# 服务器监控

## 指标

### cpu

1. 负载， uptime指令获取

一般而言，服务器的合理负载是CPU核数\*2。也就是说对于8核的CPU，负载在16以内表明机器运行很稳定流畅。如果负载超过16了，就说明服务器的运行有一定的压力了。对于负载升高的另一重要原因——IO没有清晰明确的展示。linux提供了iostat命令，可以了解io的开销。

示例：uptime

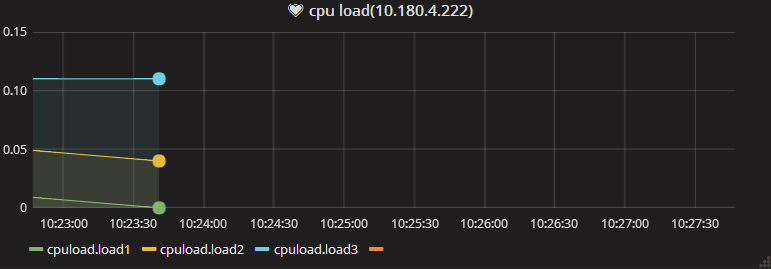
10:19:04 up 257 days, 18:56, 12 users, load average: 2.10, 2.10,2.09

　 　1、10:19:04 //系统当前时间

　　 2、up 257 days, 18:56 //主机已运行时间,时间越大，说明你的机器越稳定。

　　 3、12 user //用户连接数，是总连接数而不是用户数

4、load average // 系统平均负载，统计最近1，5，15分钟的系统平均负载



Load1对应1分钟负载，Load2对应2分钟负载。

1. CPU分配情况，top获取

用户进程占用CPU百分比，us

内核进程占用CPU百分比，sy

用户进程空间内改变过优先级的进程占用CPU百分比，ni

空闲CPU百分比，id

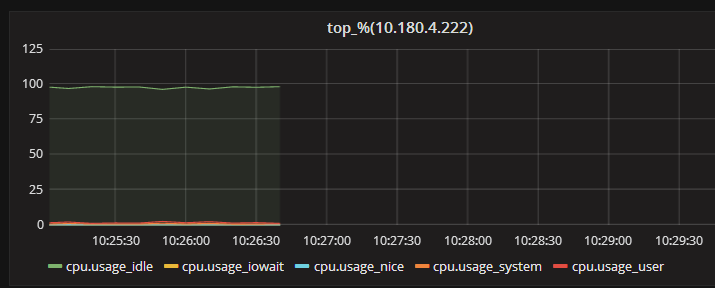
等待输入输出的CPU时间百分比，wa

硬件CPU中断占用百分比，hi

软中断占用百分比, si

虚拟机占用百分比，st

* wa表示IO等待所占用的CPU时间的百分比。wa占用超过30%则表示IO压力很大。
* nice值一个形象比喻，假设在一个CPU轮转中，有2个runnable的进程A和B，如果他们的nice值都为0，假设内核会给他们每人分配1k个cpu时间片。但是假设进程A的为0，但是B的值为-10，那么此时CPU可能分别给A和B分配1k和1.5k的时间片。故可以形象的理解为，nice的值影响了内核分配给进程的cpu时间片的多少，时间片越多的进程，其优先级越高，其优先级值（PRI）越低。%nice，就是改变过优先级的进程的占用CPU的百分比，如上例中就是0.5k/2.5k=1/5=20%。
* 若 %id 的值高但系统响应慢时，有可能是 CPU 等待分配内存，此时应加大内存容量
* 若 %id 的值持续低于1，则系统的 CPU 处理能力相对较低，表明系统中最需要解决的资源是 CPU



