分布式监控调研

监控实现：Prometheus + Grafana + Influxdb

Reference

https://yunlzheng.gitbook.io/prometheus-book/（对应github 仓库）

https://cloud.tencent.com/document/product/1416

Introduction

简而言之，prometheus 实现监控，grafana 实现数据可视化，influxdb 扩展 prometheus 本地存储

- prometheus

  Prometheus是一个开源的完整监控解决方案，其对传统监控系统的测试和告警模型进行了彻底的颠覆，形成了基于中央化的规则计算、统一分析和告警的新模型。 相比于传统监控系统Prometheus具有以下优点：

  - 易于管理：Prometheus核心部分只有一个单独的二进制文件，不存在任何的第三方依赖(数据库，缓存等等)。唯一需要的就是本地磁盘，因此不会有潜在级联故障的风险。
  - 监控服务的内部运行状态
  - PromQL查询：轻松回答 “在过去一段时间中95%应用延迟时间的分布范围”、“预测在4小时后，磁盘空间占用大致会是什么情况？”、“CPU占用率前5位的服务有哪些？(过滤)”等问题
  - 可扩展
  - 易于集成：使用Prometheus可以快速搭建监控服务，并且可以非常方便地在应用程序中进行集成。目前支持： Java， JMX， Python， Go，Ruby， .Net， Node.js等等语言的客户端SDK，基于这些SDK可以快速让应用程序纳入到Prometheus的监控当中，或者开发自己的监控数据收集程序。同时这些客户端收集的监控数据，不仅仅支持Prometheus，还能支持Graphite这些其他的监控工具。

    同时Prometheus还支持与其他的监控系统进行集成：Graphite， Statsd， Collected，Scollector， muini， Nagios等。

    Prometheus社区还提供了大量第三方实现的监控数据采集支持：JMX， CloudWatch， EC2，MySQL， PostgresSQL， Haskell， Bash， SNMP， Consul， Haproxy， Mesos，Bind， CouchDB， Django， Memcached， RabbitMQ， Redis， RethinkDB，Rsyslog等等。

    说明我们可以有很多扩展的方面
  - 可视化

- grafana

  Prometheus中的Graph面板可查询数据形成图表。但是缺点也很明显，这些查询结果都是临时的，无法持久化的，更别说我们想实时关注某些特定监控指标的变化趋势。

为了简化这些问题Prometheus内置了一个简单的解决方案Console Template，它允许用户通过Go模板语言创建任意的控制台界面，并且通过Prometheus Server对外提供访问路径。

Console Teamplet虽然能满足一定的可视化需求，但是也仅仅是对Prometheus的基本能力的补充。同时使用也会有许多问题，首先用户需要学习和了解Go Template模板语言，其它其支持的可视化图表类型也非常有限，最后其管理也有一定的成本。而Grafana则可以让我们以更简单的方式创建更加精美的可视化报表。

- influxdb

  Prometheus Server本身就是一个时序数据库，将采集到的监控数据按照时间序列的方式存储在本地磁盘当中。

  Prometheus的本地存储设计可以减少其自身运维和管理的复杂度，同时能够满足大部分用户监控规模的需求。但是本地存储也意味着Prometheus无法持久化数据，无法存储大量历史数据，同时也无法灵活扩展和迁移。

  为了保持Prometheus的简单性，Prometheus并没有尝试在自身中解决以上问题，而是通过定义两个标准接口(remote_write/remote_read)，让用户可以基于这两个接口对接将数据保存到任意第三方的存储服务中，这种方式在Promthues中称为Remote Storage。

  目前Prometheus社区也提供了部分对于第三方数据库的Remote Storage支持：

存储服务	支持模式
AppOptics	write
Chronix	write
Cortex:	read/write
CrateDB	read/write
InfluxDB	read/write
OpenTSDB	write
PostgreSQL/TimescaleDB:	read/write
SignalFx	write

  其中 influxdb 时序数据库是较为常见的选择

Web 服务搭建：OpenResty

OpenResty 是一个基于 Nginx 与 Lua 的高性能 Web 平台，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。用于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

OpenResty通过汇聚各种设计精良的 Nginx 模块（主要由 OpenResty 团队自主开发），从而将 Nginx 有效地变成一个强大的通用 Web 应用平台。这样，Web 开发人员和系统工程师可以使用 Lua 脚本语言调动 Nginx 支持的各种 C 以及 Lua 模块，快速构造出足以胜任 10K 乃至 1000K 以上单机并发连接的高性能 Web 应用系统。

OpenResty的目标是让你的Web服务直接跑在 Nginx 服务内部，充分利用 Nginx 的非阻塞 I/O 模型，不仅仅对 HTTP 客户端请求,甚至于对远程后端诸如 MySQL、PostgreSQL、Memcached 以及 Redis 等都进行一致的高性能响应。

但我其实不是很懂 openresty 是怎么和 prometheus 产生关联的，看得最多的是 nginx 暴露 exporter 给 prometheus 监控，那是不是要我们用 nginx 重新搭建 disgrafs 的整个集群?

具体项目

伴鱼数据库之监控系统

prometheus-lua-nginx

https://github.com/gebiWangshushu/Hei.PrometheusFileBaseServiceDiscover

github上还有不少 openresty 结合 prometheus 的项目，但是对于如何利用 prometheus 实现拦截的案例暂未找到

个人想法

鉴于 disgrafs 的不稳定性，可以考虑另找现有的较为稳定分布式系统实现监控系统。当然，如果我们能将监控系统实现在 disgrafs 上并利用该监控系统找出 disgrafs 容易宕机的原因，再解决它，这自然是非常好的。这还是得找老师要点建议