Summary

Currently, Zabbix technically supports ingesting Prometheus metrics of all types, but some of the metrics are difficult to work with. Specifically, metrics of type Histogram can be presented in Zabbix as multiple items with the same key names but different parameters. However, even though such items are logically related and represent the same data, it's difficult to analyze this data without specialized functions.

Zabbix must support PromQL rate() and histogram_quantile() functions for easier integration with Prometheus-compatible exporters (such as Kubernetes). The rate() function calculates per-second rates (such as the number of API requests per second) and can be used either separately or along with histogram_quantile() function. histogram_quantile() accepts a quantile (percentage / 100), analyzes buckets (derived from Histogram metrics), and returns an approximation expressed in bucket units.

Use cases

1. I want to calculate rates and speeds from Prometheus monotonic counters
2. I want to calculate percentiles from Prometheus histograms

Zabbix acceptance

1. New history trigger function rate(/host/key, time period:<time shift>) must be added for triggers and calculated items
    a. It calculates the per-second rate for monotonically increasing counters
    b. It must produce the same result as PromQL rate()
        i. Corner-cases (e.g. counter resets, missing data) must be processed the same way
        ii. It must fail if there is only one data point in the time period
2. New foreach function bucket_rate_foreach(item filter, time period, <parameter number>) must be added for aggregate calculations
    a. The default for "parameter number" is 1 (parameters numbered from 1)
    b. The function must return an array of pairs (bucket upper bound, rate value) suitable for use in histogram_quantile()
        i. "bucket upper bound" is a value of item key parameter defined by "parameter number"
3. New aggregate function histogram_quantile(quantile, bucket1, value1, bucket2, value2, ...) must be added
    a. It must produce the same result as PromQL histogram_quantile()
        i. It must check the same corner cases. It must fail if:
            1. number of buckets < 2
            2. there is no +Inf bucket
            3. number of observations (value in bucket +Inf) is 0
            4. quantile < 0 or quantile > 1
    b. It must not assume any particular bucket order
4. New aggregate function bucket_percentile(item filter, time period, percentage) must be added
    a. It is an alias for histogram_quantile(percentage/100, bucket_rate_foreach(item filter, time period, 1))
5. Documentation must mention that rate() and histogram_quantile() have PromQL counterparts

Currently, only item ids are cached for item filters. The implementation should consider item parameter caching along with item ids. Macro expansion in item parameters is optional.

Examples:

Prometheus metric	Zabbix items	Expression

Prometheus metric	Zabbix items	Expression
zbx_requests_bucket{le="1.0"} zbx_requests_bucket{le="2.5"} zbx_requests_bucket{le="5.0"} zbx_requests_bucket{le="7.5"} zbx_requests_bucket{le="10.0"} zbx_requests_bucket{le="+Inf"}	zbx_requests_bucket[1.0] zbx_requests_bucket[2.5] zbx_requests_bucket[5.0] zbx_requests_bucket[7.5] zbx_requests_bucket[10.0] zbx_requests_bucket[+Inf]	Rate (number of requests per second) over 5 minutes for 1.0 bucket: Zabbix: rate(/host/zbx_requests_bucket[1.0],5m) PromQL: rate(zbx_requests_bucket{le="1.0"}[5m]) Maximum request time for 90% of all requests over the last 5 minutes: Zabbix: bucket_percentile(/host/zbx_requests_bucket[*], 5m, 90) PromQL: histogram_quantile(0.90, rate(zbx_requests_bucket[5m]))

Zabbix UI changes

1. N/A

Decisions made

1. Support for type Summary is out of scope. It's not as popular as Histogram and there is not much to help with in terms of processing.
2. Will not support timeshift for bucket_rate_foreach() and bucket_percentile() from day one. Timeshifts in foreach functions should be a subject of a separate endeavor.