Show Table of Contents
Home > Administrator Guide > Performance > **Improve Server Performance**

Improve Server Performance

Use the topics below for guidance on how to improve the performance of deployments that are extract-intensive, user-intensive, or both:

What's your goal?

Optimizing for Extracts

The data engine process stores extracts and answers queries; the background process refreshes extracts. Because both are demanding of CPU resources, the best approach to improving performance for an extract-intensive deployment is to isolate these two processes from one another, and from the other server processes. This may take three machines. If you don't have three machines to work with, there are still strategies you can use (see the deployment examples below).

Optimizing for Users and Viewing

The VizQL server process handles loading and rendering views for Tableau Server users. If you are trying to optimize your deployment for a high number of users and a lot of view interaction, this is the process you should focus on. If you can run additional VizQL Server processes on additional machines instead of on the same machine, do so. You'll see better performance since the VizQL Server processes will have fewer processes to compete with for system resources.

How Many Processes to Run

In general, more processes means more processing power and better performance, but too many can overwhelm your system or even decrease server performance. You can run up to 8 instances each of the VizQL Server, application server, data server, or background processes on any given machine—but it's likely that 8 aren't necessary (learn more). As you consider how many instances of each process to run, pay special attention to the VizQL Server process and the background process.

VizQL Server Process

The VizQL Server process is multi-threaded and multi-processed on a machine. There can be over 16 threads running on a single

machine.

Caching is something to keep in mind with the VizQL Server process. Each instance has its own cache, and caching affects server performance. This is because a cached view loads faster than a view that's being requested for the first time. Too many unique caches makes it less likely that a request will be handled by a process that already has the result in its cache. As you look at the suggested VizQL ranges in this topic, start with the low end of the range and see how views perform before you increase to the maximum number.

Here are some general, conservative guidelines for determining the minimum and maximum number of VizQL Server processes to run:

- **Minimum number per deployment**: 1 VizQL Server process for every 100 concurrent viewers using your deployment.
- **Maximum number per machine**: The machine's RAM divided by 4 (64-bit) or 2 (32-bit), with an upper limit of 8 per machine.

  Note: The deployment examples later in this topic may use slightly different formulas to address specific performance goals.

As an example, a single 64-bit machine with 32 GB of RAM that's handling 300 concurrent viewers should run a minimum of 3 VizQL Server processes and a maximum of 8. If you have two such machines to work with, you should have at least 3 VizQL Server processes across both machines, with each machine being a capable of running up to 8 VizQL Server processes each. A single 32-bit machine with 8 GB of RAM that's handling 200 concurrent viewers can run 2-4 VizQL Server processes. If this same machine is handling 500 concurrent viewers, it's probably time to add more hardware.

Background Process

A single background process can consume 100% of a single CPU core, and sometimes even more for certain tasks. As a result of this, the total number of instances you should run depends on the machine's available cores—as well as on what you're trying to improve. The deployment examples below uses N to represent the machine's total number of cores, and each suggests a different strategy where the background process is concerned. When in doubt, start with the low end of the suggested range and assess performance before increasing the number.

Data Engine and Repository Processes

There are scenarios where the data engine process should be isolated on its own node—such as if you are trying to improve an extract-intensive deployment and you want to emphasize querying more than extract refreshes. The deployment examples provide specifics. Because the data engine stores real-time data, transferring it is a multi-phased procedure. Move the Data Engine and Repository Processes describes how to do it.

Another reason to isolate the data engine (and/or the repository) is to minimize your deployment's potential for downtime. See High Availability for details. Unless you're configuring for high availability, the repository can usually remain on the primary Tableau Server.

Where to Configure Processes

You configure the type and number of processes any machine is running using the Tableau Server Configuration dialog box. If you are adding new machines as part of your reconfiguration, they must already have Tableau Worker software installed on them. Refer to Install and Configure Worker Servers for steps.

If you are reconfiguring the processes on your primary or standalone Tableau Server, see Reconfigure Processes.

