07 | 如何解决Prometheus的存储容量问题？

2023-01-23 秦晓辉 来自北京

《运维监控系统实战笔记》

讲述：秦晓辉

时长 15:02　大小 13.73M

你好，我是秦晓辉。

前几讲我们介绍了 Prometheus 的关键设计，这些设计都非常优秀。在云原生监控领域，有不可撼动的江湖地位，那这样说来 Prometheus 是不是就没有缺点了呢？是否可以满足所有使用场景呢？显然也不是。

一个软件如果什么问题都想解决，就会导致什么问题都解决不好。所以 Prometheus 也存在一些不足之处，其中一个广受诟病的问题就是**单机存储不好扩展**。所以这一讲我们就针对这个问题来聊聊如何扩展 Prometheus 的存储。

所有场景都需要扩展容量吗？

虽然我们聊的是 Prometheus 容量扩展问题，不过我必须先说明一点，大部分场景其实不需要扩展，因为一般的数据量压根达不到 Prometheus 的容量上限。很多中小型公司使用单机版本

的 Prometheus 就足够了，这个时候不要想着去扩展，容易过度设计，引入架构上的复杂度问题。

Prometheus 单机容量上限是多少？根据我的经验，每秒接收 80 万个数据点，算是一个比较健康的上限，一开始也无需用一台配置特别高的机器，随着数据量的增长，可以再升级硬件的配置。当然，如果想要硬件方便升配，就需要借助虚拟机或容器，同时需要使用分布式块存储。

每秒接收 80 万个数据点是个什么概念呢？每台机器每个周期大概采集 200 个系统级指标，比如 CPU、内存、磁盘等相关的指标。假设采集频率是 10 秒，平均每秒上报 20 个数据点，可以支持同时监控的机器量是 4 万台。

$$800000 \div 20 = 40000$$

可以看出，每秒接收 80 万数据点，其实是一个很大的容量了。当然，如果使用 node-exporter，指标数量要多于 200，800 左右，那也能支持 1 万台机器的监控。

不过刚刚我们只计算了机器监控数据，如果还要用这个 Prometheus 监控各类中间件，那就得再做预估计算了。有些中间件会吐出比较多的指标，有些指标其实用处不大，可以丢掉（drop），当然这就是另一个话题了，后面监控实战部分我们会详细讲解，这里暂不展开。

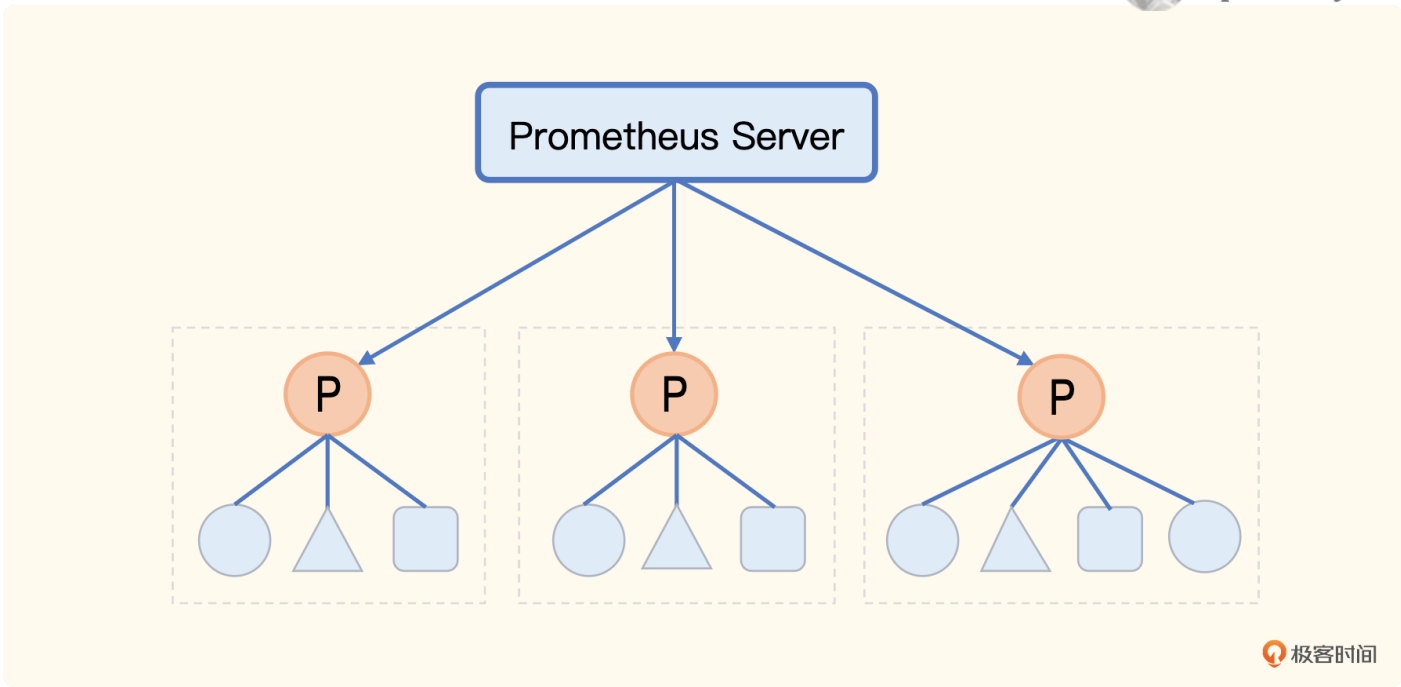
如果单个 Prometheus 实在是扛不住，也可以拆成多个 Prometheus，根据业务或者地域来拆都是可以的，这就是下面要介绍的 Prometheus 联邦机制。

