11 | 机器监控：操作系统有哪些指标需要重点关注？

2023-02-01 秦晓辉 来自北京

《运维监控系统实战笔记》

讲述：秦晓辉

时长 14:51　大小 13.57M

你好，我是秦晓辉。

前面两讲我从方法论和技术实现角度给你介绍了监控数据的采集原理及方法，这些方法论是我们搞定后面各种监控需求的基础，这里你可以再结合总结图复习一下。有了这些理论基础之后，我们就可以动手实操了。

监控方向里我们最耳熟能详的，就是机器监控，也就是我们前面说的设备监控中的一种。机器是进程运行的基础环境，在制作中间件、应用监控仪表盘的时候，我们一般会把机器核心指标，比如 CPU、内存、磁盘、网络、IO 等，作为仪表盘的一部分，USE 方法论主要就是针对机器监控提出的，其重要性不言而喻，所以今天我们就从机器监控开始聊起。

机器监控手段

机器层面的监控分为两部分，带内监控和带外监控。带内监控就是通过带内网络来监控，主要是以在 OS 里部署 Agent 的方式，来获取 OS 的 CPU、内存、磁盘、IO、网络、进程等相关监控指标。随着云时代的到来，普通运维研发人员主要关注带内监控即可，IDC 运维人员才会关注带外监控。不过为了让你的知识网络更加完整，带外监控我也浅聊几句。

带外监控走的是带外网络，通常和业务网络不互通，通过 IPMI、SNMP 等协议获取硬件健康状况。

IPMI 可用于监控硬件的物理参数，如系统温度、风扇速度、电源电压等，可以有效地利用 IPMI 监控硬件温度、功耗、启动或关闭服务器和系统，以及进行日志记录。IPMI 的一个主要亮点是，它的功能独立于服务器的 CPU 和操作系统。因为固件是直接在服务器主板上运行的，所以不管安装的操作系统是什么，它都可以用于管理各种远程位置的服务器。

BMC（服务器基板管理控制器）也可以开启 SNMP 的支持，通过 SNMP Trap 做硬件监控是一个很好的思路。不过目前没有看到比较好的产品，可能一些老牌的国外监控产品可以做，但是那些产品都偏老套且收费昂贵。不过现在大都在上公有云，传统的 SNMP Trap 的监控，已经是一个存量需求了，不懂也不用太过焦虑。

我们再回来看带内监控，常见的 Agent 有 Categraf、Telegraf、Grafana-agent、Datadog-agent、Node-exporter 等，这一讲我们除了介绍基本的 CPU、内存、硬盘相关的监控，还会介绍一下进程监控和自定义监控脚本，所以我们选择 Categraf，相对比较方便。

Categraf 介绍

Categraf 作为一款 Agent 需要部署到所有目标机器上，因为采集 CPU、内存、进程等指标，是需要读取 OS 里的一些信息的，远程读取不了。采集到数据之后，转换格式，传输给监控服务端。

Categraf 推送监控数据到服务端，走的是 Prometheus 的 RemoteWrite 协议，是基于 Protobuf 的 HTTP 协议，所以所有支持 RemoteWrite 的后端，都可以和 Categraf 对接，比如 Prometheus、Nightingale、TDEngine 等。

Categraf 在 Github 的 Releases 页面提供了所有的版本信息，每个版本的发布，都会提供 changelog 和二进制发布包。Categraf 默认提供 Linux 和 Windows 两个平台的发布包，当然，Categraf 也可以跑在 macOS 上，如果有兴趣你也可以自行编译测试。

Categraf 配置

为了方便看到效果，我把 Categraf 采集的数据推送给 Nightingale，即 n9e-server 的 `/prometheus/v1/write` 地址，你可以看一下相关的配置。

复制代码

```
[[writers]]
url = "http://127.0.0.1:19000/prometheus/v1/write"
```

注：Categraf 的主配置文件是在 conf 目录下的 config.toml，解压缩发布包就可以看到了。

当然，如果你想让 Categraf 把数据推给 Prometheus，也可以。但需要把 127.0.0.1:19000 修改为你的环境的 Prometheus 地址，还要修改 URL 路径，因为 Prometheus 的 RemoteWrite 数据接收地址是 /api/v1/write。

最后还要注意，Prometheus 进程启动的时候，需要增加一个启动参数 `--enable-feature=remote-write-receiver`，重启 Prometheus 就能接收 RemoteWrite 数据了。

