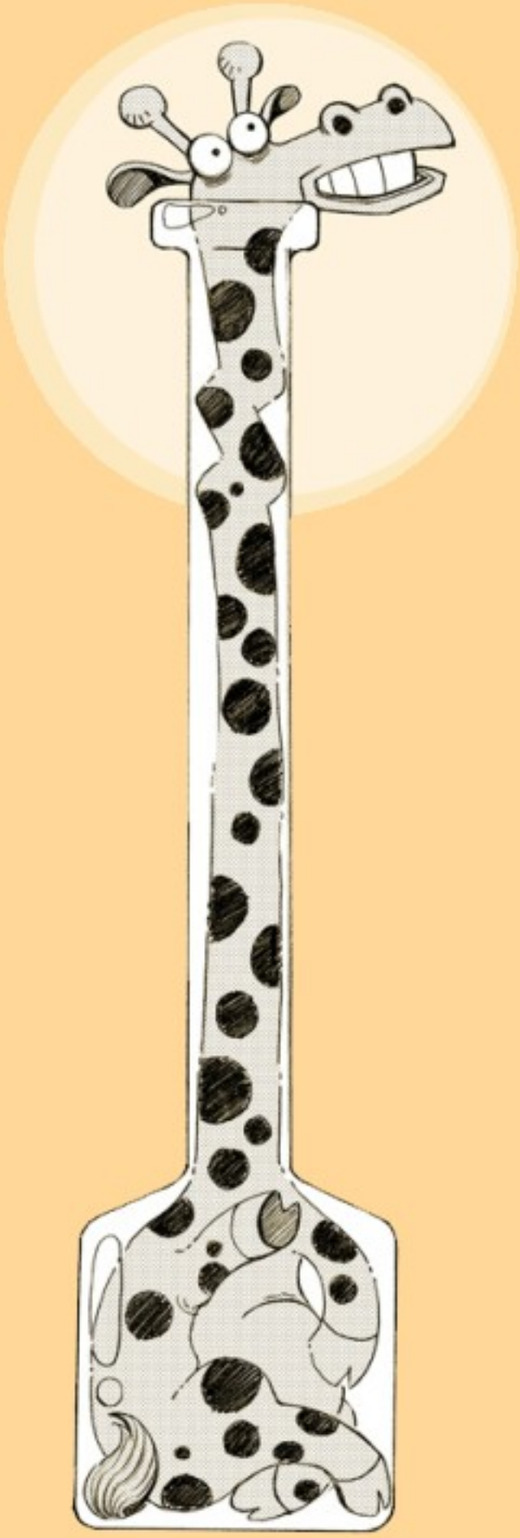
51CTO.com
技术成就梦想



Linux运维趋势

第四期

2011年1月号 关键字：CPU, 内存, IO, 网络, load

性能瓶颈

内容目录

Linux 运维趋势

杂志策划：51CTO 系统频道

本期主编：杨赛

封面制作：高鹏飞

交流圈子：
http://g.51cto.com/linuxops

邮件群组：
groups.google.com/group/linuxops-cn

订阅方式：发送 Email 到
linuxops-cn+subscribe@googlegroups.com

专题页面：
http://os.51cto.com/art/201011/233915.htm
http://down.51cto.com/zt/71

投稿信箱：
yangsai@51cto.com

“守住每一天”的意思就在这里：每天要做一件满意的事件，那怕一天就解决一个问题。

系统运维经验分享：守住每一天

文字整理/杨赛

人物简介：

刘宇，网名守住每一天，现新浪运维工程师，LinuxTone.org管理员之一，擅长CDN运维。目前关注集中化管理，分布式监控，以及分布式日志系统。

他的博客：http://liuyu.blog.51cto.com/

运维是一个所含范围很广泛的职业，在不同的企业、不同的工作环境下，运维的职责可能是完全不同的。即使单就系统运维而言，有些运维可能专注于内网服务器的维护，工作偏向网管和帮助台的职责；有些运维可能从开发环境、数据库到线上应用部分都负责；有些可能仅仅负责特定应用的运维。所以，即使是在职的系统运维，也可能对这个行业的同行们的工作内容了解有限。

为此，51CTO系统频道计划展开一项长期活动，请各个岗位上的、有数年运维经验的在职系统运维们分享他们自己的成长经验。本次我们邀请到新浪系统运维工程师、LinuxTone的管理员刘宇（守住每一天）来分享一下他的运维经验。

51CTO：首先，简单的跟我们介绍一下您现在的工作情况吧。您现在负责哪方面的工作？

守住每一天（以下简称守住）：我是在门户网站做Linux运维嘛，现在做CDN运维这块。近期核心主要放在集中化管理，分布式监控，和分布式日志系统上。

51CTO：您最初接触Linux是什么时候？最开始做运维这个工作是从什么时候开始的？

守住：我是03年开始接触linux，那时主要是自学。也没怎么去论坛交流之类的。从事工作是在07年，这个过程倒是还挺顺利的。主要是诚恳，踏实，肯学，底子要打好。这样的小伙一般企业都会要的。

51CTO：一开始做运维的几年，有什么印象深刻的事情吗？

守住：刚开始做的时候很难，遇到问题时，总是喜欢问人。不先查文档。这是一个很不好的习惯。后来慢慢改了：）

最痛苦的是最开始工作那会儿，刚接手设备的时候，一下遇到3台机器陆续宕机。不是硬盘故障，就是主板故障，最终换新设备，迁移解决。这么折腾一大圈，一下子就让我接触到了故障分析、服务器迁移方面的很多东西。

51CTO：您自己感觉在您的运维经历中，哪段时间是您成长最快的？那段时间您关注的技术点主要是哪些？有什么人提供指导或是交流吗？

守住：压力越大，成长越快。我成长也就是08年的时候吧。当时整个公司的网站迁移、升级，从机房选择一直到全部迁移升级完成，都是我一个人，压力相当大。那会儿主要是关注负载均衡的部署。秋香给我帮助挺大的。

51CTO：有什么对您帮助特别大的技术书籍么？或者文章/讲座？

守住：书我看得少~人懒，网上的文案看得多。然后再实践，再找相关的书进行一个系统的学习。我喜欢反着来，对我来说效率更高吧。

有关文案这方面可以给大家一些建议：我一般的学习方法是，先阅读官网README，再看WIKI，再看INSTALL，这就差不多了，可以准备动手操作。所以英文一定要有些基础。如果感觉英文吃力，那就只好搜索相关的中文做参考了。国内很多文档都有问题，尤其是那些瞎搜索出来的，所以建议大家选择比较靠谱的论坛或网站搜索，比如国外有 howtoforge，allcommands，cyberciti 等，国内就是 CU，51CTO，LinuxTone 这些我比较常去。

51CTO：说说 LinuxTone.org 这个站吧。当初建立这个社区是怎样的一个情况？主要是针对哪些层次的 Linux 技术人的？

守住：2008 年那会儿，netseek 创立了这个站，我也是同年加入管理团队的。正如同 LinuxTone 的建站宗旨说的，我们主要专注于系统服务、集群架构、安全监控、性能调优程序设计这些方面，算是比较偏中高端的综合 IT 运维人员吧。

51CTO：您认为一个理想的 Linux 运维交流社区应该是怎样的？

守住：还是那句话：“我为人人，人人为我”。每个人因为分享自己的经验获得快乐，我认为就是理想的社区。

51CTO：最后，介绍一下您现在的关注方向吧。2011 年您在技术上和个人成长上有什么计划么？

守住：技术方面，主要还是集中化管理，分布式监控，和分布式日志系统方面的继续深入具体计划方面其实我是没有的。我觉得计划不如变化。“守住每一天”的意思就在这里：每天要做一件满意的事件，那怕一天就解决一个问题。计划得再好，也赶不上你遇到的问题。有些时候一个问题解决就需要几天，那你的计划就全打乱了。今日事今日毕。早上在地铁里想好今天做什么，下班地铁记录今天做的工作这样就 OK 了。手机和 GOOGLE 如此方便。有什么不方便的呢？

