OSH 大作业第三次调研

监控系统

彻底吃透监控系统，就这一篇！_zl1zl2zl3的博客-CSDN博客

监控系统的功能

- 监控运行系统、平台等的实时状态，保证系统能稳定安全地运行
- 收集运行时的信息
- 通过这些信息，分析结果，预测风险，采取行动
- 如果风险较大，进行故障预警
- 一旦发生故障，立刻发出告警信息
- 通过监控的数据，找到故障在哪里，协助生成解决方案
- 可视化监控数据，便于统计、导出 与 分析

`Prometheus` 的本质是 时间序列数据库（后面会展开说），能够很好地支持 大量数据的写入。

它采用拉（ Pull ）的模式拉取数据，并通过 Alert 模块实现监控预警。

`Prometheus` 的一些特点：①后端采用 `Golang` 开发，前端是 `Grafana`，`JSON`编辑即可解决，定制化难度低；②适合云环境的监控，对 `OpenStack` 、`Kubernetes` 有更好的集成；③安装相对复杂，监控、告警 和 界面 分属不同组件；④图形化界面不强，有些配置需要修改文件。

Openresty

可能用得上的教程 **tutorial · OpenResty** 官方博客

B站官号 **OpenResty**官方的个人空间哔哩哔哩**bilibili**

Openresty的简介与优势

浅谈 OpenResty (linkedkeeper.com)

`Openresty` 是基于 `nginx` 打造的 Web 服务器，可以运行 `Lua` 脚本语言。

(OpenResty is a web platform based on nginx which can run Lua scripts using its LuaJIT engine. 来自OpenResty - Wikipedia)

`Openresty`帮我们实现了可以用`Lua`的规范开发，实现各种业务，并帮我们弄清各模块的编译顺序。

`Openresty`的`master-worker`模型

- `master`进程管理多个`worker`进程
- 基本的事件处理都是放在`worker`中
- `master`负责全局初始化，以及对`worker`的管理
- 请求被分配到`worker`，`worker`创建`coroutine`（协程），协程之间数据隔离，且每个协程都有独立的全局变量

`Openresty`的优势

- `Nginx`核心 + 第三方模块，默认集成`Lua`环境
- 可借助`Nginx`事件驱动模型和非阻塞IO

非阻塞**IO**

Linux 网络编程的5种IO模型：阻塞IO与非阻塞IO - schips - 博客园 (cnblogs.com)

当用户线程发起IO操作后，无需等待，马上可得到结果。结果如果是error，表明数据还没有准备好，于是用户线程可再次发送IO操作。一旦内核中的数据准备好了，并再次收到用户线程的请求，内核就将数据拷贝到了用户线程，然后返回。

在非阻塞IO模型中，用户线程需不断询问内核数据是否就绪，也就说非阻塞IO不会交出CPU，而会一直占用CPU。

高并发

什么是高并发？ - 知乎 (zhihu.com)

高并发（High Concurrency）是互联网分布式系统架构设计中必须考虑的因素之一，它通常是指，通过设计保证系统能够同时并行处理很多请求。

并发用户数：同时承载正常使用系统功能的用户数量。例如一个即时通讯系统，同时在线量一定程度上代表了系统的并发用户数。

Lua

优势

Lua脚本可以很容易的被C/C++代码调用，也可以反过来调用C/C++的函数，这使得Lua在应用程序中可以被广泛应用。不仅仅作为扩展脚本，也可以作为普通的配置文件，代替XML,Ini等文件格式，并且更容易理解和维护。

在目前所有脚本引擎中，Lua的速度是最快的。这一切都决定了Lua是作为嵌入式脚本的最佳选择。

Coroutine(协同程序)

Lua 协同程序(coroutine) | 菜鸟教程 (runoob.com)

Lua协同程序，与线程类似，有独立的堆栈、局部变量、指令指针，但也与其他协同程序共享全局变量和其他大部分东西。

在任一指定时刻只有一个协同程序在运行，并且它只有在明确被要求挂起时才会被挂起。

协同程序类似于同步的多线程。

`coroutine`方法：

- `create`-创建（可以基于函数创建）
- `resume`-重启， `yield`-挂起（遇到`yield`的时候就代表挂起当前线程，等到再次`resume`）

- `status`-查看状态（dead，suspended，running）
- `wrap`-返回函数，调用该函数就相当于创建`coroutine``
- `running` -返回正在跑的`coroutine`，实际上是返回一个coroutine的线程号

