QCon
全球软件开发大会

单机 20 亿指标，知乎 Graphite 极致优化！

知乎 SRE / 熊豹 (faceair)

InfoQ PRO

关注 InfoQ Pro 服务号

你将获得：

- InfoQ 技术大会讲师 PPT 分享
- 最新全球 IT 要闻
- 一线专家实操技术案例
- InfoQ 精选课程及活动
- 定期粉丝专属福利活动

极客时间

极客时间 App

轻松学习 · 高效学习

大会到场福利

新老用户

可获得课程优惠口令 立减10元

QCon66666

新用户

可免费领取一门课

扫码立领 >>>

1 Graphite 在知乎的应用

2 现有的存储方案都有哪些不足？

3 我们的方案：Graphite On Prometheus

4 着眼未来！Graphite 已死，Statsd 永生！

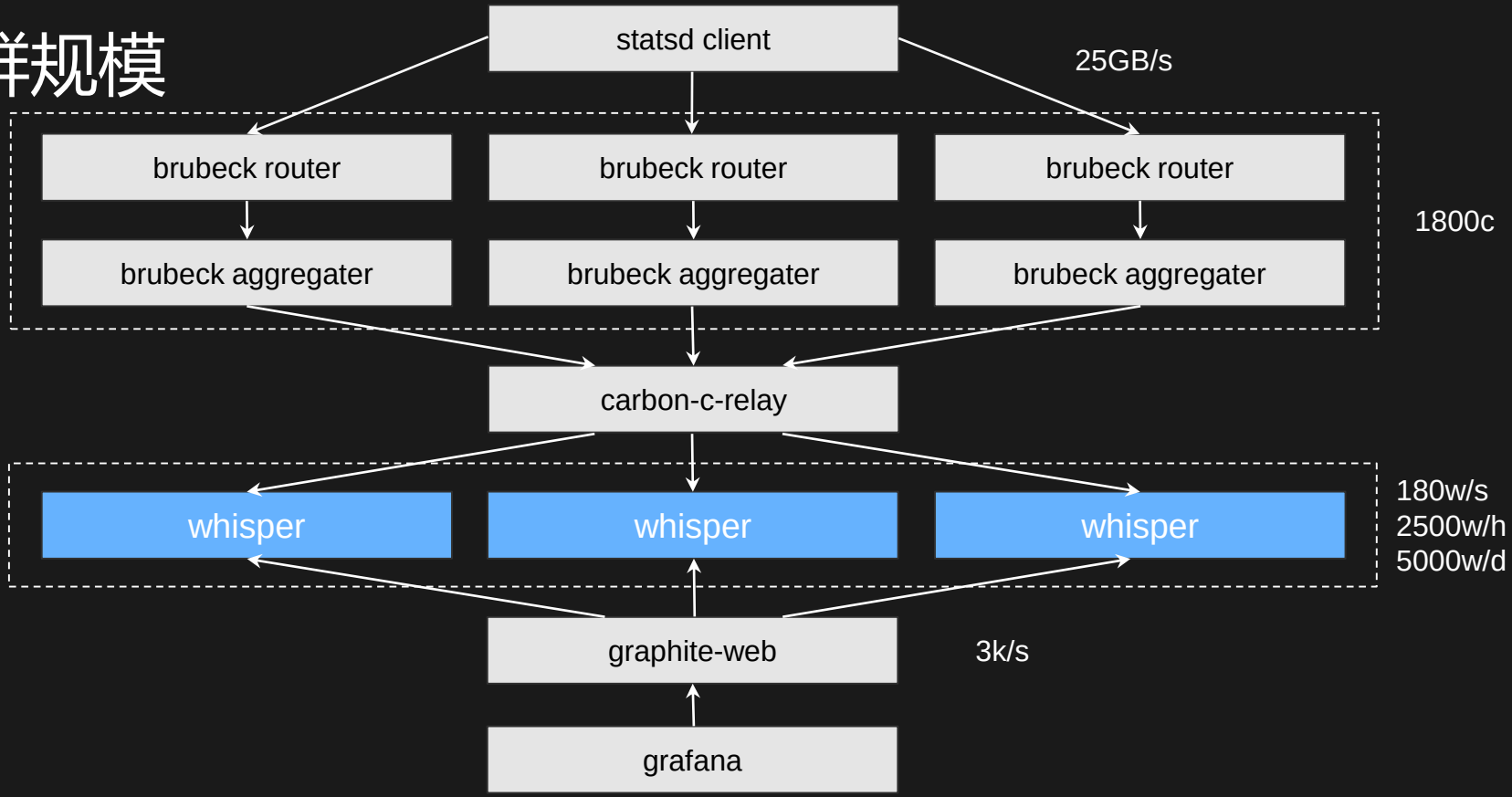
Graphite 在知乎的应用

应用场景 → 集群规模 → 效果对比

应用场景

1. CDN、动态加速分地区、运营商监控，长短连接网关状态码、出入流量监控
2. 数千微服务运行监控，支撑故障定位、依赖分析、性能分析等场景
3. 大部分业务场景 (商业广告投放、前端资源加载、推送全链路等) 实时监控
4. 全量基础资源实例 (Redis、MySQL、Kafka、HBase 等) 运行监控
5. 多机房万级物理机多维度资源利用率监控

集群规模


statsd client
25GB/s
brubeck router
brubeck router
brubeck router
brubeck aggregater
brubeck aggregater
brubeck aggregater
1800c
carbon-c-relay
whisper
whisper
whisper
180w/s
2500w/h
5000w/d
graphite-web
3k/s
grafana

Whisper 的问题

1. 查询性能差，长时间跨度或复杂条件难出结果
2. 集群读写合计 500w IOPS，对磁盘性能要求极高
3. 存储节点 CPU、内存利用率高，磁盘存储空间占用大
4. 临时和无效的指标难区分和清理，维护成本高昂

