UCLOUD优刻得 | intel

容器监控架构演进

王琼-YY直播-容器研发

UCLOUD优刻得 | intel

1. 监控遇上 Kubernetes

2. Prometheus 优化

3. VictoriaMetrics 实践

4. 性能优化

监控遇上 Kubernetes

YYMS 是公司内通用的监控报警系统，有完善的数据收集，展示和报警机制，但是 YYMS 并不支持k8s这种采集方案。k8s对Container进行了封装，拥有了Pod、Deployment、Namespace、Service等众多概念。与传统集群相比，k8s集群监控更加复杂：

- 监控维度更多，除了传统物理集群的监控，还包括核心服务监控（API server，Etcd等）、容器监控、Pod监控、Namespace监控等
- 监控对象动态可变，在集群中容器的销毁创建十分频繁，无法提前预置
- 监控指标随着容器规模爆炸式增长，如何处理及展示大量监控数据
- 随着集群动态增长，监控系统必须具备动态扩缩的能力

UCLOUD优刻得 | intel

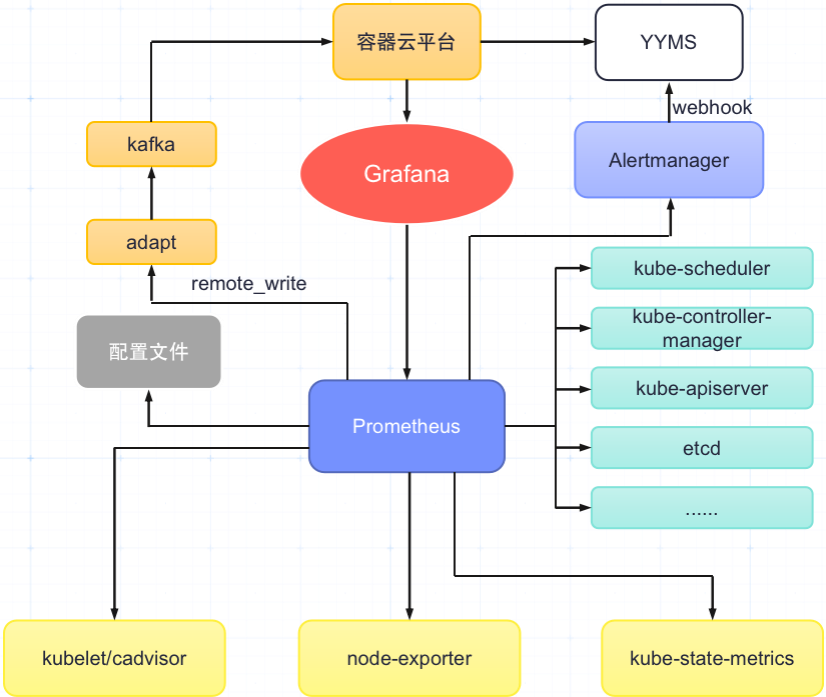
监控遇上 Kubernetes

目前容器云平台提供的k8s集群包括：

- 10+集群（云+物理机房+边缘）
- 1000+机器
- 2W+ Pod

监控系统架构

- kubernetes_sd_config 自动发现服务
- 通过remote_write协议将数据写至kafka，容器云平台通过消费kafka topic 获取容器基础监控数据。容器云平台定制告警规则发送给相应的服务负责人
- prometheus通过定期执行集群级别的告警规则，将触发告警的信息发送至alertmanger，alertmanager通过webhook将告警信息发送至YYMS
- grafana 可视化展现