Prometheus

Prometheus原理详解大数据老司机的博客-*CSDN*博客prometheus

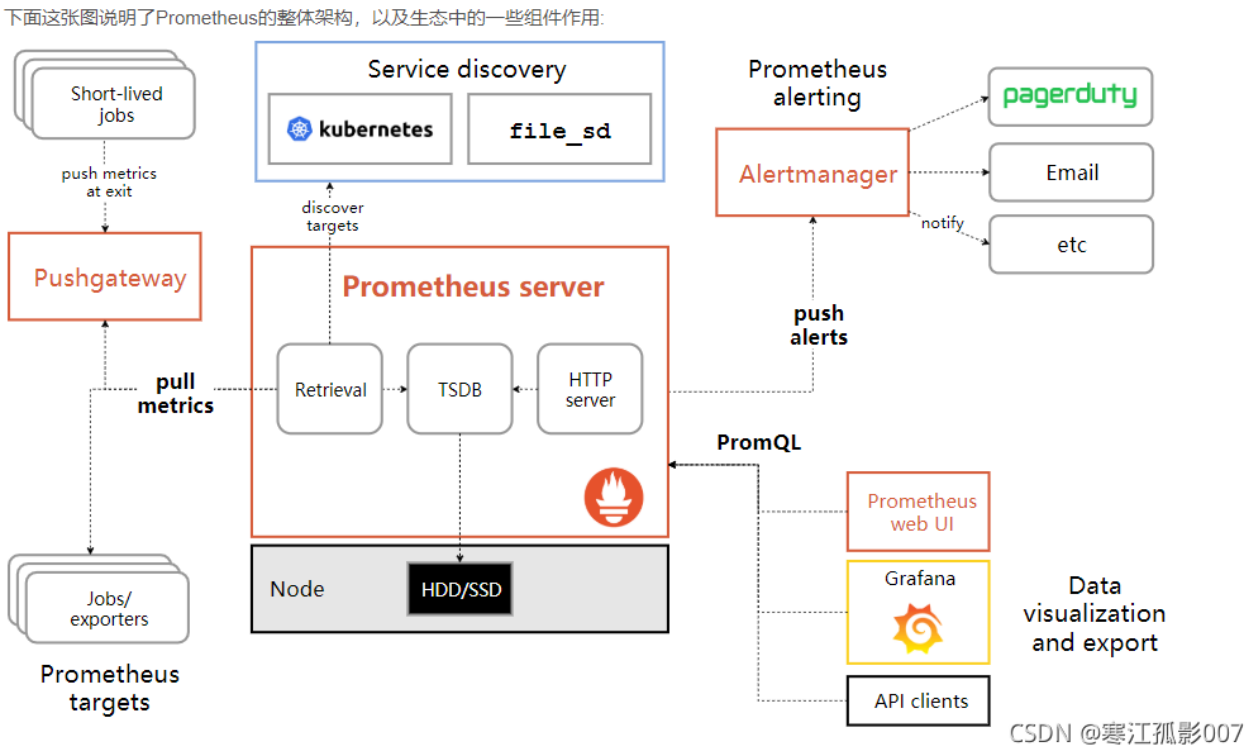
可能用得到的教程 14. Prometheus 快速入门教程 - 随笔分类 - 陈树义 - 博客园 (cnblogs.com)

`Prometheus` 最开始是由 `SoundCloud` 开发的开源监控告警系统。随着 `Kubernetes` 在容器编排领头羊地位的确立，`Prometheus` 也成为 `Kubernetes` 同期监控的标配。

`Prometheus` 是一个用于事件监控和告警的免费软件应用。它将实时的指标数据（metrics）记录并存储在通过 Http 拉取模型构建的时间序列数据库（允许高维度）中，有着较灵活的询问功能和实时告警功能。（翻译自维基百科Prometheus (software) - Wikipedia）

(**Prometheus** is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database (allowing for high dimensionality) built using a HTTP pull model, with flexible queries and real-time alerting.)

