# linux性能优化学习

## 环境

CentOS Linux release 7.5.1804 (Core)

CPU 2核

内存 4G

## 工具

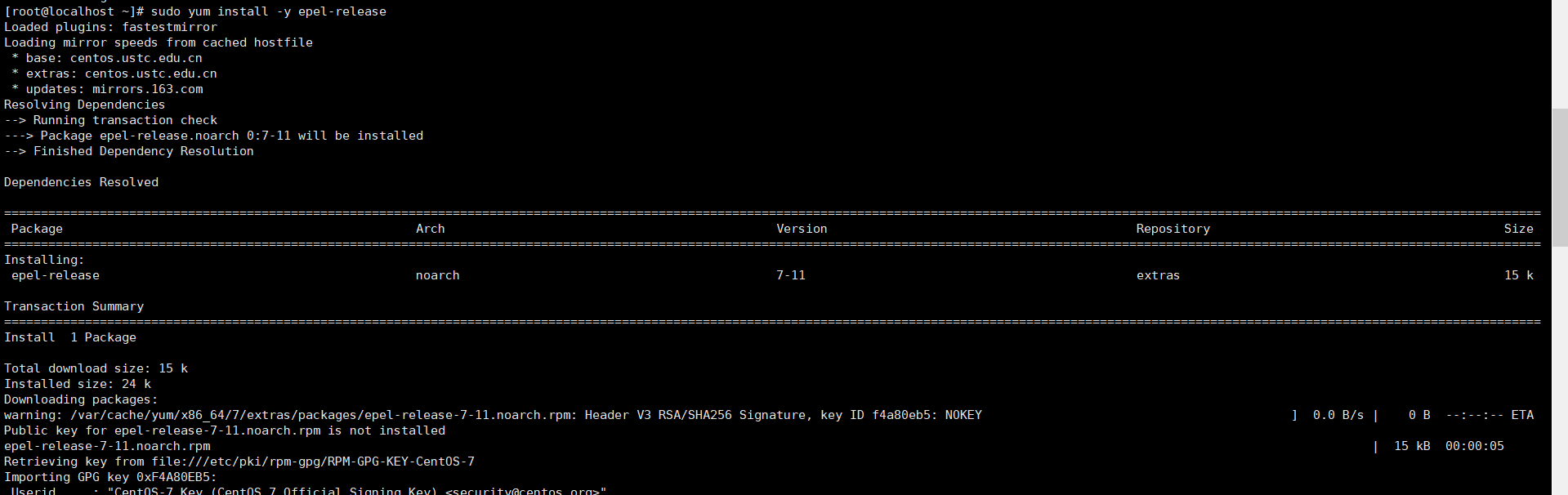
### stress安装

命令

sudo yum install -y epel-release

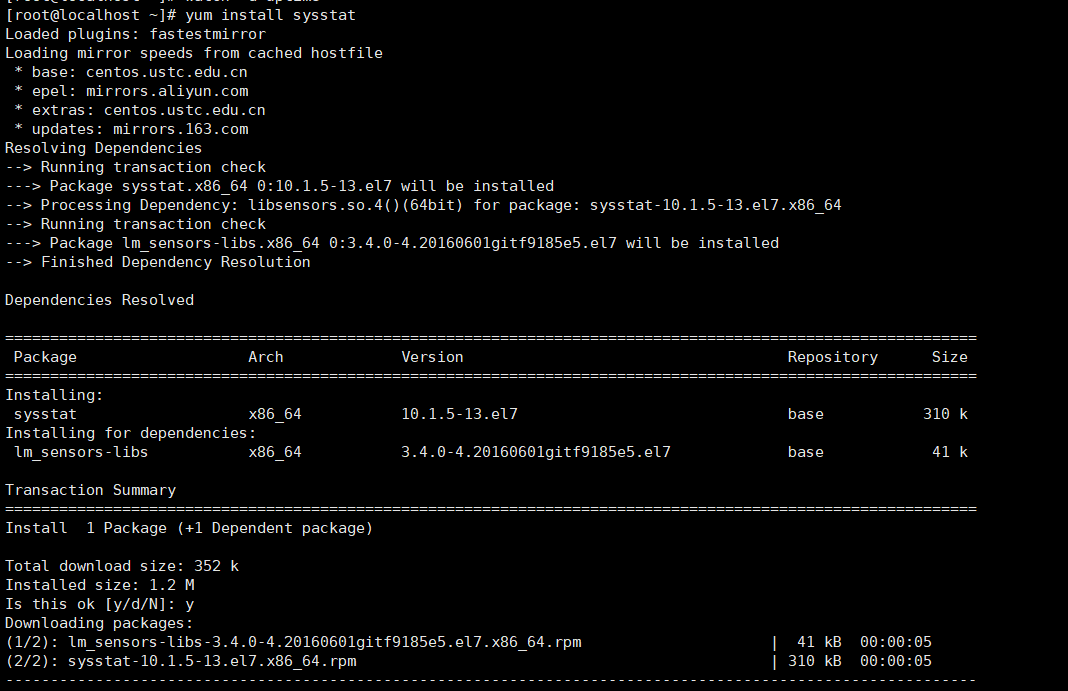
sudo yum install -y stress

sudo yum install -y stress-ng





### sysstat安装



## 命令

### ps命令

#### 作用:

用于显示当前进程 (process) 的状态。

参数:

-A 列出所有的行程

-w 显示加宽可以显示较多的资讯

-au 显示较详细的资讯

-aux 显示所有包含其他使用者的行程

au(x) 输出格式 :

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

USER: 行程拥有者

PID: pid

%CPU: 占用的 CPU 使用率

%MEM: 占用的记忆体使用率

VSZ: 占用的虚拟记忆体大小

RSS: 占用的记忆体大小

TTY: 终端的次要装置号码 (minor device number of tty)

STAT: 该行程的状态:

D: 不可中断的静止 (最常见的是在进行 I/O操作)

R: 正在执行中

S: 静止状态

T: 暂停执行

Z: 不存在但暂时无法消除

W: 没有足够的记忆体分页可分配

<: 高优先序的行程

N: 低优先序的行程

L: 有记忆体分页分配并锁在记忆体内 (实时系统或捱A I/O)

START: 行程开始时间

TIME: 执行的时间

COMMAND:所执行的指令

例如

[root@osmin1 ~]# ps -au

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1455 0.0 0.0 115436 2044 pts/0 Ss 10:58 0:00 -bash