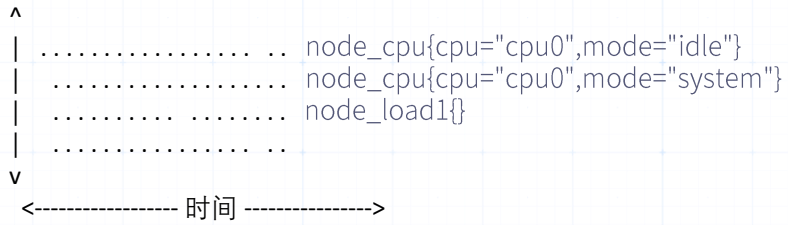


UCLOUD优刻得 | intel

存在的问题

- 无法横向扩展，数据无法长时间保存，大内存问题导致OOM
- 单点故障
- 无法聚合查询
- 维护成本太高

内存优化



在time_series中的每一个点称为一个样本（sample），样本由以下三部分组成：

- 指标(metric)：metric name和描述当前样本特征的labelsets
- 时间戳(timestamp)：一个精确到毫秒的时间戳;样本值(value)
- 一个float64的浮点型数据表示当前样本的值

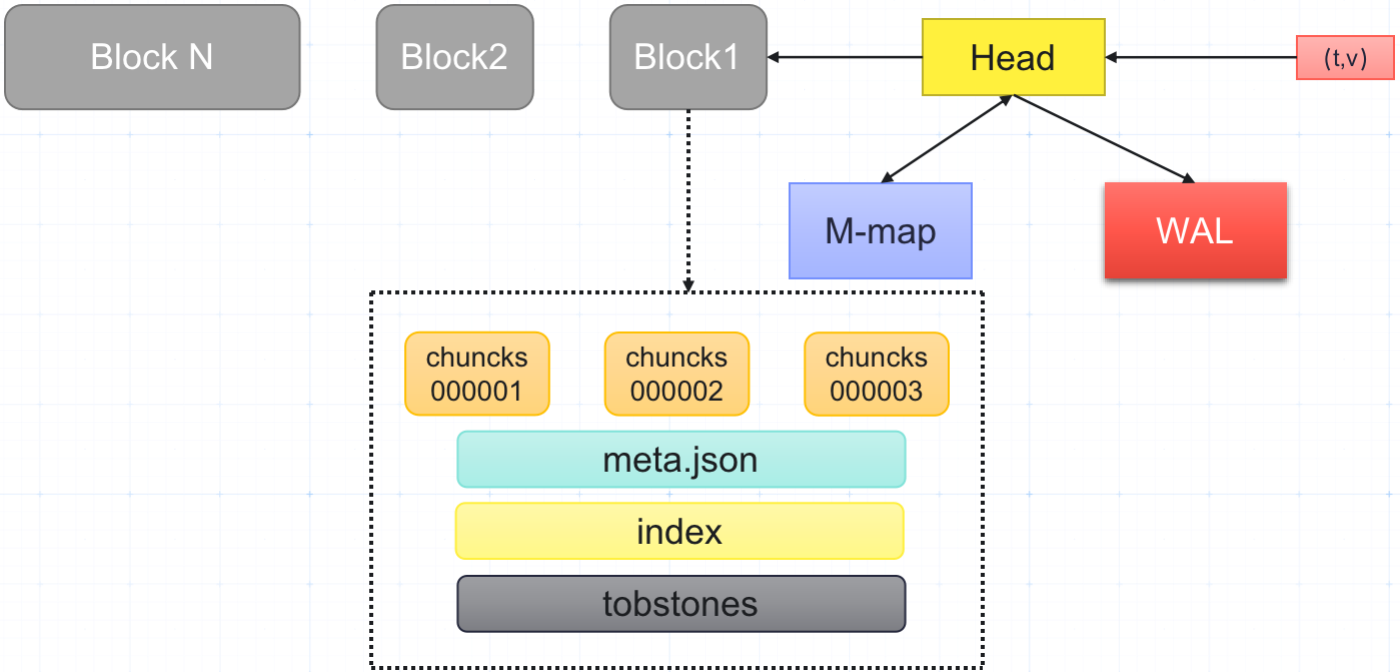
内存优化

<---------- metric ---------------------><-timestamp -><-value->
http_request_total{status="200", method="GET"}@1434417560938 => 94355
http_request_total{status="200", method="GET"}@1434417561287 => 94334