Prometheus 联邦机制

联邦机制可以理解为是 Prometheus 内置支持的一种集群方式，核心就是 Prometheus 数据的级联抓取。比如某公司有 8 套 Kubernetes，每套 Kubernetes 集群都部署了一个 Prometheus，这 8 个 Prometheus 就形成了 8 个数据孤岛，没法在一个地方看到 8 个 Prometheus 的数据。

当然，我们用 Grafana 或 Nightingale，把这 8 个 Prometheus 作为数据源接入，然后就可以在一个 Web 上通过切换数据源的方式查看不同的数据了，但本质还是分别查看，没法做多个 Prometheus 数据的联合运算。

而联邦机制，一定程度上就可以解决这个问题，把不同的 Prometheus 数据聚拢到一个中心的 Prometheus 中，你结合着这个架构图来理解一下我说的这种方法。



原本一个 Prometheus 解决不了的问题，拆成了多个，然后又把多个 Prometheus 的数据聚拢到中心的 Prometheus 中。但是，中心的 Prometheus 仍然是个瓶颈。所以在联邦机制中，中心端的 Prometheus 去抓取边缘 Prometheus 数据时，不应该把所有数据都抓取到中心，而是应该**只抓取那些需要做聚合计算或其他团队也关注的指标**，大部分数据还是应该下沉在各个边缘 Prometheus 内部消化掉。

怎么做到只抓取特定的指标到中心端呢？通过 `match[]` 参数，指定过滤条件就可以实现，下面是中心 Prometheus 的抓取规则。

复制代码

```
scrape_configs:
  - job_name: 'federate'
    scrape_interval: 30s
    honor_labels: true
    metrics_path: '/federate'
    params:
      'match[]':
        - '{__name__=~"aggr:.*"}'
    static_configs:
      - targets:
        - '10.1.2.3:9090'
        - '10.1.2.4:9090'
```

边缘 Prometheus 会在 `/federate` 接口暴露监控数据，所以设置了 metrics_path，honor_labels 设置为 true，意思是在标签重复时，以源数据的标签为准。过滤条件中通过正则匹配过滤出所有 `aggr:` 打头的指标，这类指标都是通过 Recoding Rules 聚合出来的关键指标。当然，这是我假设的一个规范。

联邦这种机制，可以落地的核心要求是，**边缘 Prometheus 基本消化了绝大部分指标数据**，比如告警、看图等，都在边缘的 Prometheus 上搞定了。只有少量数据，比如需要做聚合计算或其他团队也关注的指标，被拉到中心，这样就不会触达中心端 Prometheus 的容量上限。这就要求公司在使用 Prometheus 之前先做好规划，建立规范。说实话可能实施起来会有点儿难，所以我更推荐下面的远程存储方案。

远程存储方案

默认情况下，Prometheus 收集到监控数据之后是存储在本地，在本地查询计算。由于单机容量有限，对于海量数据场景，需要有其他解决方案。最直观的想法就是：既然本地搞不定，那就在远端做一个集群，分治处理。

Prometheus 本身不提供集群存储能力，可以复用其他时序库方案。时序库挺多的，如果挨个儿去对接比较费劲，于是 Prometheus 建立了统一的 Remote Read、Remote Write 接口协议，只要满足这个接口协议的时序库都可以对接进来，作为 Prometheus remote storage 使用。