root 1474 0.0 0.0 110088 852 tty1 Ss+ 10:58 0:00 /sbin/agetty --noclear tty1 linux

root 1529 0.0 0.0 155324 1848 pts/0 R+ 11:17 0:00 ps -au

### uptime命令

作用:

打印系统总共运行了多长时间和系统的平均负载

详解

[root@osmin1 ~]# uptime

10:58:44 up 1 min, 1 user, load average: 0.14, 0.10, 0.04

08:41:04 当前时间

up 1 min 系统运行时间

1 user 当前登录系统的用户数

load average: 0.14, 0.10, 0.04 过去1分钟 ，过去5分钟，过去15分钟的平均负载(load average)

何为平均负载？

系统处于可运行状态和不可中断状态的平均进程数，也就是平均活越进程数。

可运行运行的状态

就是正在使用CPU或者等待CPU的进程，即ps命名中处于R状态(running或者runable)的进程

不可中断状态

正处于内核态关键流程中的进程，并且这些流程是不可打断的比较常见的为等待系统硬件设备的I/O响应,也就是ps命令中处于D状态(Uninterruptible sleep)也称disk sleep的进程

### mpstat命令

mpstat是sysstat软件套件的一部分

是一个常用的多核 CPU 性能分析工具，用来实时查看每个 CPU 的性能指标，以及所有 CPU 的平均指标。

mpstat是MultiProcessor Statistics的缩写，是实时系统监控工具。报告CPU的一些统计信息，这些信息存放在/proc/stat文件中。在多CPUs系统里，其不但能查看所有CPU的平均状况信息，而且能够查看特定CPU的信息。

语法：

mpstat [-P {|ALL}] [internal [count]]

