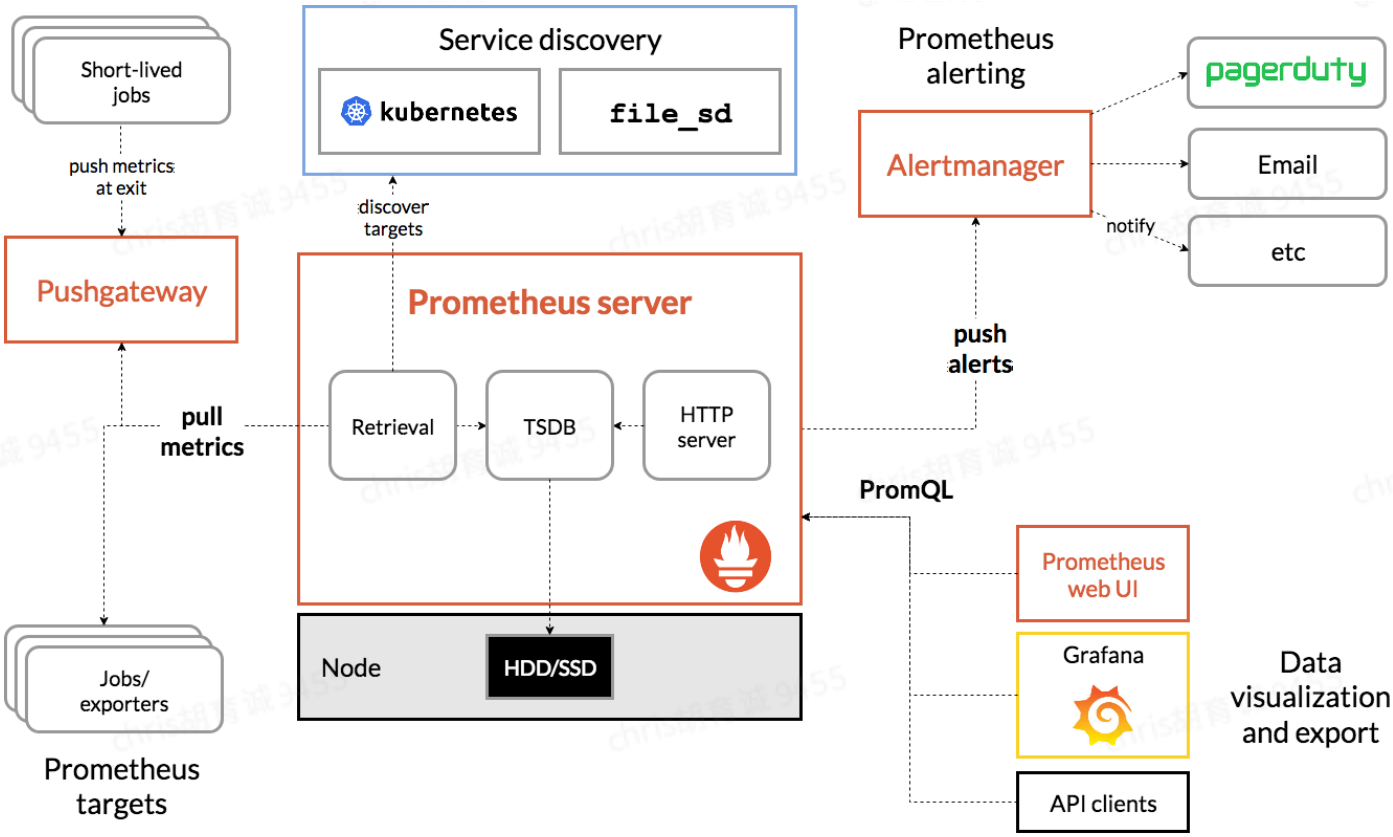
promethus监控系统设计与实现

1.需求

a.对内部组件无依赖,部署简单可用性高
b.对开源框架如spring-actuator、grafana监控有较好的支持
c.扩展性强,支持常用中间件的监控和自定义服务的监控
d.部署成本较低,涉及组件较少,无额外组件依赖
e.有活跃的github社区和生态,问题支持度高
f.部署分布式服务简单,利于后续动态扩展
g.对数据监控有一定安全措施

2.方案实现

- 应用服务使用micrometer+spring-actuator进行打点
- prometheus会定期从应用服务抓取打点指标数据(根据prometheus中配置的ip+端口)
- grafana查看指标视图时,grafana会定期(默认15s)从prometheus抓取时序数据绘图,grafana支持对原始时序指标使用函数和变量进行绘制(grafana中配置prometheus数据源ip+端口)
- prometheus抓取指标时会根据用户配置的告警规则(告警规则文件路径需在prometheus配置文件中指定),进行评估计算,对于满足告警规则的时序数据prometheus会推送到alertmanager
- alertmanager中管理告警通知方式,目前采用webhook方式,当alertmanager收到prometheus推送的告警,alertmanager会把告警指标发送到webhook地址
- 目前webhook地址会打到自定义告警项目中,自定义告警项目会对告警json数据处理成飞书机器人能支持的格式



