基于 Linux 的集群可视化运维与管理

潘永德 杨将天* 吕灿

南宁学院人工智能学院，南宁，530200

摘 要 为保障集群的安全性和可靠性，更好地提升集群生产的安全性，本研究设计并且实现了集群可视化运维的监控可视化系统。首先，使用 Prometheus Exporter 对服务器的基本信息进行监测，使用 pushgateway、exporter 对中间件指标进行采集，使用 Prometheus 进行系统的优化设计和二次开发，完成指标数据采集、指标数据存储、告警功能模块，使用 Grafana 作为展示面板；其次，实现了平台中的主机性能指标、大数据集群状态以及性能指标的采集、存储和展示功能；最后，对各指标进行自定义告警规则，邮箱方式的实时告警功能进行了优化设计，提供了主机级别、应用级别等多维度多级别的监控数据查询功能，形成一套一站式的基于 Linux 的平台可视化运维监控告警服务解决方案，可以有效为集群的各个环节提供安全保障。

关键字 Prometheus，Hadoop，Grafana，监控告警，安全

Linux Based Cluster Visual Operation Maintenance and Management

Pan Yongde Yang Jiangtian LvCan
School of Artificial Intelligence Nanning University
Nanning 530200, China;
393243084@qq.com jtyang2@iflytek.com canlv@iflytek.com

Abstract—In order to ensure the security and reliability of the cluster and better improve the security of cluster production, this study designs and implements a monitoring visualization system for cluster visual operation and maintenance. First, use the tool node_exporter to monitor the basic information of the server, use pushgateway and exporter to collect the middleware indicators, use Prometheus to optimize the design and secondary development of the system, complete the indicator data collection, indicator data storage, alarm function modules, and use grafana as the display panel; Secondly, the functions of collecting, storing and displaying host performance indicators, big data cluster status and performance indicators in the platform are realized; Finally, the user-defined alarm rules for each index, the real-time alarm function of mailbox mode are optimized, and the multi-dimensional and multi-level monitoring data query function such as host level and application level is provided, forming a one-stop visual operation and maintenance monitoring alarm service solution based on Linux platform, which can effectively provide security for all links of the cluster.

Key words—Prometheus Hadoop Grafana Alarm Security

1 引 言

本研究设计并实现了一套大数据平台监控可视化系统，对平台运行指标数据进行采集、存储和可视化，评估系统当前的运行状态，解决分布式系统的状态建模和可视化问题。同时关注业务任务的忙闲规律，给出合理的任务调度建议。实现运行管理、状态可视化监控管理、异常报警等功能。从而直观展示大数据平台的运行状态，简化运维，达到简化运维工作的目的。

现阶段有云服务平台的企业都选择自己开发设计监控软件。例如 GoogleAppEngine 开发设计了一个专用工具，可以让消费者随时随地查询云服务平台中分布式系统程序流程的运行状态[1]。此外，愈来愈多的企业逐渐科学研究云服务平台监控解决方法，并为有必须的人和企业研发了一些第三方专用工具。现阶段用以云服务平台的开源系统监控专用工具有 SALSA、MRTG、Cacti、Nagios、Ganglia、Zabbix 等。这种专用工具给予不一样的监管作用，他们相互之间出现差别和重合。详细介绍了日志解决专用工具 SALSA。在分布式系统自然环境中，它可以追踪日志的控制流和数据流，进而形成分布式系统执行状态的视图。文献[2]描述了监管平台技术系统和基础设施规模的不断扩大，由于技术系统和基础设施的设备类型多、数量大、运维方式和手段差异大，当前的运维工作存在故障排查时间长、故障定位不准确、故障处理效率低以及对业务影响大等问题。为此，结合系统的实际情况，建设一套基于大数据的监控管理平台统一运维管理系统，主要介绍系统设计的目标、系统的逻辑结构以及主要功能模块。文献[3]设计并实现了工业云平台监控