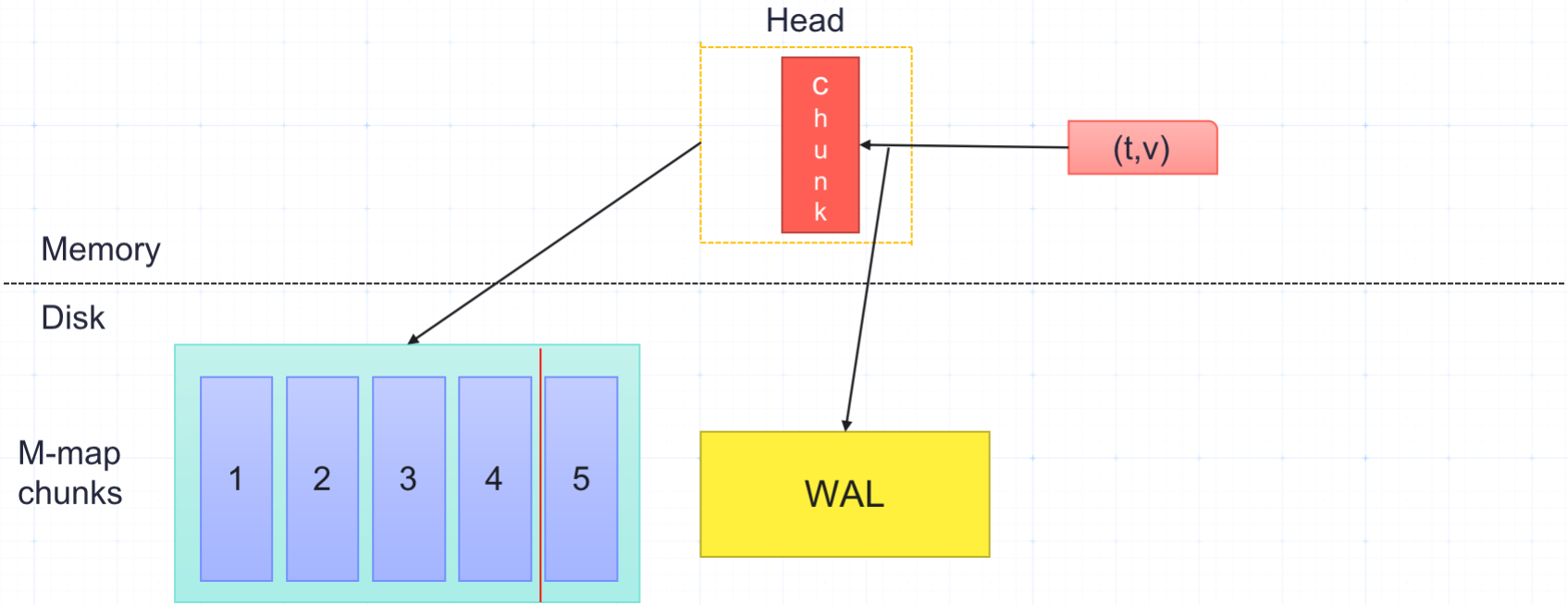
http_request_total{status="404", method="GET"}@1434417560938 => 38473
http_request_total{status="404", method="GET"}@1434417561287 => 38544

http_request_total{status="200", method="POST"}@1434417560938 => 4748
http_request_total{status="200", method="POST"}@1434417561287 => 4785

- http_request_total 通常称为 metric_name，大括号内的kv对称为labels，metric_name 是一个特殊的label，可以转换为{__name__="http_request_total"}，所有labels称为labelset
- 一个series由两部分组成：labelset + 一系列sample（timestamp，value）
- 一个series由labelset唯一确定



参考文档：https://ganeshvernekar.com/blog/prometheus-tsdb-the-head-block/


Head
C
h
u
n
k
(t,v)
Memory
Disk
M-map
chunks
1
2
3
4
5
WAL

UCLOUD优刻得 | intel

内存优化

- 降低样本数量
  - 查看当前每秒的样本数 rate(prometheus_tsdb_head_samples_appended_total[1h])
  - 调整 scrape_interval 参数
  - 减少指标数量
  - 减少单个样本数据的大小（一般情况下1-2个字节）
- 加载历史数据时，是从磁盘加载到内存，查询范围越大，内存越大
  - 查询尽量避免大范围查询，注意时间范围和 Step 比例
  - 不合理的查询条件，如 Group、大范围的 Rate
  - 大查询可以使用 RecordRule
- 加快block内存落盘时间
  - storage.tsdb.min-block-duration
  - maxSamplesPerChunk

