均衡负载

使用top或者uptime来了解负载情况

uptime后三个数字表示过去1分钟、过去5分钟、过去15分钟的平均负载

平均负载指单位时间内，系统处于可运行状态和不可中断状态的平均进程数，也就是平均活跃进程数

可运行状态的进程：指正在使用cpu或者正在等待cpu的进程，常用ps命令看到处于R状态的进程

不可中断状态的进程则是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，比如常见的等待硬件设备的i/o响应，ps命令中D状态的进程

平均负载就是平均活跃进程数，就是单位时间内的活跃进程数，实际上是活跃进程数的指数衰减平均值

理想情况平均负载等于cpu个数，可以通过top或者从/proc/cpuinfo中读取:grep 'model name' /proc/cpuinfo | wc -l

如果过去1分钟、5分钟、15分钟的值基本相同，或者相差不大，说明负载很平稳

如果过去1分钟的值远小于15分钟的值，就说明系统最近1分钟的负载在减少，而过去15分钟内却又很大的负载

反过来，如果1分钟的值远大于15分钟的值，说明最近1分钟的负载在增加，增加可能是临时性的，可以会持续增加下去，需要持续观察，一旦1分钟的平均负载接近或超过cpu的个数，说明系统正在发生过载的问题

平均负载高于cpu数量70%的时候，就应该分析排查负载高的问题，负载过高，导致进程响应变慢，进而影响服务的正常功能

推荐方法：把系统的平均负载监控起来，根据历史数据，判断负载的变化趋势。

平均负载与cpu使用率的关系：

1、cpu密集型进程，使用大量cpu会导致平均负载升高，此时两者一致

2、I/O密集型进程，等待I/O 也会导致平均负载升高，但cpu使用率不一定很高

3、大量等待CPU的进程调度也会导致平均负载升高，此时的cpu使用也会很高

使用mpstat实时查看每个cpu的性能指标，以及所有cpu的平均指标

使用pidstat实时查看进程的cpu、内存、i/o以及上下文切换等性能指标

watch -d uptime查看平均负载的变化情况

mpstat -P ALL 5查了cpu的使用率的变化情况 -P ALL表示监控所有cpu，每个5s输出一组数据

pidstat -u 5 1 每隔5s输出一组数据

cpu上下文切换

cpu寄存器，是cpu内置的容量小、但速度极快的内存

程序计数器，用来存储cpu正在执行的指令位置、或者即将执行的下一条指令位置

cpu上下文切换：把前一个任务的cpu上下文（cpu寄存器和程序计数器）保存起来，然后加载新的任务的上下文到寄存器和计数器，最后在跳转到程序计数器所指的位置，运行新任务，保存下来的上下文，会存储在系统内核中，并在任务重新调度执行再次加载进来

cpu上下文切换场景：进程上下文切换、线程上下文切换以及中断上下文切换

进程上下文切换

进程空间分为内核空间和用户空间，进程既可以在用户空间运行又可以在内核空间中运行，进程在用户空间时，被称为进程的用户态，陷入内核空间被称为进程的内核态

从用户态转为内核态需要系统调用来完成

进程上下文切换是指从一个进程切换到另一个进程

系统调用过程中一直是同一个进程在运行，被称为特权模式切换

进程是由内核来管理和调度，进程切换只能发生在内核态，进程上下文切换不仅包括虚拟内存、栈、全局变量扽给用户的资源，还包括内核堆栈、寄存器等内核空间的状态

进程上下文切换比系统调用多了一步：在保存当前进程的内核状态和cpu寄存器之前，需要先把改进程的虚拟内存、栈等保存下来；而加载了下一进程的内核态，还需刷新进程的虚拟内存和用户栈