效果对比

100 倍

复杂查询响应时间提升 100 倍
响应时间平均提升 10 倍，P95 提升 8 倍
查询压力不变，开销降低 80%

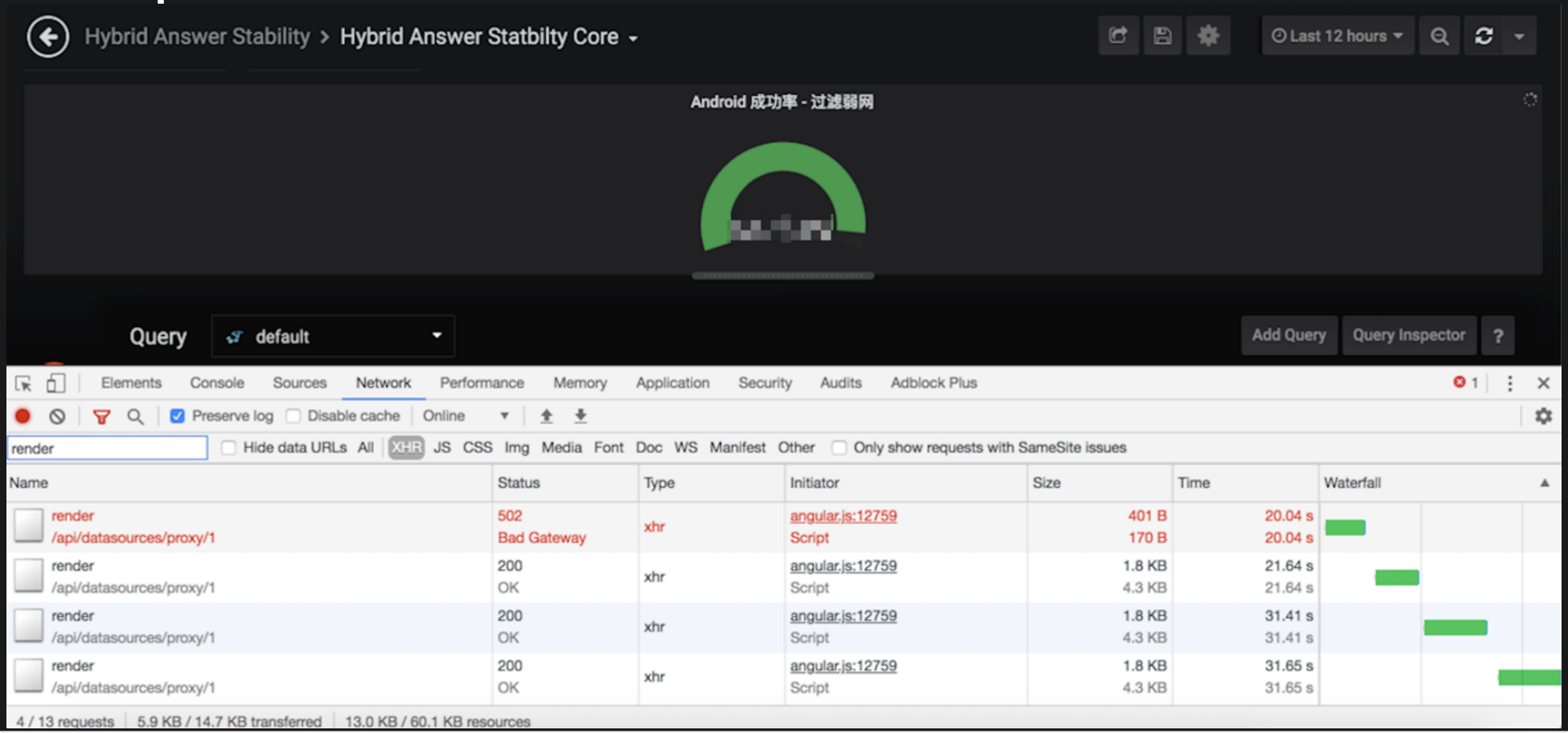
10000 倍

IOPS 降低 1w 倍
存储空间占用降低 90%
单机存储指标量高出业界 2000 倍

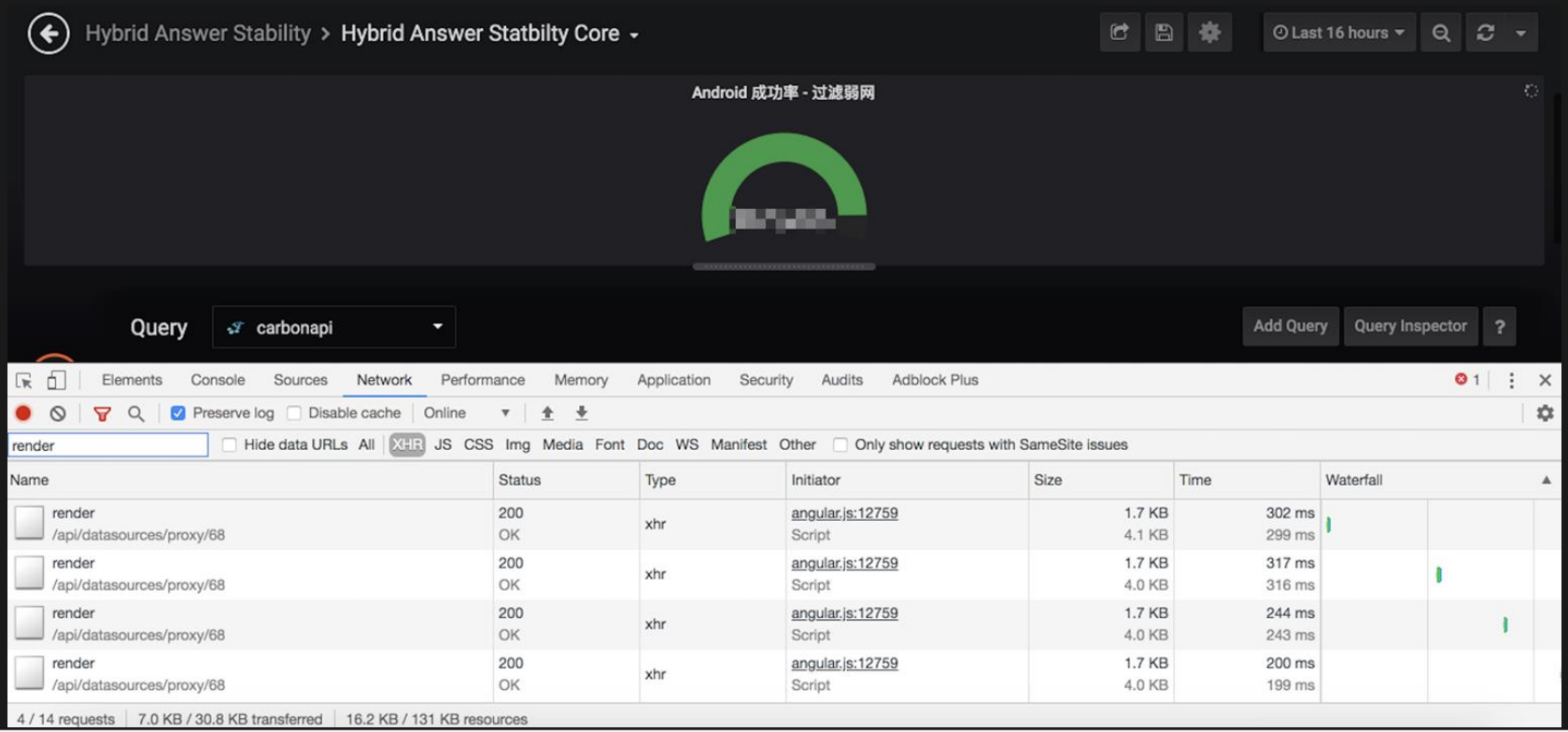
Example: Android 某加载成功率

```
asPercent(
    integral(
        transformNull(
            sumSeriesWithWildcards(statsc.production.Zhihu.Android*[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadSuccess, 3, 4, 7)
        , 0)
    ),
    sumSeries(
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadSuccess, 3, 4, 7)
            , 0)
        ),
        integral(
            scale(
                transformNull(
                    sumSeriesWithWildcards(statsc.production.Zhihu.Android*[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadAbortedTiming.*.count_ps, 3, 4, 7, 9)
                    , 0)
            , 10)
        ),
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadFail.*, 3, 4, 7, 9)
            , 0)
        ),
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadFail.ResponseError.*, 3, 4, 7, 9, 10)
            , 0)
        ),
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*.[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadFail.RequestError, 3, 4, 7)
            , 0)
        ),
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*.[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadFail.RequestError.timeout, 3, 4,7)
            , 0)
        ),
        integral(
            transformNull(
                sumSeriesWithWildcards(statsc.production.Zhihu.Android*.[redacted].LoadStabilityWithNetwork.{GOOD,MEDIUM}.PageLoadFail.*.*.*, 3, 4, 7, 9, 10, 11)
            , 0)
        )
    )
)
```