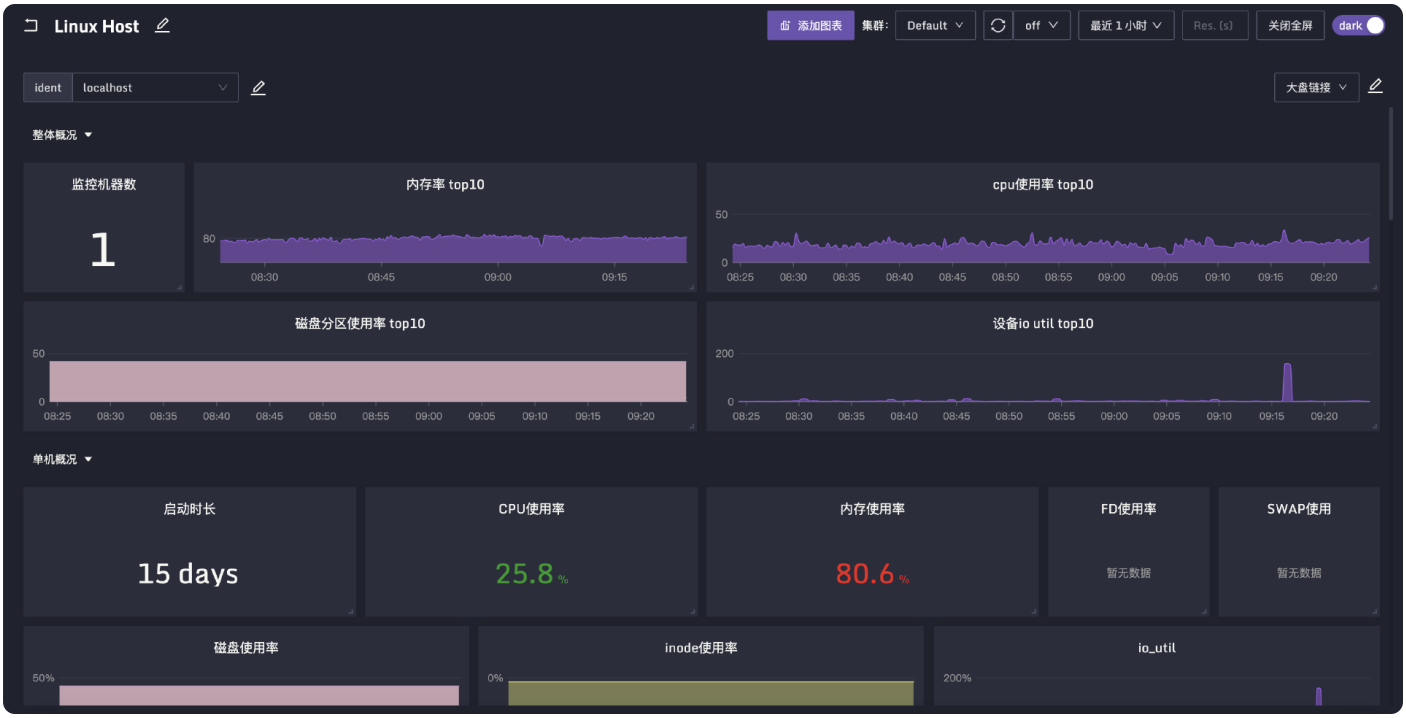
配置完就可以启动了，不过启动之前，我们先来简单测试一下。通过 `./categraf --test` 看看 Categraf 有没有报错，正常情况会在命令行输出采集到的监控数据，下面是我的环境下的运行结果，你可以参考下。

```
[root@tt-fc-dev01.nj categraf]# ./categraf --test --inputs mem:system
2022/11/05 09:14:31 main.go:110: I! runner.binarydir: /home/work/go/src/categra
2022/11/05 09:14:31 main.go:111: I! runner.hostname: tt-fc-dev01.nj
2022/11/05 09:14:31 main.go:112: I! runner.fd_limits: (soft=655360, hard=655360
2022/11/05 09:14:31 main.go:113: I! runner.vm_limits: (soft=unlimited, hard=unl
2022/11/05 09:14:31 config.go:33: I! tracing disabled
2022/11/05 09:14:31 provider.go:63: I! use input provider: [local]
2022/11/05 09:14:31 agent.go:85: I! agent starting
2022/11/05 09:14:31 metrics_agent.go:93: I! input: local.mem started
2022/11/05 09:14:31 metrics_agent.go:93: I! input: local.system started
2022/11/05 09:14:31 prometheus_scrape.go:14: I! prometheus scraping disabled!
2022/11/05 09:14:31 agent.go:96: I! agent started
09:14:31 system_load_norm_5 agent_hostname=tt-fc-dev01.nj 0.3
09:14:31 system_load_norm_15 agent_hostname=tt-fc-dev01.nj 0.2675
09:14:31 system_uptime agent_hostname=tt-fc-dev01.nj 7307063
09:14:31 system_load1 agent_hostname=tt-fc-dev01.nj 1.66
09:14:31 system_load5 agent_hostname=tt-fc-dev01.nj 1.2
09:14:31 system_load15 agent_hostname=tt-fc-dev01.nj 1.07
09:14:31 system_n_cpus agent_hostname=tt-fc-dev01.nj 4
09:14:31 system_load_norm_1 agent_hostname=tt-fc-dev01.nj 0.415
09:14:31 mem_swap_free agent_hostname=tt-fc-dev01.nj 0
09:14:31 mem_used agent_hostname=tt-fc-dev01.nj 5248593920
09:14:31 mem_high_total agent_hostname=tt-fc-dev01.nj 0
09:14:31 mem_huge_pages_total agent_hostname=tt-fc-dev01.nj 0
09:14:31 mem_low_free agent_hostname=tt-fc-dev01.nj 0
...
```