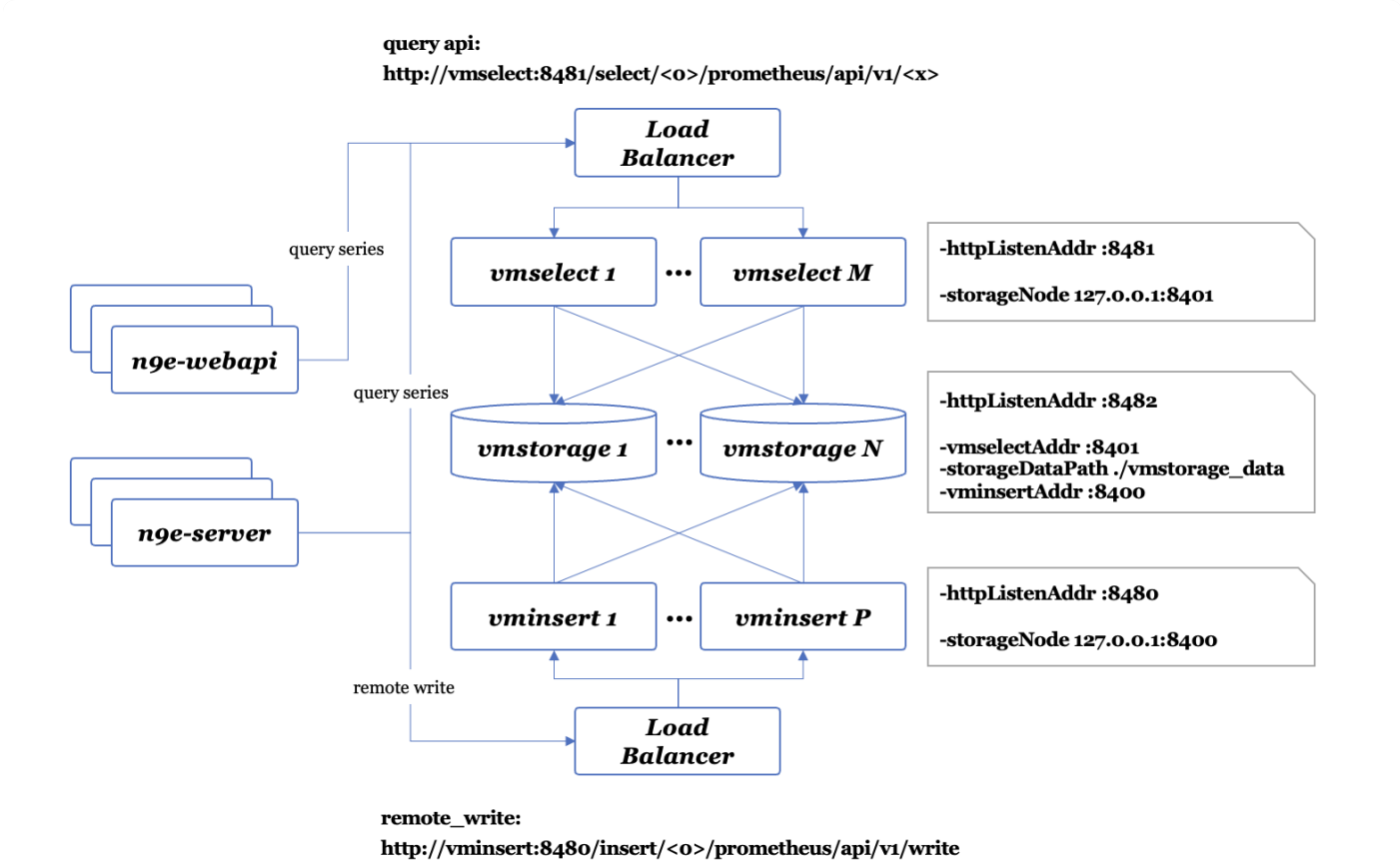
目前国内使用最广泛的远程存储主要是 VictoriaMetrics 和 Thanos。下面我们简单介绍一下。

VictoriaMetrics

VM 虽然可以作为 Prometheus 的远程存储，但它志不在此。VM 是希望成为一个更好的 Prometheus，所以它不只是时序库，它还有抓取器、告警等各个组件，体系非常完备，不过现在国内基本上还只是把它当做时序库在用。

VM 作为时序库，核心组件只有 3 个，vmstorage、vminsert 和 vmselect。其中 vmstorage 用于存储时序数据，vminsert 用于接收时序数据并写入到后端的 vmstorage，Prometheus 使用 Remote Write 对接的就是 vminsert 的地址，vmselect 用于查询时序数据，它没有实现 Remote Read 接口，反倒实现了 Prometheus 的 Querier 接口。这是为什么呢？

因为 VM 觉得 Remote Read 设计太“挫”了，性能不好，还不如直接实现 Querier 接口。这样设计的话，Grafana 这类前端应用直接可以和 vmselect 对接，而不用在中间加一层 Prometheus ，我们可以看一下 VM 的架构。