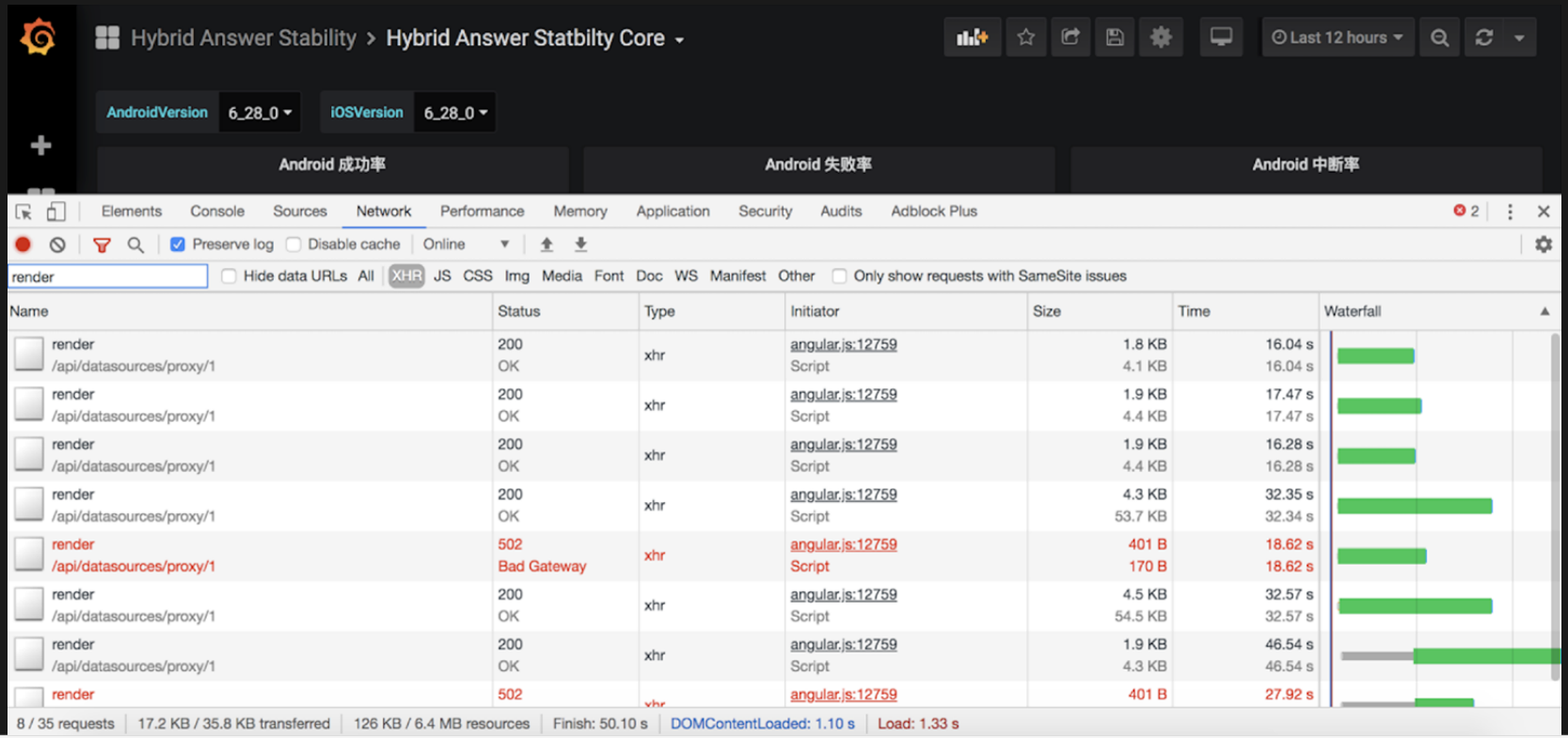
Example: Android 某加载成功率



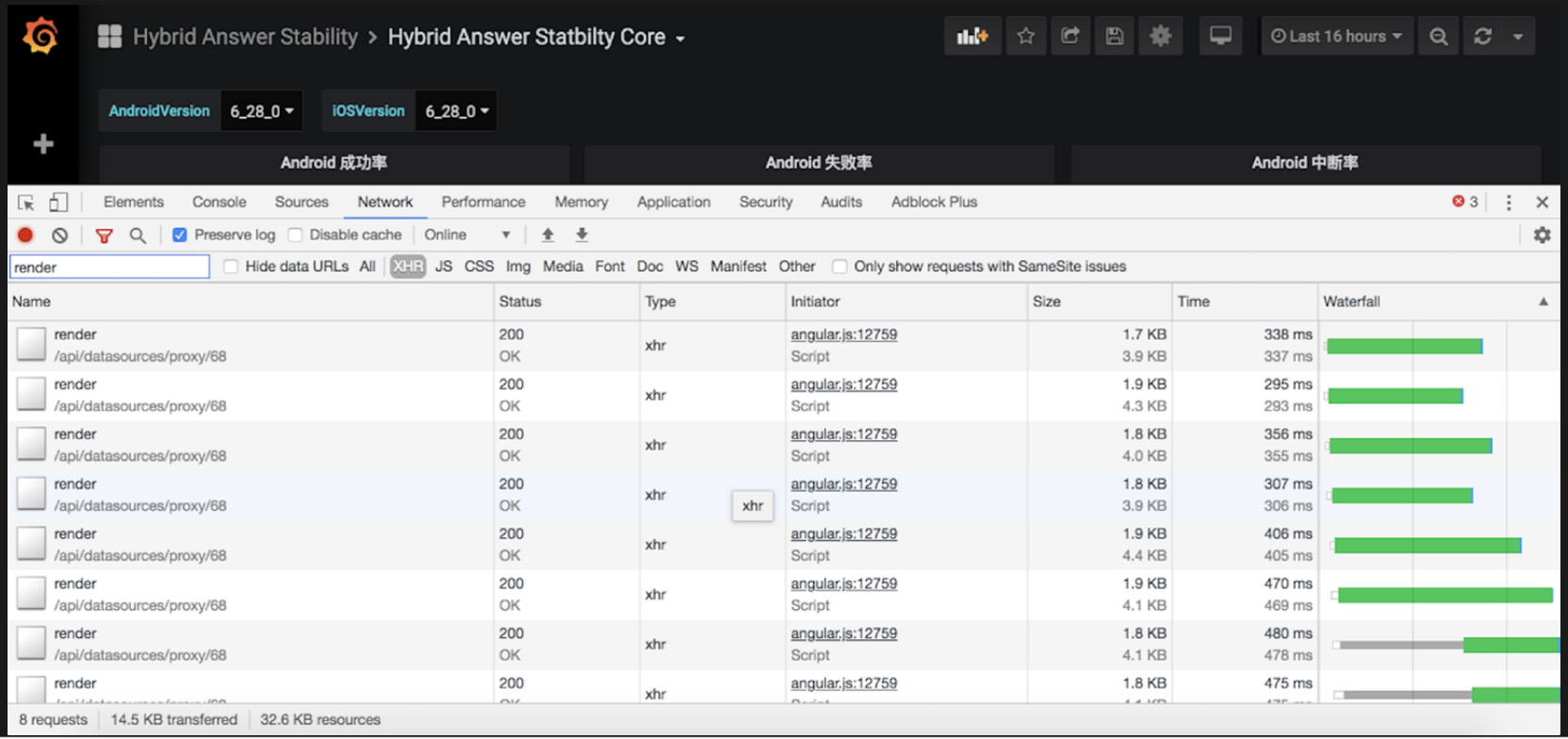
Example: 提升 100 倍



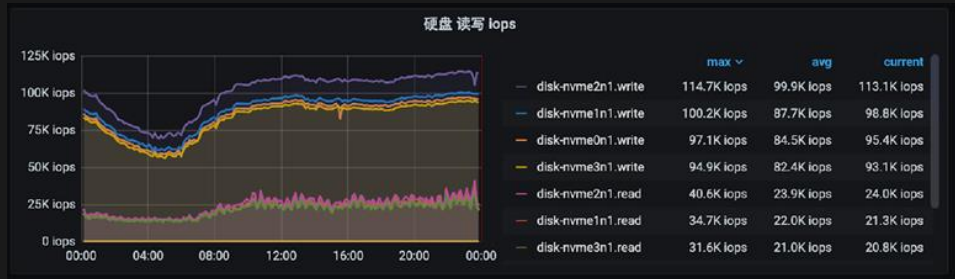
Example: Hybrid 某监控大盘

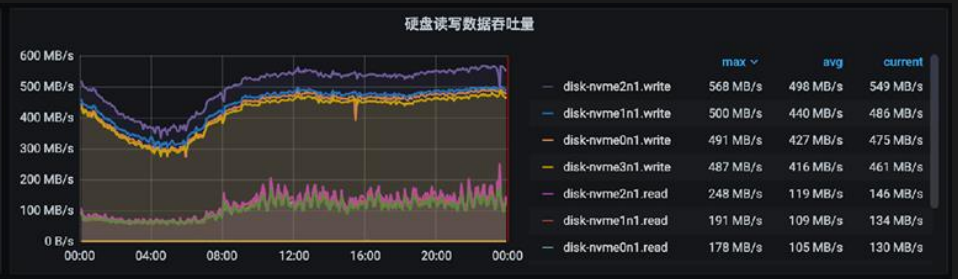


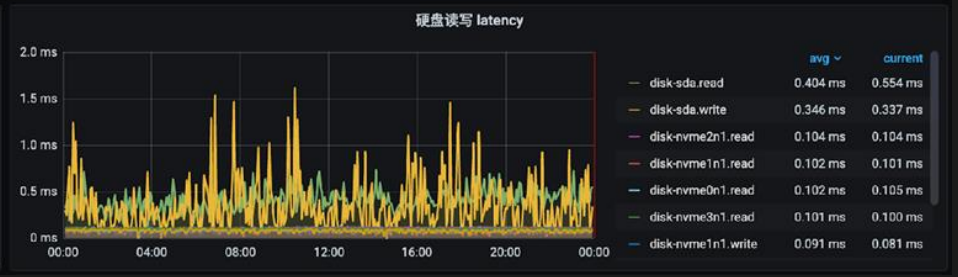
Example: 提升 66 倍



Example: 改造前 NVMe 单节点







Graphite 存储方案效果对比

2020-05-08 00:00:00 to 2020-05-08 23:59:59

Whisper NVMe 集群读写 IOPS

	max	avg
write	4.155M iops	3.595M iops
read	1.479M iops	1.061M iops

Promate HDD 硬盘读写 iops

	max	avg
write	454 iops	268 iops
read	236 iops	87 iops

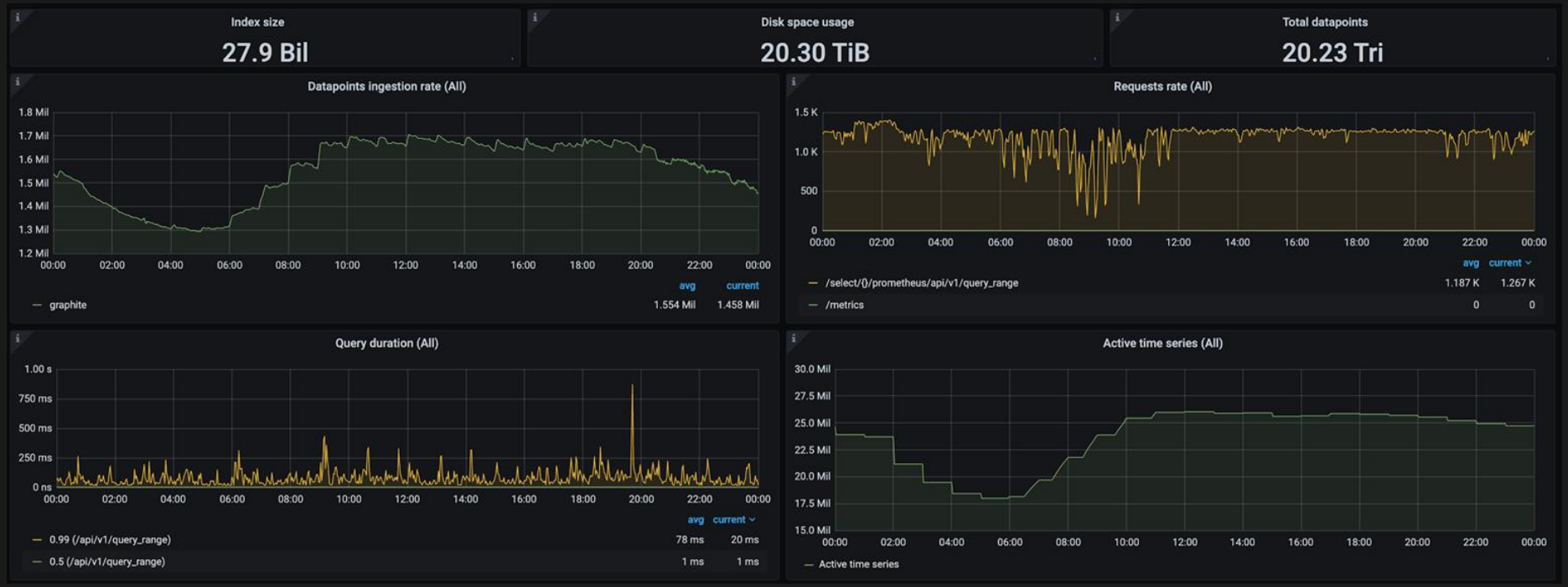
Whisper NVMe 集群硬盘读写吞吐量

	max	avg
write	21.01 GB/s	18.26 GB/s
read	9.81 GB/s	5.76 GB/s

Promate HDD 硬盘读写吞吐量

	max	avg
write	111.9 MB/s	52.0 MB/s
