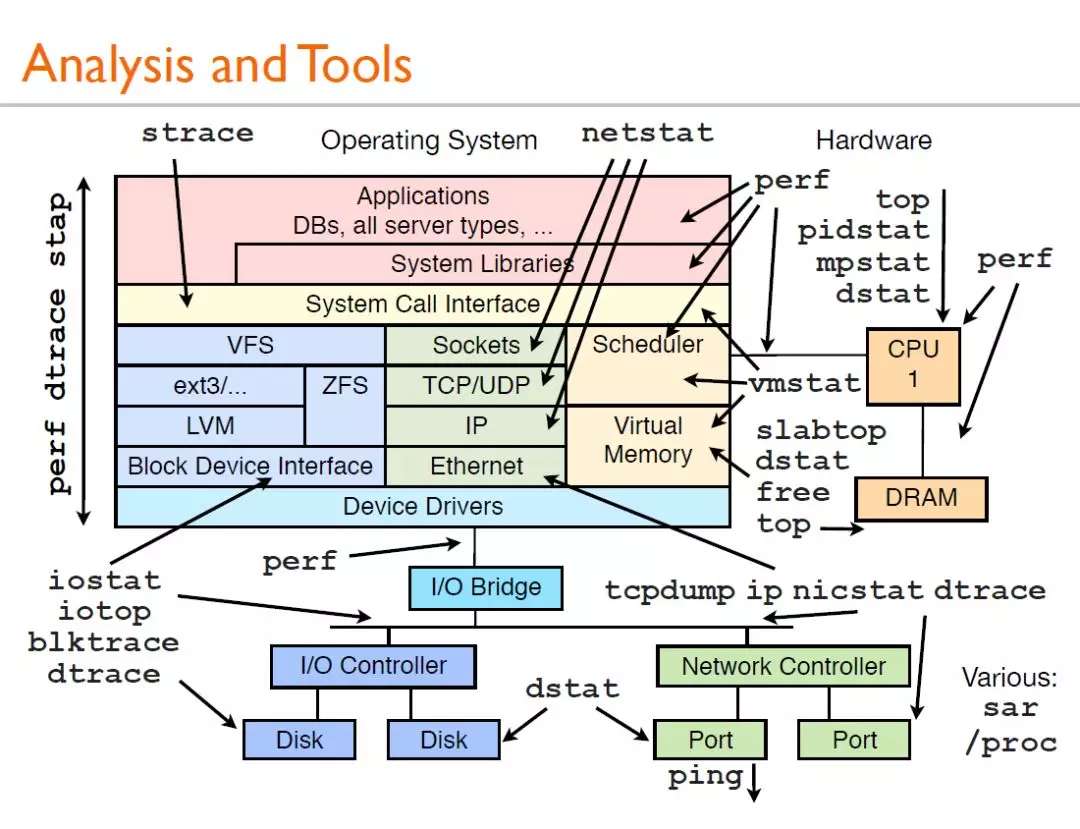
# 背景

背景知识：具备背景知识是分析性能问题时需要了解的。比如硬件cache；再比如操作系统内核。应用程序的行为细节往往是和这些东西互相牵扯的，这些底层的东西会以意想不到的方式影响应用程序的性能，比如某些程序无法充分利用cache，从而导致性能下降。比如不必要地调用过多的系统调用，造成频繁的内核/用户切换等。

# 性能分析工具



## uptime

$ uptime

23:51:26 up 21:31, 1 user, load average: 30.02, 26.43, 19.02

该命令可以大致的看出计算机的整体负载情况，load average后的数字分别表示计算机在1min、5min、15min内的平均负载。

## dmesg | tail

$ dmesg | tail

[1880957.563150] perl invoked oom-killer: gfp\_mask=0x280da, order=0, oom\_score\_adj=0

[...]

[1880957.563400] Out of memory: Kill process 18694 (perl) score 246 or sacrifice child

[1880957.563408] Killed process 18694 (perl) total-vm:1972392kB, anon-rss:1953348kB, file-rss:0kB

[2320864.954447] TCP: Possible SYN flooding on port 7001. Dropping request. Check SNMP counters.

