Graphite Quota/Usage: Metrics of Metrics

Like an inception.

This is about a new neat thing for the Go-Graphite storage program: Go-Carbon.

(Xiaofan Hu @ Time Series)

Why

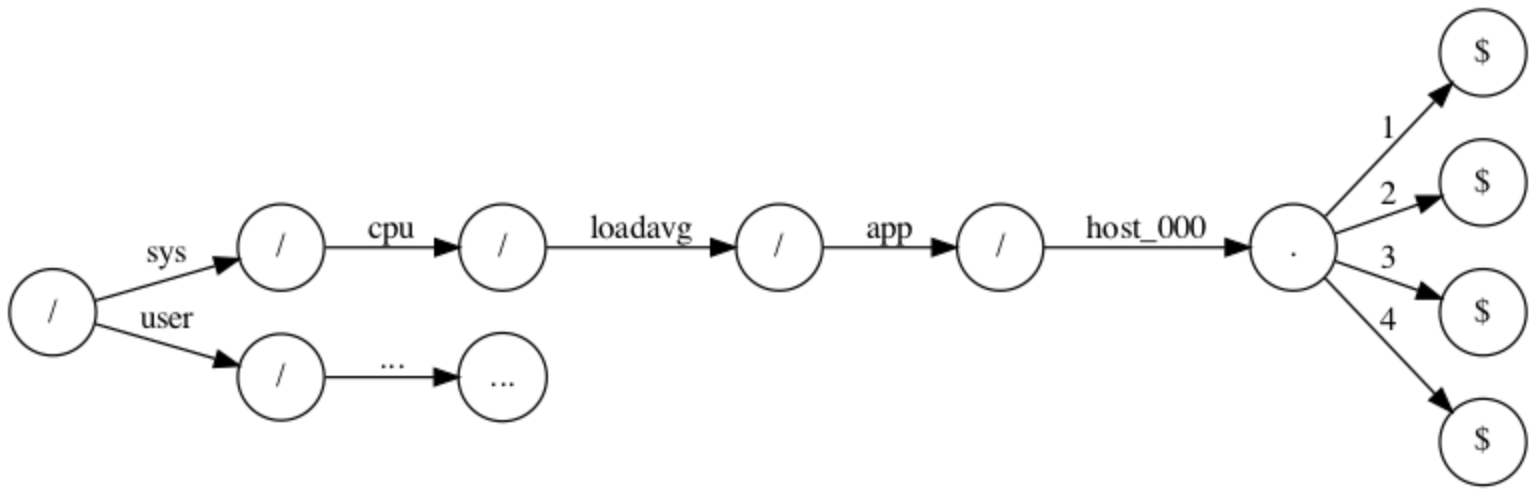
To keep Graphite alive when there are "unscheduled" metric growth or bugs that produces high amount of metrics.

(and it started with a slack chat with Anton Timofieiev.)

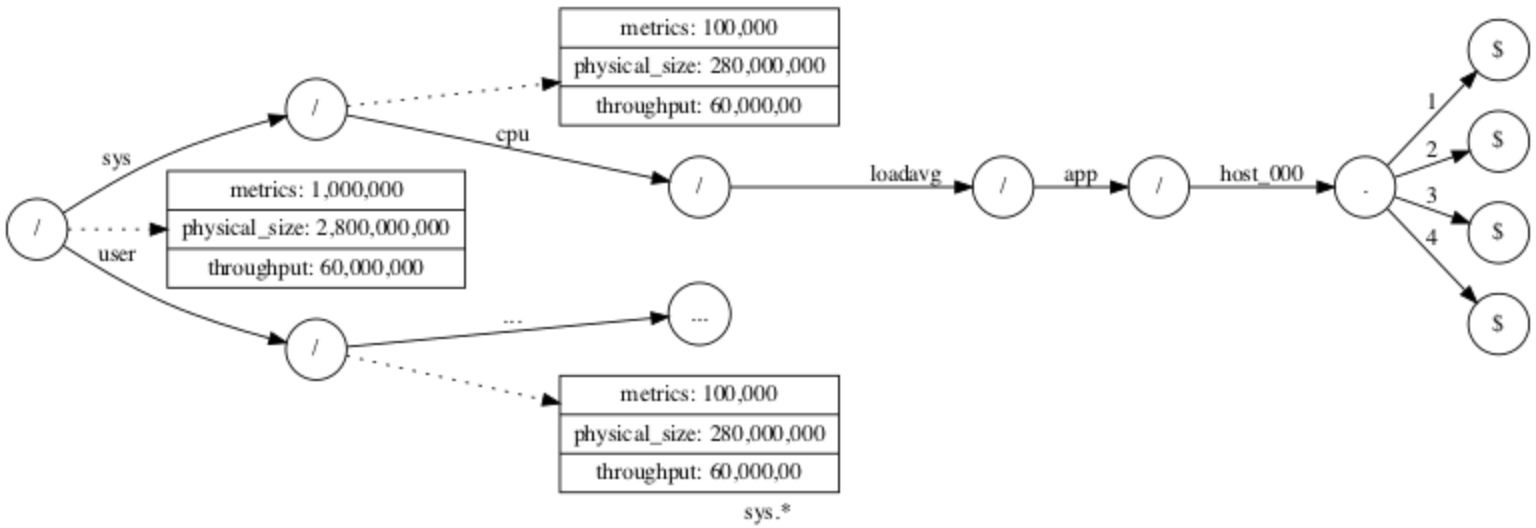
How-1

Overall idea: maintaining the usage on the directory nodes on the Go-Carbon's internal trie index tree, and whenever metrics are being sent to the system, Go-Carbon would check if it's a new metric and if there is enough quota left for its matching rules in the system.

