The cluster nodes all share the same root folder, which also contains the Notification System. The nodes also share the same set of Source Documents as they all need access to the documents in order to run the tasks.

Horizontal or Vertical Scaling

QlikView Publisher can utilize either horizontal or vertical scaling. The benefits of either variant are explained below. With Publisher Groups you can do both – use different node sizes in the same environment and allow the nodes to run with different configuration parameters.

Horizontal Scaling

When horizontally scaling your QlikView Publisher cluster, you add more nodes of the same type to the cluster to increase the throughput of tasks or reduce the time needed for a full reload window. The maximum amount of cluster nodes is defined by your license and is referenced in the "NUMBER_OF_XS;<x>;;" tag (where <x> is the maximum number of nodes in your cluster) in the license file. The benefits of horizontal scaling are a higher level of redundancy and a fairly easy way of scaling, as more nodes can be put into place quickly to immediately assist with the workload. The potential downside is a higher load on the files in the "ApplicationData" folder, which are used to keep the cluster in sync.

Horizontal vs. Vertical

Horizontal scaling: Adding more nodes, of the same spec, to the cluster

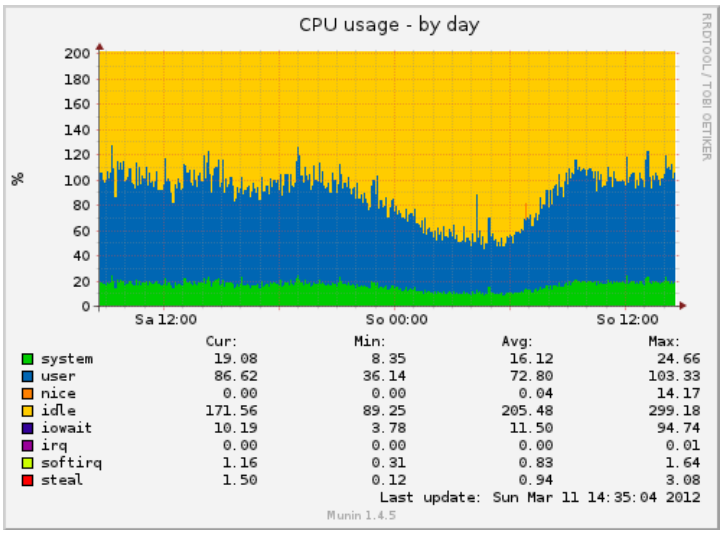
Vertical scaling: Increasing the size of individual nodes in the cluster, in terms of higher CPU speed and more memory

Vertical Scaling

When vertically scaling your QlikView Publisher cluster, you increase the size of the nodes in the cluster instead of adding more nodes. By adding primarily CPU resources to the nodes you can either have your tasks run faster or run more tasks in parallel per node. The benefit of this is that you do not need to add more nodes, which translates to not increasing license costs. The potential downside is that your data sources may no longer be able to deliver data at the rate that the cluster node can handle it, due to limitations at driver or data source level. This also requires tuning of Windows so that many processes of the same type can run in parallel. This is related to the Desktop Heap Space for non-interactive services.

Identifying Your Bottlenecks

As with any other backend software, QlikView Publisher is designed to utilize as much of the resources (CPU, memory, network) as possible to meet demand. The resources consumed by a task are returned to the system upon task completion. As the system is designed to run multiple tasks in parallel some tasks might end up competing for the same resources. The resources needed to run a task can vary greatly and depending on how the reload script is constructed, how quickly the data can be pulled into QlikView and how much data is extracted. For example; extracting a large set of data might mean the task primarily consumes RAM resources, while a task that performs lots of data aggregation at script level might consume lots of CPU resources. Whereas a task that extracts data from a slow data source, such as web files, might consume a small amount of resources on the Publisher machine, but run for a very long time. Factors such as these should be considerations when identifying bottlenecks and finding ways to overcome them.