* 通讯作者：杨将天，工程师，jtyang2@iflytek.com

告警服务系统。首先,基于 Prometheus 进行系统的优化设计和二次开发,完成了指标数据采集、指标数据存储、指标数据展示、告警历史查询、告警规则设置、告警联系人管理、一键管理等功能;其次,实现了平台中的主机性能指标、Kubernetes 集群状态以及性能指标的采集、存储和展示功能;最后,对各节点各指标自定义告警,邮件、钉钉、微信等多种方式的实时告警功能进行了优化设计,提供了主机级别、应用级别等多维度多级别的监控数据查询功能,形成了一套工业云平台的监控告警服务,可以有效为工业生产的各个环节提供安全保障。

2 系统总体架构的设计

本系统主要有四大功能模块,其一为指标数据的采集模块,分别采集主机硬件(node)指标、MySQL 数据库指标、大数据组件指标。其二为指标数据持久化存储模块,使用时序数据库 Influx DB[6]进行远端存储。其三为告警功能模块,使用钉钉插件 dingtalk 进行告警[5]。其四为指标数据可视化模块,使用 Grafana 对大数据组 namenode、datanode、hdfs、resourcemanager 等进行指标的可视化[6]。系统架构图如下图 1 所示。

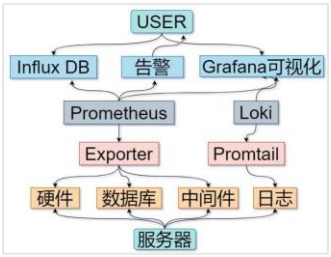

图 1 系统整体架构图

3 模块功能详细设计与实现

3.1 指标数据采集的实现

本设计使用 Prometheus 多服务联邦模式[7],使用不同的 Prometheus 服务采集不同服务类型的监控数据,本系统使用 1 个主 Prometheus+2 个从属的 Prometheus(一个用来采集 node_exporter 的 metrics、一个用来采集 mysql_exporter 的 metric)最终汇总两个 Prometheus 采集的数据到主的服务中,以便在一个 Prometheus 服务中对两个数据集启用告警和查询。从功能分区上来启用联邦集群。其架构图如图 2 所示。

(1)实现 Prometheus 联邦模式,两个从属的 Prometheus 服务启动后,需要对 Prometheus 的 master 节点进行配置,即配置文件与内容如下所示。

图 2 分层指标采集架构图

```
- job_name: 'federate'
  scrape_interval: 15s
  honor_labels: true
  metrics_path: '/federate'
  params:
    'match[]':
      - '{job="mysql"}'
      - '{job="node"}'
      - '{job="pushgateway"}'
  static_configs:
    - targets:
      - '10.90.104.62:9091'
      - '10.90.104.62:9092'
```

(2)上述服务完全启动之后,可在 master 服务中看到两个 slave 服务,如图 3 所示。

图 3 slaver 服务图示

(3)部署 Prometheus 服务与 mysqld_exporter 服务,首先创建配置文件.my.cnf,配置数据库服务节点、端口号、用户名以及密码,如下所示。

```
[client]
host=10.90.104.58
port=3306
user=exporter
password=123456
```

(4) 部署 Prometheus 服务与 node_Exporter 服务,node_Exporter 解压即可运行,需配置 Prometheus 的 prometheus.yml 文件,如下所示。

```
  - job_name: 'node'
    "labels": {
      "job": "node"
    },
    static_configs:
-targets:["10.90.104.53:9100","10.90.104.59:9100"]
```

(5)测试在 masterPrometheus 服务中可以对 slaver 中收集的数据进行查询。在 slave1 的 Prometheus 中查询,只有 node 监控数据。如图 4 所示。