linux通过tlb来管理虚拟内存到物理内存的映射关系，当虚拟内存更新后，TLB也需要更新，内存的访问会随之变慢

多个处理器系统上，缓存是被多个处理器共享的，刷新缓存不仅会影响当前处理器的进程，还会影响共享缓存的其它处理器的进程

活跃进程：正在运行和等待运行的进程

触发进程调度的场景：

1、为了保证所有进程可以得到公平的调度，cpu时间被划分一段段时间片，轮流分配给各个进程，当时间片耗尽，就会被系统挂起，切换到其它正在等待cpu的进程运行

2、进程在系统资源不足（比如内存不足）时，要等到资源满足后才可以运行，这时进程也会被挂起，并由系统调度其它进程运行

3、当进程通过睡眠函数sleep这样的方法将自己主动挂起时，自然也会重新调度

4、当由优先级更高的进程运行，为了保存高优先级进程的运行，当前进程会被挂起，由高优先级进程来运行

5、发生硬件中断时cpu上的进程会被中断挂起，转而执行内核中的中断服务程序

线程与进程最大区别：线程是调度的基本单位，进程则是资源拥有的基本单位，内核中的任务调度，实际上的调度对象是线程；进程只是给线程提供了虚拟内存、全局变量等资源

当进程只有一个线程时，进程等于线程

当进程拥有多个线程时，线程会共享相同的虚拟内存和全局变量等资源，这些资源在上下文切换是不需要修改的

线程也有自己的私有数据，比如栈和寄存器，这些在上下文切换时，也是需要保存的

线程的上下文切换分为两种：

1、前后两个线程属于不同的进程，因为资源不共享，所有切换过程跟进程上下文一致

2、前后两个线程属于同一个进程，虚拟内存是共享的，在切换时，虚拟内存这些资源就保持不动，只需要切换线程的私有数据、寄存器等不共享的数据

中断上下文切换：

同一个CPU，中断处理比进程拥有更高的优先级，所以中断上下文切换不会与进程上下文切换同时发生，由于中断会打断正常进程的调度和执行，大部分中断处理程序都短小精悍，以便快速执行结束

中断上下文切换也需要消耗cpu，切换次数过多也会耗费大量的cpu，甚至严重降低系统的整体性能，当中断次数过多，需要注意去排查是否会给系统带来严重的性能问题

使用vmstat查看内存使用情况，也常用来分析cpu上下文切换和中断的次数（系统总体上下文切换情况）

vmstat 5每间隔5s输出一组数据

cs是每秒上下文切换的次数

in则是每秒中断的次数

r是就绪队列的长度，也就正在运行和等待cpu的进程数

b处于不可中断睡眠状态的进程数

pidstat -w 5每间隔5s输出1组数据（查看每个进程的详情情况）

cswch表示每秒自愿上下文切换的次数

nvcswch表示每秒非自愿上下文切换的次数

自愿上下文切换，是指进程在无法获取所需资源。导致的上下文切换，比如i/o、内存等系统资源不足，就会发生自愿上下切换

非自愿上文切换，则是指进程由于时间片已等原因，被系统强制调度，进而发生的上下文切换。比如说，大量进程都在争抢cpu时，就容易发生非自愿上下切换

us和sy列为cpu的使用率，当sy占比重多时，说明cpu被内核占用，当us占比重多时，说明cpu被用户核占用

系统的就绪队列过长（正在运行和等待cpu的进程数过多）导致大量的上下文切换，而上下文切换又导致系统的cpu的占用率升高

pidstat -w -u 1（w表示进程切换指标，-u表示输出cpu使用指标）

pidstat -wt 1（-wt表示输出线程的上下文切换指标）每隔1s输出一组数据

查看中断发生的类型：/proc/interrupts文件中读取，proc是Linux的一个虚拟文件系统，用户内核空间与用户空间之间的通信，proc/interrupts就是这种通信机制的一部分，提供一个只读的中断使用情况

wacth -d cat /proc/interrupts

RES重调度中断，表示唤醒空闲状态cpu来调度新的任务运行，这是多处理系统(smp)中，调度器用来分散任务到不同cpu的机制，通常被称为处理器中断（lpl）

上下文切换多少正常，这个数值取决于系统本身的cpu性能。如果系统的上下文切换次数比较稳定，那么从数百到一万以内，都应该算是正常的。但是上下文切换次数超过一万次，或者切换次数出现数量级的增长，有可能出现性能问题

上下文切换的类型：

自愿上下文切换变多了，说明进程都在等待资源，有可能发生i/o等其他问题

非自愿上下文切换变多了，说明进程都在被强制调度，都在争抢cpu，说明cpu的确成了瓶颈

中断次数变多，说明cpu被中断处理程序占用，还需要通过查看/proc/interrupts文件来分析具体的中断类型

应用cpu使用率过高

为了维护cpu时间，linux通过事先定义的接拍率（内核中表示hz），触发时间中断，并使用全局变量jiffie记录开机以来的节拍数。每发生一次中断,jiffies的值就加1

可以通过查询/boot/config内核选项来查看节拍率的配置值

grep 'CONFIG\_HZ=' /boot/config-$(uname -r)