计算指标需要多少内存
https://www.robustperception.io/how-much-ram-does-prometheus-2-x-need-for-cardinality-and-ingestion

- 去掉不需要的labels

```
metric_relabel_configs:
 - separator: ;
   regex: label_node_role_kubernetes_io_AutoCordon
   replacement: $1
   action: labeldrop
```

- 收集指定的metrics

```
relabel_configs:
 - source_labels: [__meta_kubernetes_service_label_k8s_app]
   separator: ;
   regex: etcd
   replacement: $1
   action: keep
```

- RuleRecord

```
groups:
- name: node-rules
  rules:
  - expr: (kube_pod_container_resource_requests_memory_bytes) * on (pod,cluster)
group_left(phase,label_node_role_kubernetes_io_NoSchedule) (kube_pod_status_phase{phase=~"Pending|Running"} == 1) * on
(node,cluster) group_left(label_biz_type,label_node_role_kubernetes_io_NoSchedule)
     max (kube_node_labels{label_biz_type!="",label_node_role_kubernetes_io_ingress=""}) by
(label_biz_type,label_node_role_kubernetes_io_NoSchedule,node,cluster)
    record: resource_requests_memory:label_biz_type
```

UCLOUD优刻得 | intel

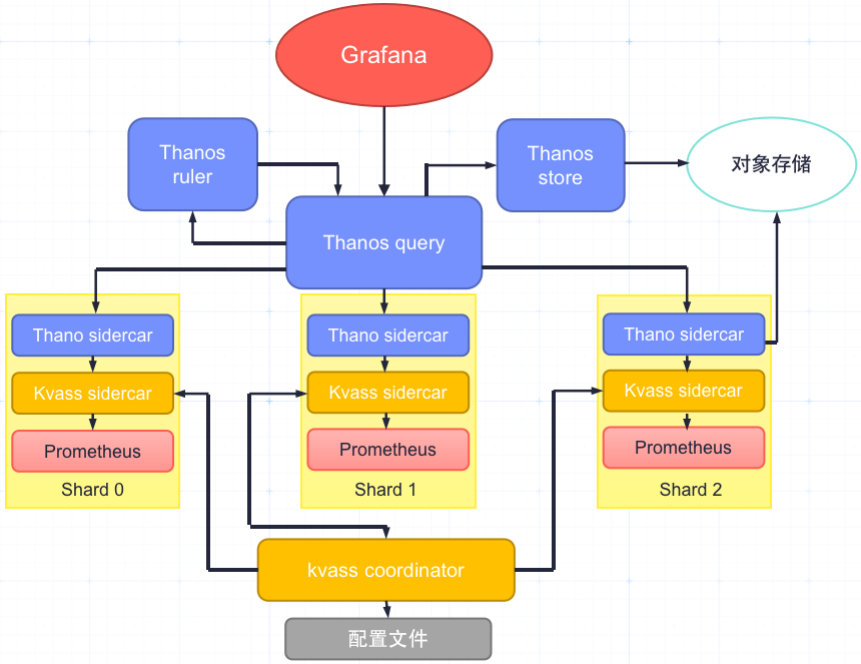
Thanos + Kvass

Kvass是一个Prometheus横向扩缩容解决方案

他使用Sidecar动态得根据Coordinator分配下来的target列表来为每个Prometheus生成只含特定target的配置文件，从而将采集任务分散到各个Prometheus分片。

Coordinator用于服务发现，target分配和分片扩缩容管理

- 配置管理简单：无需针对配置文件做任何特殊工作，单一小集群时候怎么用，现在就怎么用。
- 负载可控：由于Kvass在向分片配置target的时候，会根据target的实际规模来，而不是通过hashmod来，所以每个分片的总负载会被控制在一个阈值以下，不会出现某个分片OOM的情况。
- 自动扩缩容：Kvass会根据当前集群的规模，动态调整分片个数