打印内核环形缓存区中的内容，可以用来查看一些错误；

上面的例子中，显示进程18694因引内存越界被kill掉以及TCP request被丢弃的错误。通过dmesg可以快速判断是否有导致系统性能异常的问题。

## vmstat

vmstat(VirtualMeomoryStatistics,虚拟内存统计) 是Linux中监控内存的常用工具，可对操作系统的虚拟内存、进程、CPU等的整体情况进行监视。

vmstat的常规用法：vmstat interval times即每隔interval秒采样一次，共采样times次，如果省略times，则一直采集数据，直到用户手动停止为止。

简单举个例子：

可以使用ctrl+c停止vmstat采集数据。

第一行显示了系统自启动以来的平均值，第二行开始显示现在正在发生的情况，接下来的行会显示每5秒间隔发生了什么，每一列的含义在头部，如下所示：

▪ procs：r这一列显示了多少进程在等待cpu，b列显示多少进程正在不可中断的休眠（等待IO）。

▪ memory：swapd列显示了多少块被换出了磁盘（页面交换），剩下的列显示了多少块是空闲的（未被使用），多少块正在被用作缓冲区，以及多少正在被用作操作系统的缓存。

▪ swap：显示交换活动：每秒有多少块正在被换入（从磁盘）和换出（到磁盘）。

▪ io：显示了多少块从块设备读取（bi）和写出（bo）,通常反映了硬盘I/O。

▪ system：显示每秒中断(in)和上下文切换（cs）的数量。

▪ cpu：显示所有的cpu时间花费在各类操作的百分比，包括执行用户代码（非内核），执行系统代码（内核），空闲以及等待IO。

内存不足的表现：free memory急剧减少，回收buffer和cacher也无济于事，大量使用交换分区（swpd）,页面交换（swap）频繁，读写磁盘数量（io）增多，缺页中断（in）增多，上下文切换（cs）次数增多，等待IO的进程数（b）增多，大量CPU时间用于等待IO（wa）

## iostat

iostat用于报告中央处理器（CPU）统计信息和整个系统、适配器、tty 设备、磁盘和 CD-ROM 的输入/输出统计信息，默认显示了与vmstat相同的cpu使用信息，使用以下命令显示扩展的设备统计：

第一行显示的是自系统启动以来的平均值，然后显示增量的平均值，每个设备一行。

常见linux的磁盘IO指标的缩写习惯：rq是request,r是read,w是write,qu是queue，sz是size,a是verage,tm是time,svc是service。

▪rrqm/s和wrqm/s：每秒合并的读和写请求，“合并的”意味着操作系统从队列中拿出多个逻辑请求合并为一个请求到实际磁盘。

▪r/s和w/s：每秒发送到设备的读和写请求数。

▪rsec/s和wsec/s：每秒读和写的扇区数。

▪avgrq –sz：请求的扇区数。

▪avgqu –sz：在设备队列中等待的请求数。

▪await：每个IO请求花费的时间。

▪svctm：实际请求（服务）时间。

▪%util：至少有一个活跃请求所占时间的百分比。

## dstat

dstat显示了cpu使用情况，磁盘io情况，网络发包情况和换页情况，输出是彩色的，可读性较强，相对于vmstat和iostat的输入更加详细且较为直观。在使用时，直接输入命令即可，当然也可以使用特定参数。

## iotop

iotop命令是专门显示硬盘IO的命令，界面风格类似top命令，可以显示IO负载具体是由哪个进程产生的。是一个用来监视磁盘I/O使用状况的top类工具，具有与top相似的UI，其中包括PID、用户、I/O、进程等相关信息。

可以以非交互的方式使用：iotop -bod interval，查看每个进程的I/O，可以使用pidstat，pidstat -d instat。

## pidstat

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。

使用方法：pidstat –d interval；pidstat还可以用以统计CPU使用信息：pidstat –u interval；统计内存信息：Pidstat –r interval。

## top

top命令的汇总区域显示了五个方面的系统性能信息：

1、负载：时间，登陆用户数，系统平均负载；

2、进程：运行，睡眠，停止，僵尸；

3、cpu:用户态，核心态，NICE,空闲，等待IO,中断等；

4、内存：总量，已用，空闲（系统角度），缓冲，缓存；

5、交换分区：总量，已用，空闲

任务区域默认显示：进程ID，有效用户，进程优先级，NICE值，进程使用的虚拟内存，物理内存和共享内存，进程状态，CPU占用率，内存占用率，累计CPU时间，进程命令行信息。