图片来自网络

n9e-webapi 和 n9e-server 是 Nightingale 的两个模块，都可以看做 VM 的上层应用。通过这张图片，我们可以比较清楚地看出 VM 的架构以及与上层应用的交互方式。

n9e-webapi 通过 vmselect 查询时序数据，vmselect 是无状态模块，可以水平扩展，通常部署多个实例，前面架设负载均衡，所以 n9e-webapi 通常是对接 vmselect 的负载均衡。n9e-server 也有一些查询 VM 的需求，先不用关注，重点关注 Remote Write 那条线，n9e-server 通过 Remote Write 协议，把数据转发给 vminsert 的负载均衡，vminsert 也是无状态的，可以水平扩展。

vmstorage 是存储模块，可以部署多个组成集群，只要把所有 vmstorage 的地址告诉 vmselect 和 vminsert，整个集群就跑起来了，非常简单。

问题是数据通过 vminsert 进来之后，如何分片？vmselect 和 vminsert 之间没有任何关系，vmselect 在查询具体某个指标数据的时候，怎么知道数据位于哪个 vmstorage 呢？这个架构

虽然看起来简单，但总感觉跟常见的分布式系统不太一样呢。

主要是因为 VM 采用了一种叫做 merge read 的方案，一个查询请求发给 vmselect 之后，vmselect 会向所有的 vmstorage 发起查询请求。注意，是所有的 vmstorage，然后把结果合并在一起，返回给前端。所以，vmselect 压根就不用关心数据位于哪个 vmstorage。此时 vminsert 用什么分片算法都无所谓了，数据写到哪个 vmstorage 都行，反正最后都会 merge read。

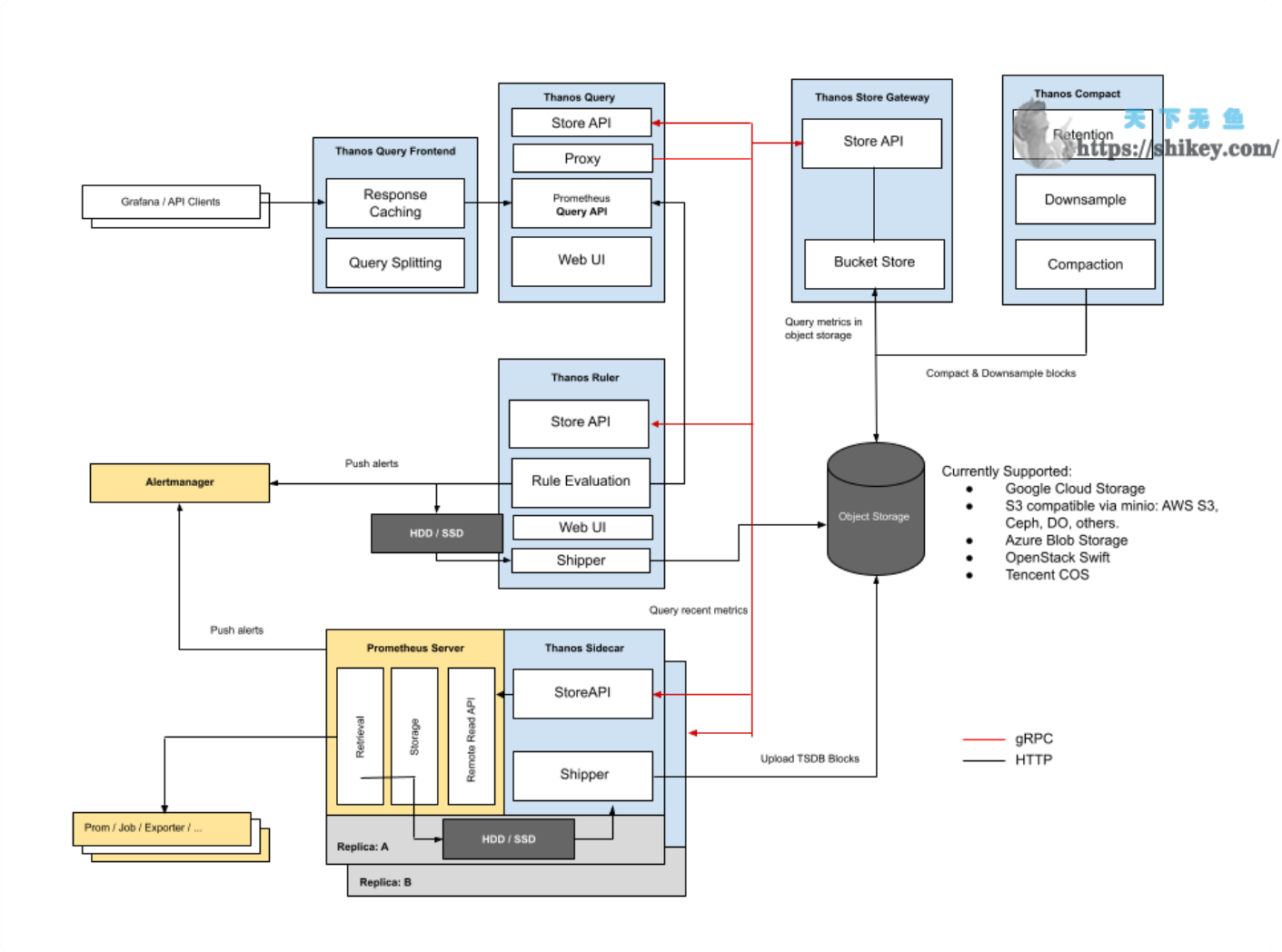
这个奇葩设计，看起来还挺有效的。有没有问题呢？显然是有的，就是 vmstorage 集群不能太大。如果有几千个节点，随便一个查询过来，vmselect 都会向所有 vmstorage 发起查询，任何一个节点慢了都会拖慢整体速度，这就让人无法接受了。一般来讲，一个 vmstorage 集群，有一二十个节点还是比较健康的，这个容量就已经非常大了，能满足大部分公司的需求，所以这不是个大问题。