Optimizing the Extracts and Workbooks

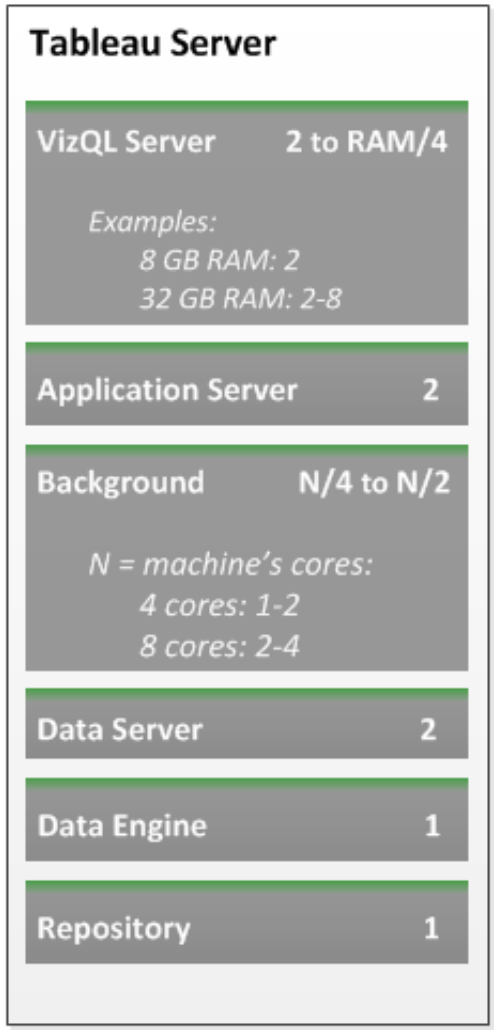
Fast server performance with extracts is partly a function of the extracts and workbooks themselves. Workbook authors can help improve server performance by keeping the extract's data set short, through filtering or aggregating, and narrow, by hiding unused fields. Use the Tableau Desktop options **Hide All Unused Fields** and **Aggregate data for visible dimensions** to do this. For steps, see Creating an Extract (Tableau Desktop help). For general tips on building well-performing workbooks, search for *"performance"* in the Tableau Desktop help. To see how workbooks perform after they've been published to Tableau Server you can create a performance recording. See Create a Performance Recording for details.

Assessing View Responsiveness

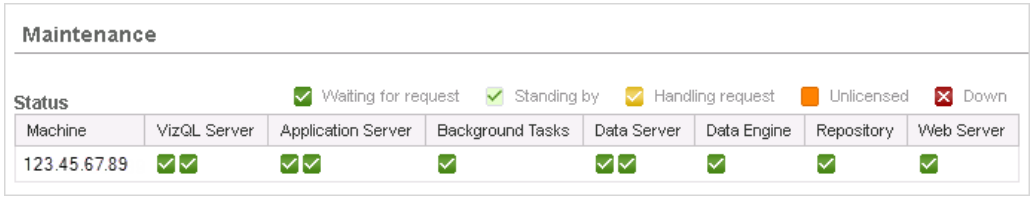
When a user opens a view, the components of the view are first retrieved and interpreted, then displayed in the user's web browser. For most views, the display rendering phase occurs in the user's web browser and in most cases, this yields the fastest results and highest level of interactive responsiveness. Handling most interactions in the client web browser reduces bandwidth and eliminates round-trip request latencies. If a view is very complex, Tableau Server handles the rendering phase on the server instead of in the client web browser—because that generally results in the best performance. If you find that views aren't as responsive as you'd like, you can test and change the threshold that causes views to be rendered by the server instead of in the client web browser. See About Client-Side Rendering for more information.

One-Machine Example: Extracts

A Tableau Server installation with heavy extract usage can run on a single 64-bit machine configured as follows:



The above configuration would look like the following on the Tableau Server Maintenance page:



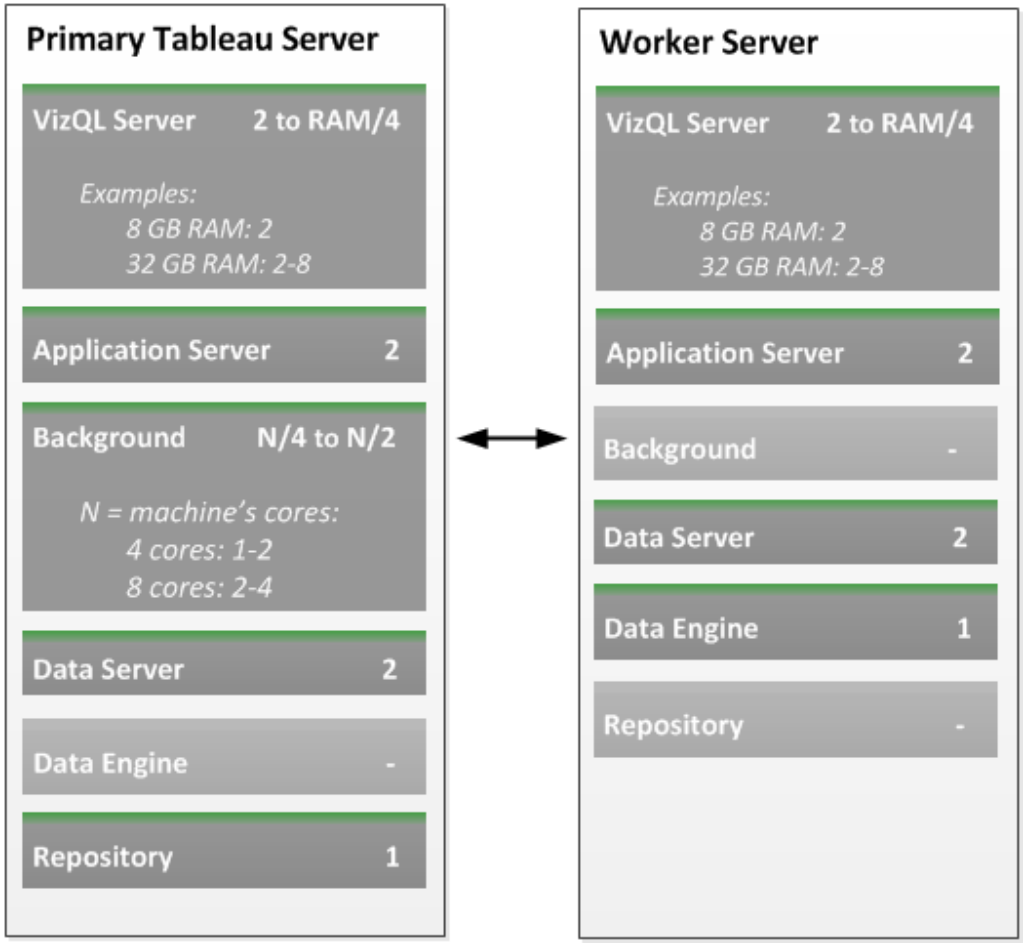
Configuration Notes:

- Run at least 2 VizQL server processes and determine the maximum number to run by dividing the machine's RAM by 4.

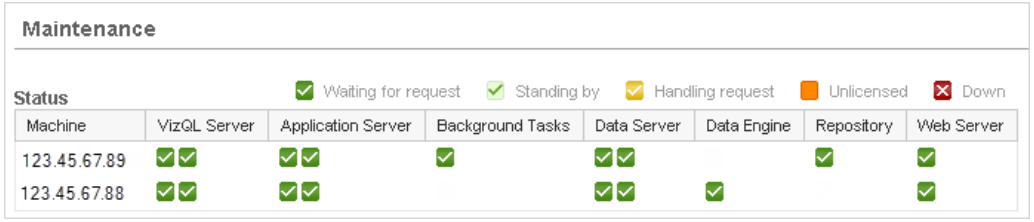
- Calculate the least number of background processes to run by taking the machine's total number of cores and divide it by 4. To determine the maximum number, divide by 2.
- Both the background and data engine processes are CPU-intensive and the above configuration balances them.
- Scheduling extract refreshes for off-peak times helps the data engine and background processes not compete with one another for system resources.
- Because you are not trying to optimize for a high number of concurrent users here, start off with 2 VizQL Server processes and observe how views perform before you increase the number.