## htop

htop是Linux系统中的一个互动的进程查看器，一个文本模式的应用程序(在控制台或者X终端中)，需要ncurses。

Htop可让用户交互式操作，支持颜色主题，可横向或纵向滚动浏览进程列表，并支持鼠标操作。

与top相比，htop有以下优点：

▪ 可以横向或者纵向滚动浏览进程列表，以便看到所有的进程和完整的命令行。

▪ 在启动上，比top更快。

▪ 杀进程时不需要输入进程号。

▪ htop支持鼠标操作。

## mpstat

mpstat是Multiprocessor Statistics的缩写，是实时系统监控工具。其报告与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。常见用法：mpstat –P ALL interval times。

## netstat

Netstat用于显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于检验本机各端口的网络连接情况。

常见用法：

netstat -npl 可以查看你要打开的端口是否已经打开。

netstat -rn 打印路由表信息。

netstat -in 提供系统上的接口信息，打印每个接口的MTU,输入分组数，输入错误，输出分组数，输出错误，冲突以及当前的输出队列的长度。

## ps

ps参数太多，具体使用方法可以参考man ps，常用的方法：ps aux #hsserver；ps –ef |grep #hundsun

▪ 杀掉某一程序的方法：ps aux | grep mysqld | grep –v grep | awk ‘{print $2 }’ xargs kill -9

▪ 杀掉僵尸进程：ps –eal | awk ‘{if ($2 == “Z”){print $4}}’ | xargs kill -9

## strace

跟踪程序执行过程中产生的系统调用及接收到的信号，帮助分析程序或命令执行中遇到的异常情况。

举例：查看mysqld在linux上加载哪种配置文件，可以通过运行下面的命令：strace –e stat64 mysqld –print –defaults > /dev/null

## lsof

lsof(list open files)是一个列出当前系统打开文件的工具。通过lsof工具能够查看这个列表对系统检测及排错，常见的用法：

查看文件系统阻塞 lsof /boot

查看端口号被哪个进程占用 lsof -i : 3306

查看用户打开哪些文件 lsof –u username

查看进程打开哪些文件 lsof –p 4838

查看远程已打开的网络链接 lsof –i @192.168.34.128

## perf

perf是Linux kernel自带的系统性能优化工具。优势在于与Linux Kernel的紧密结合，它可以最先应用到加入Kernel的new feature，用于查看热点函数，查看cashe miss的比率，从而帮助开发者来优化程序性能。

性能调优工具如perf，Oprofile等的基本原理都是对被监测对象进行采样，最简单的情形是根据tick中断进行采样，即在tick中断内触发采样点，在采样点里判断程序当时的上下文。假如一个程序90%的时间都花费在函数foo()上，那么90%的采样点都应该落在函数foo()的上下文中。运气不可捉摸，但我想只要采样频率足够高，采样时间足够长，那么以上推论就比较可靠。因此，通过tick触发采样，我们便可以了解程序中哪些地方最耗时间，从而重点分析。

想要更深的了解本工具可以参考：

<http://blog.csdn.net/trochiluses/article/details/10261339>

# 性能测试工具

## perf\_events

▪ perf\_events: 一款随 Linux 内核代码一同发布和维护的性能诊断工具，由内核社区维护和发展。Perf 不仅可以用于应用程序的性能统计分析，也可以应用于内核代码的性能统计和分析。

更多参考：<http://blog.sina.com.cn/s/blog_98822316010122ex.html。>

## eBPF tools

▪ eBPF tools: 一款使用bcc进行的性能追踪的工具,eBPF map可以使用定制的eBPF程序被广泛应用于内核调优方面，也可以读取用户级的异步代码。重要的是这个外部的数据可以在用户空间管理。这个k-v格式的map数据体是通过在用户空间调用bpf系统调用创建、添加、删除等操作管理的。more: <http://blog.csdn.net/ljy1988123/article/details/50444693。>