我个人比较建议你在选型远程存储的时候使用 VictoriaMetrics，架构简单，更有掌控力。像 M3 虽然容量比 VM 大得多，但是架构复杂，出了问题反而无从着手，不建议使用。

在 Prometheus 存储问题的解决方案中，除了 VM，还有一个影响力比较大的就是 Thanos，也就是传说中的灭霸。下面我们看看灭霸是个什么招式。

Thanos

Thanos 的做法和 VM 不同，Thanos 完全拥抱 Prometheus，对 Prometheus 做了一个增强，核心特点是使用**对象存储**做海量时序存储。你看下它的架构图。

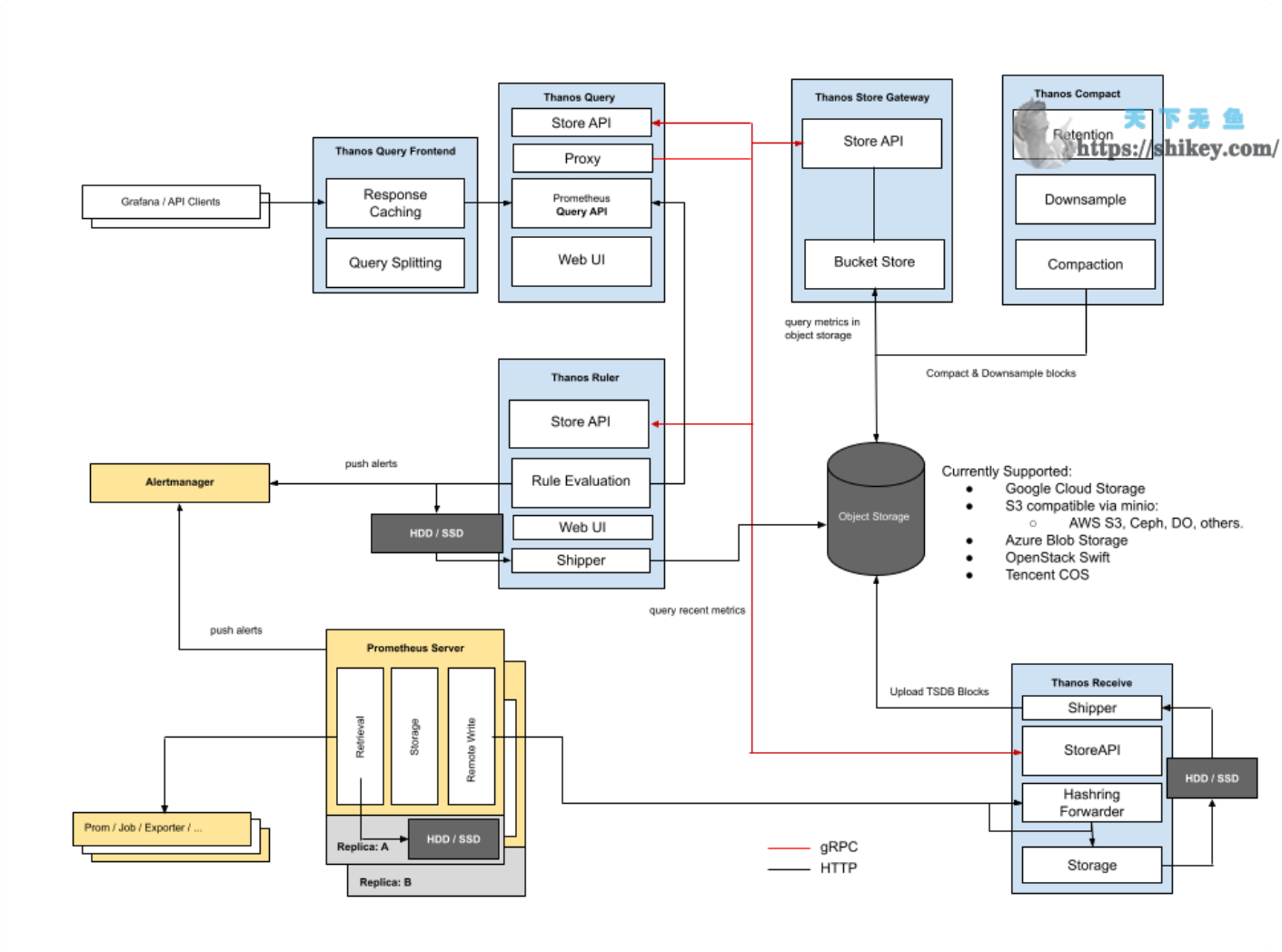


图片来自 Thanos 官网

这个架构图初看起来比较复杂，黄色部分是 Prometheus 自身，蓝色部分是 Thanos，黑色部分是存储。这里边有几个核心点。

1. 每个 Prometheus 都要伴生一个 Thanos Sidecar 组件，这个组件有两个作用，一是响应 Thanos Query 的查询请求，二是把 Prometheus 产生的块数据上传到对象存储。