测试没问题，就可以启动了。通过 `nohup ./categraf &> stdout.log &` 简单启动即可，如果是生产环境，建议你使用 systemd 或 supervisor 等托管进程。

导入配置

Categraf 内置 Linux 监控大盘，导入 Nightingale 就能看到效果，当然它也内置了告警规则，你可以看一下规则的 JSON 文件。下面是我的环境的大盘效果图。



走完上面的几步，我们就可以看到监控效果了。下面我们就来看一下这些数据是怎么来的。

Categraf 插件

Categraf 和 Telegraf、Datadog-agent 类似，都是插件架构，采集 CPU 指标的是 cpu 插件，采集内存指标的是 mem 插件，采集进程指标的是 procstat 插件，每个插件都有一个配置目录，在 conf 目录下，以 input. 打头。

复制代码

```
ulric@localhost conf % ls
categraf.service             input.exec                input.mem
config.toml                  input.greenplum           input.mongodb
example.input.jolokia_agent  input.http_response       input.mtail
input.arp_packet             input.ipvs                input.mysql
input.conntrack              input.jenkins             input.net
input.cpu                    input.kafka               input.net_response
input.disk                   input.kernel              input.netstat
input.diskio                 input.kernel_vmstat       input.netstat_filter
input.dns_query              input.kubernetes          input.nfsclient
input.docker                 input.linux_sysctl_fs     input.nginx
input.elasticsearch          input.logstash            input.nginx_upstream_ch
```

OS 的各类指标采集，大都是读取的 /proc 目录，上一讲我们已经介绍过了，这里不再赘述。对于常见的插件，Categraf 中有哪些配置可以控制其行为，下面我们一起看一下。

cpu

cpu 插件的配置文件在 input.cpu 目录，只有两个配置项，interval 和 collect_per_cpu，其中 interval 表示采集频率，所有的插件都有这个配置。而和 CPU 采集相关的配置实际只有一个，就是 collect_per_cpu，它用于控制是否采集每个单核的指标。默认值是 false，表示不采集单核的指标，只采集整机的指标。

CPU 相关的指标，最核心的就是使用率。大型互联网公司，为了应对突发流量，CPU 的平均利用率一般就是 30% 左右。如果平时 CPU 利用率总是超过 60%，就比较危险了。

mem

mem 插件的配置文件在 input.mem 目录，核心配置只有一个 collect_platform_fields，它表示是否采集各个平台特有的指标，什么意思呢？内存相关的指标，Linux、Windows、BSD、Darwin 都不是完全一样的，内存总量、使用量这类指标所有平台都有，但是其余很多指标都是和平台相关的，比如 Linux 下特有的 swap、huge_page 相关的指标，其他平台就是没有的。

内存相关的指标，最核心的是可用率，即 mem_available_percent。新版本的 OS 内置 available 指标，老版本的姑且可以把 free + cached + buffer 看做 available。这个值如果小于 30% 就可以发个低级别告警出来了，然后着手扩容。

disk

disk 插件的配置文件在 input.disk 目录，默认不用调整，如果某个挂载点不想采集，可以配置到 ignore_mount_points 中，这样就会自动忽略掉对应的挂载点。