Idle，空闲，iowait对应wa指标,usage\_system对应us，usage\_user对应sy

### 内存

1. 物理内存，top指令获取

内存的大小

可用的大小

使用的大小

目录缓存大小

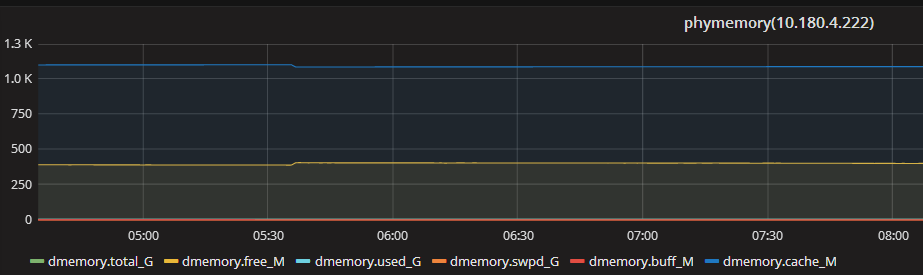
其它指标：vmstat -SM -n 3 获取

swap，交换到虚拟内存的大小

buffer cache的内存大小，用于提高磁盘存取效率，针对磁盘块的读写的缓存

page cache的内存大小，用于提高磁盘存取效率，针对文件inode的读写的缓存。

* 如果SWAP的值不为0，或者还比较大，比如超过100M了，但是SI，SO的值长期为0，这种情况我们可以不用担心，不会影响系统性能。
* 如果cache的值大的时候，说明cache处的文件数多，如果频繁访问到的文件都能被cache处，那么磁盘的读IO bi会非常小。



total\_G，总物理内存(G)，free\_M，空闲内存，used\_G，使用内存，swpd\_G，交换内存，buff\_M对应buffer cache，cache\_M对应page cache

1. 虚拟内存，top指令获取

内存的大小

可用的大小

使用的大小

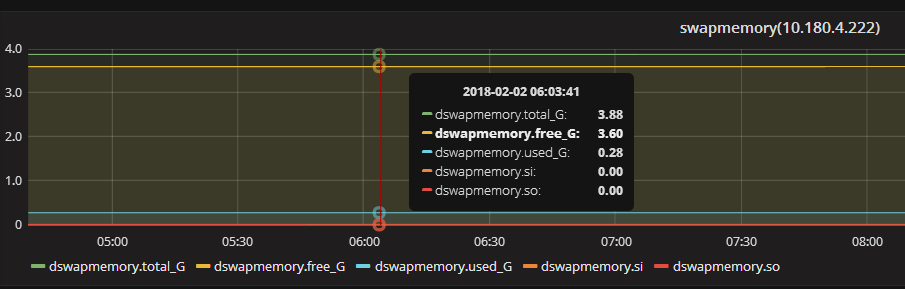
buffers/page cache大小

其它指标，vmstat -SM -n 3 获取

交换内存使用(si)，由磁盘调入内存

交换内存使用(so)，由内存调入磁盘

* 内存够用的时候，si/so这2个值都是0，如果这2个值长期大于0时，系统性能会受到影响，磁盘IO和CPU资源都会被消耗。
* 如果Swap的used很高，则表示系统内存不足。



si，交换内存使用，so，交换内存使用

1. 内存分页监控,sar -B

pgpgin/s：表示每秒从磁盘或SWAP置换到内存的字节数(KB)

pgpgout/s：表示每秒从内存置换到磁盘或SWAP的字节数(KB)

fault/s：每秒钟系统产生的缺页数,即主缺页与次缺页之和(major +minor)

majflt/s：每秒钟产生的主缺页数.

pgfree/s：每秒被放入空闲队列中的页个数

pgscank/s：每秒被kswapd扫描的页个数

pgscand/s：每秒直接被扫描的页个数

pgsteal/s：每秒钟从cache中被清除来满足内存需要的页个数

%vmeff：每秒清除的页(pgsteal)占总扫描页(pgscank+pgscand)的百分比

### I/O

1. 整体I/O情况,sar -b 3 2

tps: 每秒向磁盘设备请求数据的次数，包括读、写请求，为rtps与wtps的和。出于效率考虑，每一次IO下发后并不是立即处理请求，而是将请求合并(merge)，这里tps指请求合并后的请求计数。