图 4 slaver1 查询图示

（6）在 slave2 的 Prometheus 中查询，只有 msyql 的监控数据，如图 5 所示。

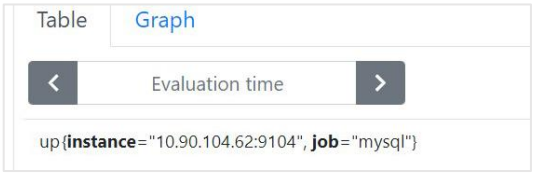

图 5 slaver2 查询图示

（7）在 master 的 Prometheus 中查询，可以看到 slave1 和 slave2 中的所有数据。如图 6 所示。

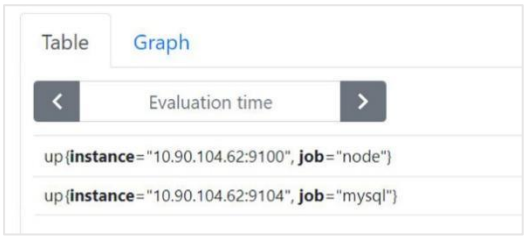

图 6 master 节点查询示例

采用上述服务联邦的形式，在服务类型和节点数量很大的情况下，分组监控不同类型服务，最终通过全局的 Prometheus 服务聚合到一起进行数据的告警和可视化展示。

3．2 指标数据存储的实现

Prometheus 数据的本地存储的方式在读写上虽然很高效，但是无法持久化数据，无法存储大量历史数据，同时也无法灵活扩展和迁移。为了解决上述问题同时又保证 Prometheus 的简单性。本系统使用两种接口，remote_write 与 remote_read，将数据抛到 InfluxDB，完成指标数据持久化[8]。

（1）配置 Prometheus。

```
remote_write:
-url:
"http://localhost:8086/api/v1/prom/write?db=prometheus"
remote_read:
-url: "http://localhost:8086/api/v1/prom/read?db=prometheus"
```

（2）验证 influxdb 是否有数据写入。

```
> use prometheus
Using database prometheus
> select time,__name__,value from prometheus_http_request
name: prometheus_http_requests_total
time                __name__                          value
----                --------                          -----
1641106626138000000 prometheus_http_requests_total 94
1641106626138000000 prometheus_http_requests_total 2
1641106626138000000 prometheus_http_requests_total 7
1641106626138000000 prometheus_http_requests_total 770
1641106626138000000 prometheus_http_requests_total 292
>
```

图 7 Influx DB 存在 Prometheus 数据

3．3 告警模块的实现

为了提升 Promthues 的服务可用性，本系统部署两个 Promthus Server。这样解决了 Prometheus 高可用的问题，但是 Alertmanager 目前仍然存在单点故障风险[8]。当 Alertmanager 单点失效后，告警的后续所有业务全部失效。所以需要实现 Alertmanager 的 HA。本系统启动多套 Alertmanager，但是因为多个 Alertmanager 之间并没有性息共享，所以会有告警重复发送的问题。为了解决这个问题，Alertmanager 引入了 Gossip 机制，Gossip 实现了多个 Alertmanager 间的信息共享，会合并多个告警源发来的相同告警信息，只有一个告警通知被发送给 Receiver。其架构图如下所示。

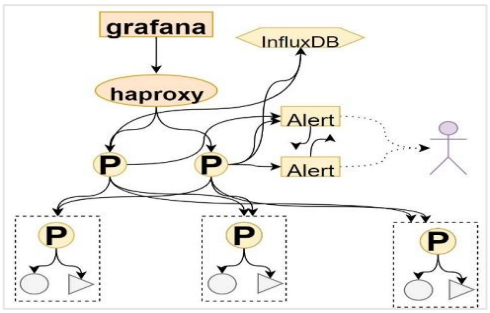

图 8 Alertmanager 高可用架构图

（1） 定义告警规则，接入告警信息，在 HA 架构基础上，把所有的告警都接入到 Alertmanager 集群中。在 Prometheus-master 节点的配置如下所示.

```
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - 10.90.104.53:9093
            - 10.90.104.53:9094
```

（2）此处配置磁盘容量的告警规则，下面在手动调小阈值，触发告警，达到测试的效果。

```
vi alertmanagers.yml
groups:
- name: DS_Monitor
  rules:
  - alert: 磁盘使用率过高
    expr:
node_filesystem_free_bytes{mountpoint !~".*pod.*",fstype=~"ext.
*|xfs"}*100/
(node_filesystem_avail_bytes{mountpoint  !~".*pod.*",fstype=~"e
```

```
xt.*|xfs"}+
(node_filesystem_size_bytes{mountpoint !~".*pod.*",fstype=~"ext.*|xfs"}-
node_filesystem_free_bytes{mountpoint !~".*pod.*",fstype=~"ext.*|xfs"})), 0.1) > 40
            for: 10s
            labels:
               severity:
            annotations:
               summary: "磁盘使用率过高"
               description: "分区点【{{ $labels.mountpoint }}】使用率大于97%,当前值【{{ $value }}%】"
```