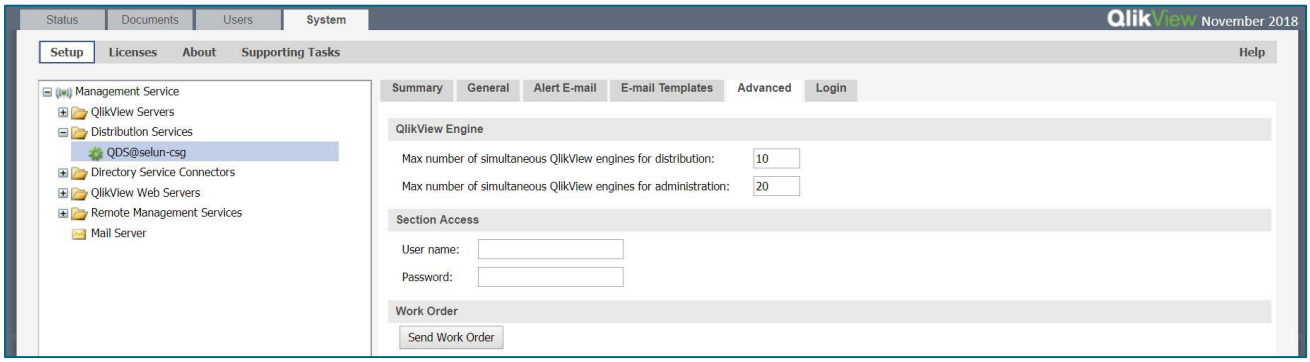
CPU Exhaustion

As mentioned, each task attempts to use as much of the resources as possible. Therefore a, a task could conceivably consume CPU resources from all available CPU cores in the server, even in a very large server. Early versions of QlikView had a 1:1 relationship between a task and a CPU core, meaning you could only run as many tasks as your server had CPU cores. While this limitation has long since been removed, it remains a best practice to not allow more tasks to run in parallel than you have CPU cores (-1). That said, more than one task can normally safely run in parallel. The tasks will combine to use the resources available in the best way possible and, provided the server does not run at 100% for extended periods of time, all tasks should run successfully.

CPU resources can produce a bottleneck when the server runs at 100% for long periods of time. In this scenario tasks may take longer to complete, fail to start, or start, but the Reload engine fails to open the Source Document to begin the script execution. You may also see tasks with incorrect status in the QlikView Management Console, as the Management service asks the Distribution service for the status and the Distribution service might not respond in a timely fashion. Windows instability may also be experienced as the operating system needs some resources to function

properly. Any given QlikView deployment has its own sweet-spot for the number of tasks that can run in parallel. Typically based on our scalability tests, running more than 14 - 16 tasks in parallel, even on large servers, can see the failure rate increase.

The configuration value set in the QlikView Management Console, “Max number of simultaneous QlikView engines for distribution,” applies to tasks per node in the cluster. For example, setting it to “10” in a 3-node cluster means that you can run a maximum of 30 tasks in parallel.



Memory Exhaustion

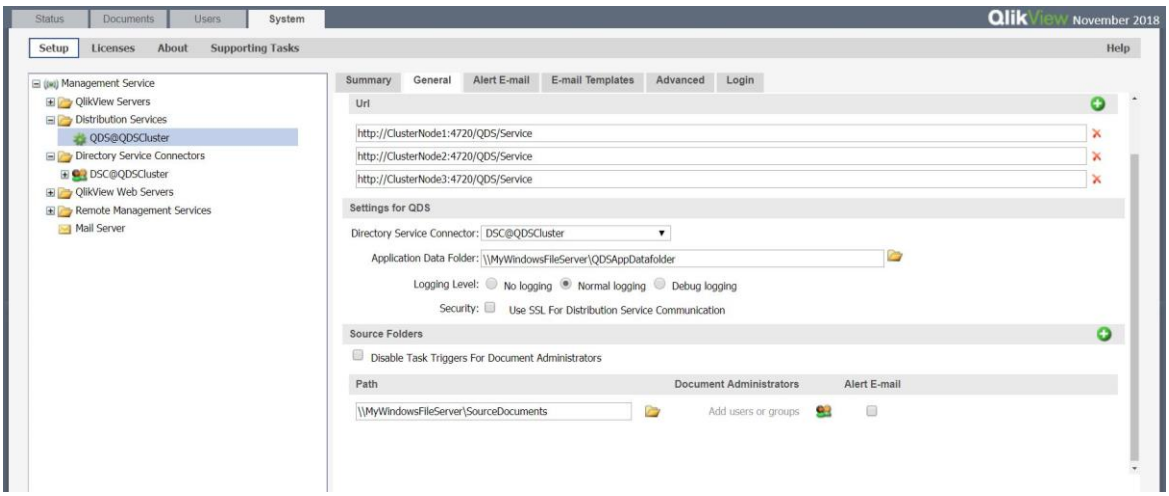
Compared to CPU exhaustion, memory exhaustion is not as common in QlikView Publisher. The amount of memory needed for an individual task largely depends on the document design. For example, the data model, the document size and the volume of data extracted into the document.

