日期： 2017-03-14 17:20　　浏览： 100 评论： 0

前言

随着 Devops、云计算、微服务、容器等理念的逐步落地和大力发展，机器越来越多，应用越来越多， 服务越来越微， 应用运行基础环境越来多样化， 容器、 虚拟机、 物理机不一而足。面对动辄几百上千个虚拟机、 容器，数十种要监控的对象， 现有的监控系统还能否支撑的住？来自于容器、虚拟机、物理机、网络设备、中间件的指标数据如何采用同一套方案快速、完整的收集和分析告警？怎样的架构、技术方案才更适合如此庞大繁杂的监控需求呢？

目录：

一、统一监控平台架构解析

先做一下回顾，统一监控平台由七大角色构成：监控源、数据采集、数据存储、数据分析、数据展现、预警中心、 CMDB(企业软硬件资产管理)。

监控源：

从层次上来分，大致可以分为三层，业务应用层、中间件层、基础设施层。业务应用层主要包括应用软件、 企业消息总线等， 中间件层包括数据库、 缓存、 配置中心、 等各种系统软件，基础设施层主要有物理机、虚拟机、容器、网络设备、存储设备等等。

数据采集：

数据源如此多样， 数据采集的任务自然轻松不了。 数据采集从指标上划分可以分为业务指标、应用指标、系统软件监控指标、系统指标。应用监控指标如：可用性、异常、吞吐量、响应时间、当前等待笔数、资源占用率、请求量、日志大小、性能、队列深度、线程数、服务调用次数、访问量、服务可用性等，业务监控指标如大额流水、流水区域、流水明细、请求笔数、响应时间、响应笔数等，系统监控指标如： CPU负载、内存负载、磁盘负载、网络 IO、磁盘 IO、 tcp 连接数、进程数等。

从采集方式来说通常可以分为接口采集、 客户端 agent 采集、通过网络协议主动抓取 （http 、snmp等）

数据存储：

采集到的数据一般都会存储到文件系统（如 HDFS）、索引系统（如 elasticsearch ）、指标库（如 influxdb ）、消息队列（如 kafka ，做消息临时存储或者缓冲）、数据库（如 mysql）

数据分析：

针对采集到的数据， 进行数据的处理。 处理分两类： 实时处理和批处理。 技术包括 Map/Reduce 计算、 全日志检索、 流式计算、 指标计算等， 重点是根据不同的场景需求选择不同的计算方式。

数据展现：

将处理的结果进行图表展现，在多屏时代，跨设备的支持必不可少。

预警：

如果在数据处理过程发现了问题， 则需要进行异常的分析、 风险的预估以及事件的触发或告警。

CMDB(企业软硬件资产管理):

CMDB在统一监控平台中是很重要的一环，监控源虽然种类繁多，但是他们大都有着关系，如应用运行在运行环境中， 应用的正常运行又依赖网络和存储设备， 一个应用也会依赖于其他的应用（业务依赖），一旦其中任何一个环节出了问题，都会导致应用的不可用。 CMDB 除了存储软硬件资产外， 还要存储这样一份资产间的关联关系， 一个资产发生了故障， 要能根据这个关系迅速得知哪些其他的资产会被影响，然后逐一解决问题。

OK，回顾到此为止，进入正题，系统监控。

二、系统监控的技术栈

系统监控的部分技术栈如下图所示， 监控技术众多， 这里自然不可能列出所有的技术， 选择了部分比较经典、受欢迎的开源技术。

系统监控不同于日志监控，有很多开源软件把数据库采集、数据存储、 数据展现、 事件告警的任务都完成了， 所以对于系统监控的技术栈中， 将这些开源软件暂且排除， 待后面章节再进行讲解。此处主要关注于如何自建一个统一系统监控平台。

数据采集：

系统监控数据采集一般分为两种方式：主动采集、客户端采集。主动采集一般是通过 SNMP、SSH、 Telnet 、 IPMI、 JMX等手段进行远程采集，客户端采集则是需要在每一个要监控的主机中部署一个客户端进行数据采集并发送到远程服务端进行接收。

数据缓冲：

和日志监控一样，在面临海量监控时，考虑到网络的压力和数据处理的瓶颈，可以在数据存储前先经过一层数据缓冲，将采集到的数据先放置到消息队列中，然后再从分布式队列中读取数据并存储。如果数据量不大的话，则可以不考虑此层。