（3）Prometheus web 页面验证

图 9 Prometheus web 页面验证

（4） 配置 Alertmanager 实现钉钉告警，测试集群的高可用。启动钉钉插件 dingtalk，修改启动配置文件，如下所示。

```
templates:
  - templates/host-dingtalk.tmpl
targets:
  webhook1:
    url:
https://oapi.dingtalk.com/robot/send?access_token=494df46bc2a2a471c2af9b7f95bae1a497ab95d5db00dc79fdf93f4cdd07206d
    secret:
SECd80f20ab84bd3d5b13b678914ba50b1285307307b65e3ba5748a5a62ff2575f0
    message:
      text: '{{ template "host.to.message" . }}'
```

（5）定义告警文件，如下所示。

```
{{ define "host.to.message" }}
{{- if gt (len .Alerts.Firing) 0 -}}
**监控告警|总数:【{{ .Alerts | len }}】**
{{- range $index, $alert := .Alerts -}}
****
****
**告警类型:**  {{ $alert.Labels.alertname }}
** 故障主机 :**  {{ $alert.Labels.instance }} {{ $alert.Labels.device }}
**  告  警  详  情  :** {{ $alert.Annotations.message }}{{ $alert.Annotations.description }}
** 告警触发时间 :**  {{ ($alert.StartsAt.Add 28800e9).Format "2006-01-02 15:04:05" }}
****
```

（6）修改集群 Alertmanager 配置文件，数据接入钉钉机器人接口，如下所示。

```
global:
  resolve_timeout: 5m
templates:
  - '/usr/local/alertmanager/*.tmpl'
# 路由分组
route:
  receiver: 'host'
  group_wait: 1s
  repeat_interval: 1m
 group_by: ['instance']
 routes:
    - match_re:
        service: 'node'
      receiver: 'host'
## 接收器指定发送人以及发送渠道
receivers:
  - name: 'host'
    webhook_configs:
-url: 'http://localhost:8060/dingtalk/webhook1/send'
      send_resolved: true
```

（7）手动依次停止 Altermanager 的 A1 和 A2 服务，告警都可以正常接收。当两个告警服务都启动的，相同的告警信息只发送一次。当在 A1 节点将告警信息静默时，A2 服务节点上的告警信息也是静默状态，由此可见，两个告警服务节点已经实现了信息共享。告警信息如下所示。

图 10 钉钉告警示例

3．4 指标可视化的实现

（1）Garfana 在创建第一个可视化仪表盘[8]之前，需要添加 Prometheus 数据源。将数据接入 grafana，可视化展示 node 的状态，下图使用 Grafana 模板 pid:11074 中的 CPU 使用率效果展示。

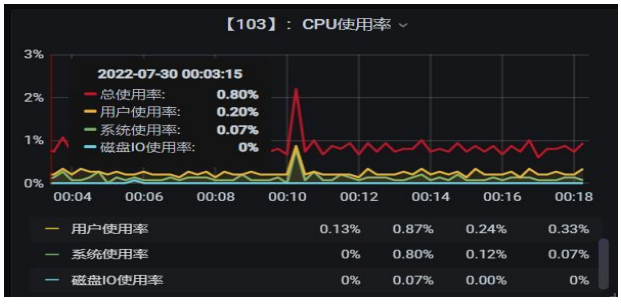

图 11 node 监控 CPU 展示

（2）使用 jmx_exporter 采集 Zookeeper 指标数据

① 启用配置项目，在 zk 的配置文件末尾追加下列配置。

```
# https://prometheus.io Metrics Exporter
metricsProvider.className=org.apache.zookeeper.metrics.prometheus.PrometheusMetricsProvider
```

```
metricsProvider.httpPort=7000
metricsProvider.exportJvmInfo=true
4lw.commands.whitelist=*
```

② 集成 jmx_exporter 到 Prometheus 服务端，Prometheus 端的配置如下所示。

```
cat prometheus.yml
  - job_name: 'zk'
    file_sd_configs:
    - files:
      - '/usr/local/prometheus/config/zk.json'
      refresh_interval: 5s
```