51CTO：感谢守住每一天的分享！本次内容到此结束。如果您有什么问题想要沟通，或者希望听到某个运维进行分享，欢迎与守住或编辑交流。

原文：
http://os.51cto.com/art/201012/240498.htm

相关推荐：

系统管理员访谈系列：Tom Limoncelli 谈交流
http://os.51cto.com/art/201010/229500.htm

不要手动构建任何东西。如果你一定需要手动构建，那么就做两遍，在做第二遍的时候把用到所有的命令都提取出来。

系统运维秘诀：变化，监控，扩展（技术篇）

文/Dormando
编译/周雪峰

在运维管理的过程中，我发现了很多有价值的秘诀，本文是这些秘诀的一个总结。虽然这些秘诀可能比较“唯心”，但是我还是把它们总结出来了，相信它们会对你有帮助的。

为变化而设计

◆Google 的秘诀是正确的——“为变化而设计”。“变化”就是不得不部署新的软件，升级现有的软件，进行扩展，设备损坏，以及人员流动等。

◆每一件事情都是在寻找平衡点。你也许会认为把你的系统和某个操作系统或某个 Linux 发行版牢牢地绑定在一起是一个好主意，但事实上这跟把它们完全隔离一样糟。如果实在有必要，你可以进行分层，并使用一点间接性。

◆这并不意味着你的系统必须是平台无关的其实我们的目的很简单：一变二，二变二十，一个系统必须可以应对各种突发事件。也就是说，如果一个系统管理员被公共汽车撞了，你有应对的方案！如果挂载的硬盘出现故障了，你有应对的方案！如果某些人运行了 rm -rf /，你也有应对的方案！增量的进行变更。记得安全更新，以及保持内容更新。

使用自动的，可重复的构建过程

◆不要手动构建任何东西。如果你一定需要手动构建，那么就做两遍，在做第二遍的时候把用到所有的命令都提取出来。

◆下面这一点十分重要：将新硬件上线到生产环境的过程不应该超过 15 分钟，而且这个过程必须足够简单。否则，当一个服务器出现故障，而没有人知道如何更换它的时候，你就该倒霉了。

◆下面这一条是普世真理：这个世界上不存在“一次性”的服务器构建。即使你的服务器只需要构建一次，但只要你构建过一次，就一定会有第二次。比如，当它损坏的时候，或者你必须进行一次重大的升级才能让它在在接下来的两年时间里更加稳定的时候。

◆测试，检查新构建好的服务器。这应该是比较容易的，因为你的构建过程都是自动化的对吧！

◆脚本化的构建，意味着从某个 Linux 发行版的 V3 升级到 V4 应该是很快的。安装 V4，对脚本进行测试。如果有问题，参考文档并修复它，直到它可以再次正常工作。这最多应该是一个星期的工作，而不是一个长达一年的浩大工程（因为那时，刚刚完成的 V5 已经发布了！）

使用冗余

◆容易重新构建，并不意味着你可以忽视冗余。跳转盒，邮件服务器，计费网关，等等。如果其中的一半挂掉了却并不造成客户的宕机生活将会变得更加简单。

◆按照以上方针来做的话，当某个设备在凌晨 3 点出现故障的时候，你可以“以后再处理那个出现故障的设备！”，把冗余的机器先替换上去。

◆下面这一条是个聊胜于无的解决方案：Rsync。DRBD 也许也不是一个完美的解决方案，但是它可以提供令人称奇的服务。

使用备份

◆备份是个严肃的话题。使用硬盘，烧录磁带。压缩它们，移动它们，并行地运行。对每一样东西进行备份！

◆如果你的构建过程是自动的，整个过程都可以被备份。如果到目前为止的几条你都做到了，那么一个真正的“灾难恢复”计划也许并不是那么遥不可及的。

监控正确的东西

◆监控你能监控的所有东西，而且要用正确的方法来进行监控。如果你的 NFS 服务器挂掉了，不要让你的监控工具发送 1000 条警报。如果对你的系统来说，超时的警报没有什么实际意义，那就别让它发。要针对各种具体的情况进行成功性测试：是的，这个服务可以进行一个新的 TCP 连接，它甚至可以响应，但是它还记得它要做什么工作吗？

◆如果你有 500 个 Web 服务器，其中一个挂掉了，你可能不必马上知道这个情况。但是，如果负载均衡器没有把这台机子踢出去，导致错误报告出现在了用户的屏幕上，那么你必须知道这个情况！

有关数据图形化，历史数据

◆图形的作用是让趋势可视化。历史数据的作用是让你对数据进行精确的分析。不要把这两者混为一谈！对图形进行目测，很容易获得错误的数值。许多站点都使用 rrd 类型的系统或其他的数据聚合系统，此类系统按照时间对数据进行平均化处理，然后保存在存储空间中这不仅仅是难以阅读的问题：这根本是错误的！

◆如果你要浏览数百张图才能精确地对一个问题进行定位，那真是糟透了。想要找出极值请使用脚本提取数据。

◆如果你必须使用图形来解决问题，尽量把各种高级的概念整合到一个单一的页面中，然后让这个页面链接到拥有具体信息的子页面中如果你在数据库负载中可以看到一个峰值，你可以点击这个页面对那些数据库进行概览，然后找到那一两台可疑的机器。基本的理念是尽快地缩小范围，尽可能的减少猜测。

日志记录，使用多个数据流

◆无论是独立工作还是与开发部门合作，都要把尽可能多的有用的信息记录到日志中。无论是分析之后再保存，还是直接扔进数据库中生成报告，这些都无所谓。信息终归是有用的。

◆有用的例子：页面呈现时间（哪个页面？哪个设备？），面向用户的错误，数据库和内部服务错误，带宽使用率等。

◆建立图表，报告，并对产生的历史数据进行比较。

◆报告是十分重要的。每周或每天对你的基础设施变更进行汇总。

数据存储方式，数据库

◆诚然，数据库运维是一套完整而独立的知识体系。但是有时，你不能把一切都丢给你的 DBA。

◆拥有多个冗余的数据库会给你带来很多好处。对于一个庞大的 Oracle 实例来说，从前，很多运维工作需要好几个小时的关机维护时间而现在，完全可以在服务运行的同时进行。MySQL 和数据库复制功能是一件奇妙的事情。

◆和 DBA 们一起努力，尽量为可能会发生问题的数据库争取到最好的硬件。RAID 10，大量的 RAM，高速硬盘，乃至于强悍的 RAM 磁盘和 SSD。运维人员对提供商要货比三家，这样可以减轻 DBA 对硬件的恐惧。从长远来看，找出哪个品牌的硬件更加优秀会节省大量的资金。

◆数据库配置一直在改变。现在出现了 HiveDB，MySQL Proxy，DPM 这些软件。我们绝对应该对巨大的数据集进行分割。我们也可以考虑一下像 starling 和 Gearman 这样具有一定创新性的软件。了解一下这些软件的用途，同时，了解一下并不是一切东西都要保存在一个数据库中的。

◆善用你的过滤器！如果这些数据很重要，应该对它们进行备份！单片的 NFS 服务器的快照很奇妙，它并不是一个备份！

◆可以虑一下替代的解决方案。MogileFS 现在变得越来越好了。实际上，还有其他类似的项目可以免费（或廉价）地维护大量的存储文件。类似的系统基本上都是是为 youtube.com、archive.org 等站点而开发的。我们最终会让廉价的 NFS 过滤器成为标准！

多一些横向扩展，少一些纵向扩展

◆横向扩展是我们应该走的路。应该使用常规的（即：可用的，价格适中的，标准的。而不是特便宜的！）硬件，然后和大家一起努力确保各方面都可以进行横向扩展。

◆横向扩展从两台机子开始。另外，进行冗余的时候也会涉及到横向扩展。

◆尽可能的横向扩展，但是不要傻乎乎的扩展。在 MySQL 复制中有一个经典的白痴扩展的例子：使用一个 master 对很多个 slave。所有的 slave 必须完成全部的写入，而写入次数是与读取次数成比例增加的（大多数应用都是这样）。也就是说，你添加的 slave 越多，通过添加 slave 扩展的资源就越少。

◆留意一下替代的解决方案。按照用户或区域对多个数据库进行划分，同时避免增加过多的 slave。实际上，有许多种方法可以达到这个目的。

◆一切都可能扩展！路由器，交换机，负载均衡器，Web 服务器，数据库，等等。

◆记得纵向扩展吗？以前那些邪恶的大型机们有很多核，很多 IO 板，配备了非常昂贵的存储设备。而现在，多核这个概念开始蔓延了。

◆RAM 是廉价的。

◆将以上两点合并起来，这意味着你只需要再次合并服务就可以了。这儿有一个负载均衡器，那儿有一个 Web 服务器……如果一个应用程序可以使用许多个 CPU（比如 apache），那么这是完美的。如果它不能（比如

memcached），那么你最终可能会由于离散的服务太多而浪费掉大量的可用资源。

◆作业系统（job systems）也许可以填补这个鸿沟。哪里的核心的越多，哪里的工作线程就越多。

缓存

◆对于开发者和系统运维人员来说，缓存可是个好东西，值得大力发展！的确，它是不可思议的。它是与众不同的。有时你可能必须要为它做一个权衡。有效地使用缓存可以让系统的整体性能提升 10 倍之多。对于你当前的系统来说，这是一个巨大的“放大镜”，并且，它的成本在总成本中只占很小的一部分。