2.1 监控技术栈

- micrometer+spring-actuator,指标采集
- prometheus,指标拉取服务和评估计算
- prometheus TSDB,时序数据存储
- prometheus alertmanager,告警通知
- grafana,指标可视化
- grafana Sqlite,grafana数据存储
- spring-security,指标端点安全校验
- 飞书机器人报警webhook

2.2 打点接口层

为了支持多种类型监控组件,抽离出打点接口层,使得具体监控组件对业务层透明.具体逻辑在mifi-spring-boot-starter

1.应用服务接入打点接口层方式:

```
<dependency>
  <groupId>com.xiaomi.mifi</groupId>
  <artifactId>mifi-spring-boot-starter</artifactId>
```

```
</dependency>
```

```
mifi.meter.config.meter.type        = prometheus
```

```
MIFIMetrics.count(metricName, 1);
MIFIMetrics.count(metricName, 1, tagA_key, tagA_value,tagB_key,tagB_value);

MIFIMetrics.histogram(metricName,value);
MIFIMetrics.histogram(metricName,value,tagA_key, tagA_value...);
```

2.接口层扩展方式:

```
CustomMeterProxy implements MeterProxy
```

```
MIFIStarterAutoConfiguration{
    @Bean
    @ConditionalOnProperty(name = "mifi.meter.config.meter.type", havingValue = "prometheus")
    public static MeterProxy customMeterProxy() {
      return new CustomMeterProxy();
    }
}
```

目前支持多种类型打点同时上报,应用项目可以配置多种打点类型.

```
mifi.meter.config.meter.type = prometheus,perfCounter
```

对于没有配置mifi.meter.config.meter.type的应用，MIFIMetrics默认使用PerfCounter打点

2.3 micrometer+spring-actuator

对mifi-spring-boot-starter引入以下依赖库

```
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
  <version>1.6.5</version>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

micrometer是一个活跃的开源指标组件facade库,支持对多种监控组件的扩展,也是prometheus推荐的接入方式

springboot-actuator已支持prometheus端点,引入micrometer-registry-prometheus库后可通过/actuator/prometheus端点抓取prometheus自定义指标+jvm指标

actuator端点安全问题通过账号验证确保,prometheus从/actuator/prometheus端点拉取数据时需要附带账号验证信息.

```
curl -u usename:password http://localhost:8010/actuator/prometheus
```

2.4 prometheus

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: "springboot"
    metrics_path: '/actuator/prometheus'
    static_configs:
    - targets: ["tj1-miui-micfc-ksc-vr1-10293-ko2v4abi4:8010"]
    basic_auth:
      username: admin
      password: admin
alerting:
  alertmanagers:
  - static_configs:
    - targets: ["tj1-miui-micfc-ksc-vr1-10293-ko2v4abi4:9093"]
rule_files:
  - ***/alert-rule.yaml
  - ***/record-rule.yaml
```

prometheus用于收集指标和时序计算评估,包括prometheus-server, alertmanager组件.搭建prometheus-server主要需要配置rule(规则),data(时序日志路径),exporter(指标客户端),alertmanager推送地址

2.4.1 rule规则

prometheus使用rule规则来评估计算时序数据,rule规则可以使用变量/模板。由于告警项目需要将alertmanage返回的告警指标拼接成prometheus原始指标,因此不建议使用报警的labels标签,防止生成指标和原始指标有出入

```
groups:
- name: example
```

```
  rules:

  # Alert for any instance that is unreachable for >5 minutes.
  - alert: InstanceDown
    expr: up == 0
    for: 5m
    labels:#不建议使用labels
      severity: page
    annotations:
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
```

2.4.2 prometheus HA

目前prometheus是单机本地数据库,会收到本机磁盘大小的限制,因此数据无法持久化,会定期清理,默认清理策略有基于时间和存储空间.
实现prometheus高可用方案，联邦集群模式

2.4.3 时序数据库

2.4.3.1 内置本地数据库

默认内置tsdb数据库,prometheus默认使用内置tsdb,属于单点本地数据库,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。

2.4.3.2 远程数据库

prometheus支持远程数据库,实现数据的持久化和一致性

2.4.4 prometheus查询

prometheus学习成本主要包括:

- promQL模板查询语言
- 指标类型计算(counter/guage/histogram/summary)
- alert 监控评估表达式

prometheus web ui控制台

2.4.5 alertManager
alertManager集群模式

2.5 grafana

grafana对各种监控组件有很好的支持,并且支持与第三方软件的联动.grafana也支持告警.目前只使用grafana的可视化功能.