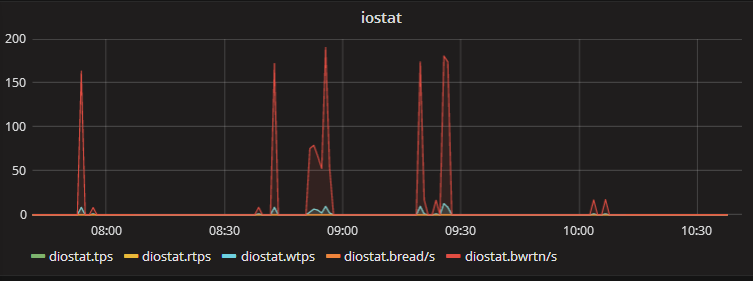
rtps: 每秒向磁盘设备的读请求次数

wtps: 每秒向磁盘设备的写请求次数

bread/s：每秒钟从物理设备读入的数据量，单位为 块/s

bwrtn/s：每秒钟向物理设备写入的数据量，单位为 块/s

rsec/s表示每秒读入，wsec/s表示每秒写入，iostat -x 获取，这两个参数某一个特别高的时候就表示磁盘IO有很大压力，



1. 各个磁盘情况，iostat

rrqm/s: 将读入请求合并后,每秒发送到设备的读入请求数.

wrqm/s: 将写入请求合并后,每秒发送到设备的写入请求数.

rkB/s：每秒读K字节数

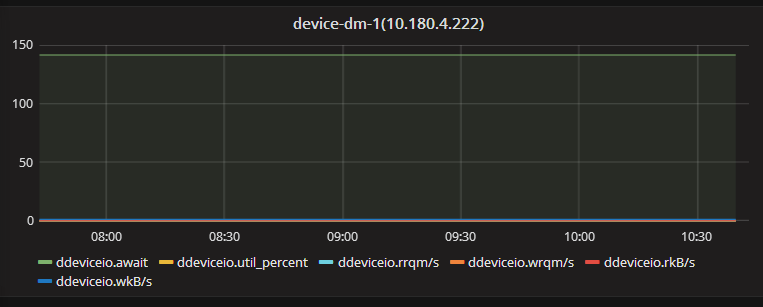
wkB/s：每秒写K字节数

await ：服务等待I/O请求的平均时间，包括请求队列等待时间 (单位毫秒)

svctm ：设备处理I/O请求的平均时间，不包括请求队列等待时间 (单位毫秒)

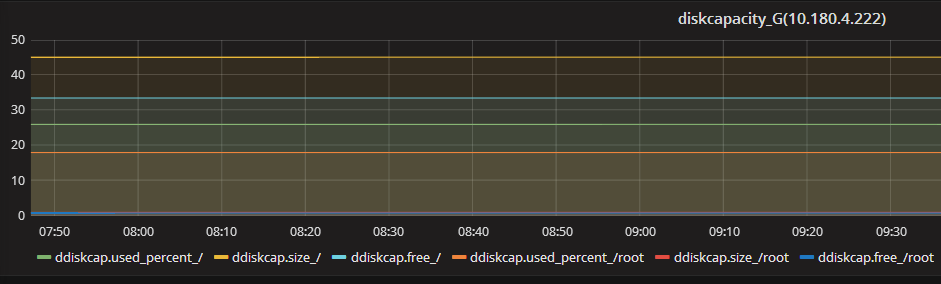
%util ：一秒中有百分之多少的时间用于 I/O 操作，即被io消耗的cpu百分比。

* 如果 %util 接近 100%，说明产生的I/O请求太多，I/O系统已经满负荷，该磁盘可能存在瓶颈。
* 如果 svctm 比较接近 await，说明 I/O 几乎没有等待时间；如果 await 远大于 svctm，说明I/O 队列太长，io响应太慢，则需要进行必要优化。
* 如果avgqu-sz比较大，也表示有当量io在等待。