◆Memcached。它可以为服务提供缓存，让数据库结构非标准化（这可以提升性能！），对 squid 缓存进行优化，甚至可以提高操作系统缓存的利用率。

◆测试它，玩弄它，并打破它。使用缓存会带来新的问题。要做好准备。

让工作异步化

◆可以使用 Starling，Gearman，The Schwartz 等工具。作业系统可以给应用程序提供更多的灵活性。工作线程可以一次性地产生出来，也可以是持久的（载入缓存数据，准备数据等）。它们可以在不同的硬件上，它们的地理位置也可以不同。它们既可以是同步的也可以是异步的。

◆维护这些东西是一个运维人员的问题。使用它们既是开发者的问题也是运维人员的问题。

◆当用户点击“给我所有的朋友发送邮件”的时候，把这个工作列入计划，然后马上说：“OK，已经完成了！你的朋友马上会收到你的邮件！”——通过异步化的方式来处理这个工作。

◆作业系统是衔接各个服务的一个场所。博客投递-〉IM 通知，定期计费-〉收费服务，网关认证等。

◆容易扩展。在请求进入的地方会有一些瓶颈，所有的工作线程必须要做的事情就是“拉”。这个是相对于 HTTP 中大量推/拉的状态而言的。

安全和巡查

◆一定要安装安全更新！这十分重要！有很多疯狂的网络专家致力于在尽可能短的时间内给你提供这些更新。不要因为你害怕改变而让他们白白地付出劳动。

◆安全性也是分层的。明白你能确保什么，以及不能做什么。MySQL 有密码访问机制，并不意味着可以允许直接通过互联网来访问它。

◆在 SSH 上禁用密码。使用经过加密的 passphrase 密钥来进行身份验证。远程的用户无法猜出你的私有密钥。他们必须从你这里才能得到它。把它保管好。做好这点，就没必要在防火墙中关闭你的 SSH 端口了。

◆搞清楚你的应用程序是如何工作的，它具体需要做些什么，并相应的进行调整。比如说如果你的应用当中，只有付费页面和 Twitter 投递服务是需要连接外部互联网的，那么就把它们做成工作线程。将这部分工作线程放在特定的设备中，让它们只能访问特定的主机。这

可以神不知鬼不觉地把你的网络的其余部分保护起来。

◆对于 PHP 站点来说，以上这些建议尤其重要，但是在其他地方，它们也可以发挥作用。如果有人要入侵，那么多半是通过你的应用。即使有人从前门入侵了系统，也要让他们花费很大精力才能进入保险箱。你需要确保的是他们无法将数据带走或上传至别的什么服务器上。

◆除了这些具体的建议之外，你还应该多读一些资料。自己判断，自己动手测试。如果你不知道一个安全模型是如何工作的，一时半会儿可能问题不大，但是这就会导致你不知道它的限制在哪里，甚至于无法判断它是否在工作。

◆基于测试，理论，攻击树的安全机制是不会在背后给你一刀的。当人们构想出模糊的安全模型的时候我也很喜欢它，但是像我这样的普通人都可以把它弄的支离破碎。

◆尽可能地进行巡查！登录，退出，以及使用的命令都要进行审查。对面向外部服务的所有访问，包括所有在请求中指定的参数，都要进行审查。对于你的应用程序来说，找出极值然后彻底禁止输入超出范围的值。做你能做的所有事情，让数据以可追溯的方式工作。

◆如果你怀疑某些东西可能会被破坏，应该采取适当的预防措施，最好懂得一点计算机取证方面的知识（或者聘请一个专门从事这项业务的公司）。通过移除可疑的网络访问来做出响应，通过一系列的控制台或直接通过终端来检查整个系统。在已经被破坏的机器上，避免使用任何的服务，配置文件，或数据。很多人都是“清除了一个木马”，但是不知道它是怎么进来的——这样并不算真正地清除了这个木马。

◆如果你有安全团队，取证专家，或其他人手，那么你尽量不要接触那台机器，把它隔离起来。这意味着不要重新启动它来“清除一些奇怪的进程”。他们需要这些证据。如果你必须这么做，就去做吧，但是要记得把系统彻底地清理干净，打上所有的安全更新，尽量搞清楚他们是否已经破坏了重要的数据。做你能做的所有事情。

◆安全实际上是一种权衡。如果你做错了，开发者和用户们都会“揭竿而起”：他们会找到一些方法来绕过安全机制。如果他们可以绕过它，那说明你的工作并没有做好。如果他们不能绕过它，那么他们也许会放弃，然后离开。

◆紧抓访问控制。这意味着运维人员必须要为已经锁上门的“房间”提供一些窗户。不让开发人员进入生产环境，意味着他们必须抹黑解决难题。你的确不能让开发者们直接对服务进行修改，但是你可以提供日志工具和调试工具等等。对于各种产品来说，这些都是成功的秘诀。

原文：
http://dormando.livejournal.com/484577.html

译文：
http://os.51cto.com/art/201101/241769.htm

相关阅读：

系统运维枯燥生活中的那一丝快乐
http://os.51cto.com/art/201012/241108.htm

2010 年十大 Linux 运维小窍门
http://os.51cto.com/art/201012/238961.htm

在 2010 年 12 月到 2011 年 1 月之间，发生了下面这些事儿……

八卦，趣闻与数字 2010.12 - 2011.01

收集整理/51CTO 系统频道

【Linux 安全漏洞】系统安全高手 Dan Rosenberg 发布了一段 C 程序，这段 200 多行的程序利用了 Linux Econet 协议的 3 个安全漏洞，可以导致本地帐号对系统进行拒绝服务或特权提升。

http://os.51cto.com/art/201012/237990.htm

【支付宝】支付宝正式对外发布支付宝安全控件 for Linux 版本，同期还为 iPad、iPhone、Android 等系统环境提供了无控件登录浏览模式。

http://os.51cto.com/art/201012/239221.htm

【Mint】圣诞之际，基于 Debian Testing 的滚动升级 Linux Mint 版本 2012 发布，包含 64 位版本支持。

http://os.51cto.com/art/201012/240558.htm

【LibreOffice】Ubuntu 开发者已经确认 LibreOffice 将会包括在 Ubuntu 11.04 Natty Narwhal 当中。

http://os.51cto.com/art/201101/242171.htm

【Hotmail】Hotmail 服务器健康度监控中有一种方式是通过自动化测试进行的……在 12 月 30 日，我们的一个脚本代码意外地将部分真实用户的帐号与测试帐号一并从目录服务器上删除了。

http://os.51cto.com/art/201101/242496.htm

【Apache】截止 2010 年底，互联网上有 2.55 亿站点；Apache 2009 年托管 1.09 亿个站点，而 2010 年数字变成了 1.52 亿。

http://os.51cto.com/art/201101/241932.htm

【Novell】Attachmate 公司宣布以 22 亿美元现金收购 Novell，同时 Novell 公司的 882 项专利将以 4.5 亿美元的价格出售给一家名为 CPTN Holdings LLC 的公司。而日前根据德国官方披露的资料，这家 CPTN Holdings LLC 公司的背后居然是四家业界巨头：微软、苹果、甲骨文和 EMC。

