服务治理：监控系统如何做?

> 个人学习笔记，大部分内容整理自书籍、博客和官方文档。
>
> 相关文章 &书籍：
>
> - 监控系统选型，这篇不可不读！ <https://www.jianshu.com/p/302ba018082a>
> - Prometheus vs Nagios <https://logz.io/blog/prometheus-vs-nagios-metrics/>
> - 2020 年工作上的最大收获——监控告警体系 <https://www.cnblogs.com/hunternet/p/14270218.html>
> - 《Prometheus 操作指南》 <https://yunlzheng.gitbook.io/prometheus-book/>
>
> 相关视频：
>
> - 使用Prometheus实践基于Spring Boot监控告警体系 <https://www.imooc.com/learn/1231>
> - Prometheus & Grafana -陈嘉鹏 [尚硅谷大数据] <https://www.bilibili.com/video/BV11f4y1A7aF>

监控系统有什么用?

建立完善的监控体系主要是为了：

- **长期趋势分析**：通过对监控样本数据的持续收集和统计，对监控指标进行长期趋势分析。例如，通过对磁盘空间增长率的判断，我们可以提前预测在未来什么时间节点上需要对资源进行扩容。
- **数据可视化**：通过可视化仪表盘能够直接获取系统的运行状态、资源使用情况、以及服务运行状态等直观的信息。
- **预知故障和告警**：当系统出现或者即将出现故障时，监控系统需要迅速反应并通知管理员，从而能够对问题进行快速的处理或者提前预防问题的发生，避免出现对业务的影响。
- **辅助定位故障、性能调优、容量规划以及自动化运维**

出任何线上事故，先不说其他地方有问题，监控部分一定是有问题的。

如何才能更好地使用监控使用?

1. **了解监控对象的工作原理：**要做到对监控对象有基本的了解，清楚它的工作原理。比如想对 JVM 进行监控，你必须清楚 JVM 的堆内存结构和垃圾回收机制。
2. **确定监控对象的指标：**清楚使用哪些指标来刻画监控对象的状态？比如想对某个接口进行监控，可以采用请求量、耗时、超时量、异常量等指标来衡量。
3. **定义合理的报警阈值和等级：**达到什么阈值需要告警？对应的故障等级是多少？不需要处理的告警不是好告警，可见定义合理的阈值有多重要，否则只会降低运维效率或者让监控系统失去它的作用。
4. **建立完善的故障处理流程：**收到故障告警后，一定要有相应的处理流程和 oncall 机制，让故障及时被跟进处理。

常见的监控对象和指标有哪些?

- **硬件监控**：电源状态、CPU 状态、机器温度、风扇状态、物理磁盘、raid 状态、内存状态、网卡状态
- **服务器基础监控**：CPU、内存、磁盘、网络
- **数据库监控**：数据库连接数、QPS、TPS、并行处理的会话数、缓存命中率、主从延时、锁状态、慢查询
- **中间件监控**：
  - Nginx：活跃连接数、等待连接数、丢弃连接数、请求量、耗时、5XX 错误率
  - Tomcat：最大线程数、当前线程数、请求量、耗时、错误量、堆内存使用情况、GC 次数和耗时
  - 缓存：成功连接数、阻塞连接数、已使用内存、内存碎片率、请求量、耗时、缓存命中率
  - 消息队列：连接数、队列数、生产速率、消费速率、消息堆积量
- **应用监控**：
  - HTTP 接口：URL 存活、请求量、耗时、异常量
  - RPC 接口：请求量、耗时、超时量、拒绝量
  - JVM：GC 次数、GC 耗时、各个内存区域的大小、当前线程数、死锁线程数
  - 线程池：活跃线程数、任务队列大小、任务执行耗时、拒绝任务数
  - 连接池：总连接数、活跃连接数
  - 日志监控：访问日志、错误日志
  - 业务指标：视业务来定，比如 PV、订单量等

监控的基本流程了解吗?

3915 字 无论是开源的监控系统还是自研的监控系统，监控的整个流程大同小异，一般都包括以下模块：

- **数据采集：**采集的方式有很多种，包括日志埋点进行采集（通过 Logstash、Filebeat 等进行上报和解析），JMX 标准接口输出监控指标，被监控对象提供 REST API 进行数据采集（如 Hadoop、ES），系统命令行，统一的 SDK 进行侵入式的埋点和上报等。
- **数据传输：**将采集的数据以 TCP、UDP 或者 HTTP 协议的形式上报给监控系统，有主动 Push 模式，也有被动 Pull 模式。
- **数据存储：**有使用 MySQL、Oracle 等 RDBMS 存储的，也有使用时序数据库 RRDTool、OpentTSDB、InfluxDB 存储的，还有使用 HBase 存储的。
- **数据展示：**数据指标的图形化展示。
- **监控告警：**灵活的告警设置，以及支持邮件、短信、IM 等多种通知通道。

监控系统需要满足什么要求?