Two-Machine Example: Extracts

Here's how you can configure a two-machine Tableau Server deployment so that it can handle heavy extract usage. The most important thing to note in this example is that the data engine process is isolated from the background processes. Both servers are running on 64-bit operating systems:



With the above configuration, the Status table on the Maintenance page would look like this:



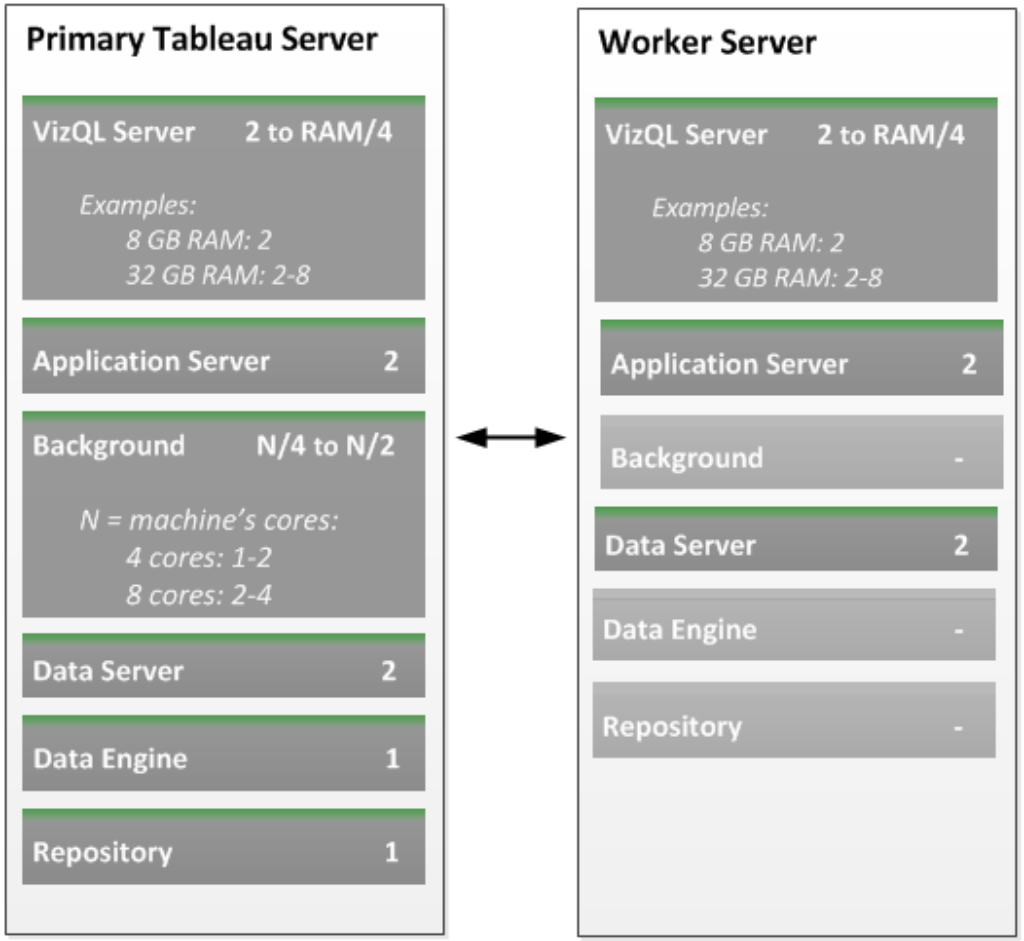
Configuration Notes:

- Once you go from a one- to two-machine deployment, your first server becomes the primary Tableau Server. In the Status table it gets a value of Gateway.

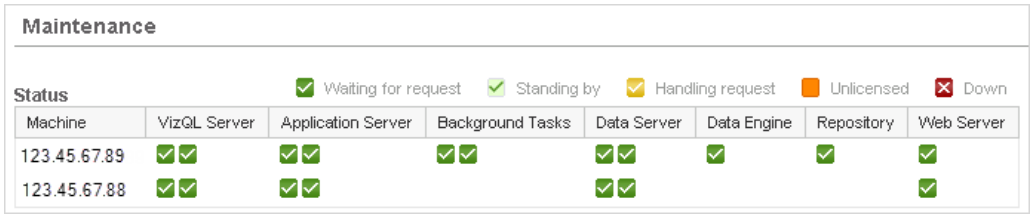
- Run at least 2 VizQL server processes on each machine and determine the maximum number to run by dividing the machine's RAM by 4.
- To figure out the minimum number of background processes to run on the primary Tableau Server, take the machine's total number of cores and divide it by 4. For the maximum number, divide by 2.
- Moving the data engine from the primary Tableau Server to the worker is a multi-phased procedure. See Move the Data Engine and Repository Processes for steps.