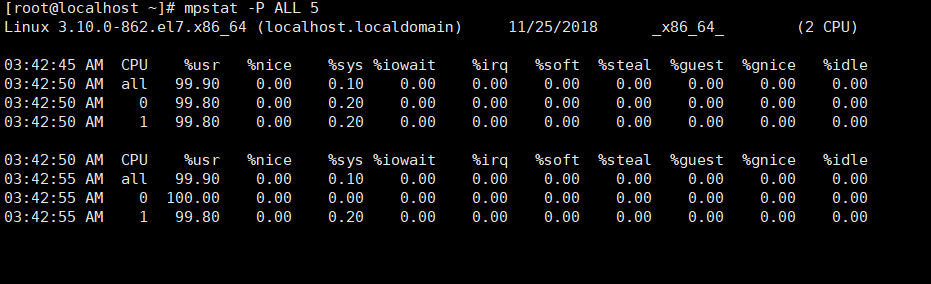
参数：

（1）-P {|ALL}：表示监控哪个CPU，在[0,cpu个数-1]中取值；

（2）internal：相邻的两次采样的间隔时间；

（3）count：采样的次数，count只能和delay一起使用；

备注：当没有参数时，mpstat则显示系统启动以后所有信息的平均值。有interval时，第一行的信息自系统启动以来的平均信息。从第二行开始，输出为前一个interval时间段的平均信息。



结果说明：

03:42:45 AM : 指mpstat运行的时间

（all）: 指所有CPU

1. user：在internal时间段里，用户态的CPU时间（%），不包含nice值为负进程，值为 (usr/total)\*100；

显示在用户级别（例如应用程序）执行时CPU利用率的百分比

1. nice：在internal时间段里，nice值为负进程的CPU时间（%），值为(nice/total)\*100；

显示在拥有nice优先级的用户级别执行时CPU利用率的百分比

1. system：在internal时间段里，核心时间（%），值为(system/total)\*100；显示在系统级别（例如内核）执行时CPU利用率的百分比

（4）iowait：在internal时间段里，硬盘IO等待时间（%），值为(iowait/total)\*100；显示在系统有未完成的磁盘I/O请求期间CPU空闲时间的百分比

（5）irq：在internal时间段里，硬中断时间（%），值为(irq/total)\*100；显示CPU服务硬件中断所花费时间的百分比

（6）soft：在internal时间段里，软中断时间（%），值为(softirq/total)\*100；显示CPU服务软件中断所花费时间的百分比,%soft : %steal : 显示虚拟机管理器在服务另一个虚拟处理器时虚拟CPU处在非自愿等待下花费时间的百分比

（7）%guest : 显示运行虚拟处理器时CPU花费时间的百分比

（8）%gnice :显示CPU或CPU运行niced guest虚拟机所花费的时间百分比。

（9）idle：在internal时间段里，CPU除去等待磁盘IO操作外的因为任何原因而空闲的时间闲置时间（%），值为(idle/total)\*100；显示CPU空闲和系统没有未完成的磁盘I/O请求情况下的时间百分比

### pidstat命令

pidstat 是sysstat软件套件的一部分

pidstat 是一个常用的进程性能分析工具，用来实时查看进程的 CPU、内存、I/O 以及上下文切换等性能指标。

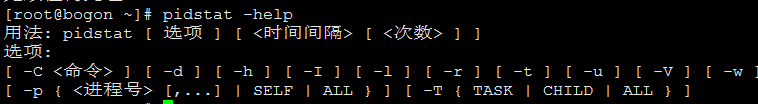
pidstat主要用于监控全部或指定进程占用系统资源的情况，如CPU，内存、设备IO、任务切换、线程等。pidstat首次运行时显示自系统启动开始的各项统计信息，之后运行pidstat将显示自上次运行该命令以后的统计信息。用户可以通过指定统计的次数和时间来获得所需的统计信息。

pidstat 示例

pidstat 的用法：

pidstat [ 选项 ] [ <时间间隔> ] [ <次数> ]