整体架构



模块介绍如下：

- Retieval (中央 Server 上) -定时去目标页面上抓取需要的数据
- Storage -将数据写入指定的时序数据库
- PromQL (右侧箭头上方) - `Prometheus` 提供的查询语言模块，可以和WehUI集成
- Jobs/Exporters (左下) - `Prometheus` 可从中拉取监控数据，Exporter以 Web API 的形式对外提供数据采集接口
- Pushgateway (左侧) - 对于 `Prometheus` 来不及处理的Job（如左上角的 Short-lived jobs ）中的数据，Job在运行时可以将它们先送到 Pushgateway 中赞UN，防止监控数据丢失。
- Service discovery (最上方) - `Prometheus` 可以动态地发现服务

  （`Prometheus` 是通过 server 端配置里的 targets 来获取监控对象的，targets 的获取又是依赖 discovery 模块进行获取的）
- AlertManager - 外部组件，独立于 `Promethues`，用于监控系统的告警

工作流程

- `Prometheus server` 定期从**jobs**、**exporters**拉取 metrics（指标，可以理解为数据），或者接收来自**Pushgateway** 发过来的数据，也可以从其他 `Prometheus server` 中获取数据 （四个数据来源）
- `Prometheus server` 在本地存储收集的数据，并运行alert.rules ，一旦满足告警规则，就向 Alertmanager 推送警报

  注：alert.rules 用于存告警规则，以下摘自官方文档。

  > *Prometheus*中的告警规则允许你基于*PromQL*表达式定义告警触发条件，*Prometheus*后端对这些触发规则进行周期性计算，当满足触发条件后则会触发告警通知
- Alertmanager 处理接收的警报，发出告警
- 在 UI 中，可视化采集的数据

省流：`Prometheus` 的服务器采集数据将，其存储在时序数据库中，监测其是否满足 alter.rules 中的告警规则，一旦满足，就向 Alertmanager 报警，后者处理警报并发出告警通知。

InfluxDB

有关时序数据和时序数据库

时序数据库 是针对 摄取（高速）、处理 和 存储 带有时间戳数据优化过的数据库，此类数据可能包括来自 服务器和应用程序的参数指标、网站或应用程序的用户交互 等内容。

时序数据：按照时间顺序记录系统、设备状态变化的数据被称为时序数据（Time Series Data），如CPU利用率、某一时间的环境温度等。

时序数据以时间作为主要的查询维度，通常会将连续的多个时序数据绘制成线，制作基于时间的多维度报表，用于揭示数据背后的趋势、规律、异常，进行实时在线预测和预警。

时序数据库是针对 摄取（高速读入）、处理 和 存储 这类时序数据有特定优化的一类数据库。

时序数据库的作用

传统数据库通常记录数据的当前值，时序型数据库则记录所有的历史数据，在处理当前时序数据时又要不断接收新的时序数据，同时时序数据的查询也总是以时间为基础查询条件。

时序数据库的特点

来自 Prometheus原理详解大数据老司机的博客-*CSDN*博客prometheus

- 大部分时间是写入操作
- 写入操作几乎都是顺序添加，多数时候数据到达后以时间排序
- 写操作很少写入风场久的数据，也很少更新数据。多数时候数据被采集后的数秒或数分钟后就会被写入数据库
- 删除操作一般为区块删除（一块一块地删），如选定开始的历史事件并指定后续的区块
- 基本数据大
- 读操作是十分典型的升序或降序地顺序读
- 高并发读操作很常见

有关**InfluxDB**

InfluxDB 介绍和使用鲁先生.的博客-*CSDN*博客influxdb数据类型

`InfluxDB` 是一个开源 分布式 时序、事件 和 指标 数据库，使用`Go`语言编写，无需外部依赖。其设计目标是实现分布式和水平伸缩扩展

InfluxDB 三大特性

- Time Series（事件序列）：可以使用与时间有关的相关函数（e.g. max、min、sum）
- Metrics （度量）：可以实时对大量数据进行计算。
- Events （事件）：支持任意的事件数据。

`InfluxDB` 的数据都有一列名为 time 的列，里面存储 UTC 时间戳。

`Prometheus` 与 `InfluxDB` 的联动案例

Prometheus + InfluxDB +Grafana 监控安装部署地球人是我哈的博客-*CSDN*博客 prometheus监控influxdb

prometheus + influxdb + grafana + mysql - 技术颜良 - 博客园 (cnblogs.com)

个人的想法

经过对复现 `DisGraFS` 的尝试，个人感觉最好还是先把在 `DisGraFS` 上部署监控系统放一放，先考虑在更简单的系统上（比如徐奥同学提到的一些工程量小的分布式文件系统）成功运用 `Prometheus` 和 `InfluxDB` 实现监控系统，再在时间允许的前提下做进一步的迭代。`Openresty` 和 `Prometheus` 应该如何一起使用或许也是要考虑的问题（似乎`Openresty` 是用来实现分布式的框架） ，网上并没有多少现成的案例可供分析。