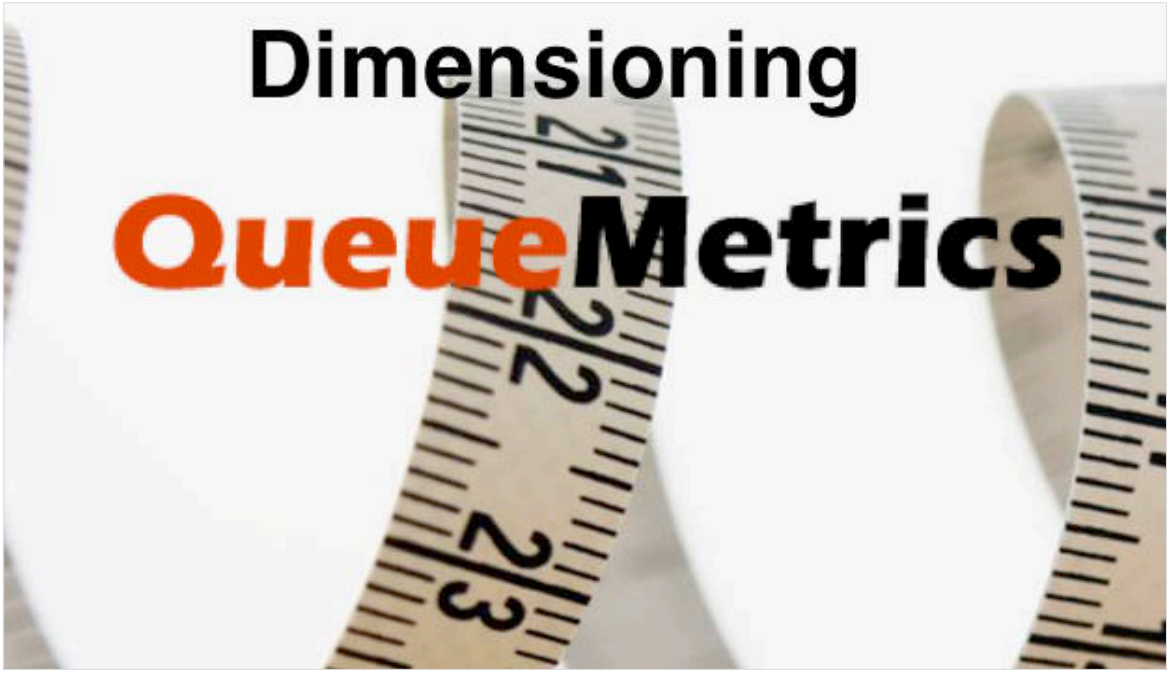
Loway presents

Dimensioning QueueMetrics Call Center Suite



Dimensioning a QueueMetrics system

QueueMetrics is not an especially resource-hungry application and is built to be quite thrifty, but as a tool that can be used in multiple ways and to run possibly very large reports, it is sometimes needed to do a little analysis beforehand. While actual numbers depend heavily on the specific usage patterns, there are some general guidelines one can follow in dimensioning.

To dimension a system you will need:

- The number of agents that will be accessing the system concurrently
- The number of expected calls per day you expect to analyze
- Whether you will be using API calls to power external systems.

What is QueueMetrics

QueueMetrics is a highly scalable monitoring and reporting suite that addresses the needs of thousands of contact centers worldwide. It offers a broad range of integrated benefits like agent

productivity monitoring, target measurement, conversion rates tracking, realtime campaign statistics analysis and an easy to use interface. It's available on premise or as a cloud hosted solution service.

Hard disk usage

QueueMetrics tracks queue events, and you can roughly expect ~5/10 events per call. Each event will be ~200 bytes on disk, so storing a million calls might require ~2Gb on disk in the worst case. This is usually the biggest item in the QM database.

Audio recordings and linked documents (e.g transcriptions) are stored externally and QueueMetrics will fetch them from a mounted file-systems, so there is no data duplication.

The QueueMetrics system itself, including the Java VM and database engine, requires ~100 Mb on a blank CentOS system.

As disk performance is very important, the faster your disks (e.g. SSD) the better QM will run.

As a general rule, you should have twice the size of your largest database table as free space on disk, to allow for maintenance (table optimizations) and temporary space for backups.

Memory usage

QueueMetrics builds a picture of the hot data-set in memory, plus any data that is required for analyses being requested. For a 100-agent system with 4/5 supervisors, 1G to 2G are usually adequate. Reporting API calls count as users.

Agents, wallboards and real-time views keep in memory a picture for the last 24 hours, so usually 10 to 20Mb for each concurrent user (agent logged in, wallboard being displayed) is adequate.

For example, a 100 agent system with about 50 users logged in concurrently and 10 wallboards open might require 20M*60 = 1.2G of memory for real-time activities plus 1G to 2G for reporting = ~3.5 G available.

Having more memory available will help the garbage collector handle peak volumes with no pauses, so a 4G Java heap might be adequate. If dimensioning a physical sever, Java will use ~200M for housekeeping, and the OS will require some more memory as a disk and DB cache, so 6G should be enough.

CPU usage

On modern multi-core systems, CPU is rarely an issue. Each QM request is run on its own thread, so QM scales out naturally on multi-core systems. It is usually not much of an issue if a real-time page or report take one second more or less when running, so CPU performance is not usually a bottleneck.

To dimension a 100 agent system with 50 agents logging on, you might expect a RT page to take ~50-100ms of CPU time, so two to four cores will be adequate.

Tips and tricks

- Virtualization is usually a good idea for QueueMetrics servers. As QueueMetrics is not especially sensitive to fine timing, as it does not handle voice itself, it works well under virtualization. With virtualization you can easily take and restore snapshots and reassign resources (CPU cores and memory) based on actual usage patterns.
- Unless the system is quite small (< 50 agents), we suggest running QueueMetrics on a separate virtual server from the main PBX. The only thing that remains to install on the PBX is a small loader tool called `uniloader` that will send queue events data to QueueMetrics.
- The number of concurrent requests can be reduced, if you use the APIs, by reducing the polling rate and/or running large analytical reports when the system is otherwise idle, e.g. after hours.
- The speed of refresh for real-time pages can be tweaked. Turning a 5-second refresh into a 7-second refresh will reduce the load by ~30% without much difference for users.
- Running large reports when the system is idle: you can have QueueMetrics generate reports automatically at night and send them over by e-mail.
- Measuring: the JVM exposes a large number of metrics through its JMX interface. If you run an important QueueMetrics service, we suggest setting up a monitoring engine to track the number and durations of garbage collection cycles so that it can be tracked over time. As a minimum, on any QueueMetrics system, you should go look for `java.lang.OutOfMemoryError` on the QueueMetrics logs.
- If you have lots of memory available, you can set the JVM to use a throughput collector - see e.g. https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/parallel.html - for the best performance at the price of increased memory usage (~2x). This should only be done if you are experiencing large collection pauses, as measured by tracking JMX statistics. By default QueueMetrics uses the G1 collector, that performs well under most workloads with no explicit tuning.

QueueMetrics Suite for Asterisk Call Centers

For more technical information about QueueMetrics call center solution please refer to the User Manual.

Visit www.queuemetrics.com for a 30 days full featured trial.

Attend our Free Webinars for a live demonstration of QueueMetrics.