③ zk.json 的配置内容如下所示。

```
[
  {
    "labels": {
      "job": "zk"
    },
"targets":
["10.90.104.53:7000","10.90.104.54:7000","10.90.104.55:7000"]
    }
]
```

④ 重启 Prometheus，Prometheus ui 显示多一个 zookeeper targets 为 up 状态。

图 12 Prometheus 采集 Zookeeper

⑤ 数据通过 grafana 进行自定义面板可视化展示

图 13 Grafana 展示 Zookeeper 状态

⑥ 核心指标如下表 1 所示。

表 1 Zookeeper 核心指标详解

指标名	指标详解
num_alive_connections	活跃连接数
outstanding_requests	堆积请求数
pending_syncs	阻塞中的 sync 操作
avg_latency	平均响应延迟
open_file_descriptor_count	打开文件描述符数
max_file_descriptor_count	最大文件描述符
	数
up{instance=~".*7000.*"}	存活数量

其中，namenode、datanode、hdfs、resourcemanager、nodemanager、yarn 资源队列均是以 jmx_exporter 进行采集。并且一个 dashboard 中来汇总其他的监控模块的和核心指标，并将此 dashboard 设置为 grafana 的 Home，即大屏展示的首页。如图 14 所示。

图 14 Grafana 展示监控大屏首页

4 结束语

该系统基 Prometheus 和 Grafana 集成，Prometheus 主要与 Grafana 结合用于监控和可视化。在系统开发之前，定义了系统需求。遇到技术难题，需及时找指导老师以及同学探讨，才能保证问题得到解决，保证开发进度。该系统如期完成了开发和测试工作。在系统研发中，深入地研究了系统监控的一些相关技术以及相关技术的使用，严格按照标准的软件系统开发流程，每个功能模块紧密连接，使得本系统具有良好的可维护性和可扩展性。这些技术包括:

(1) Prometheus，提供了多维度的数据模型和灵活的查询方式，通过将监控指标与多个标签相关联，组合任意维度的监控数据，并提供了简单的 PromQL 查询方式，还提供了 HTTP 查询接口，可以方便地与 Grafana 等 GUI 组件结合显示数据。

(2) Grafana 是一个用于可视化大规模测量数据的开源程序。它提供了强大而又优雅的方法来创建、浏览和共享数据。仪表显示来自不同指标数据源的数据。

参 考 文 献

[1] 庄磊. 基于 PaaS 云平台的数据存储方案设计与应用[J]. 信息技术, 2022(02):110-115. DOI:10.13274/j.cnki.hdzj.2022.02.020.

[2] 王羿. 基于大数据的监管平台统一运维管理系统设计[J]. 电视技术, 2022, 46(02):97-100. DOI:10.16280/j.videoe.2022.02.020.

[3] 郭建磊, 车学董, 王宁宁. 基于 Prometheus 的工业云平台监控告警服务系统[J]. 信息技术与信息化, 2021(12):142-145.

[4] 朱晓祥. 基于 InfluxDB 的时序数据库优化研究与实现

[D].电子科技大学,2022.

[5] 刘平.基于钉钉的办公自动化系统的设计与实现[J].电脑知识与技术,2020,16(15):104-106.DOI:10.14004/j.cnki.ckt.2020.1760.

[6] 夏畅.基于 Zabbix 的可视化监控告警平台的研究[J].数据通信,2020(01):32-34.

[7] 殷成建. 基于 Prometheus 的分布式监控系统的设计与实现[D].浙江大学,2021.DOI:10.27461/d.cnki.gzjdx.2021.002348.

[8] 李林飞. 基于内存数据库的持久化方案的设计与实现[D].电子科技大学,2020.DOI:10.27005/d.cnki.gdzku.2020.000441.

[9] 张旭军. 负载均衡减少单点故障[N]. 网络世界,2010-04-26(013).DOI:10.28829/n.cnki.nwlsj.2010.000288.

[10] 高亚山.基于 Web 的大数据可视化系统[J].信息与电脑(理论版),2019(15):149-150