http://os.51cto.com/art/201012/239047.htm

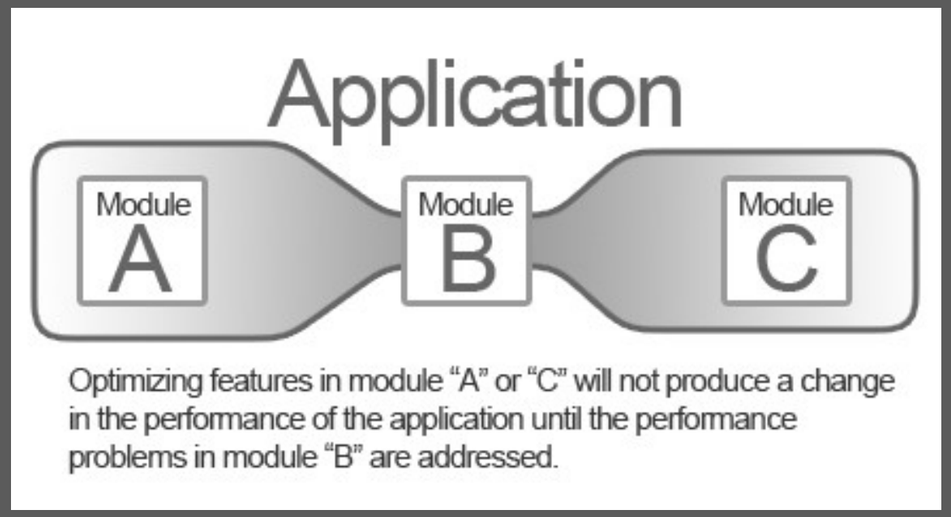
【OpenBSD】Perry 是 NETSEC 的前 CTO，负责资助 OpenBSD Crypto Framework，同时还在 2000--2001 期间担任 FBI 的咨询师。Perry 声称有些开发者在 OpenBSD 的 IPsec 栈中加入了大量后门，这是根据 FBI 的要求做的。

http://os.51cto.com/art/201012/241115.htm

【国产操作系统】民用的“中标 Linux”操作系统和解放军研制的“银河麒麟”操作系统在上海正式宣布合并，双方今后将共同以“中标麒麟”的新品牌统一出现在市场上，并将开发军民两用的操作系统。

http://os.51cto.com/art/201012/239006.htm

本期专题：Linux 性能瓶颈



瓶颈：当整个系统的性能或资源被某个或数个组件或元素限制的时候，这种现象被称之为瓶颈。该词取自水流的类比：当你从瓶子里往外倒水的时候，水流出的速度是由瓶口的宽度限制的，即所谓的瓶颈。

-自由的维基百科，Bottleneck 词条

图片文字解说：（在这个应用中），针对模块 A 或 C 进行特性优化是无法提升整个应用的性能的。只有当模块 B 的性能问题得到解决的时候，整个应用的性能才会提升。

http://polymorphicpodcast.com/shows/webperformance/

为什么进行 Linux 性能监测？

文/易龙

系统优化是一项复杂、繁琐、长期的工作，优化前需要监测、采集、测试、评估，优化后也需要测试、采集、评估、监测，而且是一个长期和持续的过程，不是说现在优化了，测试了，以后就可以一劳永逸了也不是说书本上的优化就适合眼下正在运行的系统，不同的系统、不同的硬件、不同的应用优化的重点也不同、优化的方法也不同、优化的参数也不同。性能监测是系统优化过程中重要的一环，如果没有监测、不清楚性能瓶颈在哪里，优化什么呢、怎么优化呢？所以找到性能瓶颈是性能监测的目的，也是系统优化的关键。系统由若干子系统构成，通常修改一个子系统有可能影响到另外一个子系统，甚至会导致整个系统不稳定、崩溃。所以说优化、监测测试通常是连在一起的，而且是一个循环而且长期的过程。

通常，我们监测的子系统有以下这些：

- CPU
- Memory
- IO
- Network

这些子系统互相依赖，了解这些子系统的特性，监测这些子系统的性能参数以及及时发现可能会出现的瓶颈对系统优化很有帮助。

应用类型

不同的系统用途也不同，要找到性能瓶颈需要知道系统跑的是什么应用、有些什么特点，比如 web server 对系统的要求肯定和 file server 不一样，所以分清不同系统的应用类型很重要，通常应用可以分为两种类型：

IO 相关，IO 相关的应用通常用来处理大量数据，需要大量内存和存储，频繁 IO 操作读写数据，而对 CPU 的要求则较少，大部分时候 CPU 都在等待硬盘，比如，数据库服务器、文件服务器等。

CPU 相关，CPU 相关的应用需要使用大量 CPU，比如高并发的 web/mail 服务器、图像/视频处理、科学计算等都可被视作 CPU 相关的应用。

看看实际中的例子，第 1 个是文件服务器拷贝一个大文件时表现出来的特征，第 2 个是 CPU 做大量计算时表现出来的特征：

```
$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  4    140 1962724 335516 4852308  0    0   388 65024 1442  563  0  2 47 52  0
 0  4    140 1961816 335516 4853868  0    0   768 65536 1434  522  0  1 50 48  0
 0  4    140 1960788 335516 4855300  0    0   768 48640 1412  573  0  1 50 49  0
 0  4    140 1958528 335516 4857280  0    0  1024 65536 1415  521  0  1 41 57  0
 0  5    140 1957488 335516 4858884  0    0   768 81412 1504  609  0  2 50 49  0
$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 4  0    140 3625096 334256 3266584  0    0     0    16 1054  470 100 0  0  0  0
 4  0    140 3625220 334264 3266576  0    0     0    12 1037  448 100 0  0  0  0
 4  0    140 3624468 334264 3266580  0    0     0   148 1160  632 100 0  0  0  0
 4  0    140 3624468 334264 3266580  0    0     0     0 1078  527 100 0  0  0  0
 4  0    140 3624712 334264 3266580  0    0     0    80 1053  501 100 0  0  0  0
```

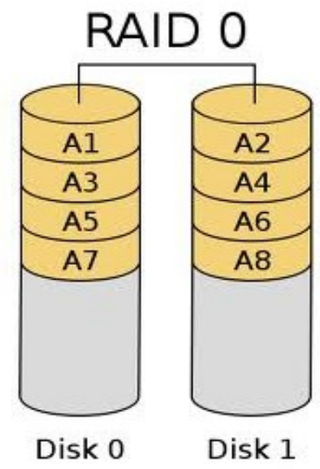
上面两个例子最明显的差别就是 id 一栏，代表 CPU 的空闲率，拷贝文件时候 id 维持在 50% 左右，CPU 大量计算的时候 id 基本为 0。

底线

我们如何知道系统性能是好还是差呢？这需要事先建立一个底线，如果性能监测得到的统计数据跨过这条线，我们就可以说这个系统性能差，如果数据能保持在线内我们就说性能好

建立这样底线需要知道一些理论、额外的负载测试和系统管理员多年的经验。如果自己没有多年的经验，有一个简单划底线的办法就是：把这个底线建立在自己对系统的期望上。自己期望这个系统有个什么样的性能，这是一个底

线，如果没有达到这个要求就是性能差。比如 VPSee 上个月有个 RAID0 的测试，期望的测试结果应该是 RAID0 的 IO 性能比单硬盘有显著提高，底线是 RAID0 的 IO 至少要比单硬盘要好（好多少不重要，底线是至少要好）测试结果却发现 RAID0 性能还不如单硬盘，说明性能差，这个时候需要问个为什么，这往往是性能瓶颈所在，经过排查发现是原硬盘有硬件瑕疵造成性能测试结果错误。



底线是 RAID0 的 IO 至少要比单硬盘好

监测工具

我们只需要简单的工具就可以对 Linux 的性能进行监测，以下是 VPSee 常用的工具：

工具	简单介绍
top	查看进程活动状态以及一些系统状况
vmstat	查看系统状态、硬件和系统信息等
iostat	查看 CPU 负载，硬盘状况
sar	综合工具，查看系统状况
mpstat	查看多处理器状况
netstat	查看网络状况
iptraf	实时网络状况监测
tcpdump	抓取网络数据包，详细分析
tcptrace	数据包分析工具
netperf	网络带宽工具
dstat	综合工具，综合了 vmstat, iostat, ifstat, netstat 等多个信息

（本文为系列第一篇，后面针对 CPU、内存 IO、网络等每一部分都各有文章叙述。更多内容可参考原文。）

原文：
http://www.vpsee.com/2009/11/linux-system-performance-monitoring-introduction/

相关阅读：

linux 进程查看连载之 linux top 命令
http://os.51cto.com/art/200910/158910.htm

Linux 系统监控工具之 vmstat 详解
http://os.51cto.com/art/201005/200672.htm

Linux 性能检测工具 iostat，ps 和 pstree
http://os.51cto.com/art/201006/203887.htm

mpstat：监测 CPU（包括多 CPU）性能
http://book.51cto.com/art/201008/217272.htm

Linux 网络性能调试工具 Netstat 命令篇
http://os.51cto.com/art/201006/203422.htm

关于内存的常规经验法则是“增加再增加”。当性能问题指向内存的时候，一般的共识是增加更多的内存。不过，这种做法只能在短期内有效。

细数十个最令人头疼的性能瓶颈

文/Kenneth Hess
编译/哲婷

当你听到“性能瓶颈”这个名词，马上出现在你脑海中的形象应该是CPU、内存、磁盘和网络。不可否认，这些是寻找性能问题所在很好的出发点，但是它们并不是所有性能问题唯一的聚集点。以下我们将列出另外六个问题可能潜在的地方，当你在出现性能故障的时候就可以对其进行一一排除。有时候，知道问题的根源所在可以很好地防止你的个人系统崩溃。