- **实时监控&告警**： 监控系统对业务服务系统实时监控，如果产生系统异常及时告警给相关人员。
- **高可用**： 要保障监控系统的可用性
- **故障容忍**： 监控系统不影响业务系统的正常运行，监控系统挂了，应用正常运行。
- **可扩展**： 支持分布式、跨 IDC 部署，横向扩展。
- **可视化**： 自带可视化图标、支持对接各类可视化组件比如 Grafana 。

监控系统技术选型有哪些？如何选择?

老牌监控系统

Zabbix 和 Nagios 相继出现在 1998 年和 1999 年，目前已经被淘汰，不太建议使用，Prometheus 是更好的选择。

Zabbix

- **介绍**： 老牌监控的优秀代表。产品成熟，监控功能很全面，采集方式丰富（支持 Agent、SNMP、JMX、SSH 等多种采集方式，以及主动和被动的数据传输方式），使用也很广泛，差不多有70%左右的互联网公司都曾使用过 Zabbix 作为监控解决方案。
- **开发语言**： C
- **数据存储**： Zabbix 存储在 MySQL 上，也可以存储在其他数据库服务。Zabbix 由于使用了关系型数据存储时序数据，所以在监控大规模集群时常常在数据存储方面捉襟见肘。所以从 Zabbix 4.2 版本后开始支持 TimescaleDB 时序数据库，不过目前成熟度还不高。
- **数据采集方式**：Zabbix 通过 SNMP、Agent、ICMP、SSH、IPMI 等对系统进行数据采集。Zabbix 采用的是 Push 模型（客户端发送数据给服务端）。
- **数据展示**： 自带展示界面，也可以对接 Grafana。
- **评价**： 不太建议使用 Zabbix，性能可能会成为监控系统的瓶颈。并且，应用层监控支持有限、二次开发难度大（基于 c 语言）、数据模型不强大。

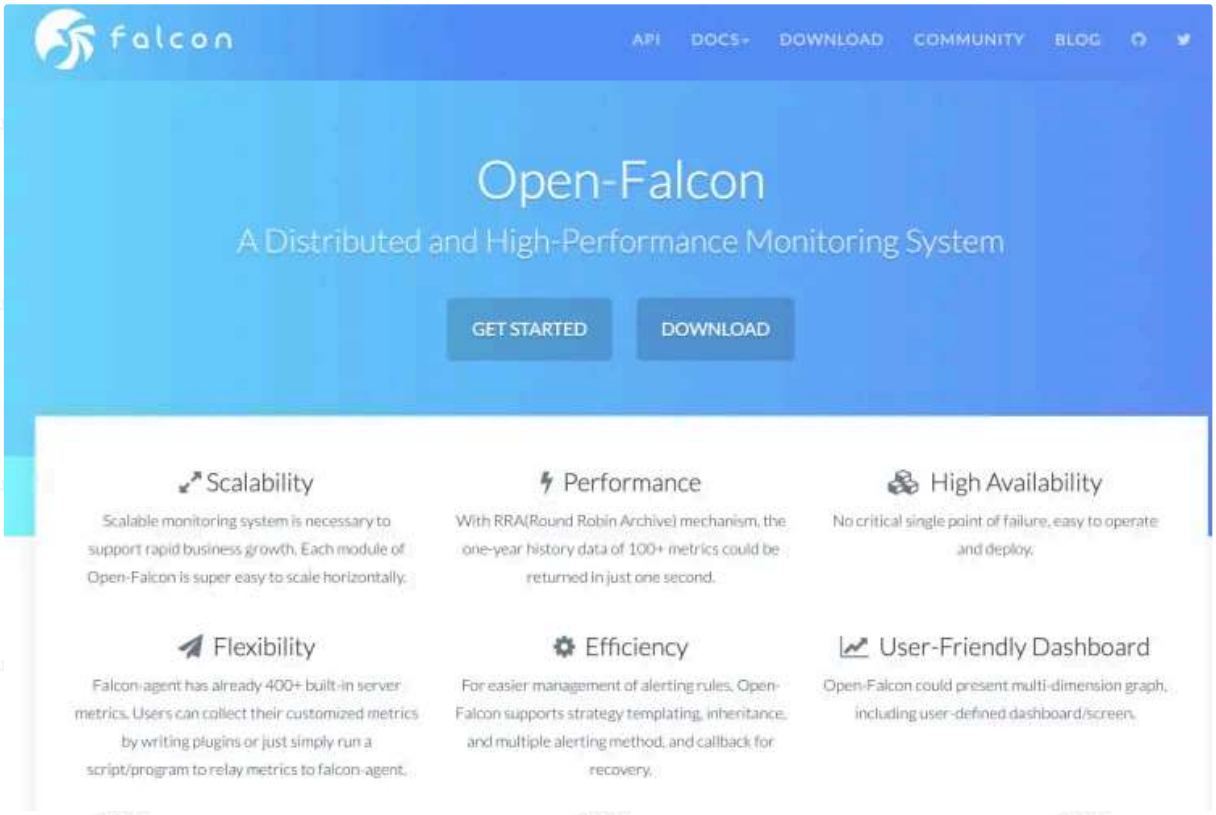
相关阅读：《zabbix 运维手册》 <http://www.sunrisenan.com/docs/zabbix>