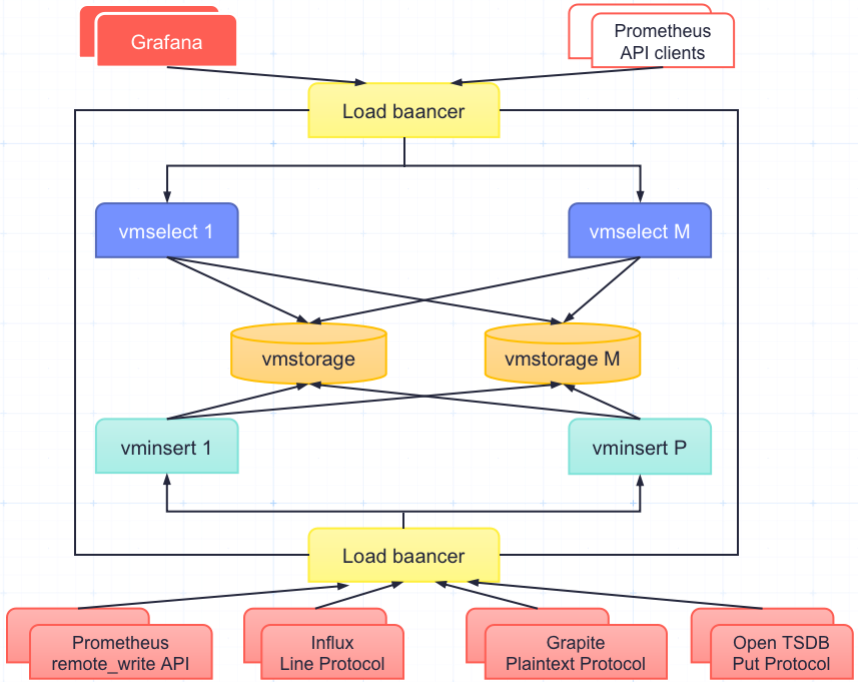
Thanos + Kvass

缺点

- 架构过于复杂，多集群的情况下不便于管理
- 分片节点prometheus挂掉导致部分数据丢失
- prometheus总体资源使用并没有减少

VictoriaMetrics 架构

- vmstorage : 存储数据
- vminsert : 通过 remote write API 接收来自 Prometheus 的数据并将其分布在可用的 vmstorage 节点上
- vmselect : 从 vmstorage 节点获取并聚合所需数据，返回给查询数据的客户端（如 Grafana）