请注意，以下排序不分先后。

1.CPU

CPU是电脑进行计算和指令操作的中枢。

CPU可以处理数以百万计的计算和指令，但是在这些操作超过负荷的时候，它的性能可能就会受到影响。当CPU支持大于75%的运算的时候，整个系统的运行速度就会减缓。CPU需要一些空间来承载不定时的数据冲击，在这个时候，工作负载可能会在很短的时间内达到100%。CPU负载是性能瓶颈的一个常见来源。

2.内存

关于内存的常规经验法则是“增加再增加”。当性能问题指向内存的时候，一般的共识是，增加更多的内存。不过，这种做法只能在短期内有效。其实，当性能瓶颈指向内存的时候，往往是因为欠缺的软件设计（内存泄露）或者其它的系统缺陷，只不过其表现为内存问题。解决内存性能问题的关键在于要在尝试增加内存之前找到真正的问题根源所在。

3.存储

磁盘速度、RAID类型、存储类型和控制器技术这些结合在一起就产生了我们所认识的磁盘I/O。磁盘I/O是系统管理员和用户们普遍焦虑的性能问题来源。这里有实际和物理的性能限制，即使是使用现在最好的磁盘技术也不例外。当在磁盘上结合和分离工作负载的时候请使用最佳做法。

4.网络

网络往往被大家认为是一种常见的性能瓶颈来源，但是事实上并非如此。除非有一个网络组件发生硬件故障，比如交换机端口损坏、电缆损坏、网卡或者路由器配置出错。所以在你怀疑“网络”性能出现问题的时候最好检查一下其它地方。网络上感知到的运行缓慢问题通常应该指向本列表中的其它九项。

5.应用程序

虽然没有应用程序开发人员愿意承认，但是劣质编码的应用程序成为了硬件问题的导火线处于静止状态的系统将严重受到应用程序启动并且没有任何迹象表明的关闭的影响。这是一

场在系统管理员和开发人员之间一直在持续的斗争。他们都声称是对方的责任。不过，在进行了无数小时的硬件性能测试之后，人们发现问题出在应用程序。

6.恶意软件

病毒、木马和间谍软件在由于恶意软件导致的性能问题中占很大一部分。在出现问题的时候，用户大多会抱怨网络、应用程序或者他们的计算机。这些性能杀手可能隐匿在一个或者多个服务器系统中、用户工作站中或者两者的结合体中。恶意软件的感染十分普遍，因此，你必须采取全方位的防御措施来阻止他们。杀毒软件、反间谍软件、本地防火墙和定期补丁能够帮助你保护系统并防止由此产生的性能问题。

7.工作负载

智能工作负载管理可以帮助你防止由于欠缺平衡的工作负载或者负载平衡计划设计不周而引起的性能问题。把另外一个系统添加到有问题的集群中去可以缓解它的工作压力。但是在虚拟环境下的物理机上似乎更容易完成这一操作。最好的建议是衡量所有系统的工作能力和性能并对报告给你的数据进行分析。转移工作负载，加强系统并小心留意它的表现。

8.故障或者过期的硬件

旧的硬件很容易出故障。硬件问题可能会导致系统重启、数据丢失，它的不可预测性让系统管理员们叫苦不迭。阻止这类悲剧最好的办法是保持硬件更新频率，使用冗余硬件，并对你的系统进行仔细的监控。

9.文件系统

你是否知道你对文件系统的选择可能会对性能产生深远的影响？答案是肯定的。有些文件系统，比如 JFS，可能只占用很少的 CPU。XFS 拥有很高的可扩展性和性能。NTFS 是一个可恢复的高性能文件系统。新的 EXT4 文件系统能够有效地支持庞大的文档。每种文件系统都有它自己的针对性，如果不能为一个应用程序选择正确的文件系统就可能导致灾难性的后果。慎重考虑你的文件系并做出适合你工作的正确选择。

10.技术

你为自己的基础架构所选择的技术在性能方面扮演着非常重要的角色。比如，如果你的服务致力于一个虚拟架构技术，你可能会遇到在物理系统中不曾见过的性能问题。另外，有些工作负载可以在虚拟技术中生机勃勃。例如，LAMP（Linux、Apache、MySQL、PHP）工作负载比在 KVW 上的速度更快。不过，集装箱式的虚拟化（OpenVZ、Parallels、Solaris Zones）拥有适合任何工作负载的性能。

原文：

http://www.serverwatch.com/trends/article.php/3912821/Uncover-Your-10-Most-Painful-Performance-Bottlenecks.htm

译文：

http://os.51cto.com/art/201011/233490.htm

相关阅读：

十大 x86 服务器常见故障——系统篇

http://os.51cto.com/art/201008/220004.htm

对 Linux 磁盘管理稍微有一些学习和经验的朋友们应该都知道 df、du 和 fdisk 这三个常用命令：df 用于检查文件系统磁盘占用情况，du 检查磁盘空间占用情况，而 fdisk 用于磁盘分区。

Linux 磁盘管理三板斧的使用心得

文/李洋

对 Linux 磁盘管理稍微有一些学习和经验的朋友们应该都知道 df、du 和 fdisk 这三个常用命令。这三个工具是本人在进行 Linux 磁盘管理时常用的工具。

1．df

df 命令可以获取硬盘被占用了多少空间，目前还剩下多少空间等信息，它也可以显示所有文件系统对 i 节点和磁盘块的使用情况。

df 命令各个选项的含义如下：

```
-a：显示所有文件系统的磁盘使用情况，包括0块（block）的文件系统，如/proc 文件系统。
-k：以 k 字节为单位显示。
-i：显示 i 节点信息，而不是磁盘块。
-t：显示各指定类型的文件系统的磁盘空间使用情况。
-x：列出不是某一指定类型文件系统的磁盘空间使用情况（与 t 选项相反）。
-T：显示文件系统类型。
```

我们先看看使用 df 命令的例子：

```
//列出各文件系统的磁盘空间使用情况
#df
Filesystem           1k-blocks
 Used   Available Use% Mounted on
/dev/hda5               381139
 332921     28540  93% /
/dev/hda1                46636
 6871     37357  16% /boot
/dev/hda3             10041144
 6632528   2898556  70% /home
none                    127372
 0    127372   0% /dev/shm
/dev/hda2             27474876
 24130460   1948772  93% /usr
/dev/hda6               256667
 232729     10686  96% /var
```

第 1 列是代表文件系统对应的设备文件的路径名（一般是硬盘上的分区）；第 2 列给出分区包含的数据块（1024 字节）的数目；第 3，4 列分别表示已用的和可用的数据块数目。

用户也许会感到奇怪，第 3，4 列块数之和不等于第 2 列中的块数。这是因为默认的每个分区都留了少量空间供系统管理员使用的缘故即使遇到普通用户空间已满的情况，管理员仍能登录和留有解决问题所需的工作空间。清单中 Use%列表示普通用户空间使用的百分比，若这一数字达到 100%，分区仍然留有系统管理员使用的空间。

最后，Mounted on 列表示文件系统的安装点。

```
//列出各文件系统的 i 节点使用情况。
#df -ia
Filesystem            Inodes   IUsed
 IFree IUse% Mounted on
/dev/hda5               98392   23919
 74473   25% /
```

```
none                         0        0
 0    -  /proc
/dev/hda1                12048       38
 12010    1% /boot
none                         0        0
 0    -  /dev/pts
/dev/hda3              1275456   355008
 920448   28% /home
none                     31843        1
 31842    1% /dev/shm
/dev/hda2              3489792   133637
 3356155    4% /usr
/dev/hda6                66264     9876
 56388   15% /var
//列出文件系统的类型。
#df -T
Filesystem    Type    1k-blocks
 Used Available Use% Mounted on
/dev/hda5     ext3       381139
 332921     28540  93% /
/dev/hda1     ext3        46636
 6871     37357  16% /boot
/dev/hda3     ext3     10041144
 6632528   2898556  70% /home
none          tmpfs      127372
 0    127372    0% /dev/shm
/dev/hda2     ext3     27474876
 24130460   1948772  93% /usr
/dev/hda6     ext3       256667
 232729     10686  96% /var2
```