await对应await，util\_percent对应%util，

1. 磁盘大小



Used\_percent\_/：path为 / 使用的百分比

Size\_/：path为 / 磁盘大小G

Free\_/： path 为 / 剩余大小G

### 进程

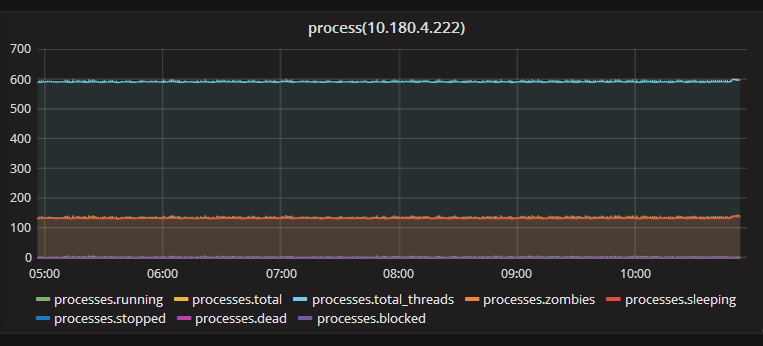
1. 进程情况，top获取

正在运行的进程数,running

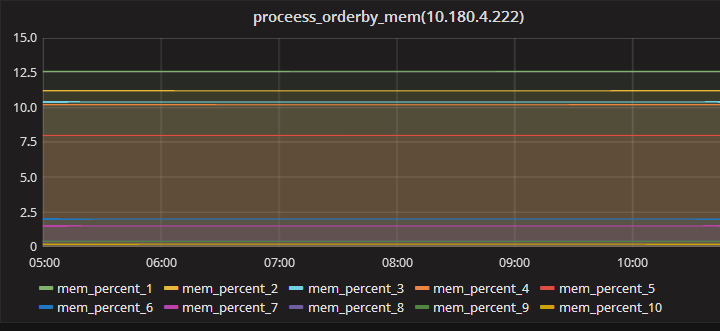
睡眠的进程数,sleeping,

停止的进程数,stopped,

僵尸进程数,表示有进程出现问题,zombie

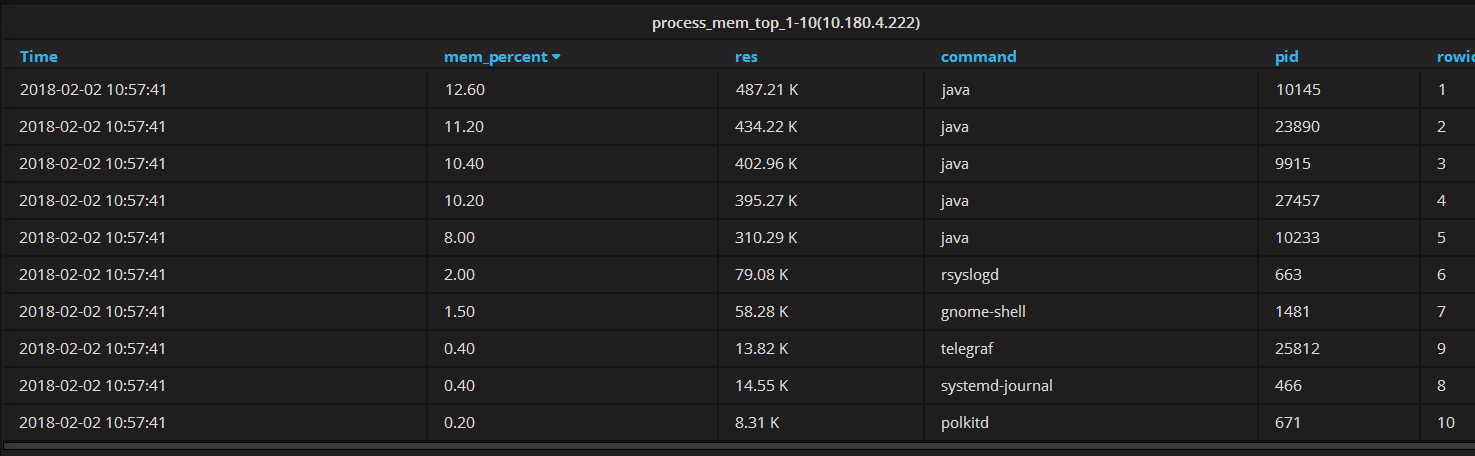


1. 进程按内存使用大小排序，前10位



Percent\_1使用第一位的占百分比

Percent\_2使用第二位的占百分比



Rowid排名

### 网络

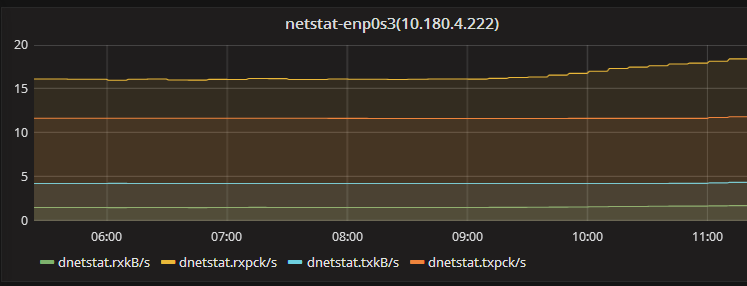
1. 网络接口统计信息,sar -n DEV

接收的数据包(rxpkg/s)