硬盘相关的指标，主要关注空间使用率和 inode 使用率。对于空间使用率，85% 一刀切的告警规则不合适，建议考虑大空间的盘和小空间的盘分别对应不同的告警规则。比如：

复制代码

```
disk_used_percent{app="clickhouse"} > 85
and
disk_total{app="clickhouse"}/1024/1024/1024 < 300
```

这个规则用了 and 关键字，除了使用率大于 85 之外，又加了一个限制条件：总量小于 300G。

diskio

diskio 插件的配置文件在 input.diskio 目录，默认不用调整，如果只想采集某几个盘的 IO 数据，可以使用 devices 配置进行限制。

硬盘 IO 相关的指标，主要关注读写延迟，所谓的 IO.UTIL 这种指标基本不用太关注。这个问题网络上讨论比较多，你也可以 Google 一下。

net

net 插件的配置文件在 input.net 目录，默认不用调整，如果只想采集某个特定网卡的数据，可以修改 interfaces 配置。

现在的 IDC 环境，动辄万兆网卡，普通业务根本不用担心网卡跑满的问题，只有专门作为接入层的机器才需要重点关注。不过网卡丢包是需要关注的，比如长期每秒几十个丢包，就有理由怀疑是硬件的问题了。

netstat

netstat 插件的配置文件在 input.netstat 目录，默认不用调整，tcp_time_wait 指标就是靠这个插件采集的。

processes

processes 插件的配置文件在 input.processes 目录，默认不用调整。最常见的指标是进程总量，如果某个机器进程总量超过 600，大概率是有问题的，常见的原因比如某个 cron 写“挫”了，每分钟 fork 一个进程，但就是不退出，时间长了就把机器压垮了。

conntrack

conntrack 插件的配置文件在 input.conntrack 目录，默认不用调整。如果你工作时间比较久，应该遇到过 conntrack table full 的报错吧。这个插件就是用来监控 conntrack 的情况的。我们

可以配置一条这样的告警规则及时发现问题：`conntrack_ip_conntrack_count / ip_conntrack_max > 0.8` 就可以告警。

kernel_vmstat

kernel_vmstat 插件的配置文件在 input.kernel_vmstat，采集的是 `/proc/vmstat` 的指标数据。这个文件内容比较多，所以配置文件中给了一个白名单，你可以按需启用，默认启用了 oom_kill。这样一来，OS 发生 OOM，我们可以及时知道。不过这需要高版本的内核，低版本是没有的。低版本的 OS 可以通过监控内核日志的“Out of memory”关键字来实现 OOM 监控。

ntp

ntp 插件的配置文件在 input.ntp 目录，需要在配置文件中给出 ntp server 的地址。每个公司都应该有对时机制，很多分布式程序都是重度依赖时间的。监控指标是 ntp_offset_ms，顾名思义，单位是毫秒，一般这个值不能超过 1000，也不能小于 -1000，需要配置告警规则。

procstat

procstat 插件的配置文件在 input.procstat 目录，用于监控进程。进程监控主要是两个方面，一个是进程存活性，一个是进程占用的 CPU、内存、文件句柄等数据。

其中特别需要注意的是文件句柄，也就是 fd 相关的指标。Linux 默认给进程分配的 fd 数量是 1024，有些人调整了 ulimit 之后发现不好使，核心原因是进程在 ulimit 调整之前就启动了，或者是因为 systemd 层面没有调好。没关系，通过 fd 这个监控指标，我们最终都可以发现问题。procstat_rlimit_num_fds_soft < 2048 就告警，需要配置这个规则。

exec

exec 插件的配置文件在 input.exec 目录，用于自定义监控脚本。因为 Agent 虽然内置了很多插件，但有些长尾需求仍然是解决不了的，此时就需要通过自定义监控脚本来扩展监控采集能力。

监控脚本可以是 Shell 的，也可以是 Python、Perl、Ruby 的，只要机器上有对应的执行环境就可以。Agent 会 fork 一个进程来执行，截获进程的 stdout（标准输出），解析成监控数据然后上报。所以脚本采集到数据之后，要格式化成某个特定的格式，打印到 stdout。Categraf 支持三种格式：Prometheus、Influx、Falcon，默认是 Influx 格式。

我们简单举个例子，监控某个目录的大小。


```
#!/bin/sh

data1=`du -sb /data1`
data2=`du -sb /data2`

echo "du,dir=/data1 size=$data1"
echo "du,dir=/data2 size=$data2"
```