2．du

du 的功能是逐级进入指定目录的每一个子目录并显示该目录占用文件系统数据块（1024 字节）的情况。若没有给出指定目录，则对当前目录进行统计。

df 命令的各个选项含义如下：

-s：对每个 Names 参数只给出占用的数据块总数。

-a：递归地显示指定目录中各文件及子目录中各文件占用的数据块数。若既不指定-s，也不指定-a，则只显示 Names 中的每一个目录及其中的各子目录所占的磁盘块数。

-b：以字节为单位列出磁盘空间使用情况（系统默认以 k 字节为单位）。

-k：以 1024 字节为单位列出磁盘空间使用情况。

-c：最后再加上一个总计（系统默认设置）。

-l：计算所有的文件大小，对硬链接文件，则计算多次。

-x：跳过在不同文件系统上的目录不予统计。

下面举例说明 du 命令的使用：

```
//查看/mnt 目录占用磁盘空间的情况
#du -abk /mnt
1       /mnt/cdrom
1       /mnt/floppy
3       /mnt
//列出各目录所占的磁盘空间，但不详细列出每个文件所占的空间
#du
3684    ./log
84      ./libnids-1.17/doc
720     ./libnids-1.17/src
32      ./libnids-1.17/samples
1064    ./libnids-1.17
4944    .
```

第 1 列是以块为单位计的磁盘空间容量，第 2 列列出目录中使用这些空间的目录名称。

清单有时很长，有时只需要一个总数。这时可在 du 命令中加-s 选项来取得总数：

```
#du -s /mnt
3       /mnt
//列出所有文件和目录所占的空间（使用 a 选项），并以字节为单位（使用 b 选项）来计算大小
#du -ab /root/mail
6144    mail/sent-mail
1024    mail/saved-messages
8192    mail
```

3、fdisk

fdisk 可以划分磁盘分区。

```
#fdisk /dev/had    //使用/dev/had作为默认的分区设备
Command (m for help): m  //选择命令选项
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

用户通过提示键入“m”，可以显示 Fdisk 命令各个参数的说明。

在 Linux 分区过程，一般是先通过 p 参数来显示硬盘分区表信息，然后根据信息确定将来的分区。如下所示：

```
Disk /dev/sda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *        41       522   3871665   83  Linux
/dev/hda2             1        40    321268+  82  Linux swap
Partition table entries are not in disk order
Command (m for help):
```

如果想完全改变硬盘的分区格式，就可以通过 d 参数一个一个地删除存在的硬盘分区。删除完毕，就可以通过 n 参数来增加新的分区。当按下“n”后，可以看到如下所示：

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
   p
   Partiton number(1-4):1
   First cylinder(1-1023):1
   Last cylinder or + size or +sizeK or + sizeM(1-1023):+258M
```

这里要选择新建的分区类型，是主分区还是扩展分区；并选择 p 或是 e。然后就是设置分区的大小。

要提醒注意的是，如果硬盘上有扩展分区，就只能增加逻辑分区，不能增加扩展分区。

在增加分区的时候，其类型都是默认的 Linux Native，如果要把其中的某些分区改变为其他类型，例如 Linux Swap 或 FAT32 等，可以通过命令 t 来改变；如果想知道系统所支持的分区类型，键入 l，如下所示：

```
Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap)
```

改变完了分区类型，就可以按下“w”，保存并退出。如果不想保存，那么可以选择“q”直接退出，如下所示：

```
Command (m for help): w
```

原文：
http://os.51cto.com/art/201012/240726.htm

> load average 是 0 的时候都认为他很低，10 的时候就觉得高，20 就不用讲了！但是除了这两种极端的情况之外，那什么时候是这两个值的临界点？当别人问起我这个问题的时候我也不知道如何回答。

系统负载：如何判断 Linux load 的值是否过高

文/色萝卜

接触过和使用过 unix 或 linux 的朋友都知道如何查看 Unix/Linux load 的值，这边我也重复一下查看 load 的方法：

```
[root@aaronw ~]# uptime
13:33:37 up 7 days, 1:52, 1 user, load
 average: 4.15, 2.00, 3.14
[root@aaronw ~]# w
13:35:35 up 1 days, 1:54, 1 user, load
 average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU
 WHAT
root pts/1 192.168.2.2 13:33 0.00s
 0.02s 0.00s w
```

load average 后面三个值代表系统在 1 分钟、5 分钟和 15 分钟的负载情况，都知道数字越高表示系统负载越大，第一直觉就是这个系统不行了。load average 是 0 的时候都认为他很低，10 的时候就觉得高，20 就不用讲了！但是除了这两种极端的情况之外，那什么时候是这两个值的临界点？当别人问起我这个问题的时候，我也不知道如何回答，在我大脑里就根本就没有考虑过。困扰了我很久，我觉得要搞明白他！

先从 linux 的 kernel 的源码开始吧！在 linux 2.6.36 版本中有这样一段代码：

```
/**
* spu_calc_load - update the avenrun
 load estimates.
*
* No locking against reading these
 values from userspace, as for
* the CPU loadavg code.
*/
static void spu_calc_load(void)
{
unsigned long active_tasks; /* fixed-
 point */

active_tasks = count_active_contexts()
 * FIXED_1;
CALC_LOAD(spu_avenrun[0], EXP_1,
 active_tasks);
CALC_LOAD(spu_avenrun[1], EXP_5,
 active_tasks);
CALC_LOAD(spu_avenrun[2], EXP_15,
 active_tasks);
}
```

CALC_LOAD 是这样定义：

```
#define LOAD_FREQ (5*HZ+1) /* 5 sec
 intervals */
#define EXP_1 1884 /* 1/exp(5sec/1min)
 as fixed-point */
#define EXP_5 2014 /* 1/exp(5sec/5min)
 */
#define EXP_15 2037 /*
 1/exp(5sec/15min) */#define
 CALC_LOAD(load,exp,n) \
load *= exp; \
load += n*(FIXED_1-exp); \
load >>= FSHIFT;
```

从这里我们能看到取负载值的最小周期 5 秒。

什么是 load？

load 的就是一定时间内计算机有多少个 active_tasks，也就是说是计算机的任务执行队列的长度，cpu 计算的队列。

load 多少是正常？

既然 load 是 cpu 计算的队列，那就应该和 cpu 个处理方式和 cpu 的个数有关系。所以我个人认为应该按系统识别的 cpu 个数来确定 load 的临界值，系统识别为 8 个 cpu，那么 load 为 8 就是临界点，高于 8 就属于 over load 了。

什么叫系统识别 cpu 个数？

我是这样认为的，这里涉及到 cpu 物理个数和超线程技术的问题。个人认为 4 个物理 cpu 和 2 个双核是不能够等同的，当然这是物理层面的事了！在系统里识别的都是 4 个 CPU.所以应该要以系统识别的为准。毕竟是系统去支配他的使用。

CPU 高不等同于 load 高

在 Unix/Linux 可能经常会遇到 cpu 的使用率为 100%，但是 load 却不高！这是为什么呢？因为几乎所有的任务和会和 CPU 进行交互，但是由于各个设备的使用频率不同，造成了不能同步进行的问题。比如说，当对硬盘进行读写的时候，出现 IO 的等待时候，事实上 cpu 已经被切换到别的进程上了。该任务就处于等待状态，当这样的任务过多，导致队列长度过大，这样就体现到负载过大了，但实际是此时 cpu 被分配去干执行别的任务或空闲，因此 CPU 高不等同于 load 高，load 高也不能于 cpu 高。

原文：
http://selboo.com.cn/post/885/

相关阅读：

查询系统负载信息 Linux uptime 命令详解
http://os.51cto.com/art/201005/200703.htm

十三个强大的 Linux 性能监测工具
http://os.51cto.com/art/201005/201618.htm

> Expect 是由 Don Libes 基于 Tcl 语言开发的，并被广泛应用于交互式操作和自动化测试的场景之中，它尤其适用于需要对多台服务器执行相同操作的环境中，可以大幅度提高系统管理人员的工作效率。