内核提供了一个用户空间节拍率USER\_HZ,固定为100也就是，1/100s（总数固定的）

linux通过/proc虚拟文件系统，向用户空间提供了系统内部状态的信息，/proc/stat 提供的就是系统的cpu和任务统计信息

只保留cpu的数据

cat /proc/stat |grep ^cpu

查询出来的数据，表示cpu在不同场景的时间

user(us)：代表用户态cpu时间，不包括nice时间，包括guest时间

nice（ni）：代表低优先级用户态cpu时间，也就是进程的nice值被调整为1-19之间时cpu时间，这里注意，nice可取值范围-20到19，数值越大，优先级反而越低

system(sys)：代表内核态cpu时间

idle（id）:代表空闲时间，不包括等待i/o的时间（iowait）

iowait(wa),代表等待i/o的cpu时间

irq(hi)：代表处理硬中断的cpu时间

softirq(si)：代表处理软中断的cpu时间

steal（st）:代表当前系统运行在虚拟机中时候，被其它虚拟机占用的cpu时间

guest(guest)：代表通过虚拟机运行其它操作系统时间，也就是运行虚拟机的cpu时间

guest\_nice(gnice)：代表以低优先级运行虚拟机的时间

cpu使用率，就是除了空闲时间外的其他时间占总cpu时间的百分比

cpu使用率=1-空闲时间/总cpu时间

为了计算cpu使用率，一般间隔3s取两次值，做差后，在计算这段时间内的平均cpu使用率

平均cpu使用率=1-（【新】空闲时间-【旧】空闲时间）/（【新】总cpu时间-【旧】总cpu时间）

linux中/proc/[pid]/stat记录每个进程运行情况的统计信息

性能分析工具给出的都是间隔一段时间的平均cpu使用率，所以注意间隔时间的设置，特别是多个工具对比分析，保证他们用的是相同的间隔时间

top（每隔3s刷新一次） 显示了系统总体的cpu和内存使用情况，以及各个进程的资源使用情况

ps 只显示每个进程的资源使用情况

%CPU表示进程的cpu使用率，是用户态和内核态cpu的使用率的总和，包括进程用户空间使用的cpu、通过系统调用执行的内核空间cpu、就绪队列等待运行的cpu。在虚拟环境中还包括虚拟机占用的cpu

pidstat 1 5每个1s输出5组数据的平均值

用户态cpu使用率（%usr）

内核态cpu使用率（%system）

运行虚拟机cpu使用率（%guest）

等待cpu使用率（%wait）

以及总的cpu使用率(%CPU)