传输的数据包,txpck/s

接收的千字节rxkB/s

传输的千字节txkB/s



网卡enp0s的网络情况

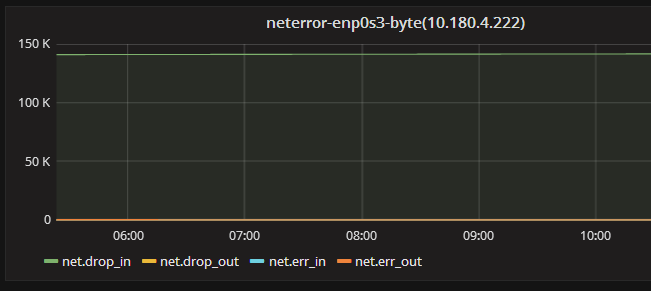
1. 错误情况,netstat -i 获取

Rx-ERR接收时，产生错误的数据包数。

RX-DRP接收时，丢弃的数据包数。

TX-ERR发送时，产生错误的数据包数。

TX-DRP发送时，丢弃的数据包数。

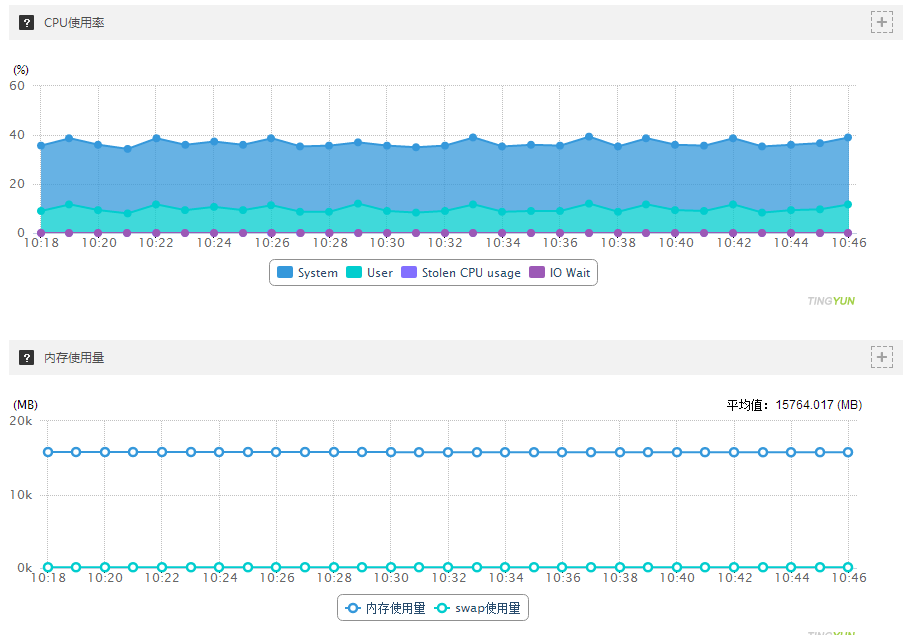