read	61.7 MB/s	8.8 MB/s

Example: 单个 HDD 节点



1 Graphite 在知乎的应用

2 现有的存储方案都有哪些不足？

3 我们的方案：Graphite On Prometheus

4 着眼未来！Graphite 已死，Statsd 永生！

现有的存储方案都有哪些不足？

Whisper 速览 → 改良派与革命派 → 高效索引

Whisper 速览

host.machine_name.memory.percent_used.value  host/machine_name/memory/percent_used/value.wsp

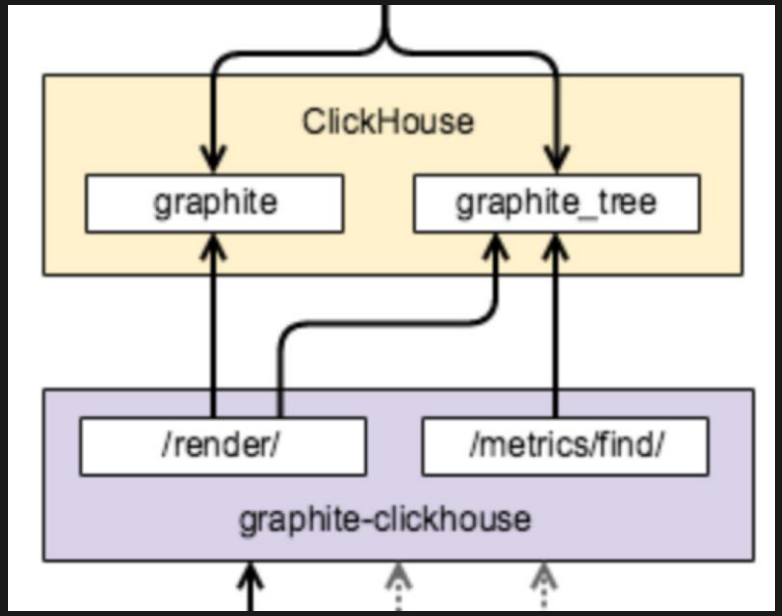
host.machine_[a-zA-Z0-9]*.memory?.{percent_used,percent_free}.value

1. 每个活跃指标对应打开一个本地文件，IOPS 随读写线性递增
2. 查询指标层级变多或量大时需要操作文件多，读取性能差
3. 更新指标数据是随机写，对磁盘性能要求高
4. 指标数据压缩能力差，读取和写入占用内存高，磁盘存储空间占用大

开源方案

1. 改良派
    a. go-carbon 使用 golang 对 whisper 重写
    b. kenshin 豆瓣出品，通过合并 whisper 指标文件降低数倍 IOPS
2. 革命派
    a. graphite-clickhouse 存储指标到 clickhouse
    b. metrictank Grafana Labs 出品，存储指标到 cassandra