## perf-tools

▪ perf-tools: 一款基于 perf\_events (perf) 和 ftrace 的Linux性能分析调优工具集。Perf-Tools 依赖库少，使用简单。支持Linux 3.2 及以上内核版本。more: <https://github.com/brendangregg/perf-tools。>

## bcc

▪ bcc(BPF Compiler Collection): 一款使用eBPF的perf性能分析工具。一个用于创建高效的内核跟踪和操作程序的工具包，包括几个有用的工具和示例。利用扩展的BPF（伯克利数据包过滤器），正式称为eBPF，一个新的功能，首先被添加到Linux 3.15。多用途需要Linux 4.1以上BCC。

更多参考：<https://github.com/iovisor/bcc#tools。>

## ktap

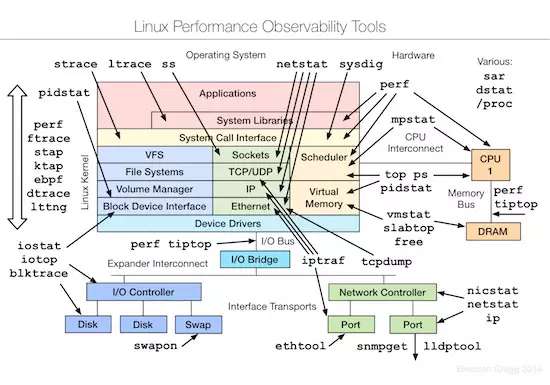
▪ ktap: 一种新型的linux脚本动态性能跟踪工具。允许用户跟踪Linux内核动态。ktap是设计给具有互操作性，允许用户调整操作的见解，排除故障和延长内核和应用程序。它类似于Linux和Solaris DTrace SystemTap。更多参考： <https://github.com/ktap/ktap。>

## Flame Graphs

▪ Flame Graphs:是一款使用perf,system tap,ktap可视化的图形软件，允许最频繁的代码路径快速准确地识别，可以是使用github.com/brendangregg/flamegraph中的开发源代码的程序生成。

更多参考：http://www.brendangregg.com/flamegraphs.html。

# 性能检测工具



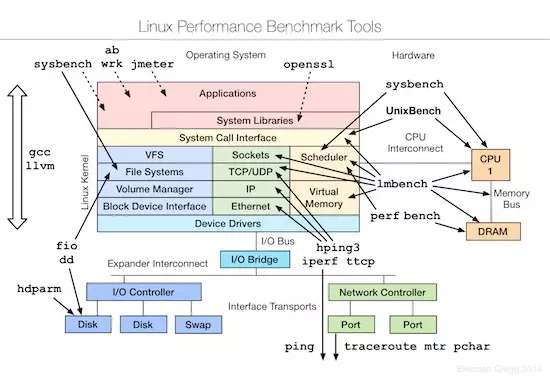
▪ 首先学习的Basic Tool有如下：

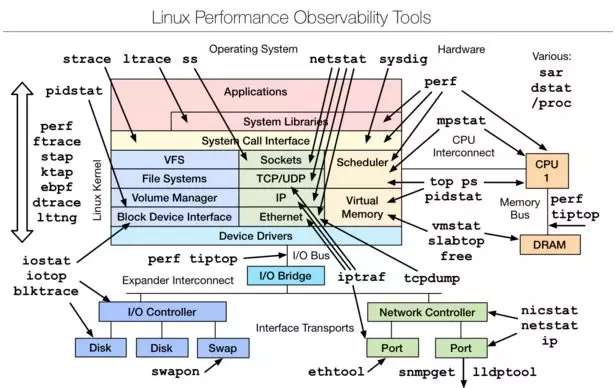
uptime、top(htop)、mpstat、isstat、vmstat、free、ping、nicstat、dstat。

▪ 高级的命令如下：

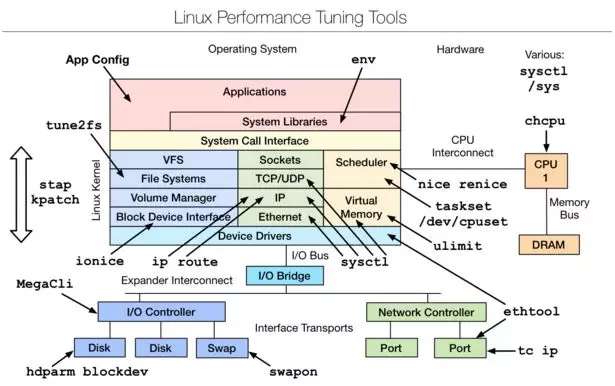
sar、netstat、pidstat、strace、tcpdump、blktrace、iotop、slabtop、sysctl、/proc。