VictoriaMetrics 功能特性

- 数据写入

  Prometheus将采集到的样本数据通过 Remote Write 的方式写入远程存储 VictoriaMetrics 中

- 数据查询

  VictoriaMetrics 提供了开箱即用的 Prometheues 查询 API

- 数据安全

  VictoriaMetrics 可以开启多副本数保障数据安全

- 可扩展性

  存储容量可以通过增加storage节点磁盘容量，或者增加storage节点数量进行扩容

- 可靠性

  vminsert与vmselect均为无状态组件，开启多副本的情况下，允许n-1个storage节点故障（n为副本数量）

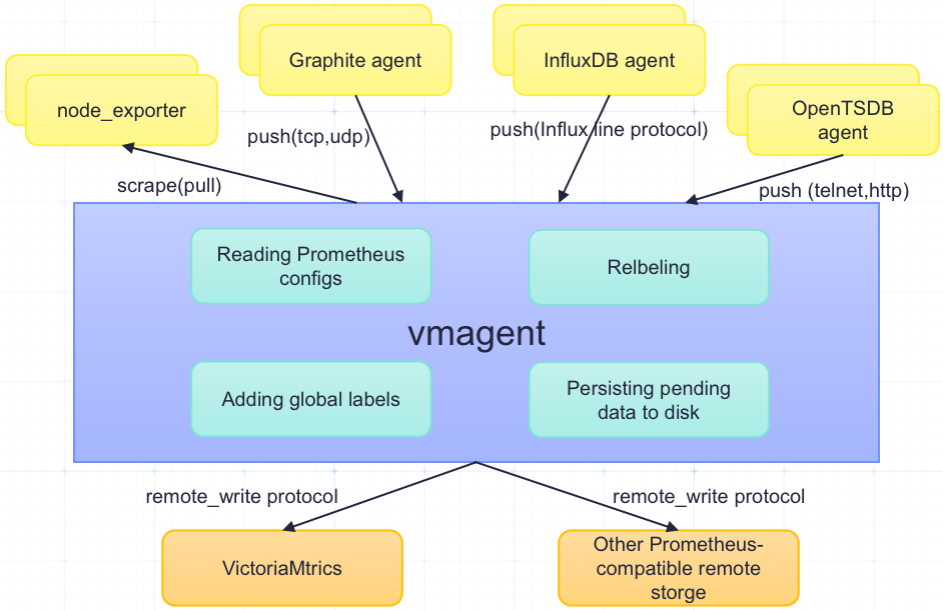
- 监控

  所有集群组件在TCP端口的/metrics页面上以Prometheus兼容的格式公开各种指标

Vmagent

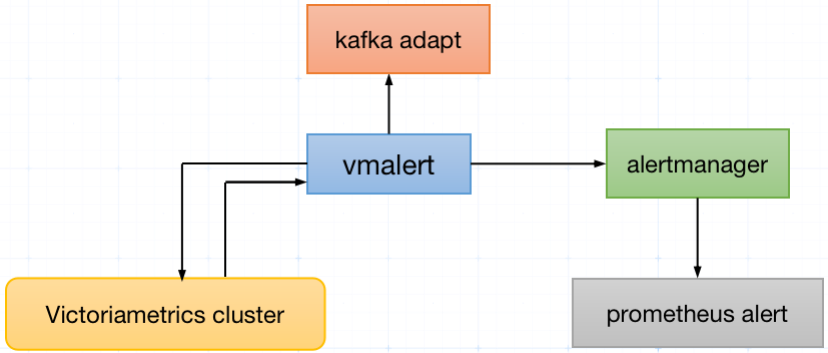
vmagent是一个轻量级但强大的代理，支持从多种来源收集指标，并将它们存储在VictoriaMetrics或任何其他支持remote_write协议的Prometheus兼容的存储系统中

- 可以作为Prometheus的直接替代品
- 可以将收集到的指标同时复制到多个远程存储系统
- 在与远程存储连接不稳定的环境中也能顺利工作
- 与Prometheus相比，使用的内存、CPU、磁盘IO和网络带宽都比较低
- 当必须抓取大量目标时，抓取目标可以分布在多个vmagent实例中
- 支持流模式