Nagios

- **介绍**： Nagios 能有效监控 Windows、Linux 和 UNIX 的主机状态（CPU、内存、磁盘等），以及交换机、路由器等网络设备（SMTP、POP3、HTTP 和 NNTP 等），还有 Server、Application、Logging，用户可自定义监控脚本实现对上述对象的监控。Nagios 同时提供了一个可选的基于浏览器的 Web 界面，以方便系统管理人员查看网络状态、各种系统问题以及日志等。
- **开发语言**： C
- **数据存储**： MySQL 数据库
- **数据采集方式**：通过各种插件采集数据
- **数据展示**： 自带展示界面，不过功能简单。
- **评价**： 不符合当前监控系统的要求，而且，Nagios 免费版本的功能非常有限，运维管理难度非常大。

新一代监控系统

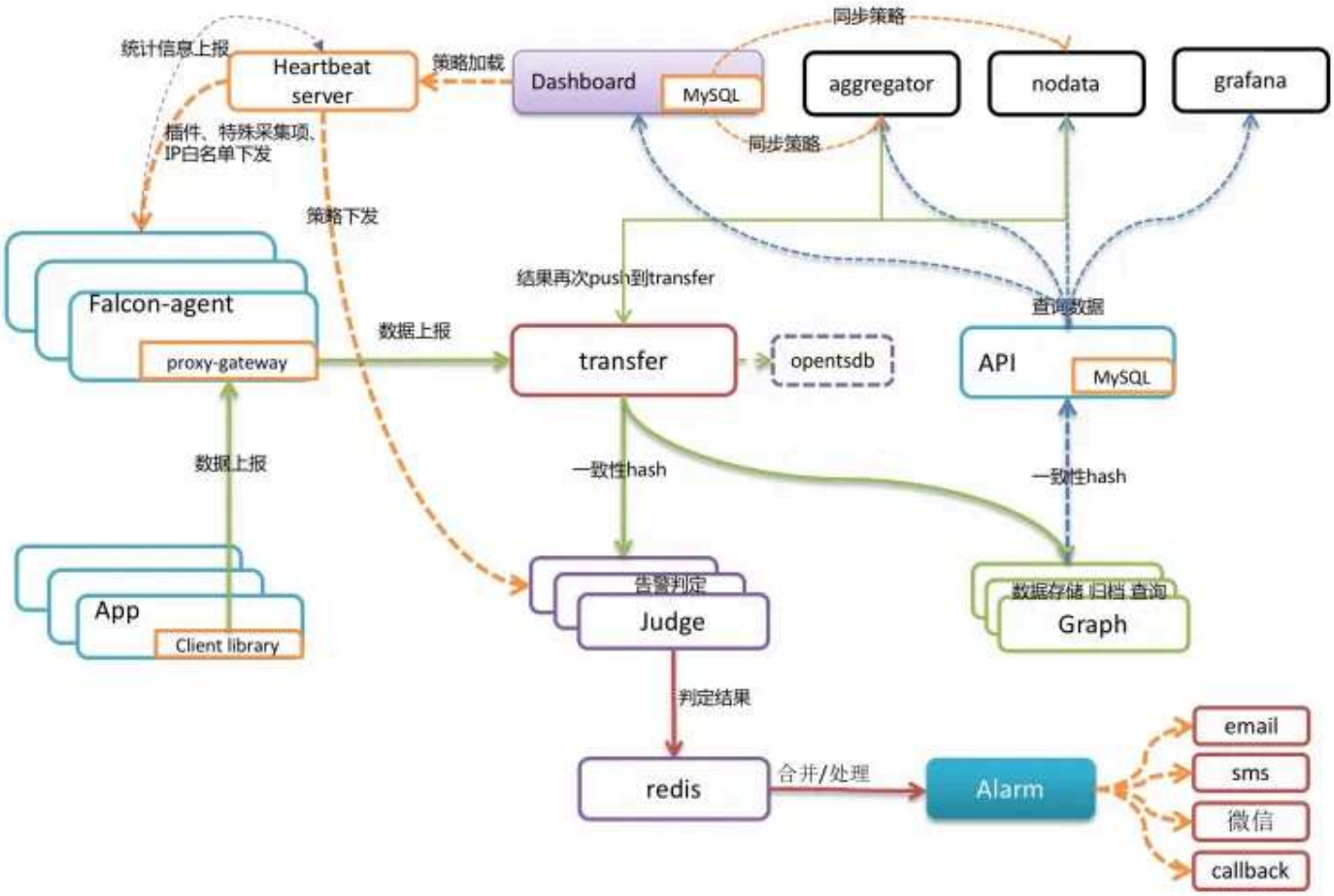
相比于老牌监控系统，新一代监控系统有明显的优势，比如：灵活的数据模型、更成熟的时序数据库、强大的告警功能。