怎么查看占用cpu是代码里的哪个函数：

GDB只适合性能分析的后期，当你找到出问题的大致函数，线下在通过它来进一步调试函数内部的问题

perf是linux内置性能分析工具，以性能事件采样为基础，不仅可以分析系统的各种事件和内核性能，还可以用来分析指定应用程序的性能问题

1、perf top实时显示占用cpu时钟最多的函数或者指令

第一行包含三个数据，分别是Samples(采样数)、event(事件类型)、Event count（事件总数量）

如果采样数过少，下面的排序和百分比没有什么实际参考价值

第一列 Overshead，是该符合的性能事件在所有采样中的比例，用百分比来表示

第二列Shared,是该函数或指令所在的动态共享对象，如内核、进程名、动态链接库名、内核模块

第三列object，是动态共享对象的类型，比如[.]表示用户空间的可执行程序、或者动态链接库，而[k]则表示内核空间

第四列Symbol是符号名，就是函数名，当函数名未知时，用十六进制的地址来表示

perf record保存系统性能信息的数据

perf report用来解析perf record保存的数据

perf top -g -p 21515 （-g开启调用关系分析，-p指定进程号）找到相应的进程回车查看里面的调用关系

用户cpu比nice cpu高，说明用户态进程占用较多的cpu，所以应该着重排查进程的性能问题

系统cpu高，说明内核态占用较多的cpu，所以应该着重排查内核线程或者系统调用的性能问题

i/o等待cpu高，说明等待i/o的时间比较长，所以应该着重排查系统存储是不是出现了i/o问题

软中断和硬中断高，说明软中断或硬中断的处理程序占用了较多的cpu，所以应该着重排查内核中的中断服务程序

碰到cpu使用率升高的问题，可以借助top\pidstat等工具，确认引发cpu性能问题的来源；在使用perf等工具排查引起性能问题的具体函数

系统的cpu使用率很高，但找不到想要的应用：

系统的cpu使用率，不仅包括进程用户态和内核态的运行，还包括中断处理、等待i/o以及内核线程，当系统的cpu使用率很高，不一定能找到相对应的高cpu使用率的进程

找不到占用高cpu应用时，查看top的tasks栏，有多少运行的进程，在看下面的进程哪些是正在运行的，具体分析运行的进程

使用pidstat查看相应的进程信息，如果没有则使用ps进行交叉确认

ps aux |grep 进程号

查看相应的进程信息，如果没有在使用top去查看确认

同一个进程出现不同进程号，要么是不断的在重启，要么是新的进程

原因1、进程不断的崩溃重启，比如因为段错误、配置错误等等，进程在退出后可能又被监控系统自动重启

原因2、这些进程都是短时进程，也就是在其它应用内部通过exec调用的外面的命令，这些命令一般只是运行很短的时间就会结束

使用pstree查看树桩形式显示所有进程之间的关系

pstree | grep stress(被查看的进程)

查看相关的父进程，将代码拷贝到本地

使用grep stress -r app查看是不是有代码在调用stress命令

execsnoop专为短时进程设计的工具。通过ftrace实时监控进程的exec()行为，并输出短时进程的基本信息，包括进程PID、父进程PID、命令行参数以及执行的结果

碰到无法解释cpu使用率情况时，首先要想到有可能是短时应用导致的问题，比如有以下两种情况

1、应用里直接调用其他二进制程序，这些程序通常运行时间比较短，通过top等工具也不容易发现

2、应用本身在不停的崩溃重启，而启动过程的资源初始化，很可能会占用相当多的cpu

我们可以用pstree或者exescnoop找到他们的父进程，在从父进程所在的应用入手，排查问题的根源

系统出现大量不可中断进程和僵尸进程：

当iowait升高，进程很可能因为得不到硬件的响应，长时间处于不可中断状态，从ps或者top命令的输出中，状态处于D时，则为不可中断状态

进程状态：

R表示进程在cpu的就绪队列中，正在运行或者正在等待运行

D表示不可中断状态睡眠，一般表示进程正在跟硬件交互，并且交互过程不允许被其他进程中断或打断

Z表示僵尸进程，就是进程实际上已经结束，但是父进程并没有回收它的资源（比如进程的描述符、PID）

S表示可中断状态睡眠，进程因为等待某个事件而被系统挂起，当进程等待的时间发生时，它就会被唤醒进入R状态

I表示空闲状态，用在不可中断睡眠的内核线程上，硬件交互导致不可中断进程用D表示，但对某些内核线程来说，在实际上并没有任何负载，用idel正是为了区分这种情况，D状态的进程会导致平均负载升高，I状态的进程并不会

当系统或硬件发生故障，进程可能会在不可中断状态保持很久，甚至导致系统中出现大量不可中断进程，这时需要注意系统是不是出现了i/o等性能问题

正常情况，当一个进程创建子进程，他应该通过系统调用wait(）或者waitpid()等待子进程结束，回收子进程的资源，而子进程在结束时，会向父进程发送SIGHLD信号，父进程还可以注册SIGHLD信号来处理函数，异步回收资源

如果父进程没有回收，或者子进程进行太快，父进程没有来得及处理子进程，子进程已经退出，子进程就会变成僵尸进程

僵尸进程持续时间都毕竟短，在父进程回收它的资源后就会消亡，或者在父进程退出后，由init进程回收也会消亡，一旦父进程没有处理子进程的终止，还一直保持运行状态，那么子进程就会一直处于僵尸状态，大量的僵尸进程会用尽PID进程号，导致新的进程不能创建

dstat性能测试工具，可以同时观察系统的cpu、磁盘i/o、网络以及内存使用情况、

s+中s表示这个进程是一个会话的领导进程，而+表示前台进程组

进程组便是一组相互关联的进程，比如每个子进程都是父进程所在的成员

会话是指共享同一个控制终端的一个或多个进程

比如通过ssh登录服务器，就会打开一个控制终端（TTY），这个控制终端就对应一个会话，在终端运行的命令以及他们的子进程，就构成一个个的进程组，后台运行的命令，构成后台进程组，前台运行的命令，构成前台进程组

dstat 1 10间隔1s输出10组数据

pidstat -d 输出i/o使用情况 例如：pidstat -d p 4344 1 3（-d展示i/o统计数据，-p指定进程号，间隔1s输出3组数据）

kb\_rd表示每秒读的kb数，kb\_wr表示每秒写的kb数，iodelay表示i/o的延迟

进程想要访问磁盘就必须使用系统调用

strace跟踪进程系统调用的工具

strace -p 6082(进程号)