这个例子使用的就是 Influx 输出格式，这个格式比较简单，我们第 2 讲已经详细介绍过了，这里就不再赘述。当然，我们也可以采用 Prometheus 的格式或 Falcon 的格式。把这个脚本的路径配置到 exec.toml 中，Categraf 就会周期性地执行这个脚本，采集这两个目录的大小了。

常见采集插件	相关配置及关注重点
cpu	· 可以开启单核采集 · 重点关注利用率指标
mem	· 默认开启平台特有的内存指标的采集 · 重点关注可用率指标
disk	· 磁盘利用率、inode 利用率的监控 · 不同大小的盘，可能要采用不同的规则
diskio	· 重点关注磁盘读写延迟数据 · 不用太过关注 io.util 指标
net	· 网卡流量、包量等的监控 · 重点关注丢包量的指标
netstat	· 重点关注 tcp_time_wait 指标
processes	· 重点关注机器进程总数
conntrack	· K8s 的机器大概率会启用相关内核模块，如果没有加载相关内核模块则不用监控 · 重点关注使用率
kernel_vmstat	· 重点关注 oom_kill，新版本才有
ntp	· 监控 ntp 偏移量，通常不能大于 1 秒，否则对一些分布式程序有影响 · 每个公司都应该有自己的对时机制
procstat	· 进程监控，用于监控进程是否存活，以及进程占用的 CPU、内存资源 · 重点关注进程占用的 fd 数量，这是个大坑
exec	· 自定义监控脚本，支持 Shell、Python、Perl、Ruby，甚至具备可执行权限的二进制 · 支持 Influx、Prometheus、Falcon 多种格式，Falcon 的插件可以直接复用

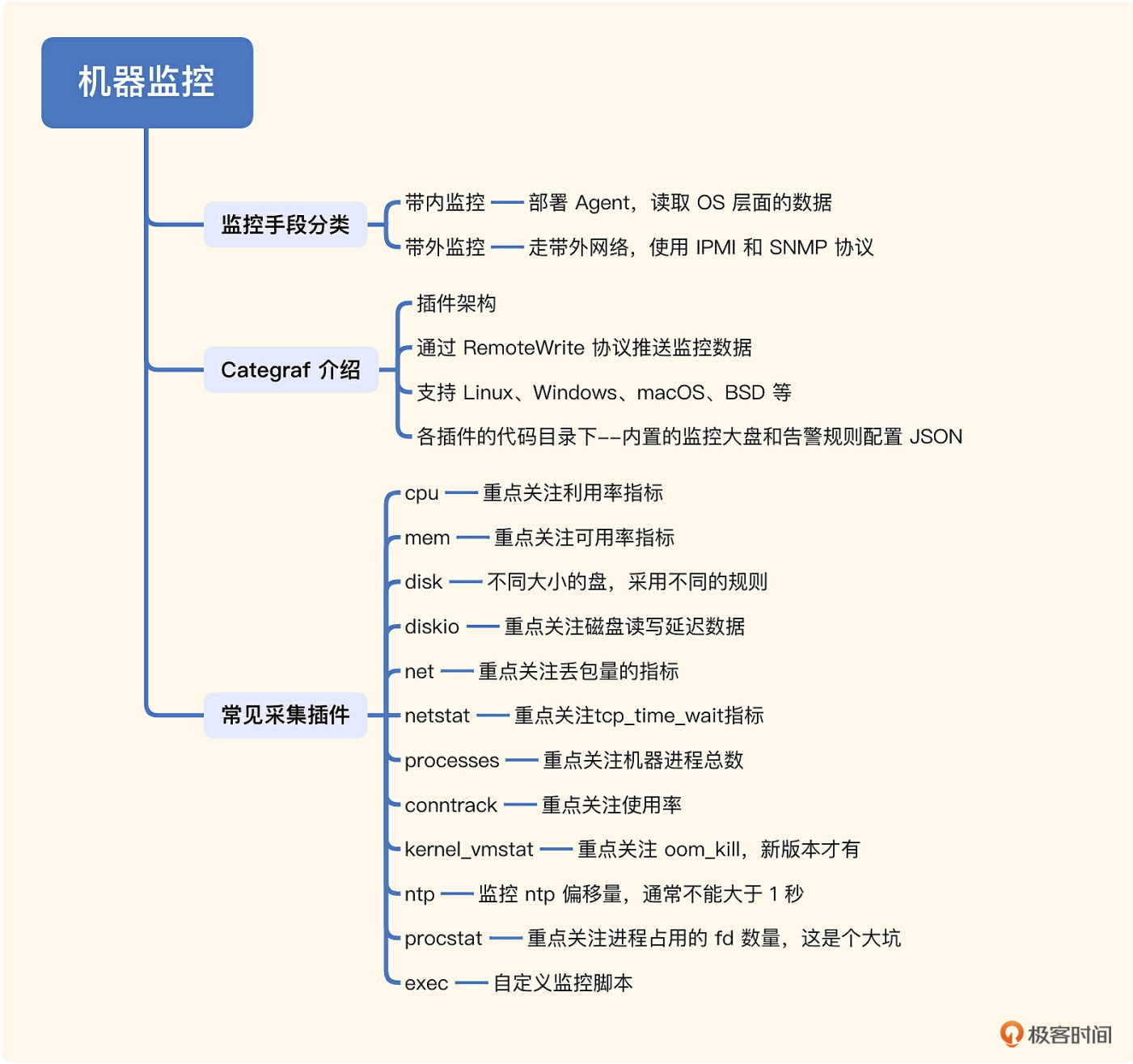
Categraf 里一些常见的跟机器相关的插件及重点关注的指标我们就介绍完了，插件比较多，所以最后我给你整理了一张表格，你可以再对照着这张表格，再在脑海中复现一下这些知识点。