数据存储：

对于系统监控数据，通常采用时序数据库来存储，时序数据库全称为时间序列数据库。时间序列数据库主要用于指处理带时间标签（按照时间的顺序变化，即时间序列化）的数据，带时间标签的数据也称为时间序列数据。如 influxdb 和 opentsdb，是其中翘楚。

OpenTSDB是用 hbase 存储所有的时序（无须采样）来构建的一个分布式、可伸缩的时间序列数据库，可以从大规模的集群（包括集群中的网络设备、操作系统、应用程序）中获取相应的 metrics 并进行存储、索引以及服务，从而使得这些数据更容易让人理解，如 web 化，图形化等。用 JAVA语言实现，对于 JAVA系的同学们是一个福音，不过其依赖 hbase 也许会让一部分同学望而却步，毕竟还要先去维护 hbase。

Influxdb 是新兴的一个时序数据库，用 go 语言编写，无需外部依赖，发展很快，最新版本已经到了。提供类 sql 的查询语法，安装方便，单点即可使用，虽然有集群的能力，不过该特性是非开源的（不过单点性能基本也都能满足企业需求了）。提供 Http API，便于调用和封装。对于想基于 influxdb 自行进行数据处理和展现的同学们而言很是友好。

数据展现：

说到时序数据的图形化展现，Grafana 是一个不得不提的利器。Grafana 是一个开源的时序数据的查询和展现软件，提供了灵活丰富的图形化选项；可以混合多种风格，有着功能齐全的度量仪表盘和图形编辑器。支持与 Graphite 、Elasticsearch 、CloudWatch、Prometheus 、InfluxdbDB 等众多数据存储对接，进行数据的查询和图表展现。一些开源的监控软件如 zabbix 、 Graphite 、 Prometheus 也都有着自己的数据图形化展现能力，但是一般也都是建议使用

Grafana 来代替它们的页面。可想而知 Grafana 的优秀。

当然， Grafana 的数据源都是来自时序数据库，在实际场景中，可能你想要查看的报表的一部分数据还来自于业务系统，这就是 Grafana 或者其他的监控软件做不到的了，去扩展是一种方式，另外一种方式就是结合自己的需求实现图表展现，通过对时序数据的计算分析以及结合业务数据，使用如 echarts 等开源图表前端框架进行展现。这时候 Influxdb 的优势就体现出来了，对外提供 http api 非常适合自主封装图形化页面。

告警：

在日志监控的分享中，确实没有对告警进行说明。像 Zabbix 、 Nagios 、 Open-Falcon 、 Prometheus 等开源监控软件，都是有些自己的告警能力的。如果你采用了他们作为监控平台，实际上告警能力就已经有了。如果是纯自建统一监控平台的话，也可以自己实现告警中心。我们自己的做法是，在数据处理时，根据配置的事件触发规则，生成相应事件扔到 kafka

中，事件处理引擎监听 kafka 中的事件数据， 进行解析并根据事件处理策略进行告警通知等处理。

三、开源系统监控软件

Zabbix VS Nagios VS Open-Falcon

上面大致介绍了运维监控的技术栈， 但是实际上已经有些开源监控软件功能都很全面， 从数据采集到数据展现都提供了支持， 如果是小团队， 不想自建监控平台的话， 选择这些开源软件其实是一个很好的选择。

Zabbix

Zabbix 是一个企业级的开源分布式监控解决方案，支持实施从数以万计的服务器、虚拟机、网络设备等收集百万的指标数据， 具备常见的商业监控软件所具备的功能 （主机的性能监控、网络设备性能监控、数据库性能监控、 FTP等通用协议监控、多种告警方式、详细的报表图表绘制） 支持自动发现网络设备和服务器； 支持分布式， 能集中展示、 管理分布式的监控点；扩展性强， server 提供通用接口，可以自己开发完善各类监控。

Zabbix 重要组件说明：

zabbix server ：负责接收 agent 发送的报告信息的核心组件，所有配置、统计数据及操作数据都由它组织进行；

database storage ：专用于存储所有配置信息，以及由 zabbix 收集的数据；

web interface ：zabbix 的 GUI 接口；

proxy ：可选组件， 常用于监控节点很多的分布式环境中， 代理 server 收集部分数据转发到 server ， 可以减轻 server 的压力；

agent ：部署在被监控的主机上，负责收集主机本地数据如 cpu、内存、数据库等数据发往 server 端或 proxy 端；

优点：

All in One ：部署相当便捷

Server 对宿主机性能要求很低。

自动发现服务器与网络设备

分布式监控，以及 WEB 集中管理功能

同时支持 agent 采集和无 agent 采集，主机通过 agent 或者 ipmi 采集数据，网络设备、存储设备等通过 SNMP 客户端采集数据， agent 支持常用的 UNIX 和 Windows 操作系统