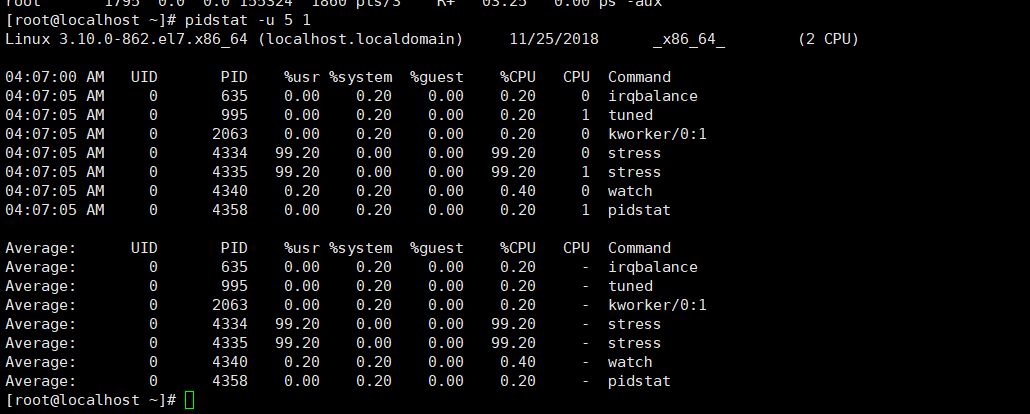
如下图：

常用的参数：

* -u：默认的参数，显示各个进程的cpu使用统计
* -r：显示各个进程的内存使用统计
* -d：显示各个进程的IO使用情况
* -p：指定进程号
* -w：显示每个进程的上下文切换情况
* -t：显示选择任务的线程的统计信息外的额外信息
* -T { TASK | CHILD | ALL }  
  这个选项指定了pidstat监控的。TASK表示报告独立的task，CHILD关键字表示报告进程下所有线程统计信息。ALL表示报告独立的task和task下面的所有线程。  
  注意：task和子线程的全局的统计信息和pidstat选项无关。这些统计信息不会对应到当前的统计间隔，这些统计信息只有在子线程kill或者完成的时候才会被收集。
* -V：版本号
* -h：在一行上显示了所有活动，这样其他程序可以容易解析。
* -I：在SMP环境，表示任务的CPU使用率/内核数量
* -l：显示命令名和所有参数

pidstat -u 5 1

显示各个进程的CPU统计，间隔5秒，仅统计一次



### stress命令

linux压力测试工具

选项

Stress 选项说明:

--version 显示软件版本信息

-v

--verbose 显示详细运行信息

-q

--quiet 不显示运行信息

-n

--dry-run 显示已经完成的指令执行情况

-t secs

--timeout secs 指定运行多少秒

--backoff usecs 等待usecs微秒后才开始执行

-c forks

--cpu forks

产生多个处理sqrt（）函数的CPU进程

-i forks

--io forks

产生多个处理sync（）函数的磁盘I/O进程

-m forks

--vm forks

产生多个处理malloc()内存分配函数的进程

--vm-bytes bytes

指定内存的Bytes数，默认值是1（与--vm-hang配合使用）

--vm-hang

指示每个消耗内存的进程在分配到内存后转入休眠状态，与正常的无限分配和释放内存的处理相反，这有利于模拟只有少量内存的机器；例如，下面命令运行时就会

分配到并一直持有256M内存，直到进程被终止；

% stress --vm 2 --vm-bytes 128M --vm-hang

-d forks

--hdd forks

产生多个执行write()函数的进程

--hdd-bytes bytes

指定写的Bytes数，默认是1GB

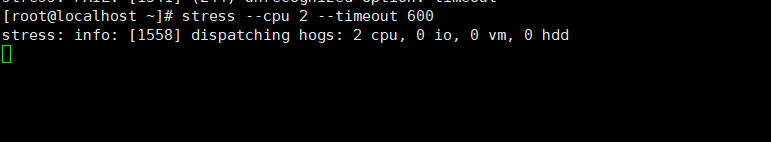
--hdd-noclean

不要将写入随机ASCII数据的文件Unlink

注意：时间单位可以为秒s，分m，小时h，天d，年y，文件大小单位可以为K，M，G

例如

模拟在10分钟内，CPU占用高的应用场景



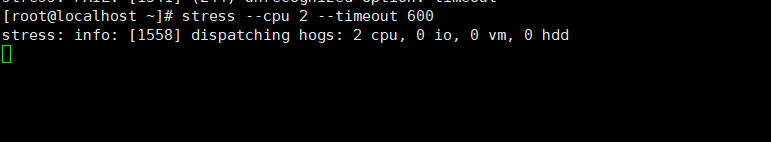
## 场景

### CPU密集型

#### 问题模拟方法

使用stress模拟,使用2个核，持续时间600秒

stress –cpu 2 –timeout 600

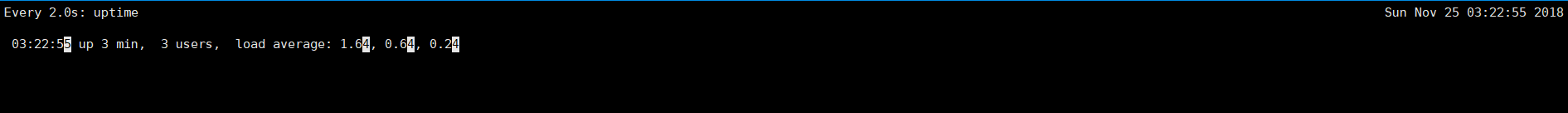


#### 问题定位方法

##### 使用uptime初步查看机器的资源消耗

uptime的变化可用使用

watch –d uptime



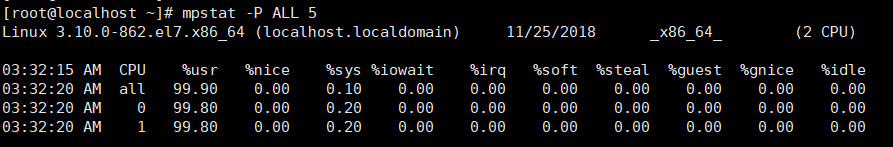
