This page describes the latency metrics that Cloud Spanner provides. If your application experiences high latency, use these metrics to help you diagnose and resolve the issue.

You can view these metrics in the Google Cloud Console (/spanner/docs/monitoring-console) and in the Stackdriver Monitoring console (/spanner/docs/monitoring-stackdriver).

The latency metrics for Cloud Spanner measure how long it took for the Cloud Spanner service to process a request. The metric captures the actual amount of time that elapsed, not the amount of CPU time that Cloud Spanner used.

These latency metrics do not include latency that occurs outside of Cloud Spanner, such as network latency and latency within your application layer. To measure other types of latency, you can use Stackdriver Monitoring to instrument your application with custom metrics (/monitoring/custom-metrics/).

You can view charts of latency metrics in the Cloud Console (/spanner/docs/monitoring-console#view-history) and in the Stackdriver Monitoring console (/spanner/docs/monitoring-stackdriver). You can view combined latency metrics that include both reads and writes, or you can view separate metrics for reads and writes.

Based on the latency of each request, Cloud Spanner groups the requests into percentiles. You can view latency metrics for 50th percentile and 99th percentile latency:

- **50th percentile latency**: The maximum latency, in seconds, for the fastest 50% of all requests. For example, if the 50th percentile latency is 0.5 seconds, then Cloud Spanner processed 50% of requests in less than 0.5 seconds.

  This metric is sometimes called the *median latency*.

- **99th percentile latency**: The maximum latency, in seconds, for the fastest 99% of requests. For example, if the 99th percentile latency is 2 seconds, then Cloud Spanner processed 99% of requests in less than 2 seconds.

When an instance processes a small number of requests during a period of time, the 50th and 99th percentile latencies during that time are not meaningful indicators of the instance's overall performance. Under these conditions, a very small number of outliers can drastically change the latency metrics.

For example, suppose that an instance processes 100 requests during an hour. In this case, the 99th percentile latency for the instance during that hour is the amount of time it took to process the slowest request. A latency measurement based on a single request is not meaningful.

The following sections describe how to diagnose several common issues that could cause your application to experience high end-to-end latency.

For a quick look at an instance's latency metrics, use the Cloud Console (/spanner/docs/monitoring-console). To examine the metrics more closely and find correlations (/spanner/docs/monitoring-stackdriver#create-charts) between latency and other metrics, use the Stackdriver Monitoring console (/spanner/docs/monitoring-stackdriver).

If your application experiences latency that is higher than expected, but the latency metrics for Cloud Spanner are significantly lower than the total end-to-end latency, there might be an issue in your application code. If your application has a performance issue that causes some code paths to be slow, the total end-to-end latency for each request might increase.

To check for this issue, benchmark your application to identify code paths that are slower than expected.

You can also comment out the code that communicates with Cloud Spanner, then measure the total latency again. If the total latency doesn't change very much, then Cloud Spanner is unlikely to be the cause of the high latency.

If your application experiences latency that is higher than expected, and the Cloud Spanner latency metrics are also high, there are a few likely causes:

- **Your instance needs more nodes.** If your instance does not have enough CPU resources, and its CPU utilization exceeds the recommended maximum (/spanner/docs/cpu-utilization#recommended-max), then Cloud Spanner might not be able to process your requests quickly and efficiently.
- **Some of your queries cause high CPU utilization.** If your queries do not take advantage of Cloud Spanner features that improve efficiency, such as query parameters (/spanner/docs/lexical#query-parameters) and secondary indexes (/spanner/docs/secondary-indexes), or if they include a large number of joins (/spanner/docs/query-syntax#join-types) or other CPU-intensive operations, the queries can use a large portion of the CPU resources for your instance.

To check for these issues, use the Stackdriver Monitoring console to look for a correlation (/spanner/docs/monitoring-stackdriver#create-charts) between high CPU utilization and high latency. Also, check the query statistics (/spanner/docs/query-statistics) for your instance to identify any CPU-intensive queries during the same time period.

If you find that CPU utilization and latency are both high at the same time, take action to address the issue:

- If you did not find many CPU-intensive queries, add nodes to the instance (/spanner/docs/create-manage-instances#change-nodes).

  Adding nodes provides more CPU resources and enables Cloud Spanner to handle a larger workload.

- If you found CPU-intensive queries, review the query execution plans (/spanner/docs/query-execution-plans) to learn why the queries are slow, then update your queries to follow the SQL best practices for Cloud Spanner (/spanner/docs/sql-best-practices).

  You might also need to review the schema design (/spanner/docs/schema-design) for the database and update the schema to allow for more efficient queries.

- Monitor your instance with the Cloud Console (/spanner/docs/monitoring-console) or the Stackdriver Monitoring console (/spanner/docs/monitoring-stackdriver).
- Learn how to find correlations between high latency and other metrics (/spanner/docs/monitoring-stackdriver#create-charts).
- Understand how to reduce read latency by following SQL best practices (/spanner/docs/sql-best-practices) and using timestamp bounds (/spanner/docs/timestamp-bounds).
- Find out about latency metrics in query statistics tables (/spanner/docs/query-stats-tables), which you can retrieve using SQL statements.
- Understand how instance configuration affects latency (/spanner/docs/instances).