2. Thanos Sidecar 调用 Prometheus 的接口查询数据，暴露为 StoreAPI，Thanos Store Gateway 调用对象存储的接口查询数据，也暴露为 StoreAPI。Thanos Query 就可以从这两个地方查询数据了，相当于近期数据从 Prometheus 获取，比较久远的数据从对象存储获取。

虽然对象存储比较廉价，但这个架构看起来还是过于复杂了，没有 VM 看起来干净。另外这个架构是和 Prometheus 强绑定的，没法用作单独的时序存储，比较遗憾。不过好在 Thanos 还有另一种方案，不用 Sidecar，使用 Receive 模块，来接收 Remote Write 协议的数据，写入本地，同时上传对象存储，你看一下这个架构。



图片来自 Thanos 官网

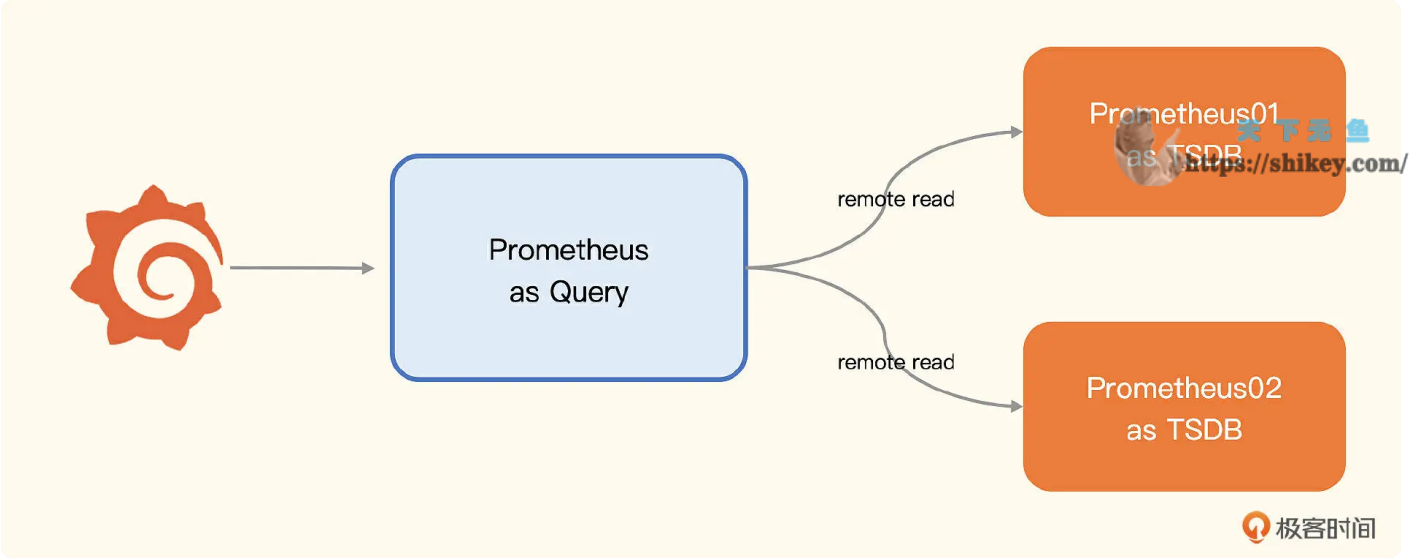
从存储角度来看，这个架构和 Prometheus 就没有那么强的绑定关系了，可以单独用作时序库。关键点还是对象存储，虽然对象存储是海量、廉价的，但是延迟较高，而且一个请求拆成两部分，一部分查本地，一部分查对象存储，也没有那么可靠。