在 linux 下灵活使用 expect 脚本的小窍门

文/Balakrishnan Mariyappan
译/周雪峰

对于喜爱自动化的 Linux 系统管理员而言，一定使用过 expect 这个命令行工具。Expect 是由 Don Libes 基于 Tcl 语言开发的，并被广泛应用于交互式操作和自动化测试的场景之中，它尤其适用于需要对多台服务器执行相同操作的环境中，可以大幅度提高系统管理人员的工作效率。本文是 thegeekstuff.com 最近更新的一篇技术分享文章，其中详细讲述了如何通过不同的命令行选项来执行一个 expect 脚本。具体有什么用，大家可以自由发挥想象力。

本文假设您对 expect 的基本使用方法已经有一定的了解。

如果你是 expect 脚本语言的新手，可以首先从我们的 expect 的“hello world”样例（英文）开始。

1，使用“-c”选项，从命令行执行 expect 脚本

expect 可以让你使用“-c”选项，直接在命令行中执行它，如下所示：

```
$ expect -c 'expect "\n" {send "pressed
 enter\n"}
pressed enter
$
```

如果你执行了上面的脚本，它会等待输入换行符（\n）。按“enter”键以后，它会打印出“pressed enter”这个消息，然后退出。

2，使用“-i”选项交互地执行 expect 脚本

使用“-i”选项，可以通过来自于标准输入的读命令来交互地执行 expect 脚本。如下所示：

```
$ expect -i arg1 arg2 arg3
expect1.1>set argv
arg1 arg2 arg3
expect1.2>
```

正常情况下，当你执行上面的 expect 命令的时候（没有“-i”选项），它会把 arg1 当成脚本的文件名，所以“-i”选项可以让脚本把多个参数当成一个连续的列表。

当你执行带有“-c”选项的 expect 脚本的时候，这个选项是十分有用的。因为默认情况下，expect 是交互地执行的。

3，当执行 expect 脚本的时候，输出调试信息

当你用“-d”选项执行代码的时候，你可以输出诊断的信息。如下所示：

```
$ cat sample.exp
# !/usr/bin/expect -f
```

```
expect "\n";
send "pressed enter";
```

```
$ expect -d sample.exp
expect version 5.43.0
argv[0] = expect  argv[1] = -d  argv[2]
 = sample.exp
set argc 0
set argv0 "sample.exp"
set argv ""
executing commands from command file
 sample.exp
expect: does "" (spawn_id exp0) match
 glob pattern "\n"? no
expect: does "\n" (spawn_id exp0) match
 glob pattern "\n"? yes
expect: set expect_out(0,string) "\n"
expect: set expect_out(spawn_id) "exp0"
expect: set expect_out(buffer) "\n"
send: sending "pressed enter" to { exp0
 pressed enter}
```

4，使用“-D”选项启动 expect 调试器

“-D”选项用于启动调试器，它只接受一个布尔值的参数。这个参数表示提示器必须马上启动，还是只是初始化调试器，以后再使用它。

```
$ expect -D 1 script
```

“-D”选项左边的选项会在调试器启动以前被处理。然后，在调试器启动以后，剩下的命令才会被执行。

```
$ expect -c 'set timeout 10' -D 1 -c
 'set a 1'
1: set a 1
dbg1.0>
```

5，逐行地执行 expect 脚本

通常，expect 会在执行脚本之前，把整个脚本都读入到内存中。“-b”选项可以让 expect 一次只读取脚本中的一行。当你没有写完整个脚本的时候，这是十分有用的，expect 可以开始执行这个不完整的脚本，并且，它可以避免把脚本写入到临时文件中。

```
$ expect -b
```

6，让 expect 不解释命令行参数

你可以使用标识符让 expect 不解释命令行参数。

你可以像下面这样的读入命令行参数：

```
$ cat  print_cmdline_args.exp
#!/usr/bin/expect
puts 'argv0 : [lindex $argv 0]';
puts 'argv1 : [lindex $argv 1]';
```

当执行上面的脚本的时候，会跳过命令行选项，它们会被当成参数（而不是 expect 选项），如下所示：

```
$ expect print_cmdline_args.exp -d -c
argv0 : -d
argv1 : -c
```

原文：

http://www.thegeekstuff.com/2010/12/5-expect-script-command-line-argument-examples/

译文：

http://os.51cto.com/art/201012/240260.htm

相关阅读：

Hotmail 系统故障：都是自动化脚本惹的祸
http://os.51cto.com/art/201101/242496.htm

使用 Perl 脚本彻底实现系统管理自动化
http://os.51cto.com/art/201012/240483.htm

不看后悔的 Linux 生产服务器 Shell 脚本分享
http://os.51cto.com/art/201010/229129.htm

我在配置某机房 Linux 服务器的 iptables 时，不小心设置了某一项错误参数，结果锁定了 SSH 会话，导致我们经理及另一技术员连不上服务器。

系统管理员易犯错误及解决方法汇总

文/抚琴煮酒

本文分享的都是系统管理员在工作的时候容易犯的错误，经抚琴煮酒整理并提供解决方法，希望可以给大家一些指导，避免在工作中出现此类问题。

一、安装 FreeBSD 后无法重启

问题描述：

装惯了 Linux 的人肯定知道一般会有个 boot 分区，可是在 bsd 就不那么容易了。在安装 FreeBSD 8.1 的时候遇到了问题，查阅了 chinaunix 上面，正好也有相关问题整理，特摘录如下：

我要求 FreeBSD 分区：

```
2G For /
4G For swap
10G For /root
256M For /boot
其余 for /usr
```

安装正常，结果安装重启后便出现杯具了：

```
>> FreeBSD/i386 BOOT
Default: 0:da(0,a)/boot/kernel/kernel
boot:
```

原因：

通过网上查资料，了解到手动引导的全过程，发现了问题所在：

由于独立分区/boot 造成了 FreeBSD 引导过程中无法正确找到内核引导的位置。

解决方法：

通过

```
boot: 0:da(0,e)/loader
```

可以解决引导问题，然后进入 loader 界面

*这个引导盘符根据 da0s1x 的 x 得来，因此你安装系统的时候/boot 所在分区区号，才是真正的 x 字母，如果不知道就从往后试试

同样由于默认 kernel 位置是/boot/kernel 所以依然需要手动加载

```
ok load kernel/kernel
```

获得 kernel 信息后

```
ok boot
```

这样就可以正常引导了。

但是这样还没有彻底解决问题，随后还需要在磁盘挂载的时候输入

```
mount root>ufs:/dev/da0s1a
```

才能进入系统，而且每次重启都手动一次。所以其实问题没有彻底解决。

所以，为了避免以上的/boot 问题，目前我装机一般规范化操作，一般只分三个区，避免独立分区/boot，也希望玩 Linux 的朋友们重视下这个问题。

```
2048M For /
4096M For swap
其余的均 For /usr
```

二、root 密码更改后无法远程登录

问题描述：

系统总监嫌托管的新 Linux 服务器 root 密码过于简单，吩咐公司的系统管理员将密码改复杂些，急躁的系统管理员用 passwd root 密码改掉后赶车回公司，杯具的发现密码设置得过于复杂，密码给忘了。由于机器是新装，没有配置具有 sudo 权限的用户，自己远程都进不了 root 了。这种问题就只有百分百靠系统管理员负责了。

解决方法：

这个问题只要养成良好的习惯就可以预防，就是大家更改完 root 密码后，别急着退出，可以用 ctrl+shift+F2 或 F3-F8 尝试用另一个终端进去下，如果当时就忘了，马上切换到 F1 更换。抚琴煮酒经常犯这种错误，呵呵，希望此法对大家有效。

三、锁定了 SSH 会话

问题描述：

我在配置某机房 Linux 服务器的 iptables 时，不小心设置了某一项错误参数，结果锁定了 SSH 会话，导致我们经理及另一技术员连不上服务器。

解决方法：

下面介绍的这个方法及其有用，强烈推荐给大家：为了预防此类问题出现，可以配置一计划任务 crontab，每 5 分钟运行一次，即

```
*/5 * * * *  root /bin/sh
 /root/firestop.sh
```

firestop.sh 内容为：

```
service iptables stop
```

这样即使你的脚本存在错误设置（或丢失的）规则时，也不至于将你锁在计算机外而无法返回与计算机的连接。这样你就可以放心大胆的调试你的脚本啦。这都是生产环境下逼出来的，呵呵。

四、移走硬盘造成 Emergency 模式

问题描述：

同事在处理 Linux 服务器时，移走了一块硬盘，然后就直接启动红帽 RHEL5，发现进了 Emergency 模式，焦急中他连忙跑过来找我；我第一句就是问他：你改动了硬件没，他说他移走了硬盘后就直接启动了，不是跟 windows 2003 一样嘛，有什么问题？我都无语了……