# 性能测评工具

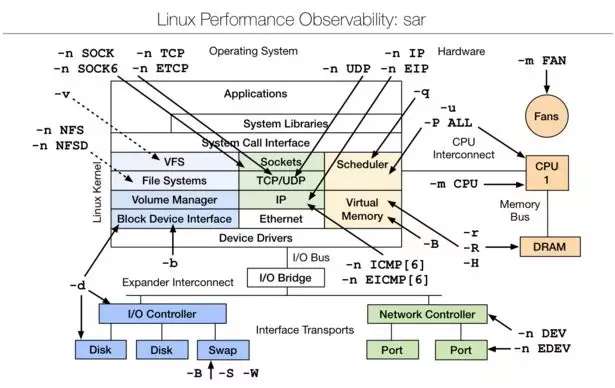




# 性能调优工具



# 性能观测工具



sar（System Activity Reporter系统活动情况报告）是目前LINUX上最为全面的系统性能分析工具之一，可以从多方面对系统的活动进行报告，包括：文件的读写情况、系统调用的使用情况、磁盘I/O、CPU效率、内存使用状况、进程活动及IPC有关的活动等方面。

sar的常归使用方式：sar [options] [-A] [-o file] t [n]

其中：

t为采样间隔，n为采样次数，默认值是1；

-o file表示将命令结果以二进制格式存放在文件中，file 是文件名。

options为命令行选项

# 性能问题

## 概述

系统的性能是指操作系统完成任务的有效性、稳定性和响应速度。Linux系统管理员可能经常会遇到系统不稳定、响应速度慢等问题，例如在linux上搭建了一个web服务，经常出现网页无法打开、打开速度慢等现象，而遇到这些问题，就有人会抱怨linux系统不好，其实这些都是表面现象。操作系统完成一个任务时，与系统自身设置、网络拓朴结构、路由设备、路由策略、接入设备、物理线路等多个方面都密切相关，任何一个环节出现问题，都会影响整个系统的性能。因此当linux应用出现问题时，应当从应用程序、操作系统、服务器硬件、网络环境等方面综合排查，定位问题出现在哪个部分，然后集中解决。

在应用程序、操作系统、服务器硬件、网络环境等方面，影响性能最大的是应用程序和操作系统两个方面，因为这两个方面出现的问题不易察觉，隐蔽性很强。而硬件、网络方面只要出现问题，一般都能马上定位。下面主要讲解操作系统方面的性能调优思路，应用程序方面需要具体问题具体对待。

以下从影响Linux性能的因素、分析性能涉及的人员、系统性能优化工具、系统性能评价标准四个方面介绍优化Linux的一般思路和方法。

## 分析

### 系统硬件资源

#### CPU

CPU是操作系统稳定运行的根本，CPU的速度与性能在很大程度上决定了系统整体的性能，因此，CPU数量越多、主频越高，服务器性能也就相对越好。但事实并非完全如此。

目前大部分CPU在同一时间内只能运行一个线程，超线程的处理器可以在同一时间运行多个线程，因此，可以利用处理器的超线程特性提高系统性能。在Linux系统下，只有运行SMP内核才能支持超线程，但是，安装的CPU数量越多，从超线程获得的性能方面的提高就越少。另外，Linux内核会把多核的处理器当作多个单独的CPU来识别，例如两个4核的CPU，在Lnux系统下会被当作8个单核CPU。但是从性能角度来讲，两个4核的CPU和8个单核的CPU并不完全等价，根据权威部门得出的测试结论，前者的整体性能要比后者低25％~30％。