相比日志之类的数据，指标数据的体量较小，一般存储 3 个月就够了，所以对象存储的优势就显得没那么大了。如果是我来选型，VictoriaMetrics 和 Thanos 之间，我会选择前者。

所谓的远程存储方案，核心就是 Remote Read/Write，其实 Prometheus 自身也可以被别的 Prometheus 当做 Remote storage，只要开启 `--enable-feature=remote-write-receiver` 这个参数即可。下面我们来看一下 Prometheus 自身怎么来搭建集群。

Prometheus 自身搭建集群

废话不多说，直接上架构图。



图上有三个 Prometheus 进程，有两个用作存储，有一个用作查询器，用作查询器的 Prometheus 通过 Remote Read 的方式读取后端多个 Prometheus 的数据，通过几行 remote_read 的配置就能实现。

```
remote_read:
  - url: "http://prometheus01:9090/api/v1/read"
    read_recent: true
  - url: "http://prometheus02:9090/api/v1/read"
    read_recent: true
```

Prometheus 的 remote_read 也是 merge read，跟 vmselect 逻辑类似。不过 VM 集群是有副本机制的，使用 Prometheus 来做集群，不太好做副本。当然，可以粗暴地为每个分片数据部署多份采集器和 PromTSDB，也基本可以解决高可用的问题。

在实际生产环境中，如果所有数据都是通过拉的方式来收集，这种架构也是可以尝试的，不过我看到大部分企业都是推拉结合，甚至推是主流，Remote Write 到一个统一的时序库集群，是一个更加顺畅的方案。

小结

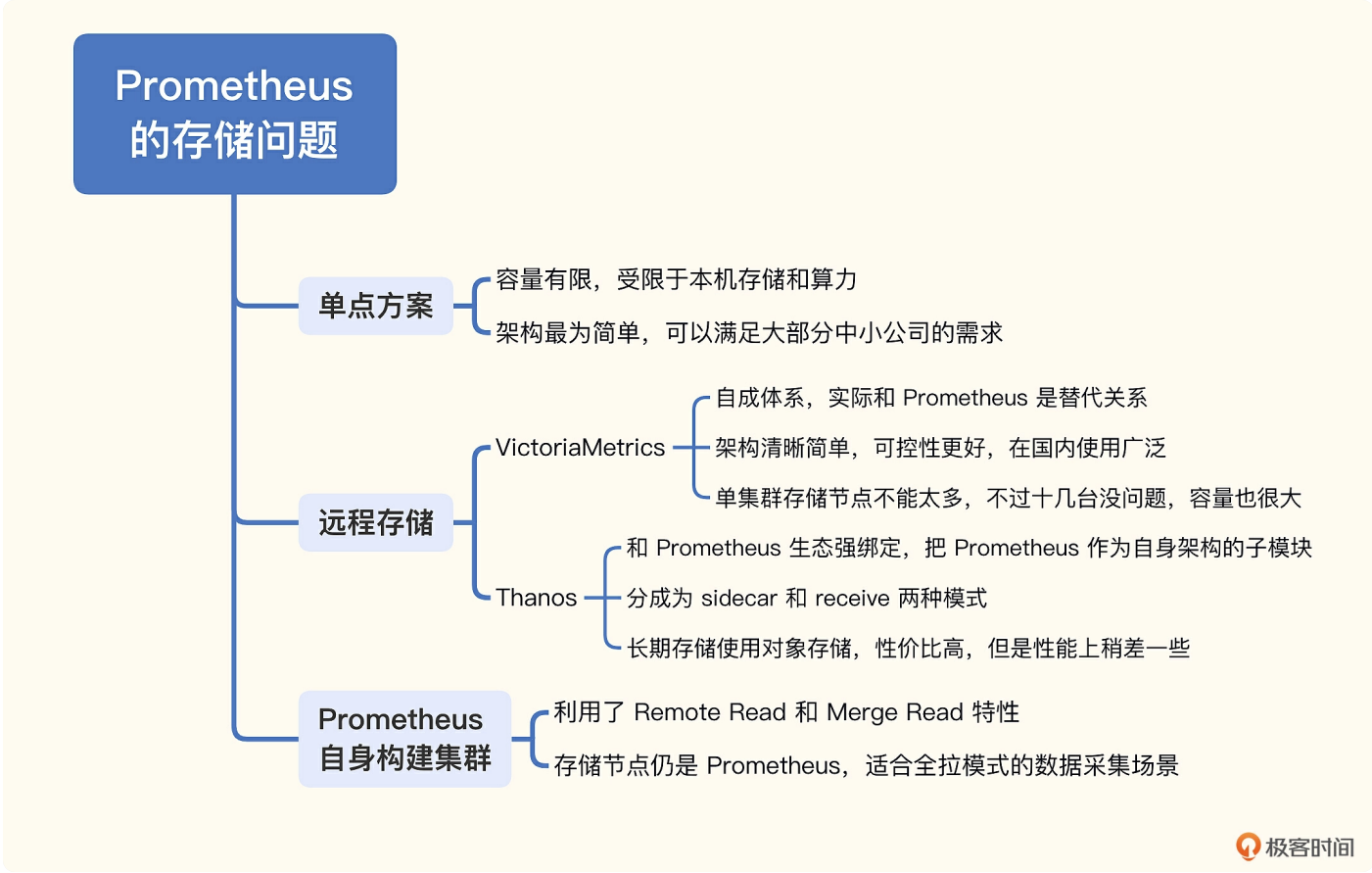
这一讲我们重点关注了 Prometheus 生态常见的存储扩展问题，并给出了 3 种集群解决方案。