功能全面，数据采集、数据存储、数据展现、事件告警。

开放式接口，扩展性强，插件编写容易

不足：

数据库瓶颈，使用 mysql 作为底层存储，大数据读写的时候，对于数据库的压力非常大

需要在主机中安装 agent

对容器监控支持不好，需要自己扩展。

Nagios

Nagios 全名为（Nagios Ain ’t Goona Insist on Saintood），最初项目名字是 NetSaint 。它是一款免费的开源 IT 基础设施监控系统， 其功能强大， 灵活性强，能有效监控 Windows 、Linux 、VMware 和 Unix 主机状态，交换机、路由器等网络设置等。 Nagios 核心功能是监控报警，告警能力很不错，但是图形展示效果很差。同时 nagios 更加灵活，很多功能都要通过插件化来实现， 对于技术能力没那么强的同学， 上手会有些困难。 当然，对于运维老手，上手会很快。

Nagios 的功能特性如下：

监控网络服务（ SMTP、POP3、HTTP、NNTP、PING等）；

监控主机资源（处理器负荷、磁盘利用率等）；

简单地插件设计使得用户可以方便地扩展自己服务的检测方法；

并行服务检查机制；

具备定义网络分层结构的能力， 用"parent" 主机定义来表达网络主机间的关系， 这种关系可被用来发现和明晰主机宕机或不可达状态；

当服务或主机问题产生与解决时将告警发送给联系人 （通过 EMail 、短信、用户定义方式） ；

可以定义一些处理程序，使之能够在服务或者主机发生故障时起到预防作用；

自动的日志滚动功能；

可以支持并实现对主机的冗余监控；

可选的 WEB界面用于查看当前的网络状态、通知和故障历史、日志文件等；

Open-Falcon

Open-Falcon 是小米运维部门开源出来的互联网企业级监控系统，目前包括小米、金山云、美团、京东金融、赶集网等都在使用 Open-Falcon 。Open-Falcon 整体可以分为两部分，即绘图组件、告警组件。“ 绘图组件 ”负责数据的采集、收集、存储、归档、采样、查询、展示（Dashboard/Screen ）等功能，可以单独工作，作为 time-series data 的一种存储展示方案。“ 告警组件 ”负责告警策略配置（ portal ）、告警判定（ judge ）、告警处理（alarm/sender ）、用户组管理（ uic ）等，可以单独工作。架构如下：

关键特性有：

数据采集免配置： agent 自发现、支持 Plugin 、主动推送模式

容量水平扩展：生产环境每秒 50 万次数据收集、告警、存储、绘图，可持续水平扩展。

告警策略自发现： Web界面、支持策略模板、模板继承和覆盖、多种告警方式、支持回调动作。

告警设置人性化：支持最大告警次数、告警级别设置、告警恢复通知、告警暂停、不同时段不同阈值、支持维护周期，支持告警合并。

历史数据高效查询：秒级返回上百个指标一年的历史数据。

Dashboard 人性化：多维度的数据展示，用户自定义 Dashboard 等功能。

架构设计高可用：整个系统无核心单点，易运维，易部署。

缺点：

支持的监控类型较少，不支持常用应用服务器如 tomcat 、apache 、jetty 等的监控。没有专门的运维支持， 代码更新较少， 没有一个较大的社区来维护， 后续想要有什么新的能力基本只能指望自己扩展。

Zabbix 、 Nagios 、 Open-Falcon 的整体对比如下：

四、基于 k8s 容器云背景下的系统监控实践：

cAdvisor+Heapster+Influxdb

上面介绍的都是比较传统的系统监控架构， 在容器时代到来后， 对于容器的支持就显得差强人意了。下面介绍下我们基于 k8s 容器云背景下的系统监控方案，首先还是介绍下我们的 DevOps平台架构， 平台运行在由 kubernetes+docker 构建的容器云中， kubernetes 、docker 等服务运行在 IaaS 平台上（我们的生产环境是阿里云）。