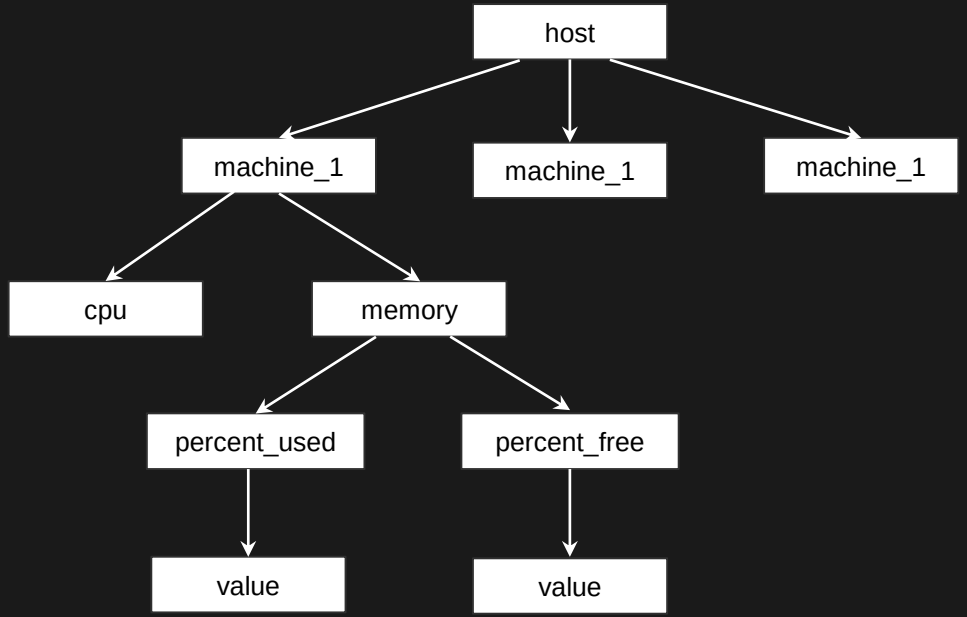
graphite-clickhouse 的索引



github.com/lomik/graphite-clickhouse/blob/v0.11.7/pkg/where/match.go

```
34
35          // Q() replaces \ with \\, so using \. does not work here.
36          // work around with [.]
37          postfix := `$`
38          if optionalDotAtEnd {
39                  postfix = `[.]?$`
40          }
41
42          if simplePrefix == "" {
43                  return fmt.Sprintf("match(%s, %s)", field, quote(`^`+GlobToRegexp(query)+postfix))
44          }
45
46          return fmt.Sprintf("%s AND match(%s, %s)",
47                  HasPrefix(field, simplePrefix),
48                  field, quote(`^`+GlobToRegexp(query)+postfix),
49          )
50  }
```

metrictank 的索引



1. 内存开销大
2. 历史指标影响查询 & 写入性能
3. 查询计划无法优化

开创者 Uber M3

host.machine_name.memory.percent_used.value

```
{
    __g0__="host",
    __g1__="machine_name",
    __g2__="memory",
    __g3__="percent_used",
    __g4__="value"
}
```

倒排索引 for Graphite

1: host.machine_1.memory.percent_used.value
2: host.machine_2.cpu.percent_used.value
3: host.machine_3.memory.percent_used.value
4: host.machine_4.memory.percent_free.value

```
__g0__:host              -> 1,2,3,4
__g1__:machine_1          -> 1
__g1__:machine_2          -> 2
__g1__:machine_3          -> 3
__g1__:machine_4          -> 4
__g2__:memory            -> 1,3,4
__g2__:cpu               -> 2
__g3__:percent_used       -> 1,2,3
__g3__:percent_free       -> 4
__g4__:value             -> 1,2,3,4
```

host.machine_*.memory.percent_free.value

```
查找 __g0__:host 得到                                -> [1,2,3,4]
在 [1,2,3,4] 中查找 __g4__:value 得到                -> [1,2,3,4]
在 [1,2,3,4] 中查找 __g2__:memory 得到               -> [1,3,4]
在 [1,3,4] 中查找 __g3__:percent_free 得到           -> [4]
在 [4] 中进行正则匹配 __g1__:machine_* 得到          -> [4]
```

倒排索引 for Graphite

1. 打破了 Graphite 指标层级关系，查询计划可优化
2. 每个索引项可独立存储和查询，也可以根据时间切片

1 Graphite 在知乎的应用

2 现有的存储方案都有哪些不足?