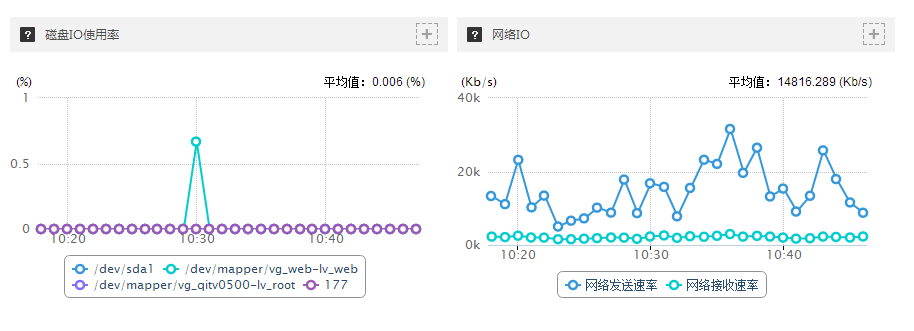
Dorp\_in对应rx\_drp

Drop\_out对应tx\_drp

## 监控内容

### 利用率：CPU、内存、磁盘、接口/带宽

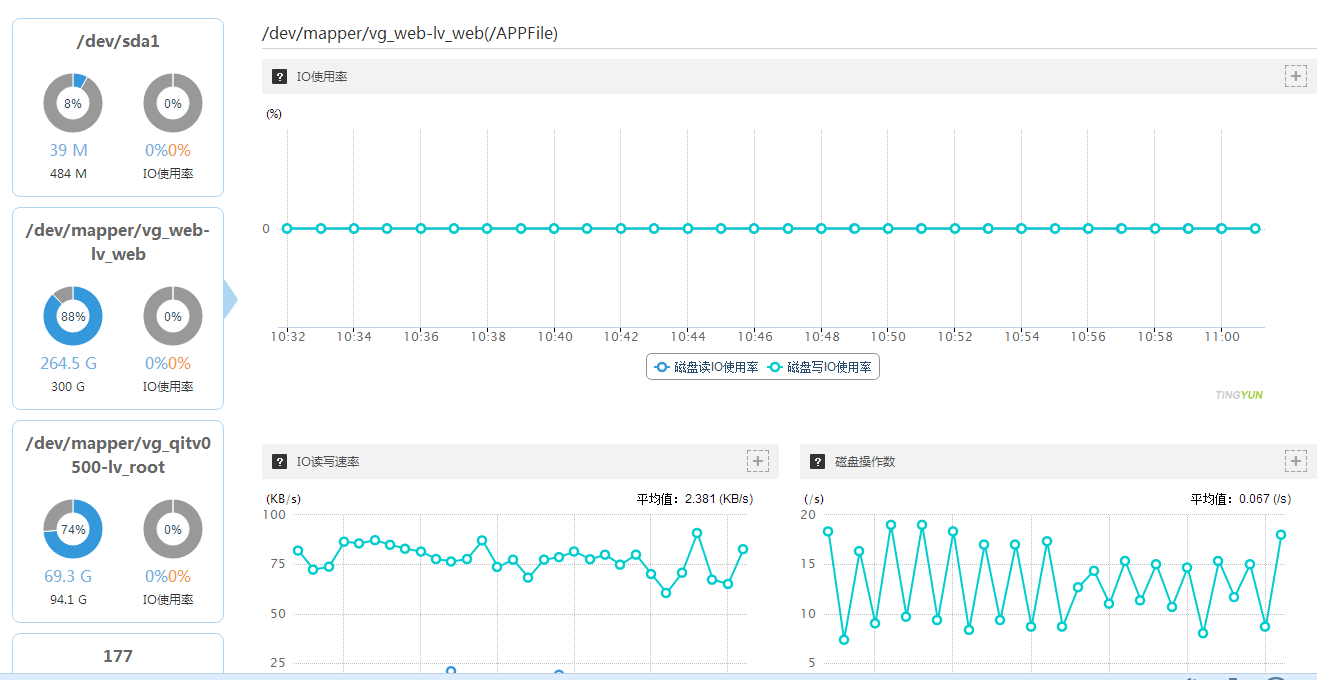




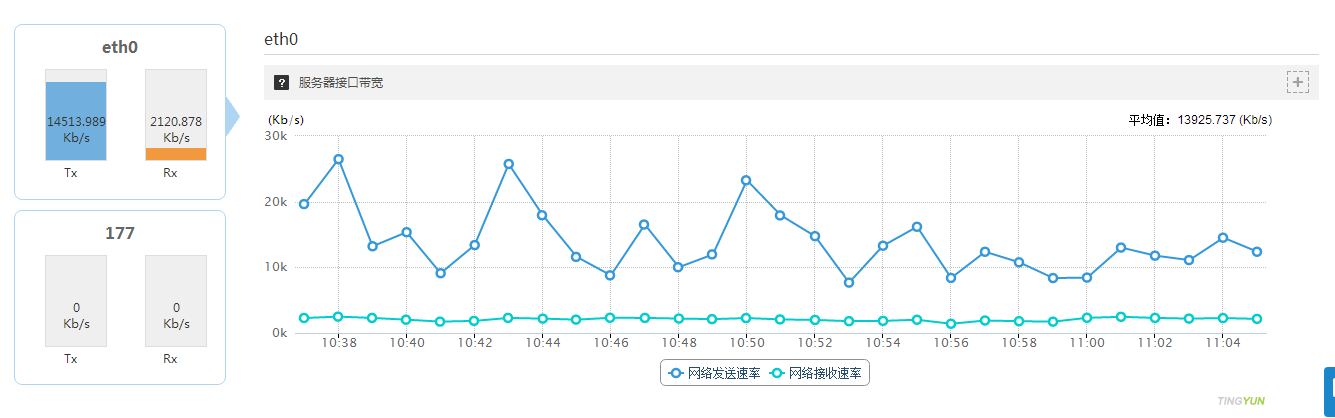
### 总量：内存、磁盘



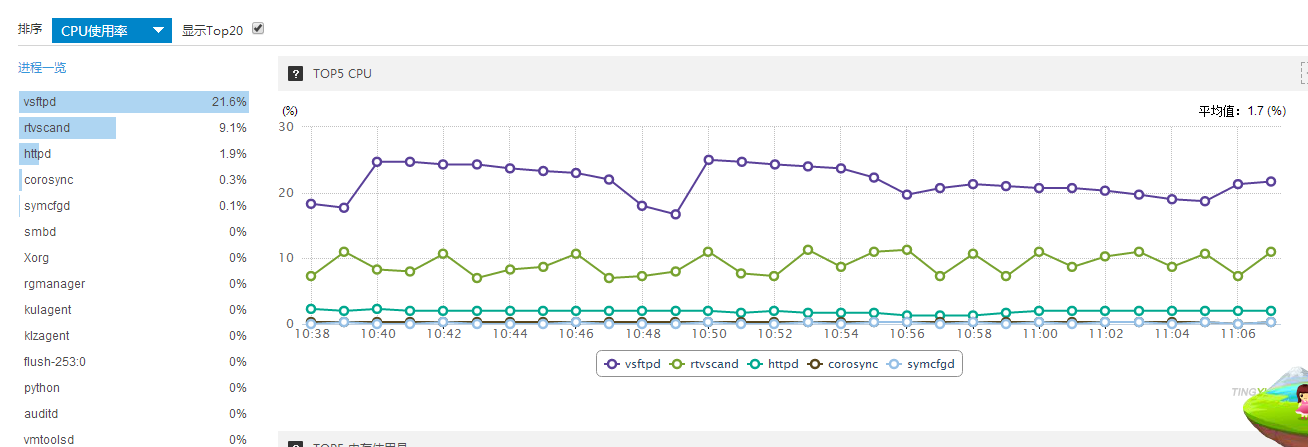
### 文件系统



### 网络



### 进程



# 全链路监控

## 背景案例

1. 大家可能了解，500 错误场景可能是大家在线上经常遇到的问题，等待很长时间，然后系统给你 500，让大家很抓狂。在部署全链路监控之前，大家可能需要使用比较古老的方式，跑到生产机上面去查问题，这个过程可能耗费大量的时间。
2. 有一个用户下单支付失败了，这个时候我们怎么样知道，为什么这个支付失败了，到底是支付整个链路上面哪里出了问题，这个是需要我们还原现场。