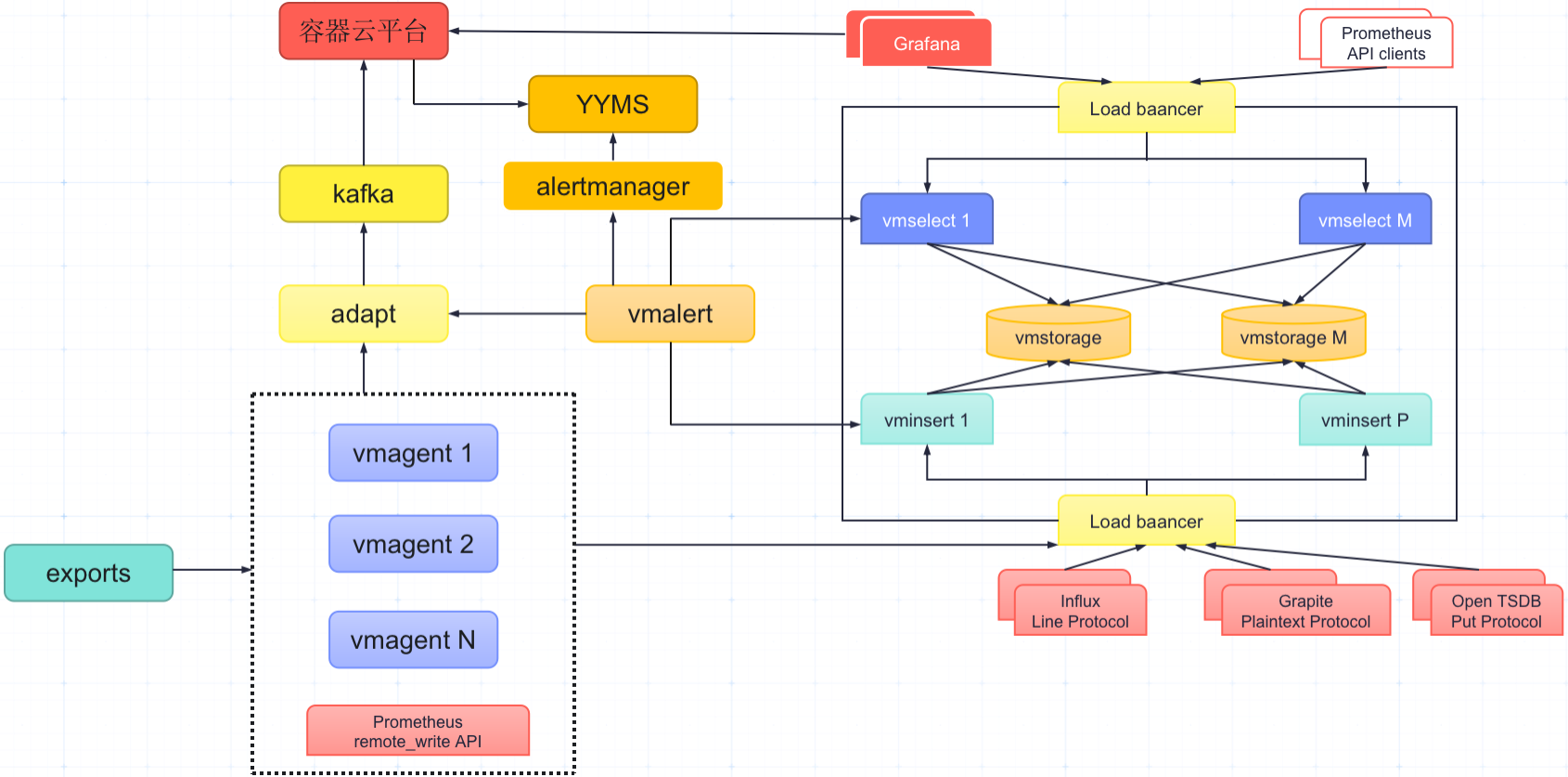


Vmalert

- 通过remoteWrite将告警状态写入远端存储持久化存储
- 通过remoteWrite将ruleRecord写入远端存储
- 通过remoteRead将告警发送至alertmanager



prometheus alert: https://github.com/feiyu563/PrometheusAlert.git
prometheus kafka adapt: https://github.com/Telefonica/prometheus-kafka-adapter.git


容器云平台
Grafana
Prometheus
API clients
YYMS
Load baancer
kafka
alertmanager
vmselect 1
vmselect M
adapt
vmalert
vmstorage
vmstorage M
vminsert 1
vminsert P
vmagent 1
vmagent 2
vmagent N
Load baancer
exports
Influx
Line Protocol
Grapite
Plaintext Protocol
Open TSDB
Put Protocol
Prometheus
remote_write API

UCLOUD优刻得 | intel

性能优化

Ingestion rate: 280K samples/sec
Active time series: 2.8 million
Samples scraped and stored: 24.5 billion

- 存储
  VictoriaMetrics: 7.2GB . This translates to 0.3 bytes per sample (7.2GB/24.5 billion samples)
  Prometheus: 52.3GB (32.3GB data plus 18GB WAL). This translates to 52.3GB/24.5 billion samples = 2.1 bytes per sample

