Tune Application Performance

Contents:

This section provides general guidelines for tuning the configuration parameters of the Trifacta® node for varying loads.

> **NOTE:** These guidelines are estimates of what should provide satisfactory performance. You should review particulars of the variables listed below in detail prior to making recommendations or purchasing decisions.

For more information on tuning the performance of the connected cluster, see *Tune Cluster Performance*.

Application Performance

Some Trifacta platform services running on the Trifacta node can use multiple processes to serve more requests in parallel (e.g., `webapp` and `vfs-service`). By increasing the number of processes, these services are able to serve more requests in parallel and improve the application's response time under load.

Other services, such as `batch-job-runner`, `data-service`, and `scheduling-service` use multiple threads within a single process as needed to serve concurrent requests. Each service may have tuning parameters that can be adjusted according to load.

> **Tip:** Unless specific recommendations apply below, you should use the default configuration.

For an expected number of peak concurrent (active) users, *P*, set the following parameters.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter	Default Value	Recommended Value	Example
`webapp.numProcesses`	2	*P* / 15, rounded up to the nearest integer, minimum of 2	for *P* = 40, set to 3
`webapp.database.pool.maxConnections`	10	5 * `webapp.numProcesses`	for `webapp.numProcesses` = 3, set to 15
`vfs-service.numProcesses`	2	*P* / 50, rounded up to the nearest integer, minimum of 2	for *P* = 225, set to 5

Other applicable parameters

The following configuration parameters also affect application performance.

> **NOTE:** Avoid modifying these parameters unless instructed by *Trifacta Support*.

Parameter	Default Value
batch-job-runner.systemProperties.httpMaxConnectionsPerDestination	50

Limit Application Memory Utilization

Several Trifacta node services allow limitations on memory utilization by varying their JVM configuration's -Xmx value. These can be limited by modifying the following parameters:

Parameter	Default Configuration
batch-job-runner.jvmOptions	-Xmx1024m
diagnostic-server.jvmOptions	-Xmx512m
data-service.jvmOptions	-Xmx128m
spark-job-service.jvmOptions	-Xmx128m

Other services have low memory requirements.

Photon Running Environment Performance

Jobs run on mthe Trifacta node and "Quick Scan" samples are executed by the Photon running environment embedded on the Trifacta node, running alongside the application itself. Two main parameters can be used to tune concurrency of job execution and throughput of individual jobs:

Parameter	Description	Default
batchserver.workers.photon.max	Maximum number of simultaneous Photon processes; once exceeded, jobs are queued	2
photon.numThreads	Number of threads used by each Photon process	4

Increasing the number of concurrent processes allows more users' jobs to execute concurrently. However, it also leads to resource contention among the jobs and the application services.

Photon's execution is purely in memory. It does not spill to disk when the total data size exceeds available memory. As a result, you should configure limits on Photon's memory utilization. If a job exceeds the configured memory threshold, it is killed by a parent process tasked with monitoring the job.

Parameter	Description	Default
photon.memoryMonitorEnabled	Set true to enable the monitor, and set to false (limited only by operating system) otherwise	true
photon.memoryPercentageThreshold	Percentage of available system memory that each Photon process can use (if photon.memoryMonitorEnabled is true)	50

> **A reasonable rule of thumb:** the input data size should not exceed one tenth of the job's memory limit. This rule of thumb accounts for joins and pivots and other operations that can increase memory usage over the data size. However, this parameter is intended as a safeguard; it is unlikely that all running jobs would approach the memory limit simultaneously. So you should "oversubscribe" and use slightly more than (100 / batchserver.workers.photon.max) for this threshold.

In addition to a memory threshold, execution time of any Photon job can also be limited via the following parameters:

Parameter	Description	Default
`photon.timeoutEnabled`	Set `true` to enable the timeout, and set to `false` (unlimited) otherwise	`true`
`photon.timeoutMinutes`	Time in minutes after which to kill the Photon process (if `photon.timeoutEnabled` is `true`)	`180`

For more information, see *Configure Photon Running Environment*.