如果有一个系统它能够帮你还原这次调用，到底哪一步出了问题，所以你需要去反查，你需要拿着一些业务上面预先埋的点，比如说定单号或者用户ID，通过这些信息我能够反查到当时用户下单操作的那次调用，再去看那次调用到底哪里出了问题。

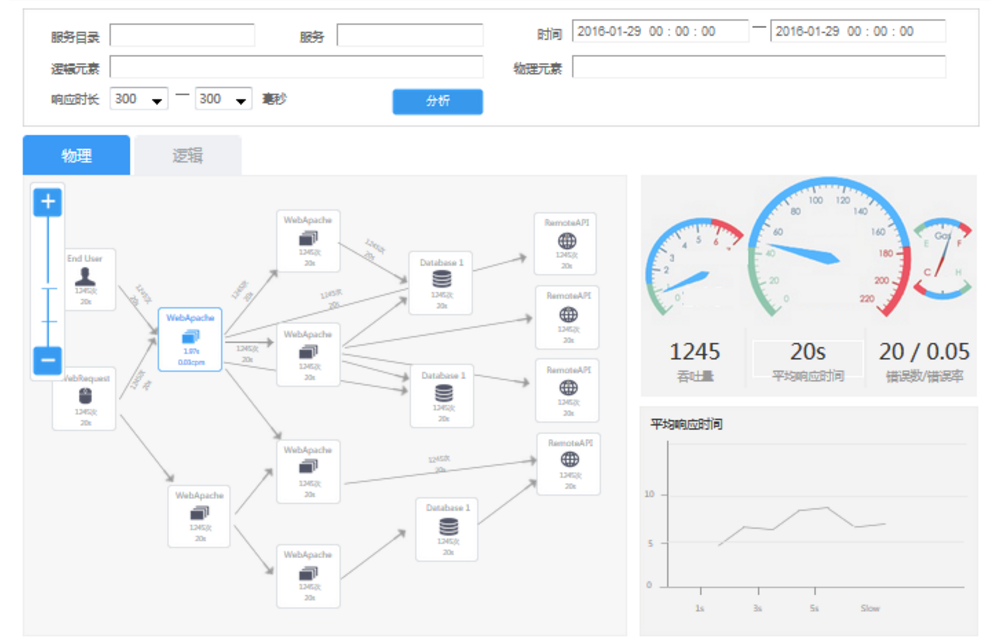
## 链路监控核心

1. 跟踪Http请求的整个链路，从Nginx、Tomcat和微服务之间的调用链路。
2. 跟踪各个调用的响应情况。
3. 跟踪各个调用的Bug情况。
4. 收集各个调用的基础数据(登录账号/调用时间)和业务数据(单据编号)
5. 收集SQL。

## 监控功能

1. PV/UV、定单量、支付成功/失败率等重要业务数据
2. 关注各个调用的各项性能指标，比如吞吐量（TPS）、响应时间及错误记录等
3. **根据调用耗时查询系统调用，快速定位应用性能瓶颈(修复pinpoint的bug即可)**。
4. **根据业务关键字查询系统调用，调用分布图只显示与关键字相关的调用链路**。
5. **链路包含Nginx，选择nginx节点后可查看请求时间、响应时间分布图。**
6. 链路包含App。
7. 告警，根据告警关键字定位问题。
8. 用户视角：查看用户某段视角内的uri分布
9. Uri视角：查看某个URI的调用统计
10. BUG视角：系统产生bug的链路

## 示例





# 报警

## 监控策略

监控服务器、接收对象



## 报警方式