- 内存
  VictoriaMetrics: 4.3GB
  Prometheus: 23GB

参考文档：https://valyala.medium.com/prometheus-vs-victoriametrics-benchmark-on-node-exporter-metrics-4ca29c75590f

总体架构

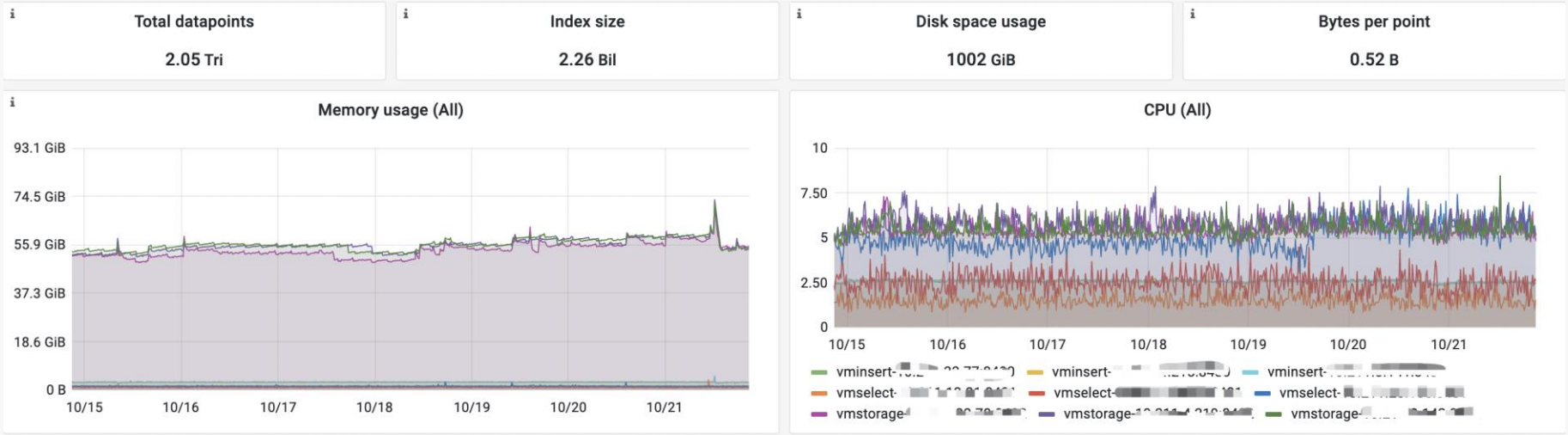

平均每秒抓取的samples数量
2 Mil
1.50 Mil
1 Mil
500 K
0
10/14
10/15
10/16
10/17
10/18
10/19
10/20
avg
current
{type="promremotewrite"}
1.52 Mil
1.53 Mil


平均查询时间
40 s
30 s
20 s
10 s
0 s
10/14
10/15
10/16
10/17
10/18
10/19
10/20
avg
current
0.99 (/api/v1/labels)
1.41 s
23.0 s
0.5 (/api/v1/labels)
1.40 s
22.2 s


平均每秒新 time-series 创建数量
50 K
40 K
30 K
20 K
10 K
0
10/14
10/15
10/16
10/17
10/18
10/19
10/20
churn rate


过去一小时活动的 time series
100 Mil
80 Mil
60 Mil
40 Mil
20 Mil
0
10/14
10/15
10/16
10/17
10/18
10/19
10/20
Active time series

总体架构


Total datapoints
2.05 Tri
Index size
2.26 Bil
Disk space usage
1002 GiB
Bytes per point
0.52 B
Memory usage (All)
93.1 GiB
74.5 GiB
55.9 GiB
37.3 GiB
18.6 GiB
0 B
10/15
10/16
10/17
10/18
10/19
10/20
10/21
CPU (All)
10
7.50
5
2.50
0
10/15
10/16
10/17
10/18
10/19
10/20
10/21
vminsert-
vminsert-
vminsert-
vmselect-
vmselect-
vmselect-
vmstorage-
vmstorage-
vmstorage-

UCLOUD优刻得 | intel

THANKS!