Disk Access Latency

The least obvious bottleneck in a QlikView Publisher cluster is disk access latency and the number of read and write operations. This is because two folders, “ApplicationData” and “Source Document,” must be available as network-accessible file shares for a Publisher cluster to function properly. Note that QlikView only supports Windows-based file shares, i.e. folders shared from a Windows file server.

ApplicationData Folder

The “ApplicationData” folder is configured in the QlikView Management Console.



The “ApplicationData” folder functions as the root folder for the Publisher cluster. It allows the cluster nodes to share information related to the running of tasks, as well as all the logging done by the cluster. While each node has a local copy of the Workorder held in the QlikView Distribution Service process cache and the full list of all tasks configured in the environment, each node needs to check which node should run a task when it is triggered. This is done by the QlikView Distribution Service, which constantly scans the files in the “ApplicationData” folder to keep track of which tasks are running and the load on each node and records the outcome of tasks and resets triggers so they reflect already executed tasks as well as the next instance for that task.

Each QlikView Distribution Service constantly scans the files in the “ApplicationData” folder, and also updates the files so that all the other nodes are notified and share the same understanding of reality. This normally happens many times per second. When disk access latency becomes an issue you may notice symptoms such as, tasks are not triggered when they should, the QlikView Management Console does not display the correct task status or the next start time for a task or the nodes are unable to accept new tasks to be run.

If the QlikView Distribution Service process has failed to reach the files in the Notification System for more than 10 minutes then log files, with the name “Cluster_date.txt,” will be created in the “ApplicationData” subfolder for each QlikView Distribution Service node. This is an indication of disk access latency which can occur when there is a disk I/O request queue of more than 2 - 3 requests over time.

Source Document Folder

The “Source Document” folder also needs to be shared, so that each node has equal access to the QlikView documents it needs to perform actions on. The documents are loaded from and saved to this folder when a reload task takes place.

An example of high disk latency, or a bottleneck, occurring anywhere between the server running the task and the file share would be long load times for documents prior to the actual script execution (which are recorded in the task log).

Best Practices

Distribution of Services

In a QlikView Publisher cluster, you distribute the QlikView Distribution Service across multiple nodes. To maximize throughput and resiliency, the following best practices apply.

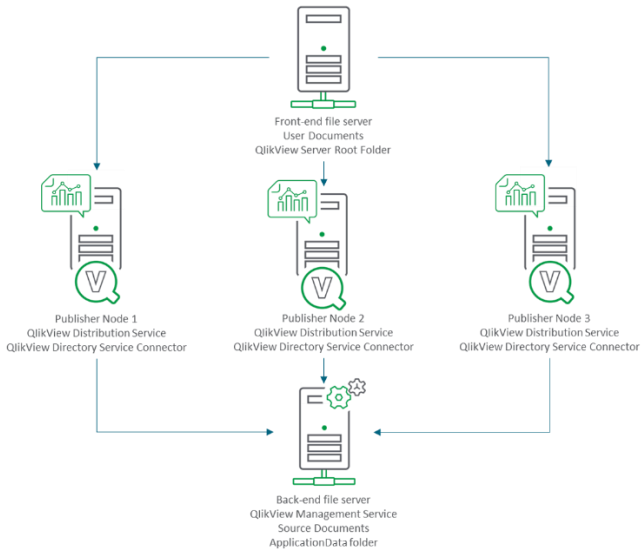
Dedicated Host

Qlik recommends running a QlikView Publisher node with as few other resource-consuming services as possible. Ideally, a Publisher node should only run the QlikView Distribution Service and the QlikView Directory Service Connector. This configuration allows the Publisher to consume as much of the resources as possible without affecting or being affected by anything else on the same server. void running the QlikView Management Service, which hosts the QlikView Management Console, on the same server as the Publisher services.