可能出现CPU瓶颈的应用有db服务器、动态Web服务器等，对于这类应用，要把CPU的配置和性能放在主要位置。

#### 内存

内存的大小也是影响Linux性能的一个重要的因素，内存太小，系统进程将被阻塞，应用也将变得缓慢，甚至失去响应；内存太大，导致资源浪费。Linux系统采用了物理内存和虚拟内存两种方式，虚拟内存虽然可以缓解物理内存的不足，但是占用过多的虚拟内存，应用程序的性能将明显下降，要保证应用程序的高性能运行，物理内存一定要足够大；但是过大的物理内存，会造成内存资源浪费，例如，在一个32位处理器的Linux操作系统上，超过8GB的物理内存都将被浪费。因此，要使用更大的内存，建议安装64位的操作系统，同时开启Linux的大内存内核支持。

由于处理器寻址范围的限制，在32位Linux操作系统上，应用程序单个进程最大只能使用4GB的内存，这样以来，即使系统有更大的内存，应用程序也无法“享”用，解决的办法就是使用64位处理器，安装64位操作系统。在64位操作系统下，可以满足所有应用程序对内存的使用需求 ，几乎没有限制。

可能出现内存性能瓶颈的应用有NOSQL服务器、数据库服务器、缓存服务器等，对于这类应用要把内存大小放在主要位置。

#### 磁盘I/O性能

磁盘的I/O性能直接影响应用程序的性能，在一个有频繁读写的应用中，如果磁盘I/O性能得不到满足，就会导致应用停滞。好在现今的磁盘都采用了很多方法来提高I/O性能，比如常见的磁盘RAID技术。

通过RAID技术组成的磁盘组，就相当于一个大硬盘，用户可以对它进行分区格式化、建立文件系统等操作，跟单个物理硬盘一模一样，唯一不同的是RAID磁盘组的I/O性能比单个硬盘要高很多，同时在数据的安全性也有很大提升。

根据磁盘组合方式的不同，RAID可以分为RAID0，RAID1、RAID2、RAID3、RAID4、RAID5、RAID6、RAID7、RAID0+1、RAID10等级别，常用的RAID级别有RAID0、RAID1、RAID5、RAID0+1，这里进行简单介绍。

RAID 0：通过把多块硬盘粘合成一个容量更大的硬盘组，提高了磁盘的性能和吞吐量。这种方式成本低，要求至少两个磁盘，但是没有容错和数据修复功能，因而只能用在对数据安全性要求不高的环境中。

RAID 1：也就是磁盘镜像，通过把一个磁盘的数据镜像到另一个磁盘上，最大限度地保证磁盘数据的可靠性和可修复性，具有很高的数据冗余能力，但磁盘利用率只有50%，因而，成本最高，多用在保存重要数据的场合。

RAID5：采用了磁盘分段加奇偶校验技术，从而提高了系统可靠性，RAID5读出效率很高，写入效率一般，至少需要3块盘。允许一块磁盘故障，而不影响数据的可用性。

RAID0+1：把RAID0和RAID1技术结合起来就成了RAID0+1，至少需要4个硬盘。此种方式的数据除分布在多个盘上外，每个盘都有其镜像盘，提供全冗余能力，同时允许一个磁盘故障，而不影响数据可用性，并具有快速读/写能力。

通过了解各个RAID级别的性能，可以根据应用的不同特性，选择适合自身的RAID级别，从而保证应用程序在磁盘方面达到最优性能。

#### 网络宽带

Linux下的各种应用，一般都是基于网络的，因此网络带宽也是影响性能的一个重要因素，低速的、不稳定的网络将导致网络应用程序的访问阻塞，而稳定、高速的网络带宽，可以保证应用程序在网络上畅通无阻地运行。幸运的是，现在的网络一般都是千兆带宽或光纤网络，带宽问题对应用程序性能造成的影响也在逐步降低。

### 操作系统相关资源

基于操作系统的性能优化也是多方面的，可以从系统安装、系统内核参数、网络参数、文件系统等几个方面进行衡量，下面依次进行简单介绍。

#### 系统安装优化

系统优化可以从安装操作系统开始，当安装Linux系统时，磁盘的划分，SWAP内存的分配都直接影响以后系统的运行性能，例如，磁盘分配可以遵循应用的需求：对于对写操作频繁而对数据安全性要求不高的应用，可以把磁盘做成RAID 0；而对于对数据安全性较高，对读写没有特别要求的应用，可以把磁盘做成RAID 1；对于对读操作要求较高，而对写操作无特殊要求，并要保证数据安全性的应用，可以选择RAID 5；对于对读写要求都很高，并且对数据安全性要求也很高的应用，可以选择RAID10/01。这样通过不同的应用需求设置不同的RAID级别，在磁盘底层对系统进行优化操作。