Two-Machine Example: Viewing

A two-machine deployment with light extract usage and heavier viewing can be configured as follows:



The Status table for the above configuration would look like this:



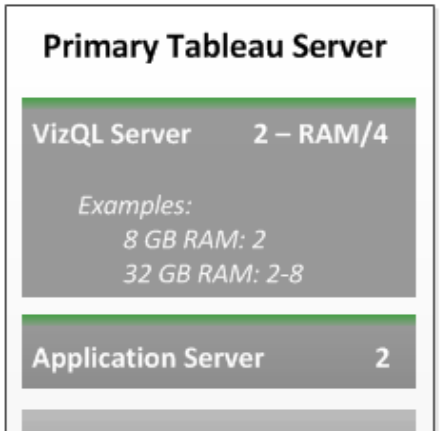
Configuration Notes:

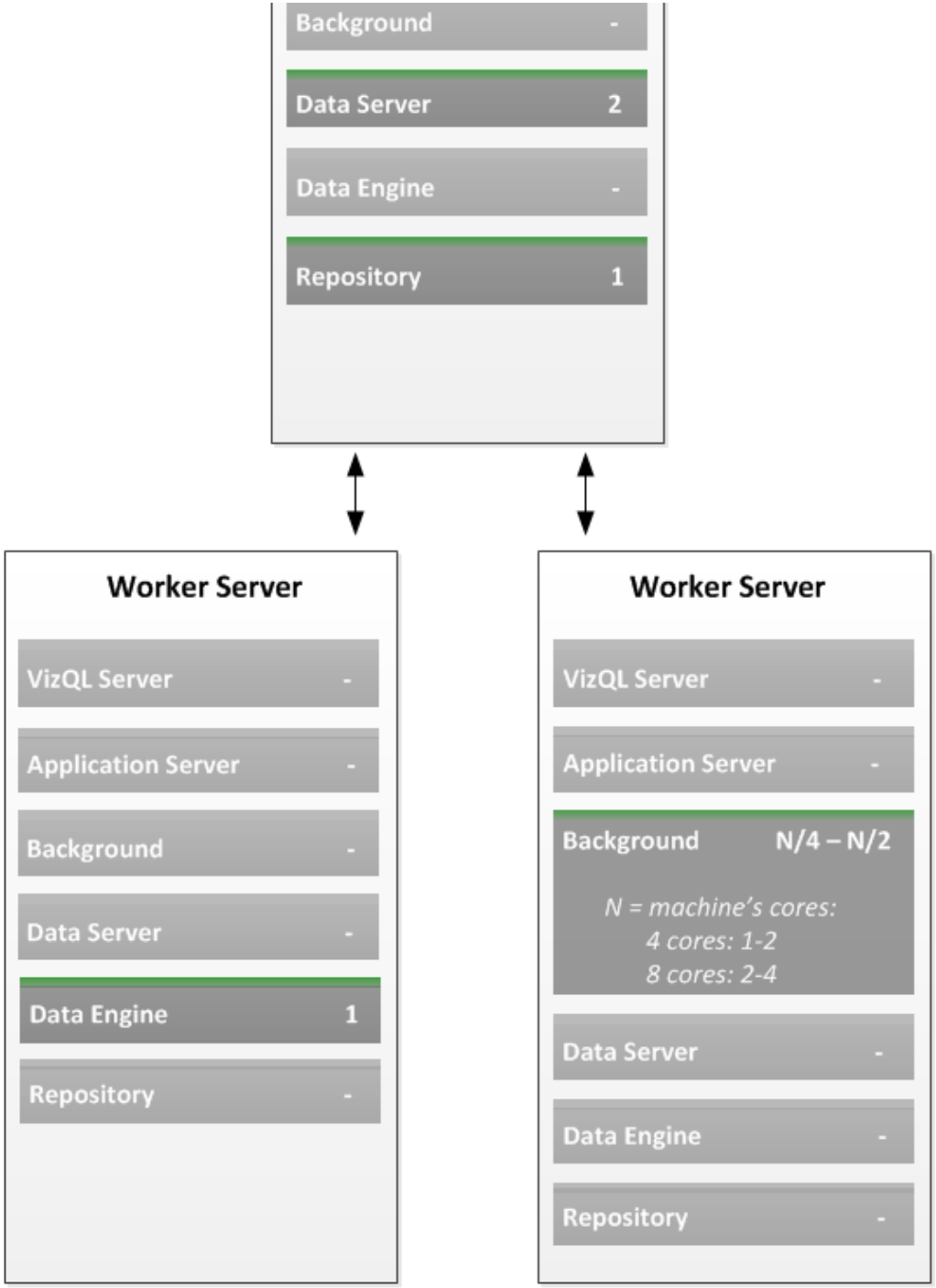
- Run at least 2 VizQL server processes on each ma2chine and determine the maximum number to run by dividing the machine's RAM by 4.
- A minimum of 2 background processes should be run on the primary Tableau Server. The maximum number you should run is equal to the machine's total number of cores.
- In a deployment where extracts are refreshed infrequently, the data engine and background processes can be on the same machine as the other processes.
- If extract refresh jobs will be only run during off hours, many background processes can be placed on each machine to maximize their parallelism.
- The number of machines in the cluster is solely determined how many VizQL Server processes are needed to support the number of concurrent viewers.

