Remote write tuning

- Remote write characteristics
  - Resource usage
- Parameters
  - capacity
  - max_shards
  - min_shards
  - max_samples_per_send
  - batch_send_deadline
  - min_backoff
  - max_backoff

Prometheus implements sane defaults for remote write, but many users have different requirements and would like to optimize their remote settings.

This page describes the tuning parameters available via the remote write configuration.

Remote write characteristics

Each remote write destination starts a queue which reads from the write-ahead log (WAL), writes the samples into an in memory queue owned by a shard, which then sends a request to the configured endpoint. The flow of data looks like:

```
       |-->  queue (shard_1)   --> remote endpoint
WAL --|-->  queue (shard_...) --> remote endpoint
       |-->  queue (shard_n)   --> remote endpoint
```

When one shard backs up and fills its queue, Prometheus will block reading from the WAL into any shards. Failures will be retried without loss of data unless the remote endpoint remains down for more than 2 hours. After 2 hours, the WAL will be compacted and data that has not been sent will be lost.

During operation, Prometheus will continuously calculate the optimal number of shards to use based on the incoming sample rate, number of outstanding samples not sent, and time taken to send each sample.

Resource usage

Using remote write increases the memory footprint of Prometheus. Most users report ~25% increased memory usage, but that number is dependent on the shape of the data. For each series in the WAL, the remote write code caches a mapping of series ID to label values, causing large amounts of series churn to significantly increase memory usage.

In addition to the series cache, each shard and its queue increases memory usage. Shard memory is proportional to the `number of shards * (capacity + max_samples_per_send)`. When tuning, consider reducing `max_shards` alongside increases to `capacity` and `max_samples_per_send` to avoid inadvertently running out of memory. The default values for `capacity: 10000` and `max_samples_per_send: 2000` will constrain shard memory usage to less than 2 MB per shard.

Remote write will also increase CPU and network usage. However, for the same reasons as above, it is difficult to predict by how much. It is generally a good practice to check for CPU and network saturation if your Prometheus server falls behind sending samples via remote write (`prometheus_remote_storage_samples_pending`).

Parameters

All the relevant parameters are found under the `queue_config` section of the remote write configuration.

`capacity`

Capacity controls how many samples are queued in memory per shard before blocking reading from the WAL. Once the WAL is blocked, samples cannot be appended to any shards and all throughput will cease.

Capacity should be high enough to avoid blocking other shards in most cases, but too much capacity can cause excess memory consumption and longer times to clear queues during resharding. It is recommended to set capacity to 3-10 times `max_samples_per_send`.

`max_shards`

Max shards configures the maximum number of shards, or parallelism, Prometheus will use for each remote write queue. Prometheus will try not to use too many shards, but if the queue falls

behind the remote write component will increase the number of shards up to max shards to increase throughput. Unless remote writing to a very slow endpoint, it is unlikely that `max_shards` should be increased beyond the default. However, it may be necessary to reduce max shards if there is potential to overwhelm the remote endpoint, or to reduce memory usage when data is backed up.

`min_shards`

Min shards configures the minimum number of shards used by Prometheus, and is the number of shards used when remote write starts. If remote write falls behind, Prometheus will automatically scale up the number of shards so most users do not have to adjust this parameter. However, increasing min shards will allow Prometheus to avoid falling behind at the beginning while calculating the required number of shards.

`max_samples_per_send`

Max samples per send can be adjusted depending on the backend in use. Many systems work very well by sending more samples per batch without a significant increase in latency. Other backends will have issues if trying to send a large number of samples in each request. The default value is small enough to work for most systems.

`batch_send_deadline`

Batch send deadline sets the maximum amount of time between sends for a single shard. Even if the queued shards has not reached `max_samples_per_send`, a request will be sent. Batch send deadline can be increased for low volume systems that are not latency sensitive in order to increase request efficiency.

`min_backoff`

Min backoff controls the minimum amount of time to wait before retrying a failed request. Increasing the backoff spreads out requests when a remote endpoint comes back online. The backoff interval is doubled for each failed requests up to `max_backoff`.

`max_backoff`

Max backoff controls the maximum amount of time to wait before retrying a failed request.

□□□ #2
□ □□□ □□□ 2026-10-31 13:21:16 UTC
□ □□□ □□□ 2026-10-31 13:21:17 UTC