通过uptime可以看出系统的负载在呈现一个上升的趋势，而且机器的负载异常的高

从过去15分钟 的24%，到5分钟的64%，到最近 的164%，说明机器的负载已经很高了

检查当前机器的资源消耗情况

监控资源消耗

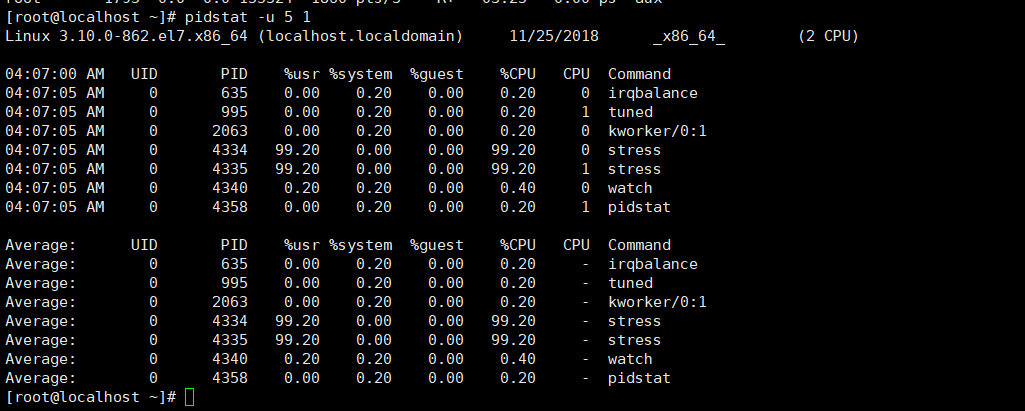
mpstat -P ALL 5 以5秒的为间隔，打印一次资源情况



从图中可以看到，机器的资源消息消耗是仅是CPU占用过高，其他的资源占用都较低

检查是哪个进程占用CPU

使用pidstat命令



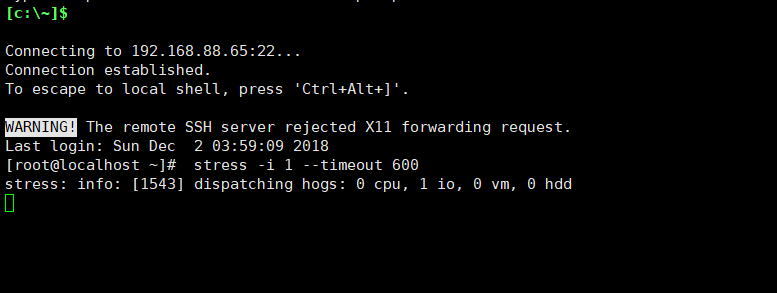
通过检查pid命令，发现占用CPU非常高的进程为stress程序，基本的问题已经定位

### IO密集型

#### 问题模拟方法(教程方法)

模拟方法

|  |
| --- |
| stress -i 1 --timeout 600 |



但在我本地虚拟机中测试发现不行，需要采用新的方法

iowait无法升高的问题，是因为案例中stress使用的是 sync() 系统调用，它的作用是刷新缓冲区内存到磁盘中。对于新安装的虚拟机，缓冲区可能比较小，无法产生大的IO压力，这样大部分就都是系统调用的消耗了。所以，你会看到只有系统CPU使用率升高。解决方法是使用stress的下一代stress-ng，它支持更丰富的选项，比如 stress-ng -i 1 --hdd 1 --timeout 600（--hdd表示读写临时文件）。

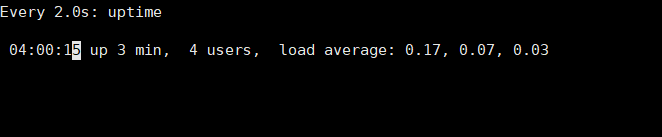
#### 新方法

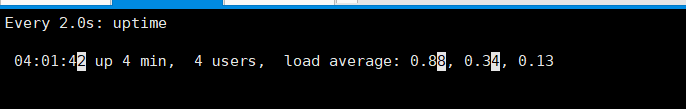
|  |
| --- |
| stress-ng -i 2 --hdd 1 --timeout 600 |

#### 定位方法

查看平均负载的变化情况

watch -d uptime

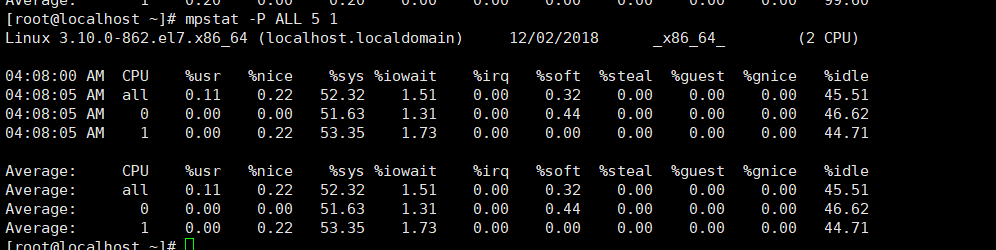




统计系统当前各资源占用情况

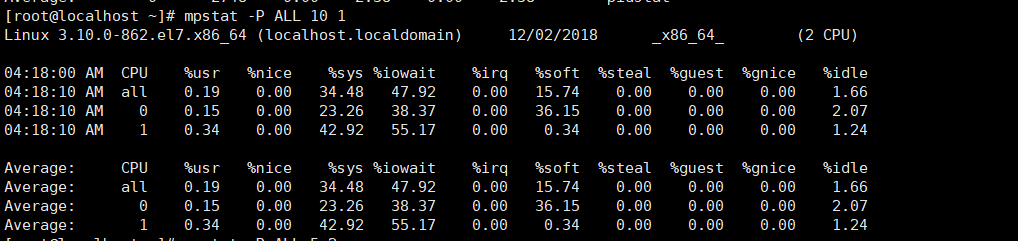
mpstat -P ALL 5 1

(原资源使用显示)