File Server

The Windows file server hosting the “ApplicationData” folder should be a separate and dedicated server. Any other service with heavy resource consumption running on this server will affect the file accessibility and potentially impact the Publisher cluster. However, if needed, the same server could be used to host the QlikView Management Service as this service has a small resource footprint.

The ideal hard drive configuration for the file server is to separate the shares hosting the “ApplicationData” folder and the “Source Document” folder onto individual disks. Preferably, a third small and fast hard drive (SSD ideally) should be used to host the Notification System to ensure immediate access to these files. This normally becomes a factor in larger clusters where 3 or more nodes update the files constantly. In a large-scale deployment, the file server should be dedicated to the Publisher file access and not host any of the files used by the QlikView Server front-end.



Optimizing for Stability and Performance

How Many Reload Engines (QVB) to Allow in an Environment

By default, QlikView Publisher sets “Max number of QlikView engines to use for distribution” to 4. This value determines the number of QlikView Publisher tasks that can run in parallel and can be adjusted to suit any environment. The number is per cluster node, so a setting of 8 in a 3-node cluster allows a maximum of 24 tasks in parallel.

When determining how many Reload engines to allow per QlikView Publisher, the following factors should be considered:

Resource Availability

The first and fundamental factor that affects the number of reloads that can safely take place simultaneously is that the Reload engine, the QVB.exe process, is a fully multi-threaded process. This means that a single executing task using a Reload engine will try to use as much of the resources from the hardware as possible, without limit. The limiting factors are only script efficiency and data transfer rates. For example, a reload script that performs lots of data aggregation at script level will have a large impact on the CPU utilization for that reload task. Tasks that run in parallel on a server compete for the same pool of resources containing CPU cycles and RAM.

With this fundamental factor in mind, we can start discussing the possible settings.

CPU Cores -1

The first consideration is that you should never allow for more Reload engines than the server has CPU cores -1. E.g. For a server with 8 CPU cores, maximum Reload engines would be 7

Start Low and Increase until Finding Your Limits

The second consideration is that even on big servers with large number of CPU cores (16+) it rarely pays off to go higher than approximately 13 simultaneous reload engines allowed. This is partly due to the first fundamental factor (every running task tries to consume all resources) and partly due to other bottlenecks (such as wheremultiple data connections using the same database driver become slower as the number of concurrent connections increase). Going for a high number also often increases the failure rate significantly. Qlik recommends starting with a low number and then increase gradually until you hit the sweet-spot for your environment, where you get a high throughput without introducing a high failure rate. If your reload schedule calls for a large number of tasks to run simultaneously, you will be better off clustering the Publisher role over more servers than increasing the number of engines on a single node.

DeskTop Heap Size

A third consideration is the Windows settings that affect the number of concurrent processes of the same type. If you enable a large number of Reload engines, you might encounter errors related to Windows running out of DeskTop Heap Size. This will be visible in the Windows System Event log, with errors like “DeskTop Heap exhausted”. If so, you need to adjust the Windows DeskTop Heap Size in the windows registry to allow for a higher server load.

IMPORTANT: Windows registry changes are made at your own risk and are not supported by Qlik. Please make a backup of the Windows registry before applying changes to enable rollback if the setting change fails or introduces Windows instability.