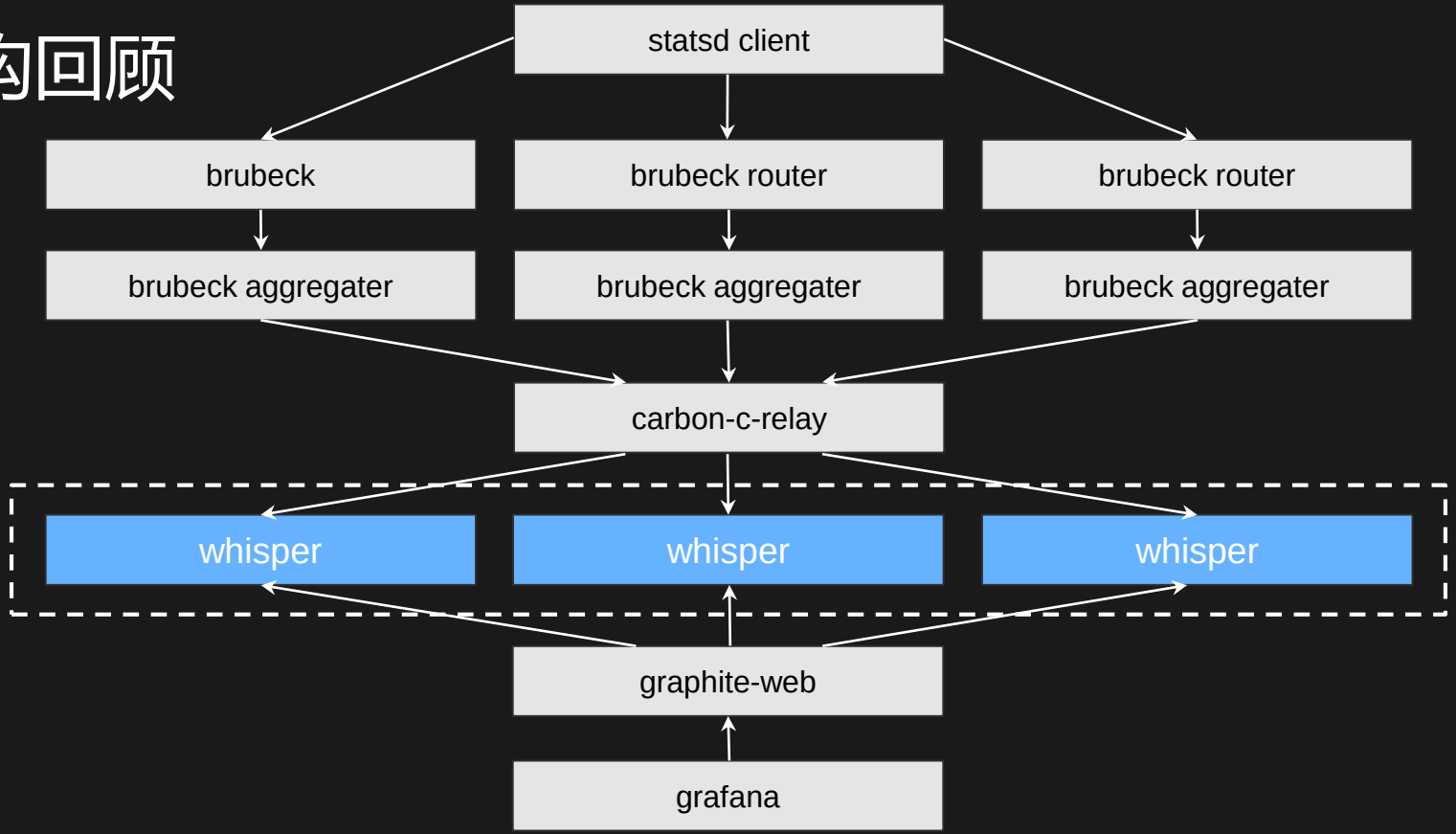
3 我们的方案：Graphite On Prometheus

4 着眼未来！Graphite 已死，Statsd 永生！

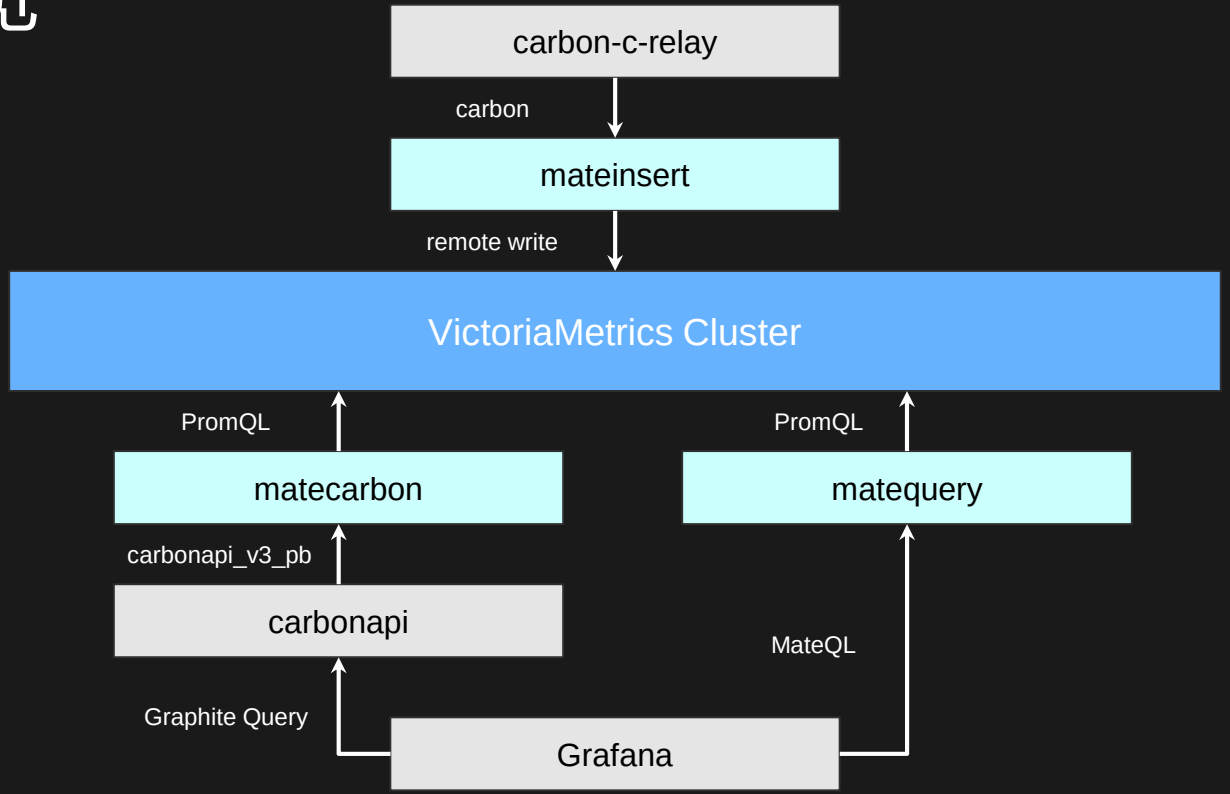
Graphite On Prometheus

架构速览 → 高性能 VictoriaMetrics → 兼容 Graphite → MateQL

架构回顾


statsd client
brubeck
brubeck router
brubeck router
brubeck aggregater
brubeck aggregater
brubeck aggregater
carbon-c-relay
whisper
whisper
whisper
graphite-web
grafana

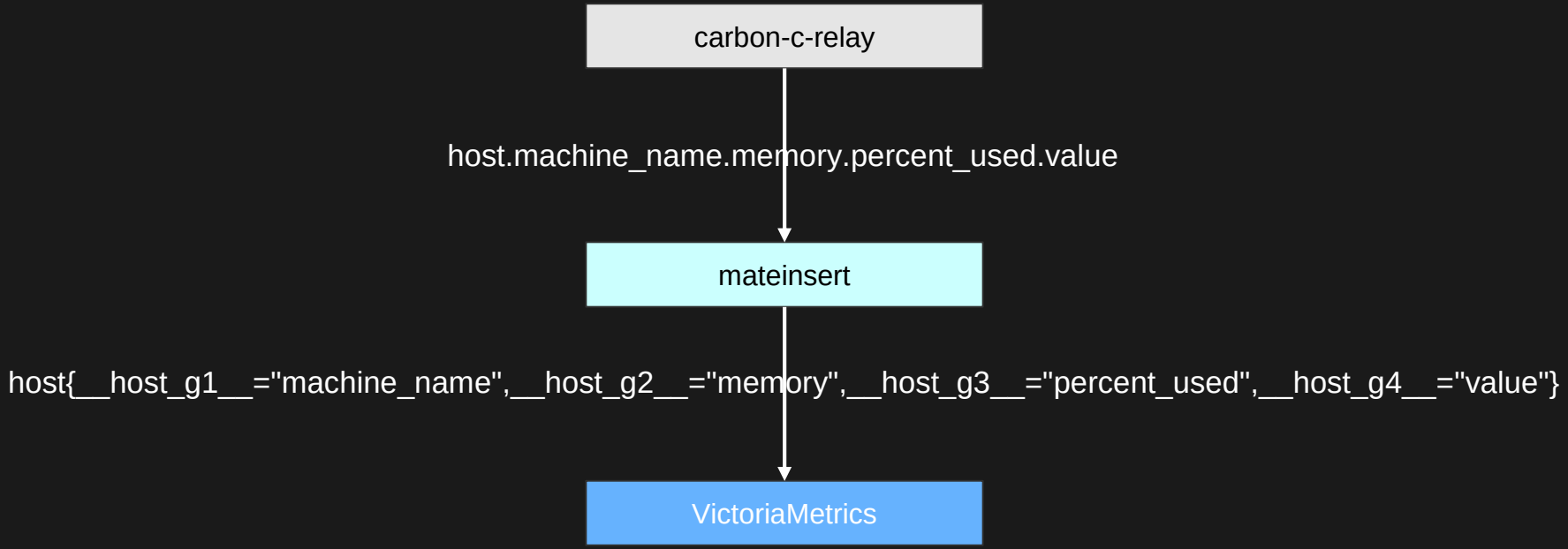
架构速览


carbon-c-relay
carbon
mateinsert
remote write
VictoriaMetrics Cluster
PromQL
PromQL
matecarbon
matequery
carbonapi_v3_pb
carbonapi
MateQL
Graphite Query
Grafana

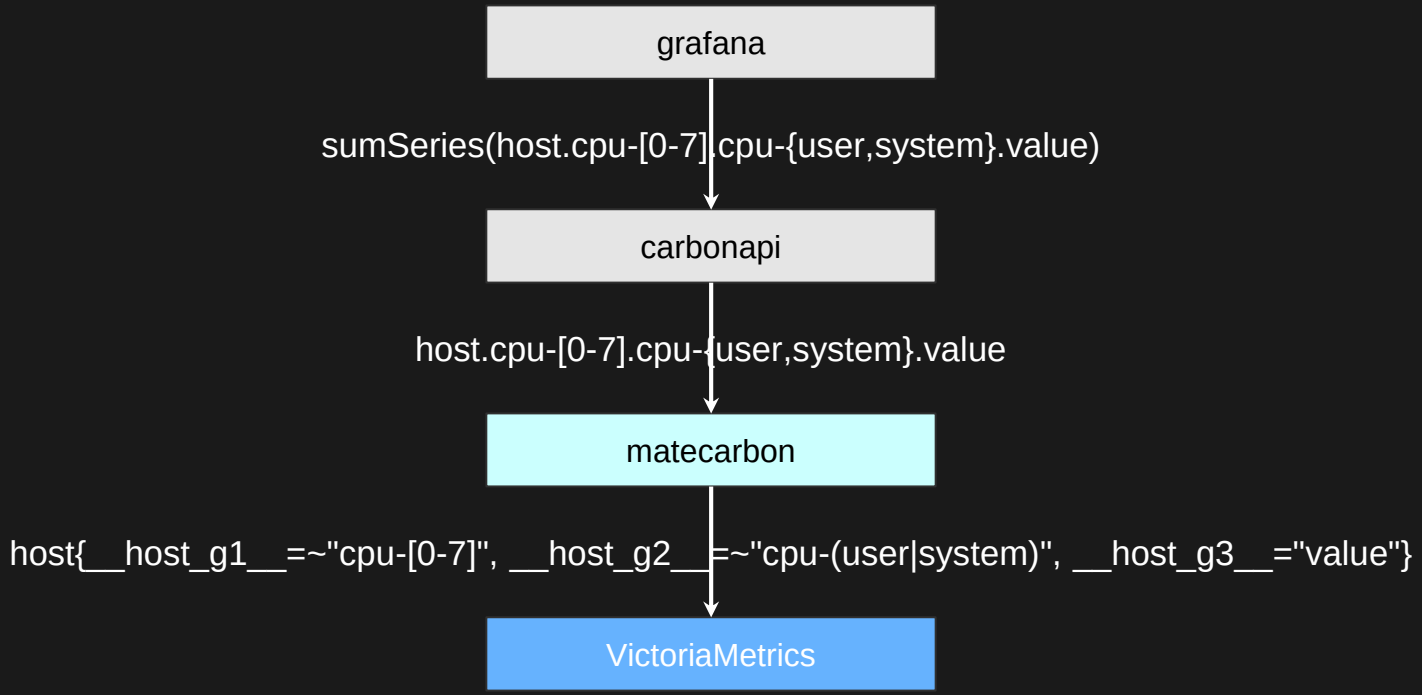
高性能 VictoriaMetrics

1. Prometheus 远端存储，各项性能测试中都名列前茅
2. 创始人 Aliaksandr Valialkin (@valyala)，著名项目 fasthttp、chproxy 作者
3. 维护活跃，积极响应用户需求
4. 集群架构简单，无外部依赖，易维护