(The devils are in the details.)



sys.*


/
sys
user
/
/
metrics: 1,000,000
physical_size: 2,800,000,000
throughput: 60,000,000
metrics: 100,000
physical_size: 280,000,000
throughput: 60,000,00
cpu
/
...
...
metrics: 100,000
physical_size: 280,000,000
throughput: 60,000,00
sys.*
loadavg
/
app
/
host_000
.
1
2
3
4
$
$
$
$

How-2

There are 6 controls available: `throughput`, `physical-size`, `logical-size`, `metrics`, `namespaces`, `data-points`.

An Example

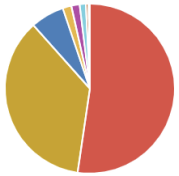
```
# For all namespaces bellow root
[*]
metrics        =       1,000,000
throughput     =      60,000,000
physical-size = 50,000,000,000 # 50GB

# For root level throttling
[/]
metrics        =       10,000,000
throughput     =      600,000,000
physical-size = 250,000,000,000 # 250GB
```

With every quota rules, metrics for quotas, usages, and throttles are generated. Nice dashboards and alarms could be built on top of them.

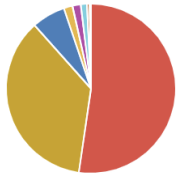
Current

data_points



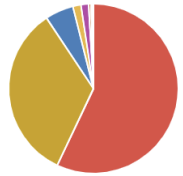
	avg
general	2.314 Tri
security	1.598 Tri
stats	283 Bil
cs	74.4 Bil
http	69.0 Bil

logical_size



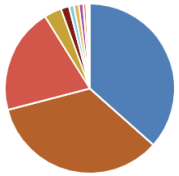
	avg
general	27.77 Tri
security	19.18 Tri
stats	3.40 Tri
cs	893 Bil
http	828 Bil

metrics



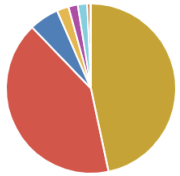
	avg
general	48.8 Mil
security	28.8 Mil
stats	4.64 Mil
cs	1.326 Mil
http	1.250 Mil

namespaces



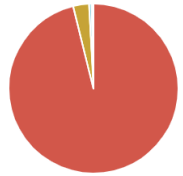
	avg
stats	5.20 K
stats_counts	4.89 K
general	2.875 K
security	471
storage	222

physical_size



	avg
security	14.45 Tri
general	12.73 Tri
stats	1.748 Tri
cs	696 Bil
http	554 Bil

throughput



	avg
general	817 Mil
security	25.8 Mil
ddos	4.06 Mil
stats	2.447 Mil
http	413 K

Quota Usages

metrics quota usage

86%

physical_size quota usage

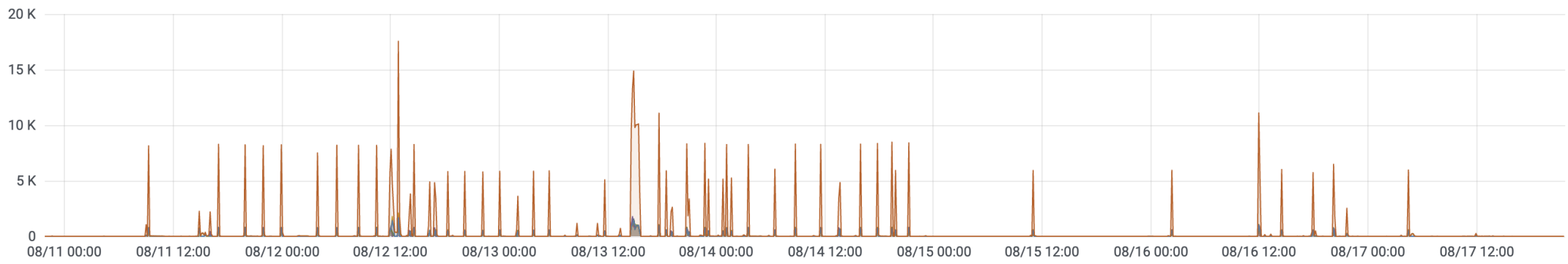
87%

throughput quota usage

63%

Details

Throttled Metrics/DataPoints



Where

It's still being slowly released on our production and for now we are looking at enforcing the quotas on root level.

However, the usages of various namespaces/prefixes that are being collected are giving us great visibility on our users.

If needed, we can quickly enable throttling on the targeted namespaces.

Fun Fact

Go map is much faster than a home-baked trie tree traversing.

It's so good that it actually saves us 5 CPU cores when it comes to throttling on over 1 million data points per second.

(In hindsight, it makes sense. I was just not expecting multiple map lookup are still faster than one trie tree traversal.)

Links

- The Quota Pull Request on Github.
- How the Go map helps cutting down our cpu usage.
- More Quota docs for Go-Graphite/Go-Carbon.