目前飞书机器人告警通知 已支持 跳转到 grafana 特定指标面板,实现动态绘制指标视图.主要通过生成一个固定的临时面板,所有告警通知都跳转至该临时面板,跳转时将指标变量传入面板,面试对指标变量进行匹配,匹配到对应时序数据时会拉取该时序数据进行绘制

临时面板根据指标动态绘制

2.5.1 grafana查看95-percent,99-percent

histogram_quantile

```
histogram_quantile(0.99, dubbo_invoke_seconds_bucket)
```

2.6 spring-security

springboot-actuator 端点涉及到数据安全问题,目前在mifi-spring-boot-starter中只打开/actuator/prometheus端点,并使用账号密码限制其他应用抓取数据.主要引入spring-boot-starter-security库

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

配置如下

mifi-spring-boot-starter

```
management.endpoints.web.exposure.include = prometheus
management.endpoint.prometheus.enabled    = true
spring.security.user.name                 = admin
spring.security.user.password             = admin
```

prometheus-sever

```
scrape_configs:
  - job_name: "springboot"
    metrics_path: '/actuator/prometheus'
    static_configs:
    - targets: ["localhost:8010"]
    basic_auth:
      username: admin
      password: admin
```

2.7 飞书webhook

使用现有的机器人发群消息,来实现告警,具体查看飞书群:[智能决策平台报警]

3.QA review

3.1 现有PerfCounter指标名不符合惯例

- 现有PerfCounter指标名包含"/"字符,目前建议修改为".",prometheus打点不支持"/"字符.
- 现有PerfCounter每个打点一个指标名,不太适合告警规则正则匹配,建议一个功能统一固定指标名,具体打点名附加在label/tags中

3.2 打点接口tag需成对出现

```
1 public void count(String name, long count, String... tags)
```

tags必须为偶数,符合label规范key=value,否则会抛异常,exg:

```
1 public void count("dubbo.invoke", 1,"method","tostring");
```

3.3 未申明打点类型时MIFIMetrics默认使用PerfCounter打点

3.4 actuator端点安全

目前只开放了/actuator/prometheus端点,其他端点已禁用,另外访问/actuator/prometheus需进行账号验证.

3.5

1.prometheus指标名只支持[a-zA-Z_:][a-zA-Z0-9_:]*,现有指标名中有无效字符需要更改

2.PerfCounterFilter中原有指标名改为label,指标名作为固定值方便模板计算

```
1     old:PerfCounter.count(successCountName, 1);
2     new:MIFIMetrics.count("dubbo.invoke", 1, "method", successCountName);
```

3.perfcounter部分指标1-min/5-min暂时不支持

4.对于依赖mifi-spring-boot-starter的服务,使用MIFIMetrics打点时默认为PerfCounterMeterProxy,如需切换为PrometheusMeterProxy，需要配置

```
1 mifi.meter.config.meter.type                    = prometheus
```

4.待完成

4.1 打点接入

1. 目前prometheus打点只支持count/guage/histogram,对于perfcounter中的特殊指标1-min/5-min不支持
2. 各服务需要迁移打点功能

4.2 数据安全

1. 接入了表单验证账号密码验证方式,考虑转成HTTP Basic authentication方式,避免明文
2. grafana从prometheus拉取数据考虑进行安全验证
3. prometheus推送数据给alertmanager考虑进行安全验证

4.3 prometheus告警规则可视化

1. falcon的告警规则迁移到prometheus
2. prometheus原生告警规则通过yml进行配置,无可视化控制台,可考虑接入可视化开源规则控制台promgen

4.4 时序数据库管理

prometheus内置时序数据库,需要对时序数据库进行维护管理

4.5 告警通知

目前只支持webhook方式告警,邮箱通知方式未接入

4.6 grafana部署

目前使用现有grafana,接入消金环境需要另外部署

4.7 飞书告警相关

飞书告警文档格式

```
Endpoint: c3-miui-fi-app01.bj

Metric: 95-percentile  all(#3): 6103 >= 1000

Labels: cluster=c3,cop=xiaomi,job=mifi-info-center,name=thriftcli-MifiHBaseProxyService-ALL,owt=miui,pdl=credit,service=mifi-info-center

报警Note:

```

```
9 报警时间: 2021-04-07 19:02:13
```

5.待优化

5.1 grafana数据库

grafana内置sqlite数据库,可考虑升级为mysql

5.2 分布式

目前prometheus,alertmanager,grafana均为单点服务,可考虑升级为集群模式

5.3 容器化部署

目前通过二进制文件部署prometheus,alertmanager,grafana,可考虑容器化部署,简化部署流程