redislabs
HOME OF REDIS

Redis Enterprise Software Monitoring Guide

Introduction

This guide is intended to provide monitoring guidance to DevOps and SREs who are responsible for operating Redis Enterprise clusters either on premise or self-managed in the cloud. This is not intended for use with Redis services hosted by either Redis Inc or various cloud vendors. This data can be collected using the native Prometheus integration or Dynatrace or Datadog Redis Enterprise extensions.

Key Performance Indicators

Latency

Definition

redis_enterprise.avg_latency (unit: microseconds)

This is the average amount of time that a request takes to return from the time that it first hits the Redis Enterprise proxy until the response is returned. It does not include the full time from the remote client's perspective.

Characteristics

Due to the fact that Redis is popular due to performance, generally you would expect most operations to return in single digit milliseconds. Tune any alerts to match your SLA. It is generally recommended that you also measure Redis operation latency at the client side to make it easier to determine if a server slow down or an increase in network latency is the culprit in any performance issues.

Possible Causes

Cause	Factors
Possible spike in requests	Check both the Network Traffic and Operations Per Second metrics to determine if there is a corresponding increase
Slow Running queries	Check the slow log in the Redis Enterprise UI for the database
Insufficient compute resources	Check to see if the CPU Usage, Memory Usage Percentage, or Evictions are increasing

Remediation

Action	Method
Increase resources	The database can be scaled up online by going to the Web UI and enabling clustering on the database. In extreme cases more nodes can be added to the cluster and resources rebalanced.
Inefficient Queries	Redis allows you to view a slow log with a tunable threshold. It can be viewed either in the Redis Enterprise UI or by running redis-cli -h HOST -p PORT -a PASSWORD SLOWLOG GET 100

Memory Usage Percentage

Definition

redis_enterprise.memory_usage_percent (unit: percentage)

This is the percentage of the used memory over the memory limit set for the database

Characteristics

In Redis Enterprise, all databases have a maximum memory limit set to ensure isolation in a multi-tenant environment. This is also highly recommended when running open source Redis. Be aware that Redis does not immediately free memory upon key deletion. Depending on the size of the database, generally between 80-95% is a safe threshold.

Possible Causes

Cause	Factors
Possible spike in activity	Check both the Network Traffic and Operations Per Second metrics to determine if there is a corresponding increase
Database sized incorrectly	View the Memory Usage raw bytes over time to see if a usage pattern has changed
Incorrect retention policies	Check to see if keys are being Evicted or Expired

Remediation

Action	Method
Increase resources	The database memory limit can be raised on line with no downtime through either the Redis Enterprise UI or the API.
Retention Policy	In a caching use case setting a TTL for unused data to expire is often helpful. In addition, Eviction policies can be set, however, these may often not be able to keep up in extremely high throughput environments with very tight resource constraints.

Cache Hit Rate

Definition

redis_enterprise.cache_hit_rate (unit: percent)

This is the percentage of time that Redis is accessing a key that already exists.

Characteristics

This metric is useful **only in the caching use case** and should be ignored for all other use cases. There are tradeoffs between the freshness of the data in the cache and efficacy of the cache mitigating traffic to any backend data service. These tradeoffs should be considered carefully when determining the threshold for alerting.

Possible Causes

This is highly specific to the application caching with no general rules that are applicable in the majority of cases.Remediation

Note that redis commands return information on whether or not a key or field already exists. For example, HSET command returns the number of fields in the hash that were added.

Evictions

Definition

redis_enterprise.evicted_objects (unit: count)

This is the count of items that have been evicted from the database.

Characteristics

Eviction occurs when the database is close to capacity. In this condition, the eviction policy starts to take effect. While Expiration is fairly common in the caching use case, Eviction from the cache should generally be a matter of concern. At very high throughput and very restricted resource use cases, sometimes the eviction sweeps cannot keep up with memory pressure. Relying on Eviction as a memory management technique should be considered carefully.

Possible Causes

See Memory Usage Percentage Possible Causes

Remediation

See Memory Usage Percentage Remediation

Secondary Indicators

Network Traffic

redis_enterprise.ingress_bytes/redis_enterprise.egress_bytes (unit: bytes)

Counters for the network traffic coming into the database and out from the database

Definition

While these two metrics will not help you pinpoint a root cause, network traffic is an excellent leading indicator of trouble. Changes in network traffic patterns indicate corresponding changes in database behavior and further investigation is usually warranted.

Connection Count

Definition

redis_enterprise.conns (unit: count)

The count of current client connections to the database.

Characteristics

This metric should be monitored with both a minimum and maximum number of connections. The minimum number of connections not being met is an excellent indicator of either networking or application configuration errors. The maximum number of connections being exceeded may indicate a need to tune the database.

Possible Causes

Cause	Factors
Minimum clients not met	Incorrect client configuration, network firewall or network issues
Maximum connections exceeded	Client library is not releasing connections or an increase in the number of clients

Remediation

Action	Method
Clients Misconfigured	Confirm client configurations
Networking issue	From a client node TELNET to the endpoint and issue the PING command
Too many connections	Be sure that you are using pooling on your client library and that your pools are sized according
Too many connections	Using rladmin run "tune proxy PROXY_NUMBER threads VALUE"