Open-Falcon

- **介绍**：小米 2015 年开源的企业级监控工具，在架构设计上吸取了 Zabbix 的经验，同时很好地解决了 Zabbix 的诸多痛点。Github 地址：https://github.com/open-falcon <https://github.com/open-falcon> 。官方文档：https://book.open-falcon.org/ <https://book.open-falcon.org/> 。
- **开发语言**：Go、Python。
- **数据存储**： 环型数据库，支持对接时序数据库 OpenTSDB。
- **数据采集方式**：自动发现，支持 falcon-agent、snmp、支持用户主动 push、用户自定义插件支持、opentsdb data model like（timestamp、endpoint、metric、key-value tags）。Open-Falcon 和 Zabbix 采用的都是 Push 模型（客户端发送数据给服务端）。
- **数据展示**：自带展示界面，也可以对接 Grafana。
- **评价**：用户集中在国内，流行度一般，生态一般。

Open-Falcon 架构图如下：



- **Falcon-agent**： 采集模块。类似 Zabbix 的 agent，Kubernetes 自带监控体系中的 cAdvisor，Nagios 中的 Plugin，使用 Go 语言开发，用于采集主机上的各种指标数据。
- **Hearthbeat server**： 心跳服务。每个 Agent 都会周期性地通过 RPC 方式将自己地状态上报给 HBS，主要包括主机名、主机 IP、Agent 版本和插件版本，Agent 还会从 HBS 获取自己需要执行的采集任务和自定义插件。
- **Transfer**： 负责监控 agent 发送的监控数据，并对数据进行处理，在过滤后通过一致性 Hash 算法将数据发送到 Judge 或者 Graph。为了支持存储大量的历史数据，Transfer 还支持 OpenTSDB。Transfer 本身没有状态，可以随意扩展。
- **Jedge**： 告警模块。Transfer 转发到 Judge 的数据会触发用户设定的告警规则，如果满足，则会触发邮件、微信或者回调接口。这里为了避免重复告警，引入了 Redis 暂存告警，从而完成告警合并和抑制。
- **Graph**： RRD 数据上报、归档、存储的组件。Graph 在收到数据以后，会以 RRDtool 的数据归档方式存储数据，同时提供 RPC 方式的监控查询接口。
- **API**： 查询模块。主要提供查询接口，不但可以从 Grapg 里面读取数据，还可以对接 MySQL，用于保存告警、用户等信息。
- **Dashboard**： 监控数据展示面板。由 Python 开发而成，提供 Open-Falcon 的数据和告警展示，监控数据来自 Graph，Dashboard 允许用户自定义监控面板。
- **Aggregator**：聚合模块。聚合某集群下所有机器的某个指标的值，提供一种集群视角的监控体验。
  通过定时从 Graph 获取数据，按照集群聚合产生新的监控数据并将监控数据发送到 Transfer。