The DeskTop Memory Heap size is controlled by the following Windows registry key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\SubSystems\Windows
```

The heap size is specified by SharedSection=xxxx,yyyy,zzzz, where:

- xxxx: The shared heap size (in kilobytes) common to all desktops. This memory is not a desktop heap allocation, and the value should not be modified to address desktop heap problems.
- yyyy: The size of each desktop heap.
- zzzz: The size of the desktop heap for each desktop associated with a “non-interactive” window station.

Change the DeskTop Heap Size by setting the SharedSection value to 1024,20480,2048. After the change the registry key should look something like the following (all on one line):

```
%SystemRoot%\system32\csrss.exe
ObjectDirectory=\WindowsSharedSection=1024,20480,2048
Windows=OnSubSystemType=WindowsServerDll=basesrv,1ServerDll=winsrv:UserSe
rverDllInitialization,3ServerDll=winsrv:ConServerDllInitialization,2
ProfileControl=OffMaxRequestThreads=16
```

Depending on the size of your server, you might be able to increase the non-interactive section even higher, so the next values would be 1024,20480,4096, 1024,20480,6144 or 1024,20480,8192 (the number of CPU cores is the determining factor for how high you can go).

Note that the default value is 1024,3072,512 in x86 and 1024,20480,768 in x64 environments.

QlikView Distribution Service .config File

While many of the parameters governing how the QlikView Distribution Service runs are defined in the QlikView Management Console, some of the settings can only be adjusted by altering the .config file for it. This section outlines which settings are relevant for performance optimization and why.

The .config file is individual to each Publisher node, and therefore must be changed for each node. By default, the file is stored in the following location:
C:\Program Files\QlikView\Distribution Service\QVDistributionService.exe.config

The first two parameters of interest are:

```
<!-- ************ Task Result Cache Update ************ -->
<!-- Max number of tasks per thread updating the task result cache. Value
less than one (1) enabled automatic calibration. -->
<add key="TaskResultCacheUpdateMaxTasksPerThread" value="0"/>
<!-- Time to pause between task result cache updates, in milliseconds.-->
<add key="TaskResultCacheUpdateSleepTimeBetweenUpdates" value="200"/>
```

The first one, “TaskResultCacheUpdateMaxTasksPerThread,” defaults to “0”, which allows for automatic calibration of the number of tasks sent per thread towards the Task Results file. In environments with 300 or more tasks in total, it is recommended to disable automatic calibration as otherwise the QlikView Distribution Service process will spawn a lot of threads when trying to update the files, which in turn generates a higher than desirable load on the underlying file system. Setting it to a value that is higher than the total amount of tasks configured in the system forces the

QlikView Distribution Service to only use one thread. If the QlikView Distribution Service spawns many threads, it leads to the QlikView Distribution Service process consuming more CPU resources than it strictly needs to. This is visible in the Windows task manager by observing the CPU consumption for the QlikView Distribution Service process.

The second one, “TaskResultCacheUpdateSleepTimeBetweenUpdates,” determines how often the QlikView Distribution Service process scans the file system for updates. The default is either 200 or 1000 depending on the QlikView version. If your current default is 200, the QlikView Distribution Service looks for changes 5 times per second, which adds load to the file system. Adjusting it to 2000 or 4000 significantly reduces the load, primarily in systems with many tasks and executions per day.

In addition, the following parameter might be worth adjusting in high load environments:

```
<!-- The number of days to keep EDX Result Files -->
<add key="NbrOfDaysToKeepEDXResultFiles" value="30"/>
```

Reducing the value from the default 30 days lowers the number of files that the QlikView Distribution Service process needs to scan and keep track of, allowing it to better keep track of current events rather than historical data.

In addition to the parameters mentioned above, there are a few more that can be added in the .config file (as they are not present by default).

```
<!-- Purges deleted task and it's triggers and results. Once per day.
Should only be enabled for one QDS node in a cluster. -->
<add key="EnableAdvancedFilePurge" value="true"></add>
```

If set to “true,” the QlikView Distribution Service (QDS) process is more aggressive when cleaning out historical records from the Trigger files primarily. This reduces the size of the files, which allows for improved throughput in terms of faster opening, writing and closing of the files. This is more preeminent on tasks with frequent runs (many times per day, for example). It can be diagnosed by sorting the contents of the “TaskResults” subfolder in the “ApplicationData” folder by size. The TaskResult XML files should be small, less than 50 Kb. File sizes of hundreds of Kb or even a few Mb represent a challenge for the QDS process as a larger file takes longer to lock, open and read.