当以root用户运行时，提示没有权限，则用ps aux |grep 6082(进程号)查看进程状态

当进程为Z状态也就是僵尸进程，无法继续分析时，应该使用事件记录动态工具

使用perf record -g 收集信息(时间为15S然后退出)，然后使用perf report解析查看

pstree -aps 3084(-a表示输出命令选项，p便是pid,s指定进程的父进程)

iowait高不一定代表i/o有性能瓶颈，当系统中只有i/o类型的进程在运行时iowait也会很高，实际上，磁盘的读写远没有达到性能瓶颈的程度

当iowait升高需要先使用工具确认是不是磁盘i/o的问题

linux软中断：

中断其实是一种异步的事件处理机制，可以提高系统的并发处理能力，为了减少对正常进程运行调度的影响，中断处理过程就需要尽可能快的运行

中断处理程序在响应中断，会临时关闭中断，会导致在上一次中断处理完成之前，其他中断都不能响应，也就是中断有可能会丢失

为了解决中断处理程序的执行过长和中断丢失的问题，linux将中断处理过程分成两个阶段，也就是上半部和下半部

上半部用来快速处理中断，他在中断静止模式下运行，主要处理跟硬件紧密相关的或时间敏感的工作

下半部用来延迟处理上半部未完成的工作，通常以内核线程的方式运行

/proc/softirqs提供软中断的运行情况

/proc/interrupts提供硬中断的运行情况

使用cat /proc/softirqs查看各种类型软中断在不同的cpu上累积运行次数

1、要注意软中断的类型，也就是第一列的内容，软中断包括10个类别，分别对应不同的工作类型，比如NET\_RX表示网络接收中断，而NET\_TX表示网络发送中断

2、要注意同一种软中断在不同cpu上的分布情况，也就是同一行的内容，正常情况下，同一种中断在不同的cpu上累积次数应该差不多

TASKLET是最常用的软中断实现机制，每个TASKLET只运行一次就会结束，并且只在调用它的函数所在的cpu上运行

软中断实际上是以内核线程的方式运行，每个cpu都应一个软中断内核线程，这个软中断内核线程就叫做ksoftirqd/cpu编号，使用ps aux | grep softirq来查看，一般名字在中括号里面的就是内核线程

Linux的中断处理程序分上半部和下半部

上半部对应硬件中断，用来快速处理中断

下半部对应软中断，用来异步处理上半部未完成的工作

系统软中断cpu使用升高：

linux中的软中断包括网络收发，定时，调度，RCU锁等各种类型，可以在/proc/softirqs观察软中断的运行情况

当软中断事件的频率过高，内核线程会因为cpu使用率过高而导致软中断处理不及时，进而引发网络收发延迟，调度缓慢等性能问题

sar是一个系统报告工具，既可以实时查看系统的当前活动，又可以配置保存和报告历史统计数据

hping3是一个可以构造TCP/IP协议数据包的工具，可以对系统进行安全审计，防火墙测试

tcpdump是一个常用的网络抓包工具，常用来分析各种网络问题

hping3 -S -p 80 -i u100 192.168.0.30（-S参数便是设置TCP协议的SYN（同步序列号），-p表示端口，-i u100表示每隔100微秒发送一个网络帧）

查看中断次数的变化速率，使用watch -d cat /proc/softirqs

TIMER(定时中断)、NET\_RX（网络接收）、SCHED(内核调度)、RCU（RCU锁）

sar可以用来查看系统网络收发情况，不仅可以观察网络收发的吞吐量（BPS,每秒收发的字节数）还可以观察网络收发的PPS，即每秒收发的网络帧数

sar -n DEV显示网络收发的报告

sar -n DEV 1 （-n DEV表示显示网络收发的报告，间隔1s输出一组数据）

第一列表示报告时间

第二列表示IFACE为网卡

第三、四列：rxpck/s和txpck/s分别表示每秒接收、发送的网络帧数，也就是PPS

第5列、六列：rxkb/s和txkb/s分别表示每秒接收、发送的千字节数，也就是BPS

平均网络字节=每秒接收的千字节\*1024/每秒接收的网络帧数

tcpdump -i eth0 -n tcp port 80

-i ech0之抓取eth0网卡，-n不解析协议名和主机名

tcp port 80表示只抓取tcp协议并且端口号为80的网络帧

flags [S]表示这是一个SYN包

从系统的软中断使用率高这个现象，通过观察/proc/softirqs文件的变化情况，判断软中断类型是网络接收中断，在通过sqr和tcpdump确认这个一个SYNFLOOD问题