随着内存价格的降低和内存容量的日益增大，对虚拟内存SWAP的设定，现在已经没有了所谓虚拟内存是物理内存两倍的要求，但是SWAP的设定还是不能忽略，根据经验，如果内存较小（物理内存小于4GB），一般设置SWAP交换分区大小为内存的2倍；如果物理内存大于8GB小于16GB，可以设置SWAP大小等于或略小于物理内存即可；如果内存大小在16GB以上，原则上可以设置SWAP为0，但并不建议这么做，因为设置一定大小的SWAP还是有一定作用的。

#### 内核参数优化

系统安装完成后，优化工作并没有结束，接下来还可以对系统内核参数进行优化，不过内核参数的优化要和系统中部署的应用结合起来整体考虑。例如，如果系统部署的是Oracle数据库应用，那么就需要对系统共享内存段（kernel.shmmax、kernel.shmmni、kernel.shmall）、系统信号量（kernel.sem）、文件句柄（fs.file-max）等参数进行优化设置；如果部署的是Web应用，那么就需要根据Web应用特性进行网络参数的优化，例如修改net.ipv4.ip\_local\_port\_range、net.ipv4.tcp\_tw\_reuse、net.core.somaxconn等网络内核参数。

#### 文件系统优化

文件系统的优化也是系统资源优化的一个重点，在Linux下可选的文件系统有ext2、ext3、ReiserFS、ext4、xfs，根据不同的应用，选择不同的文件系统。

Linux标准文件系统是从VFS开始的，然后是ext，接着就是ext2，应该说，ext2是Linux上标准的文件系统，ext3是在ext2基础上增加日志形成的，从VFS到ext4，其设计思想没有太大变化，都是早期UNIX家族基于超级块和inode的设计理念。

XFS文件系统是一个高级日志文件系统，XFS通过分布处理磁盘请求、定位数据、保持Cache 的一致性来提供对文件系统数据的低延迟、高带宽的访问，因此，XFS极具伸缩性，非常健壮，具有优秀的日志记录功能、可扩展性强、快速写入性能等优点。

目前服务器端ext4和xfs是主流文件系统，如何选择合适的文件系统，需要根据文件系统的特点加上业务的需求综合来定。

### 应用程序软件资源

应用程序的优化其实是整个优化工程的核心，如果一个应用程序存在BUG，那么即使所有其他方面都达到了最优状态，整个应用系统还是性能低下，所以，对应用程序的优化是性能优化过程的重中之重，这就对程序架构设计人员和程序开发人员提出了更高的要求。

## 总结

系统性能优化是个涉及面广、繁琐、长久的工作，寻找出现性能问题的根源往往是最难的部分，一旦找到出现问题的原因，性能问题也就迎刃而解。因此，解决问题的思路变得非常重要。

例如，linux系统下的一个网站系统，用户反映，网站访问速度很慢，有时无法访问。

针对这个问题，第一步要做的是检测网络，可以通过ping命令检查网站的域名解析是否正常，同时，ping服务器地址的延时是否过大等等，通过这种方式，首先排除网络可能出现的问题；如果网络没有问题，接着进入第二步，对linux系统的内存使用状况进行检查，因为网站响应速度慢，一般跟内存关联比较大，通过free、vmstat等命令判断内存资源是否紧缺，如果内存资源不存在问题，进入第三步，检查系统CPU的负载状况，可以通过sar、vmstat、top等命令的输出综合判断CPU是否存在过载问题，如果CPU没有问题，继续进入第四步，检查系统的磁盘I/O是否存在瓶颈，可以通过iostat、vmstat等命令检查磁盘的读写性能，如果磁盘读写也没有问题，linux系统自身的性能问题基本排除，最后要做的是检查程序本身是否存在问题。通过这样的思路，层层检测，步步排查，性能问题就“无处藏身”，查找出现性能问题的环节也就变得非常简单。