我们的统一监控平台，在系统监控上，采用了 cAdvisor+Heapster+Influxdb 的方案。架构如下：

为什么采用这种方案呢？先来了解下这三个工具。

cAdvisor 是谷歌公司用来分析运行中的 Docker 容器的资源占用以及性能特性的工具 ，cAdvisor 部署为一个运行中的 daemon，它会收集、聚集、处理并导出运行中容器的信息。这些信息能够包含容器级别的资源隔离参数、 资源的历史使用状况、 反映资源使用和网络统计数据完整历史状况。对 docker 的监控能力非常强大。同时还提供了自己的 web 页面，用户可以通过 web 页面直接查看该宿主机上所有容器的监控数据。 cAdvior 功能已经被集成到了 kubelet 组件中，也就是说，安装好 kubernetes 后， cAdvisor 就已经安装到了每一个计算节点上。在每一个计算节点上都可以通过 IP+ 端口（默认为 4194）访问 cAdvisor 的页面了。

Heapster 同样是 Google 提供的， 用于对 k8s 集群的监控。 Heapster 可以通过容器启动， 传入 kubernetes master 的地址， heapster 会通过调用 kubernetes api 获取所有 kubernetes 计算节点， 然后通过 kubelet 的外部调用端口号 （默认为 10250）调用 kubelet 的 http api ，

kubelet 会进行调用 cAdvisor 接口获取当前计算节点上的容器数据以及当前主机的性能数据，返回给 heapter 。这样 heapster 就收集到了 kubernetes 集群的所有容器数据以及主机数据。Heapster 支持数据传输到 Influxdb 中进行存储。数据展现我们就是自己调用 influxdb 的 api 获取数据，结合我们的业务相关数据进行计算，用 echarts 进行前端图表展现。

可能有的同学会问， 这样只是监控到了所有计算节点的容器数据和主机性能数据， 这样有些非计算节点的主机监控该怎么办？确实， 因为 Heapster 只是针对于 kubernetes 集群去监控，非 kubelet 节点确实是拿不到数据的， 而我们又不想再用另外一种方式去单独监控主机， 那样得到的数据格式也不一样。 于是我们采取了折中的办法， 在每个非 k8s 集群节点上， 也安装 kubelet ，并且加入到 kubernetes 集群中，但是配置成不参与集群调度，也就是容器不会被部署到这些机器上。这样， heapster 就可以采集到这些主机的性能数据了。

五、容器时代的监控利器： Prometheus

除了我们实践的 cAdvisor+Heapster+Influxdb 方案可以做到容器和主机性能数据同时监控外，其实还有一个相对而言更好的方案，那就是 Prometheus。Prometheus 是一套开源的监控&报警 &时间序列数据库的组合，由社交音乐平台 SoundCloud 在 2012 年开发。随着发展，越来越多公司和组织接受采用 Prometheus，社区也十分活跃，他们便将其独立成开源项目，并且不依赖于任何公司。 Prometheus 最初是参照 google 内部监控系统 BorgMon 开发的，现在最常见的 Kubernetes 容器管理系统中，通常会搭配 Prometheus 进行监控。

2016 年 Prometheus 正式成为 Cloud Native Computing Foundation 的孵化项目，该基金会是在 Google 的支持下由一群 IT 行业巨头创建并指导 Kubernetes 容器管理系统的开发。在CNCF的主导下， Prometheus 成为该开放平台栈的第二个正式的组件。特性如下：

高维度数据模型

高效的时序数据存储能力

查询语言灵活

具体时序数据图形化展现的能力

易于运维

提供丰富的客户端开发库

告警中心功能全面

Prometheus 的架构图如下：

Prometheus Server : Prometheus 主服务器，用来收集和存储时间序列数据

client libraries : 客户端库

push gateway : 短时 jobs 的中介网关

GUI-based dashboard builder : 基于 Rails/SQL 的 GUI dashboard

Exporters ： 数据采集探针， 支持包括数据库、 主机、消息队列、 存储、应用服务器、 github 等软件、其他监控系统等多种类的探针。

Alertmanager ：告警中心

Prometheus 是 google 力捧的监控方案， 社区非常活跃， 发展很是迅速， 功能在不断的飞速补充和完善。一个监控范围覆盖容器、主机、存储、数据库、各种中间件，同时还具体完善的时序数据存储、告警中心等能力，发展又很迅速，相信 Prometheus 会越来越火热。

六、总结

系统监控的方案有很多，甚至优秀的开源兼容软件也有很多，如果需求不高，也许 zabbix 就很合适，如果想要带上容器监控，那么 Prometheus 也许是个较好的方案。总之，适合自己的才是最好的。