小结

今天我们重点介绍了机器监控，机器监控的手段有两种，基于 Agent 的这种监控手段走的是带内业务网络，也就是带内监控，这种方式最为常用；另外一种机器监控手段是带外监控，走的是带外网络，通过 IPMI、SNMP 等协议，常用于采集硬件的物理指标，比如系统温度、风扇速度、电源电压等。现在已经是云时代，使用公有云的公司越来越多，所以就算你不了解带外监控，也没关系。

我们使用 Categraf 作为采集器，把 Categraf 里常见的机器监控插件都介绍了一下，这些插件可以帮我们搞定机器常规指标的采集。比如 cpu 插件可以监控 CPU 的使用率，mem 插件可

以用来监控内存的可用率，disk 插件可以用来监控空间使用率，diskio 插件可以采集硬盘 IO 相关的指标，还有 exec 可以用来自定义监控脚本，等等。搞定这些插件，机器方面的监控也就不成问题了。



互动时刻

在 Categraf 的插件部分，我们介绍了一种自定义监控脚本的方法，这种方式灵活度很高，到底哪些场景需要自定义脚本呢？欢迎你在评论区留言，如果能同时提供脚本链接，那就更好了，我们相互交流，打开视野。也欢迎你把今天的内容分享给你身边的朋友，邀他一起学习。我们下一讲再见！

上一篇　10丨监控概论（下）：监控数据的采集方式及原理

精选留言 (5)

写留言

hshopeful
2023-02-01 来自湖北

需要使用自定义脚本的场景：
1、可以在脚本里面针对现有的监控指标进行运算得到新的监控指标（成功率）
2、可以在脚本里面执行 sql 命令从 mysql 获取一些有意义的业务指标，其他存储系统类似
3、对于一些第三方组件（不好改），有暴露文本格式的监控指标（但是不符合 prometheus 的格式标准），可以通过自定义脚本进行格式转换

作者回复:

1

怀朔
2023-02-01 来自浙江

业务场景。 如：视频转码队列 、视频转码成功 、支付成功率 等等

作者回复:

1

peter
2023-02-02 来自北京

请教老师两个问题：
Q1：设备监控在云平台上怎么实现？
我的网站计划用阿里云，请问，对于设备监控，云平台是否提供？如果提供，是否足够？

Q2：哪个云平台更好？

老师的公司，是用云平台吗？还是自建机房？ 如果选择云平台，目前国内的几个平台，包括阿里云、腾讯云、华为云等，老师倾向于选择哪一个？

leeeo

2023-02-01 来自四川

请问一下：如果从非prometheus升级到prometheus架构，老的监控的历史数据如何迁移到prometheus时序数据库中呢？

作者回复: 太笼统了没法回答。一般较难迁移，两个系统双跑一段时间

DBRE

2023-02-01 来自北京

请问Categraf 支持通过api的方式来修改自身配置和插件配置吗？

作者回复: config.toml里有个http_provider，通过这种方式支持从远端拉取配置，不过需要自行实现配置管理，然后实现categraf要求的这个接口，就可以对接起来了。之前有几个公司已经这么干了。