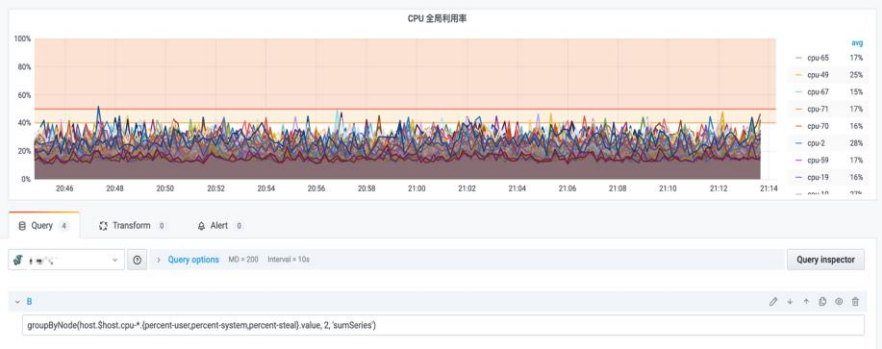
mateinsert


carbon-c-relay
host.machine_name.memory.percent_used.value
mateinsert
host{__host_g1__="machine_name",__host_g2__="memory",__host_g3__="percent_used",__host_g4__="value"}
VictoriaMetrics

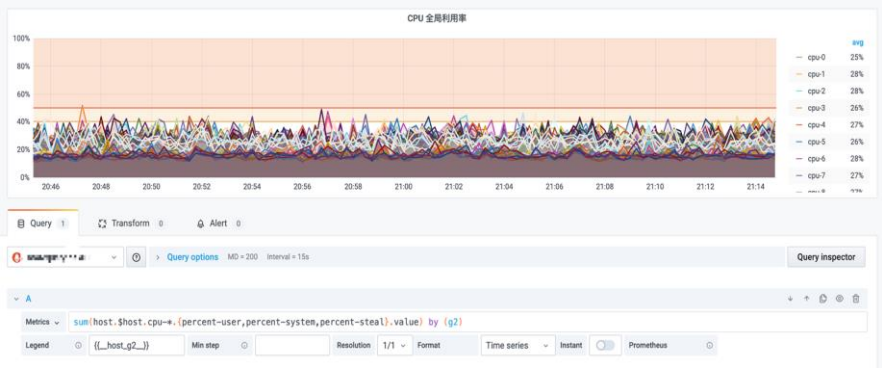
matecarbon


grafana
sumSeries(host.cpu-[0-7].cpu-{user,system}.value)
carbonapi
host.cpu-[0-7].cpu-{user,system}.value
matecarbon
host{__host_g1__=~"cpu-[0-7]", __host_g2__=~"cpu-(user|system)", __host_g3__="value"}
VictoriaMetrics

Example: CPU 全局利用率



groupByNode(host.$host.cpu-*.{percent-user,percent-system,percent-steal}.value, 2, 'sumSeries')

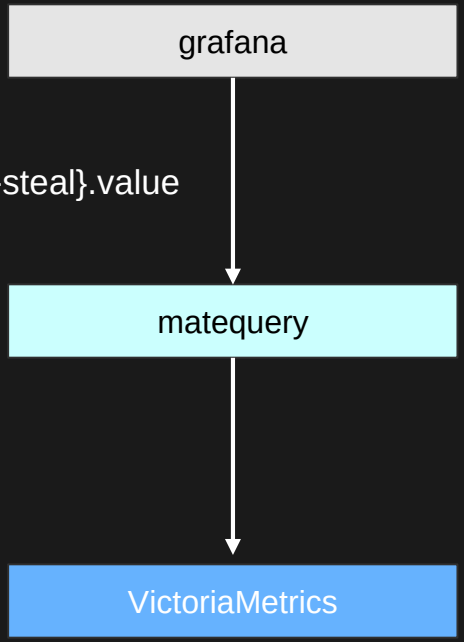


sum(host.machine_name.cpu-*.{percent-user,percent-system,percent-steal}.value) by (g2)

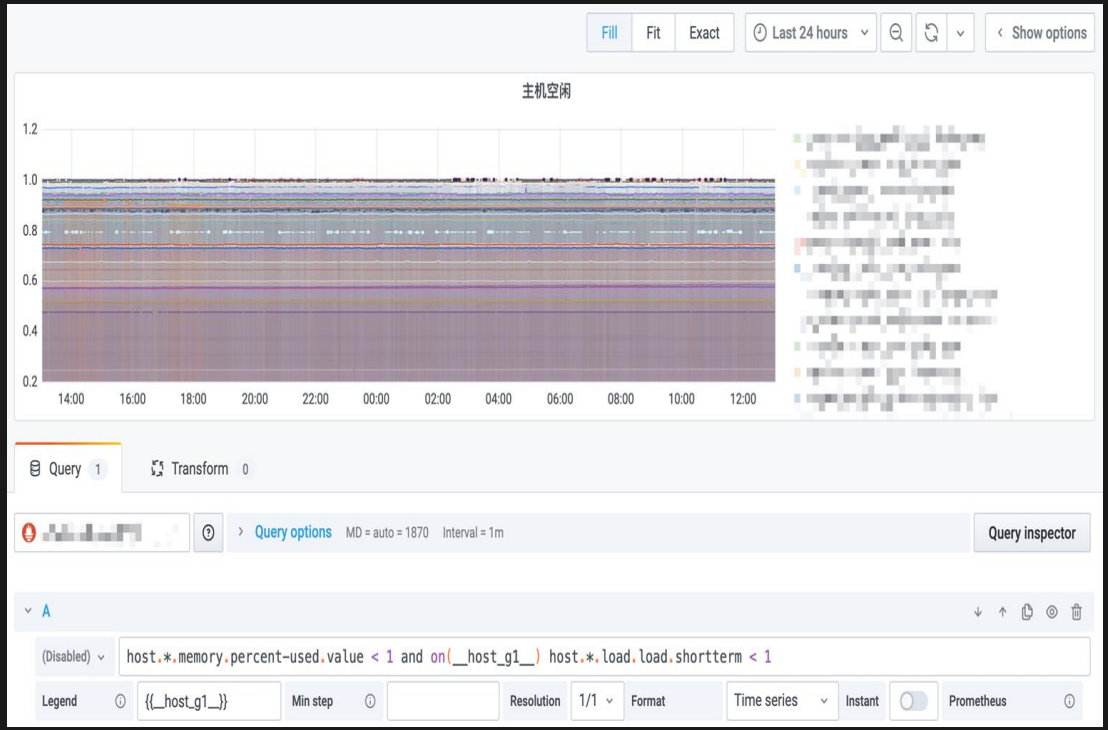
MateQL

```
sum(
        host.machine_name.cpu-*.{percent-user,percent-system,percent-steal}.value
) by (g2)
```

```
sum(
    host{
        __host_g1__="machine_name",
        __host_g2__=~"cpu-[a-zA-Z0-9_-]*",
        __host_g3__=~"(percent-user|percent-system|percent-steal)",
        __host_g4__="value"
    }
) by (__host_g2__)
```