  Each additional machine you add to support viewing can be configured like the second machine (the worker server) above.

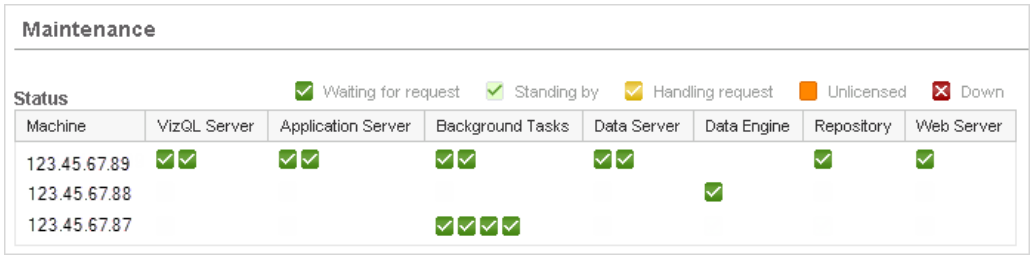
Three-Machine Example: Extracts & Viewing

A three-machine configuration is the recommended minimum number of machines to achieve the best performance if you have both a high amount of extract refreshing and usage, and a high number of concurrent users. Each of these machines is running a 64-bit operating system:





Here's the Status table for the above configuration:



Configuration Notes:

- Calculate the number of VizQL Server processes to run on the primary Tableau Server, by taking the machine's RAM and divide by 4.
- The background processes are on their own machine so that their work does not compete with that of the other processes. Because the machine is dedicated to background processes and they can consume 100% of the CPU resources, the low end of the suggested range equals the total number of cores. Depending on the size of the data being refreshed, it's possible for some deployments to run up to twice as many background processes than cores and still obtain parallel speed-up.
- Because the data engine process can consume all CPU resources on a machine, it is isolated on its own machine.
- The user loads for the application server and data server processes can typically be handled by 1 process each but they are set to 2 to provide redundancy.
- Under most conditions, the primary Tableau Server and the data engine will not be a bottleneck for the system's overall throughput as long as sufficient CPU cycles exist for them. To increase viewing capacity, add machines dedicated to the VizQL Server process. To increase capacity for refreshing extracts, add machines dedicated to the background process.

If you increase the number of VizQL instances and memory becomes an issue, there is a setting you can change. Refer to VizQL 'Out of Memory' Error for more information.

- **About Client-Side Rendering**