# Linux常用监控指标 | Open-Falcon

https://book.open-falcon.org/zh/faq/linux-metrics.html

Linux运维基础采集项

做运维，不怕出问题，怕的是出了问题，抓不到现场，两眼摸黑。所以，依靠强大的监控系统，收集尽可能多的指标，意义重大。但哪些指标才是有意义的呢，本着从实践中来的思想，各位工程师在长期摸爬滚打中总结出来的经验最有价值。

在各位运维工程师长期的工作实践中，我们总结了在系统运维过程中，经常会参考的一些指标，主要包括以下几个类别：

* CPU
* Load
* 内存
* 磁盘
* IO
* 网络相关
* 内核参数
* ss 统计输出
* 端口采集
* 核心服务的进程存活信息采集
* 关键业务进程资源消耗
* NTP offset采集
* DNS解析采集

每个类别，具体的详细指标如下，这些指标，都是open-falcon的agent组件直接支持的。falcon-agent每隔一定时间间隔（目前是60秒）会采集一次相关的指标，并汇报给server端。

## CPU相关采集项

计算方法：通过采集/proc/stat来得到，大家可以参考sar命令的统计输出来理解。

* cpu.idle：Percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.
* cpu.busy：与cpu.idle相对，他的值等于100减去cpu.idle。
* cpu.guest：Percentage of time spent by the CPU or CPUs to run a virtual processor.
* cpu.iowait：Percentage of time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.
* cpu.irq：Percentage of time spent by the CPU or CPUs to service hardware interrupts.
* cpu.softirq：Percentage of time spent by the CPU or CPUs to service software interrupts.
* cpu.nice：Percentage of CPU utilization that occurred while executing at the user level with nice priority.
* cpu.steal：Percentage of time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.
* cpu.system：Percentage of CPU utilization that occurred while executing at the system level (kernel).
* cpu.user：Percentage of CPU utilization that occurred while executing at the user level (application).
* cpu.cnt：cpu核数。
* cpu.switches：cpu上下文切换次数，计数器类型。

磁盘相关采集项

计算方法：先读取/proc/mounts拿到所有挂载点，然后通过syscall.Statfs\_t拿到blocks和inode的使用情况。每个metric都会附加一组tag描述，类似mount=$mount,fstype=$fstype，其中$mount是挂载点，比如/home，$fstype是文件系统，比如ext4。

* df.bytes.free：磁盘可用量，int64
* df.bytes.free.percent：磁盘可用量占总量的百分比，float64，比如32.1
* df.bytes.total：磁盘总大小，int64
* df.bytes.used：磁盘已用大小，int64
* df.bytes.used.percent：磁盘已用大小占总量的百分比，float64
* df.inodes.total：inode总数，int64
* df.inodes.free：可用inode数目，int64
* df.inodes.free.percent：可用inode占比，float64
* df.inodes.used：已用的inode数据，int64
* df.inodes.used.percent：已用inode占比，float64

megacli工具输出

使用 megacli 工具读取 RAID 相关信息，每个metric都会附件一组tag描述，用来标明所属PD或者 VD，PD格式为PD=Enclosure\_ID:SLOT\_ID，比如PD=32:0表明第一块磁盘 ，VD=0 表明第一个逻辑磁盘。

* sys.disk.lsiraid.pd.Media\_Error\_Count：这个及以下三个指标目前仅作为数据收集，不一定意味磁盘损坏（只是表示损坏概率变大）
* sys.disk.lsiraid.pd.Other\_Error\_Count
* sys.disk.lsiraid.pd.Predictive\_Failure\_Count
* sys.disk.lsiraid.pd.Drive\_Temperature
* sys.disk.lsiraid.pd.Firmware\_state：如果值不为0，则此物理磁盘出现问题
* sys.disk.lsiraid.vd.cache\_policy：如果值不为0，表示此逻辑磁盘缓存策略和设置不符
* sys.disk.lsiraid.vd.state： 如果值不为0，表示此逻辑磁盘出现问题

SMART工具输出

使用 smartctl 工具读取磁盘 SMART 信息，目前所有指标仅作为数据收集，不一定意味磁盘损坏（只是表示概率变大），每个metric都会有一组tag描述，表明盘符，例如device=/dev/sda。

* sys.disk.smart.Reallocated\_Sector\_Ct
* sys.disk.smart.Spin\_Retry\_Count
* sys.disk.smart.Reallocated\_Event\_Count
* sys.disk.smart.Current\_Pending\_Sector
* sys.disk.smart.Offline\_Uncorrectable
* sys.disk.smart.Temperature\_Celsius