Example: 主机空闲

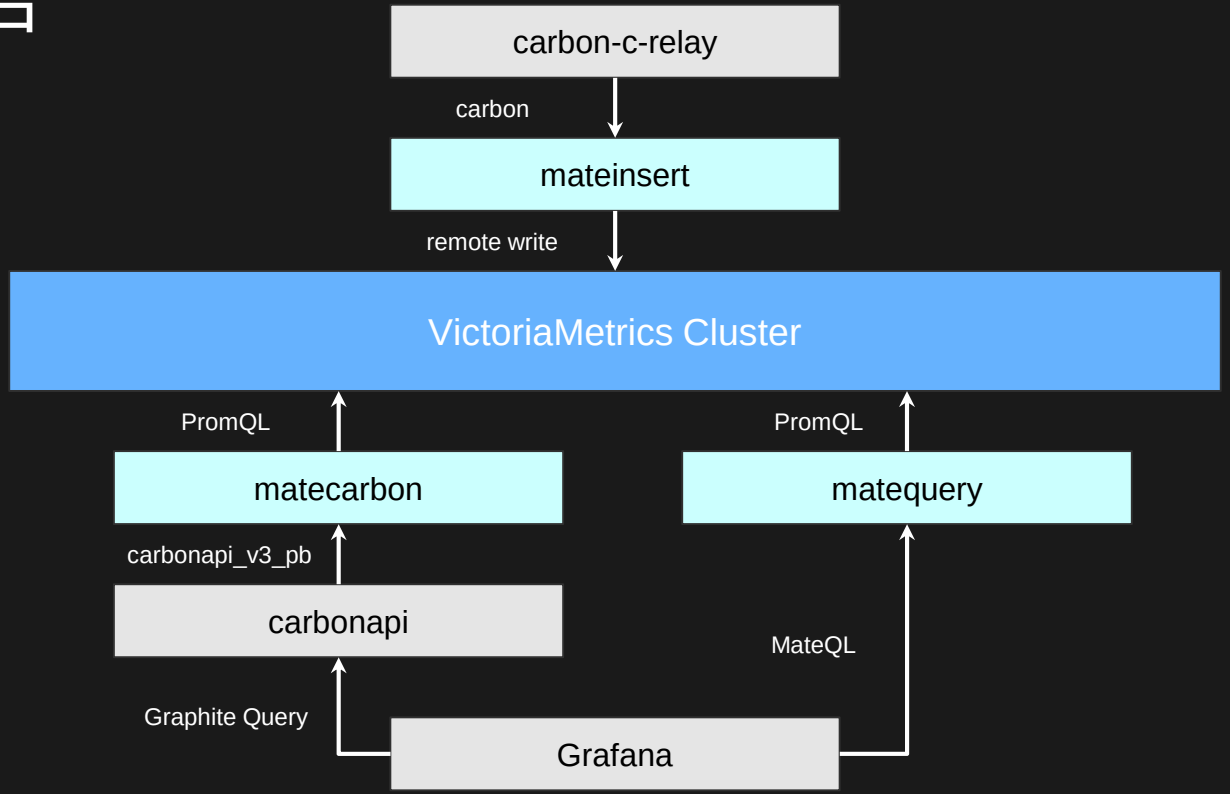


host.*.memory.percent-used.value < 1

and on(__host_g1__)

host.*.load.load.shortterm < 1

架构总结


carbon-c-relay
carbon
mateinsert
remote write
VictoriaMetrics Cluster
PromQL
PromQL
matecarbon
matequery
carbonapi_v3_pb
carbonapi
MateQL
Graphite Query
Grafana

1 Graphite 在知乎的应用

2 现有的存储方案都有哪些不足?

3 我们的方案：Graphite On Prometheus

4 着眼未来！Graphite 已死，Statsd 永生！

Graphite 已死，Graphite 永生！

指标设计 → 推、拉模型 → 统一架构

指标设计

Traditional systems

```
collectd.dfs1.df.srv-node-dfs10.df-complex.used
```

```
diskspace._srv_node_dfs10.byte_used
{
    host: dfs1
}
```

Metrics 2.0

```
{
    host: dfs1
    what: diskspace
    mountpoint: srv/node/dfs10
    unit: B
    type: used
    metric_type: gauge
}
meta: {
    agent: diamond,
    processed_by: statsd2
}
```

指标设计

OpenMetrics - Transforming the Prometheus Exposition Format into a Global Standard

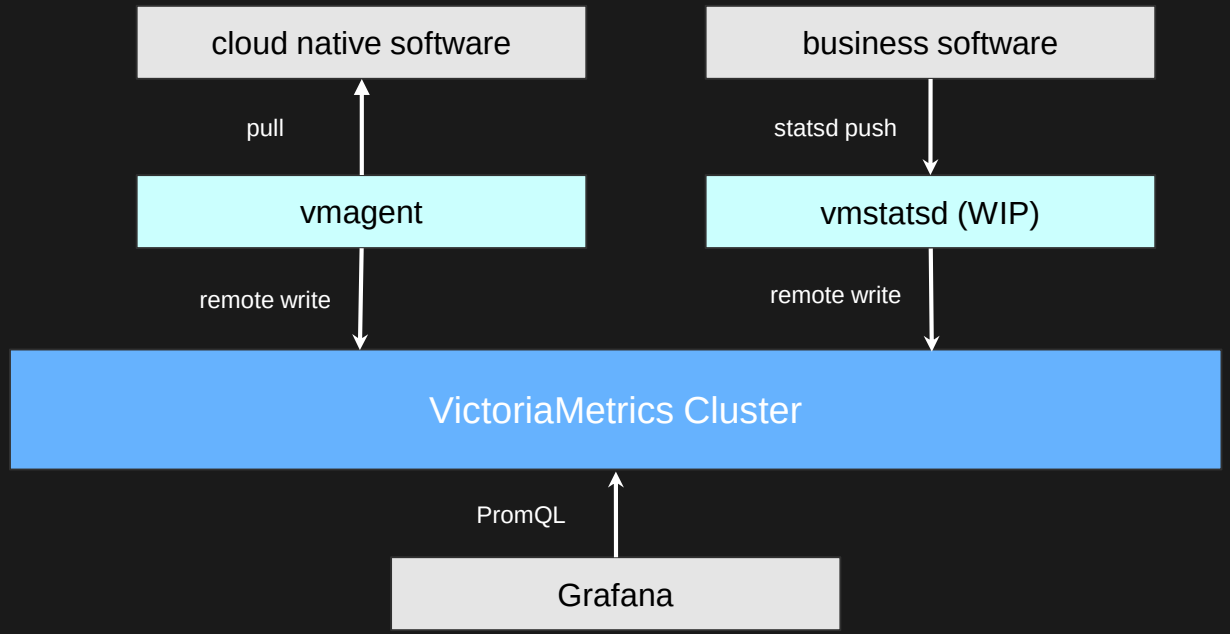
Richard Hartmann
SpaceNet

Push or Pull

Monarch: Google’s Planet-Scale In-Memory Time Series Database

Push-based data collection improves system robustness while simplifying system architecture. Early versions of Monarch discovered monitored entities and “pulled” monitoring data by querying the monitored entity. This required setting up discovery services and proxies, complicating system architecture and negatively impacting overall scalability. Push-based collection, where entities simply send their data to Monarch, eliminates these dependencies.

统一架构


cloud native software
business software
pull
statsd push
vmagent
vmstatsd (WIP)
remote write
remote write
VictoriaMetrics Cluster
PromQL
Grafana

zhihu ❤□□□□ OpenSource

1. 给 carbonapi 贡献 13 个 PR
2. 给 VictoriaMetrics 贡献 7 个 PR，包含 2 个查询计划优化 PR
3. 本项目开源在 https://github.com/zhihu/promate

总结

1. 将倒排索引引入 Graphite 体系，查询和存储性能产生质变
2. MateQL 让 PromQL 与 Graphite 融合，增强表达能力和性能
3. 推模型的 Statsd On Prometheus 对 Prometheus 生态进行补充

THANKS