- Prometheus 联邦集群：按照业务或者地域，拆成多个边缘 Prometheus，然后在中心搭建一个 Prometheus，把一些重要的多团队关注的指标或需要二次计算的指标拉到中心。

- 远程存储方案：通过 Remote Read/Write 协议，Prometheus 可以和第三方存储对接，把存储的难题抛给了第三方来解决，常用方案是 M3DB、VictoriaMetrics、Thanos。我最推荐的是 VictoriaMetrics，架构简单，更可控一些。
- Prometheus 自身搭建集群：也是利用了 Remote Read/Write 机制，只是把存储换成了多个 Prometheus，对于全部采用拉模型抓取数据的公司，是可以考虑的方案。

这三种方案很好地解决了 Prometheus 的存储问题，你可以根据实际情况来选用。但还是要注意一点，不要做过度设计。如果数据量不大，就无需搭建时序集群，徒增维护成本。

这一讲的内容我也整理了一个脑图，帮你理解和记忆。



互动时刻

业界时序存储其实远不止 VictoriaMetrics 和 Thanos，你们公司选择的是哪一款？使用体验如何？欢迎你在评论区聊一聊，我们在思想上碰撞一下，对你后续选型会有很大帮助，也欢迎你把今天的内容分享给身边的朋友，邀他一起学习。我们下一讲再见！

生成海报并分享

赞 7　提建议

上一篇 06丨PromQL有哪些常见的使用场景？

下一篇 08丨如何用 Nightingale 解决 Prometheus 的告警管理问题？

精选留言 (8)

写留言

hshopeful
2023-01-23 来自上海

几年前在的公司是参考opentsdb，底层使用hbase 自研的

作者回复: 这是个大厂

1

DBRE 酥
2023-02-02 来自北京

Prometheus 推荐的高可用方案吗？

光
2023-02-01 来自江苏

grafana 公司出品的mimir是否可以评价下设计优势和场景。

作者回复: Mimir是fork自Cortex，后面选型可以不用考虑Cortex了，Mimir和Thanos架构类似，核心的选型考虑是license，一个是agplv3，一个是apache2。VM和Mimir/Thanos相比，如果是硬盘存储，就用VM，如果是对象存储，就用Mimir/Thanos

ATony
2023-01-31 来自上海

测试了VictoriaMetrics和thanos 的数据查询，VictoriaMetrics是快点，同一条promql 在Victori

aMetrics和thanos上查询，某一个时间点数据结果不一样，对比了prometheus的监控数据，thanos反而更为准确，不知道为什么?

leeeo

2023-01-29 来自四川

VM作后端，当扩展存储节点的时候，是否会导致原有节点的数据重新分布呢？

作者回复: 不会，vm是merge read，数据在哪个node都无所谓，所以就无需挪动数据了

共 2 条评论>

胡飞

2023-01-29 来自上海

老师你好，如果是单机方案，也可以使用云存储吧，不一定非要使用第三方时许数据库搭配云存储吧？

作者回复: 单机的话，Prometheus默认把数据存到data目录，data目录如果是云存储，数据自然就是写到云存储了哈

peter

2023-01-23 来自北京

请教老师几个问题：

Q1：vmselect的负载均衡，一般用什么？

Q2：thanos的对象存储，是云存储吗？比如阿里云一类的。如果是的话，相当于thanos需要借助于第三方了。

Q3：大公司是怎么使用prometheus的？

对于中小公司，一个prometheus单机就够了，但是，对于大厂，比如阿里、京东一类的公司，会怎么使用prometheus？

Q4：单机prometheus的本地硬盘一般多大？10T吗？

作者回复: 1，一般用lvs之类的

2，是

3，一般会分成多个集群，有些集群也是单机prometheus就上了，有些是vm thanos之类做存储，还有些自研时序库

4，一般没那么大，存15天，一般也就是几百G

无名无姓

2023-01-23 来自北京

看来可以集群模式