```
<!-- Alternate Path for Notification System, UNC path, needs to be
identical on all nodes in a cluster -->
<add key="NotificationFolder" value="Your folder Path"/>
```

This parameter allows the Notification System to be separated out from the rest of the “ApplicationData” folder. Ideally, these files should be placed on a small but very fast shared disk (SSD preferably). The reason for this is that while the files in the Notification System are very small, they are accessed extremely often, and when the number of Publisher nodes increases, the load on the files multiplies.

```
<!-- Alternate Path for Temporary files used by Publisher during
distribution. Should be a local drive on each node.  -->
<add key="QdsTempFolder" value="Your folder Path"/>
```

This parameter allows you to control where the Reload engine stores the temporary copy of the document to distribute. By default, the copy is stored in the “ApplicationData” folder. Moving it to the local drive on the server running the reload and distribution task relieves the “ApplicationData” folder from a lot of disk accesses, as the User documents can be large and all nodes in the cluster transfer lots of data into that folder. Note that this setting is only available from QlikView November 2017 (12.20) SR8, and from QlikView November 2018 (12.30) SR1.

Publisher Groups

The Publisher Groups feature was introduced in QlikView 12.10. A Publisher Group is a subset of a QDS cluster. Each Publisher Group includes one or more QDS nodes and is given a unique name. A QDS node may exist in any number of Publisher Groups (zero or more).

Each task is assigned to none or one of the Publisher Groups. A task assigned to a Publisher Group is called a Dedicated Task and may only be executed by one of the QDS nodes included in the group. A task not assigned to any Publisher Group is called a Regular Task and may be executed by any of the QDS nodes.

The benefits of enabling Publisher Groups include setting different amounts of allowed Reload engines for differently sized servers within the same cluster and assigning tasks to a certain set of servers. You can also allow tasks to be prioritized above others and run with more resources if they are time-critical, for example.

https://help.qlik.com/en-US/qlikview/November2018/Subsystems/Server/Content/QV_Server/QlikView-Server/QVSRM_Clustering_DistributionService.htm

Overload Protection

With the introduction of QlikView 12.10, QlikView Publisher added an Overload Protection mechanism. It prevents more tasks from being started if the Publisher server is already running at full capacity, regardless whether or not there are available Reload engines. Any task that is triggered but prevented from immediately running due to either Overload Protection or there being no free Reload engines enters a “Queued” state. It is then run when the servers are no longer overloaded, or when a free engine becomes available. By default, the Overload Protection prevents more tasks from being launched when the CPU load is at 75% or the memory consumption is at 90%. These values can be edited as needed in the .config file for the Distribution Service.

```
    <!-- If the CPU usage % goes above this value the machine is
considered overloaded and no new tasks will be started. -->
    <add key="CpuOverloadLimit" value="75"/>
    <!-- If the memory usage % goes above this value the machine is
considered overloaded and no new tasks will be started. -->
    <add key="MemoryOverloadLimit" value="90"/>
```

Summary

QlikView Publisher is a powerful component in the QlikView family and, when properly configured, can maintain a very high throughput of tasks. In order to scale to very large deployments, it is recommended to take these configuration guidelines into consideration.

In practical experience, if all nodes are heavily loaded and using all available resources, then the maximum number of nodes is typically three. Above that, things can become erratic even given the best of circumstances. With a lower load per node, more nodes can potentially be added, ultimately the maximum number of nodes comes down to the individual load pattern of each environment.

Qlik LEAD WITH DATA

About Qlik

Qlik is on a mission to create a data-literate world, where everyone can use data to solve their most challenging problems. Only Qlik's end-to-end data management and analytics platform brings together all of an organization's data from any source, enabling people at any skill level to use their curiosity to uncover new insights. Companies use Qlik producs to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 48,000 customers around the world.

qlik.com