分区读写监控

测试所有已挂载分区是否可读写，每个metric都会有一组tag描述，表示挂载点，比如mount=/home

* sys.disk.rw： 如果值不为0，表明此分区读写出现问题

IO相关采集项

计算方法：每秒采集一次/proc/diskstats，计算差值，都是计数器类型的。每个metric都会有一组tag描述，形如device=$device，用来表示具体的设备，比如sda1、sdb。用户可以参考iostat的帮助文档来理解具体的metric含义。

* disk.io.ios\_in\_progress：Number of actual I/O requests currently in flight.
* disk.io.msec\_read：Total number of ms spent by all reads.
* disk.io.msec\_total：Amount of time during which ios\_in\_progress >= 1.
* disk.io.msec\_weighted\_total：Measure of recent I/O completion time and backlog.
* disk.io.msec\_write：Total number of ms spent by all writes.
* disk.io.read\_merged：Adjacent read requests merged in a single req.
* disk.io.read\_requests：Total number of reads completed successfully.
* disk.io.read\_sectors：Total number of sectors read successfully.
* disk.io.write\_merged：Adjacent write requests merged in a single req.
* disk.io.write\_requests：total number of writes completed successfully.
* disk.io.write\_sectors：total number of sectors written successfully.
* disk.io.read\_bytes：单位是byte的数字
* disk.io.write\_bytes：单位是byte的数字
* disk.io.avgrq\_sz：下面几个值就是iostat -x 1看到的值
* disk.io.avgqu-sz
* disk.io.await
* disk.io.svctm
* disk.io.util：是个百分数，比如56.43，表示56.43%

机器负载相关采集项

计算方法：读取/proc/loadavg，都是原始值类型的：

* load.1min
* load.5min
* load.15min

内存相关采集项

计算方法：读取/proc/meminfo 中的内容，其中的mem.memfree是free+buffers+cached，mem.memused=mem.memtotal-mem.memfree。用户具体可以参考free命令的输出和帮助文档来理解每个metric的含义。

* mem.memtotal：内存总大小
* mem.memused：使用了多少内存
* mem.memused.percent：使用的内存占比
* mem.memfree
* mem.memfree.percent
* mem.swaptotal：swap总大小
* mem.swapused：使用了多少swap
* mem.swapused.percent：使用的swap的占比
* mem.swapfree
* mem.swapfree.percent

网络相关采集项

计算方法：读取/proc/net/dev的内容，每个metric都附加有一组tag，形如iface=$iface，标明具体那个interface，比如eth0。metric中带有in的表示流入情况，out表示流出情况，total是总量in+out，支持的metric如下：

* net.if.in.bytes
* net.if.in.compressed
* net.if.in.dropped
* net.if.in.errors
* net.if.in.fifo.errs
* net.if.in.frame.errs
* net.if.in.multicast
* net.if.in.packets
* net.if.out.bytes
* net.if.out.carrier.errs
* net.if.out.collisions
* net.if.out.compressed
* net.if.out.dropped
* net.if.out.errors
* net.if.out.fifo.errs
* net.if.out.packets
* net.if.total.bytes
* net.if.total.dropped
* net.if.total.errors
* net.if.total.packets

端口采集项

计算方法，通过ss -ln，来判断指定的端口是否处于listen状态。原始值类型，值要么是1：代表在监听，要么是0，代表没有在监听。每个metric都附件一组tag，形如port=$port，$port就是具体的端口。

* net.port.listen

机器内核配置

* kernel.maxfiles： 读取的/proc/sys/fs/file-max
* kernel.files.allocated：读取的/proc/sys/fs/file-nr第一个Field
* kernel.files.left：值=kernel.maxfiles-kernel.files.allocated
* kernel.maxproc：读取的/proc/sys/kernel/pid\_max

ntp采集项

使用 ntpq -pn 获取本机时间相对于 ntp 服务器的 offset。

* sys.ntp.offset： 本机偏移时间，单位为ms，值过大或者为0则表明有异常，需要报警

进程监控

* proc.num：判断某个进程的数目，这里需要分两个场景，一种是根据进程的名字来判定，比如name=sshd；另外一种是根据cmdline来判定，比如Java的应用进程名可能都是java，根据第一种情况没法做区分，此时可以配置cmdline，如cmdline=./falcon\_agent-c./cfg.ini

进程资源监控

* process.cpu.all：进程和它的子进程使用的sys+user的cpu，单位是jiffies
* process.cpu.sys：进程和它的子进程使用的sys cpu，单位是jiffies
* process.cpu.user：进程和它的子进程使用的user cpu，单位是jiffies
* process.swap：进程和它的子进程使用的swap，单位是page
* process.fd：进程使用的文件描述符个数
* process.mem：进程占用内存，单位byte

## ss命令输出

* ss.orphaned
* ss.closed
* ss.timewait
* ss.slabinfo.timewait
* ss.synrecv
* ss.estab