IOwait等待基本为0，但系统的CPU占用很高

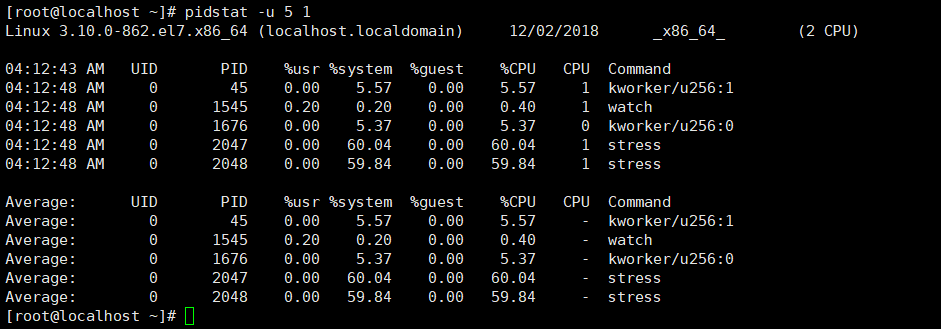
新方法后



可以很显示的查看到io占用很高以及系统调用占用很高

定位是哪个进程导致了系统占用很高以及IO占用过高

pidstat -u 5 1



通过pidstat 查看到占用资源最高的还是stress

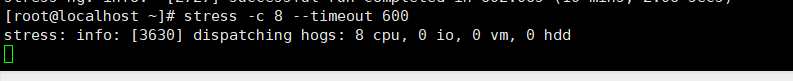
### 大量进程

#### 模拟方法

当系统中运行进程超出 CPU 运行能力时，就会出现等待 CPU 的进程

模拟方法:

使用 stress，但这次模拟的是 8 个进程

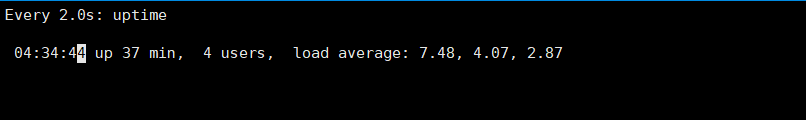


#### 定位方法

查看当前机器的负载情况

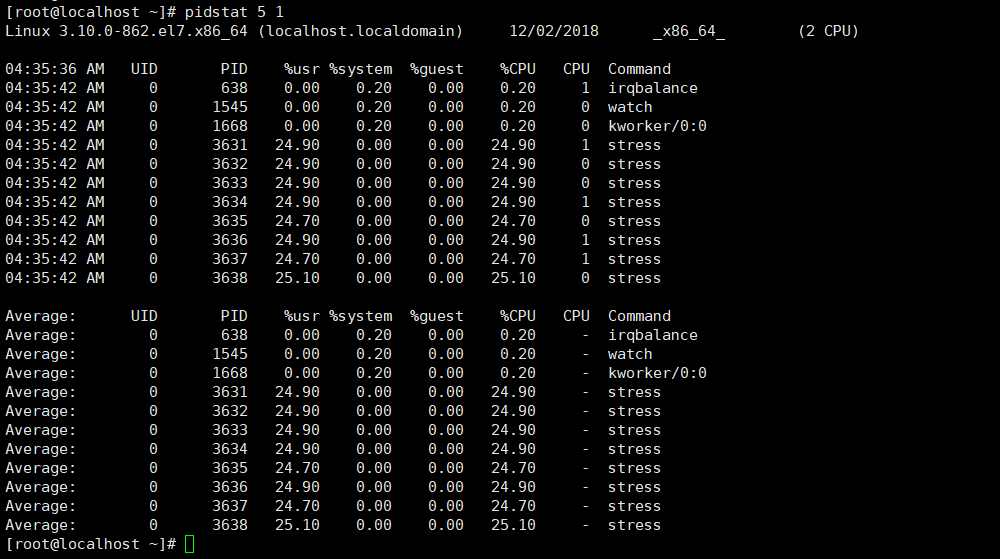
watch -d uptime

由于只有2个CPU，8个进程会使得负载非常的高



定位进程

pidstat 5 1



通过观察发现还是stress进程导致了系统的负载非常的高