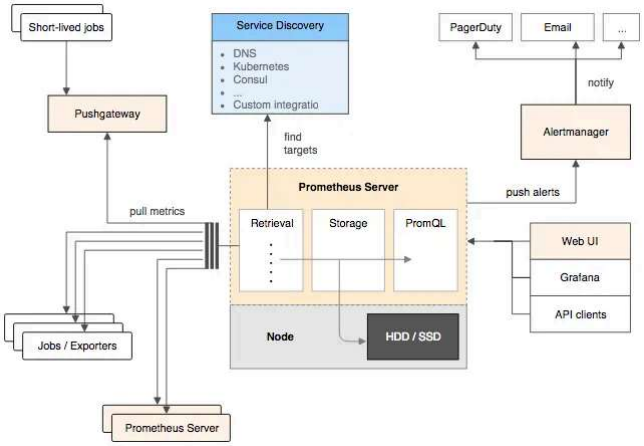
Prometheus

- **介绍**： Prometheus 受启发于 Google 的 Brogmon 监控系统，由前 Google 员工 2015 年正式发布。截止到 2021 年 9 月 2 日，Prometheus 在 Github 上已经收获了 38.5k+ Star，600+位 Contributors。 Github 地址：https://github.com/prometheus <https://github.com/prometheus> 。
- **开发语言**： Go
- **数据存储**： Prometheus 自研一套高性能的时序数据库，并且还支持外接时序数据库。
- **数据采集方式**：Prometheus 的基本原理是通过 HTTP 协议周期性抓取被监控组件的状态，任意组件只要提供对应的 HTTP 接口就可以接入监控。Prometheus 在收集数据时，采用的 Pull 模型（服务端主动去客户端拉取数据）
- **数据展示**： 自带展示界面，也可以对接 Grafana。
- **评价**： 目前国内外使用最广泛的一个监控系统，生态也非常好，成熟稳定！

Prometheus 特性：

- 开箱即用的各种服务发现机制，可以**自动发现监控端点**；
- 专为监控指标数据设计的**高性能时序数据库 TSDB**；
- 强大易用的查询语言**PromQL**以及丰富的**聚合函数**；
- 可以配置灵活的告警规则，支持**告警收敛（分组、抑制、静默）、多级路由**等等高级功能；
- **生态完善**，有各种现成的开源 Exporter 实现，实现自定义的监控指标也非常简单。

Prometheus 基本架构：



- **Prometheus Server：**核心组件，用于收集、存储监控数据。它同时支持静态配置和通过 Service Discovery 动态发现来管理监控目标，并从监控目标中获取数据。此外，Prometheus Server 也是一个时序数据库，它将监控数据保存在本地磁盘中，并对外提供自定义的 PromQL 语言实现对数据的查询和分析。
- **Exporter：**用来采集数据，作用类似于 agent，区别在于 Prometheus 是基于 Pull 方式拉取采集数据的，因此，Exporter 通过 HTTP 服务的形式将监控数据按照标准格式暴露给 Prometheus Server，社区中已经有大量现成的 Exporter 可以直接使用，用户也可以使用各种语言的 client library 自定义实现。
- **Push gateway：**主要用于瞬时任务的场景，防止 Prometheus Server 来 pull 数据之前此类 Short-lived jobs 就已经执行完毕了，因此 job 可以采用 push 的方式将监控数据主动汇报给 Push gateway 缓存起来进行中转。
- 当告警产生时，Prometheus Server 将告警信息推送给 Alert Manager，由它发送告警信息给接收方。
- Prometheus 内置了一个简单的 web 控制台，可以查询配置信息和指标等，而实际应用中我们通常会将 Prometheus 作为 Grafana 的数据源，创建仪表盘以及查看指标。

推荐一本 Prometheus 的开源书籍《Prometheus 操作指南》 <https://yunlzheng.gitbook.io/prometheus-book/> 。

总结

- 监控是一项长期建设的事情，一开始就想做一个 All In One 的监控解决方案，我觉得没有必要。从成本角度考虑，在初期直接使用开源的监控方案即可，先解决有无问题。
- Zabbix、Open-Falcon 和 Prometheus 都支持和 Grafana 做快速集成，想要美观且强大的可视化体验，可以和 Grafana 进行组合。
- Open-Falcon 的核心优势在于数据分片功能，能支撑更多的机器和监控项；Prometheus 则是容器监控方面的标配，有 Google 和 k8s 加持。