SYNFOOD问题最简单的解决方式，就是从交互机或者硬件防火墙中封掉ip

软中断cpu使用率升高是一种常见的性能问题，虽然软中断的类型很多，但实际生产中，我们遇到的性能平均大多数网络收发类型的软中断，特别是网络接收的软中断

迅速分析系统cpu的瓶颈：

cpu使用率描述了非空闲时间占总cpu时间的百分比，根基cpu上运行任务的不同，被分为用户cpu，系统cpu、等待i/oCPU、软中断和硬中断

用户cpu使用率，包括用户态cpu使用率（user）和低优先级用户态cpu使用率（nice），表示cpu在用户态运行的时间百分比，用户cpu使用率高，说明应用程序比较繁忙

系统cpu使用率，表示cpu在内核态运行的时间百分比（不包括中断）系统cpu使用率高，说明内核比较繁忙

等待i/o的cpu使用率，通常称为iowait，便是等待i/o的时间百分比，iowait高，说明系统与硬件设备的i/o交互时间比较长

软中断和硬中断的cpu使用率，分别表示内核调用软中断处理程序、硬中断处理程序的时间百分比，他们的使用率高，说明系统发生大量的中断

虚拟环境中用到的窃取cpu使用(steal)和客户CPU使用率（guest），分别表示被其它虚拟机占用cpu时间百分比，和运行客户虚拟机的cpu时间百分比

平均负载就是系统的平均活跃进程数，它反应了系统的整体负载情况，主要包括三个数值，分别是过去1分钟，过去5分钟、过去15分钟的平均负载，理想情况，平均负载等于逻辑cpu个数，这表示每个cpu都刚好被充分利用，如果平均负载大于逻辑cpu个数，就表示负载比较重了

进程上下文切换包括，无法获取资源而导致的自愿上下文切换，被系统强制调度导致的非自愿上下文切换

过多的上下文切换会将原本运行进程的cpu时间，消耗在寄存器、内核栈以及虚拟内存等数据的保存和恢复上，缩短进程真正运行的时间，成为性能瓶颈

cpu缓存的命中率，由于cpu发展的速度远快于内存的发展，cpu的处理速度就比内存的访问速度快很多，这样cpu访问内存的时，免不了要等待内存的响应，为了协调两者差距，cpu缓存（通常是多级缓存）出现了

三级缓存的大小依次增大，相应的性能一次降低，他们的命中率就是衡量的是cpu缓存的复用情况，命中率越高，则表示性能越好

查看平均负载 ：

先用uptime查看系统的平均负载，在平均负载升高，使用mpstat和pidstat分别观察每个cpu和每个进程cpu的使用情况，进程找出导致平均负载升高的进程

上下文切换：

先用vmstat，查看系统的上下文切换次数和中断次数，然后通过pidstat，观察进程的自愿上下文切换和非自愿上下切换的情况，最后通过pidstat观察线程的上下文切换情况，找出上下文切换次数增多的根源

进程cpu使用率升高的

进程cpu使用率升高：

先用top查看系统和进程的cpu使用情况，发现cpu使用率升高的进程，再用perf top观察进程的调用链，最终找出cpu升高的根源

系统的cpu使用率升高：

先用top观察系统cpu升高，但是通过top和pidstat，找不出高cpu使用率的进程，重新审视top的输出，从cpu使用率不高但处于Running状态的进程入手，找出可疑之处，最终通过perf record和perf report发现原因

不可中断进程和僵尸进程：

先用top观察到iowait升高，发现大量不可中断进程和僵尸进程，用dstst发现这是由磁盘读导致，通过pidstat找出相关的进程，用strace查看进程系统调用失败，通过perf分析进程调用链，找出原因

软中断：

通过top观察，系统的软中断cpu使用率升高，查看/proc/softirqs，找到几种变化速率快的软中断，通过sar命令，发现是网络小包问题，用tcpdump找出网络帧的类型和来源，确定问题