解决办法：

耐心跟他讲解了 Linux 下/etc/fatab 的作用及语法，告诉他可以在 Emergency 模式下输入 root 密码进入此模式，然后用

```
mount -o remount,rw /
```

将 / 分区设置成可读写，编辑/etc/fatab，将移除的硬盘用#号屏蔽掉后重启服务器，故障解除。

五、sudoer 文件损坏，无法进入 root

问题描述：

同事远程处理一台机房的 FreeBSD 8.1 机器，想加一个具有 sudo 用户的特殊用户，所以编辑了/etc/sudoer 文件，却不小心多加了一个.，然后直接保存退出了。结果杯具发生了：由于 sudoer 文件损坏，所有具有 sudo 权限的用户均不能切换到 root 模式下工作，而 FreeBSD8.1 与 Linux 不同，它默认是不允许 root 远程连接的。

解决方法：

这时只有请专人到机房去处理问题了……

六、root 密码被更改

问题描述：

一个开发小组都是用内部机房的 Linux/FreeBSD 机器，大家都知道 root 的密码；不知哪个兄弟是搞着好玩还是怎么的，偷偷的改了 root 密码却不通知大家，结果大家都用不了 root 密码，杯具了。

解决办法：

此时处理办法有 2 种，一种就是大家都知道的单用户模式修改 root，其实另一个办法也蛮简单的，系统管理员应该多配置一个具有 sudo 权限的用户，遇到此种情况时可以用 sudo 权限来修改 root 的密码，至少免得跑到机房去。毕竟有时候，机房未必在市内或在国内的。

七、库文件丢失导致 root 无法登陆

问题描述：

我们的 jail 母机 192.168.21.36，因为 root 的 shell 设置成的 bash，而其依赖的库文件 libintl.so.8 发生丢失，导致了 root 不能登陆。具体报障如下：

```
/libexec/ld-elf.so.1: Shared object
 "libintl.so.8" not found, required by
 "bash"
Connection to 192.168.21.36 closed.
```

解决方法：

①用单用户模式进入系统；

②扫描磁盘(此步非做不可，而且是安全的)

```
fsck -y
```

③将文件系统重新挂载

```
mount -a
```

④将 root 的默认 shell 切换到 sh

```
chsh -s sh
```

重启后一切正常。

八、忘记以 su 模式进入编辑器

问题描述：

普通用户用 vi 编辑 nginx.conf 等配置文件，保存的时候会提示：没有 Root Permission

解决办法：

其实只要保存时加上：

```
:w !sudo tee %
```

就可以了。

“:w !sudo tee %”这条命令的含义是把当前编辑的文件的内容当做标准输入输入到命令 sudo tee 文件名里去。也就是 sudo 保存为当前文件名，相当管用的命令，尤其适用于 FreeBSD 和 Debian 系统(我经常忘了自己原来不是 root 了)，相当 very nice.

原文：
http://os.51cto.com/art/201101/241510.htm

相关阅读：

FreeBSD 与 Linux 十个本质上的区别
http://os.51cto.com/art/201012/236797.htm

FreeBSD 系统管理员都应该知道的那点秘密
http://os.51cto.com/art/201012/236729.htm

如果你试过不小心 cat 了某个二进制文件，很可能整个终端就傻掉了，可能不会换行，没法回显，大堆乱码之类的，这时候敲入 reset 回车，不管命令有没有显示，就能恢复正常了。

系列连载：最牛 B 的 Linux Shell 命令（2）

文/Peteris Krumins
编译/BOY PT

1.用你最喜欢的编辑器来敲命令

```
command <CTRL-x CTRL-e>
```

在已经敲完的命令后按<CTRL-x CTRL-e>，会打开一个你指定的编辑器（比如 vim，通过环境变量$EDITOR 指定），里面就是你刚输入的命令，然后爱怎么编辑就怎么编辑吧，特别是那些参数异常复杂的程序，比如 mencoder/ffmpeg，一个命令动辄 3、4 行的，要修改其中的参数，这个方法最合适不过了，保存退出后自动执行这个程序。

实际上这是 readline 库的功能，在默认情况下，bash 使用的是 emacs 模式的命令行操作方式，<CTRL-x CTRL-e>是调用这个功能的一个绑定。如果你习惯使用 vi 模式，按<ESC v>可以实现同样功能。

如果你喜欢别的编辑器，可以在~/.bashrc 里面放上比如 export EDITOR=nano 的命令。

另外一个修改命令的方法是使用 fc 命令（Fix Command），在编辑器里面打开上一句命令。我们的第一辑连载提过一个^foo^bar^命令可以用 fc 来实现：fc -s foo=bar。

2.清空或创建一个文件

```
> file.txt
```

>在 shell 里面是标准输出重定向符，即把（前部个命令的）命令行输出转往一个文件内但这里没有“前部命令”，输出为空，于是就覆盖（或创建）成一个空文件了。

有些脚本的写法是:>file.txt，因为:是 bash 默认存在的空函数。

单纯创建文件也可以用 $touch file.txt，touch 本来是用作修改文件的时间戳，但如果文件不存在，就自动创建了。

3.用 ssh 创建端口转发通道

```
ssh -N -L2001:remotehost:80
 user@somemachine
```

这个命令在本机打开了 2001 端口，对本机 2001 端口的请求通过 somemachine 作为跳板转到 remotehost 的 80 端口上。

实现效果跟术语反向代理是相似的，实际上就是端口转发，注意上面的描述涉及了 3 台主机，但当然 somemachine 可以变成 localhost。

这个命令比较抽象，但有时候是很有用的，比如因为众所周知的原因国内的 IP 的 80 端口无法使用，又或者公司的防火墙只给外网开了 ssh 端口，需要访问内部服务器一个 web 应用，

以及需要访问某些限定了来源 IP 的服务，就可以用上这个方法了。

举一个具体例子，运行：

```
ssh -f -N -L
 0.0.0.0:443:twitter.com:443
 shell.cjb.net
ssh -f -N -L 0.0.0.0:80:twitter.com:80
 shell.cjb.net
```

然后在/etc/hosts 里面添加 127.0.0.1 twitter.com，好吧剩下的你懂的。

当然通常做这个功能的反向代理，应该要用 squid、nginx 之类，ssh 就算是轻量级的尝试吧！

4.重置终端

```
reset
```

如果你试过不小心 cat 了某个二进制文件，很可能整个终端就傻掉了，可能不会换行，没法回显，大堆乱码之类的，这时候敲入 reset 回车，不管命令有没有显示，就能恢复正常了。

实际上 reset 命令只是输出了一些特殊字符，我们看 BusyBox 里面最简单的 reset 程序的实现：

```
printf("\033c\033(K\033[J\033[0m\033[?
 25h");
```

输出的这些字符对 Shell 是有特殊意义的：

```
\033c: “ESC c” – 发送重置命令;
\033(K: “ESC ( K” – 重载终端的字符映射;
\033[J: “ESC [ J” – 清空终端内容;
\033[0m: “ESC [ 0 m” – 初始化字符显示属
 性;
\033[?25h: “ESC [ ? 25 h” – 让光标可见;
```

其中字符显示属性经常用来设定打印字符的颜色等，可参考这个博文。

5.在午夜的时候执行某命令

```
echo cmd | at midnight
```

说的就是 at 这个组件，通常跟 cron 相提并论，不过 at 主要用于定时一次性任务，而 cron 定时周期性任务。

at 的参数比较人性化，跟英语语法一样，可以 tomorrow，next week 之类的，详细的查看手册 man at。

6.映射一个内存目录

```
mount -t tmpfs -o size=1024m tmpfs
 /mnt/ram
```

这个命令开了一块 1G 内存来当目录用。不过放心，如果里面没文件，是不会占用内存的用多少占多少。

不过一般来说没必要手动挂载，因为多数发行版都会在 fstab 内预留了一个内存目录，挂载在/dev/shm，直接使用即可；

最常见的用途是用内存空间来放 Firefox 的配置，可以让慢吞吞的 FF 快很多，参见 Shellex 的博文：用 tmpfs 让 Firefox 在内存中飞驰，以及后来的改进：用 tmpfs 让 Firefox 在内存中飞驰 II，其中提到的脚本来自 speeding up firefox with tmpfs and automatic rsync。

那个破烂 LinuxQQ 也可以用这个方法，减少因为大量磁盘 IO 导致的问题。

原文（本文为节选）：
http://www.catonmat.net/blog/top-ten-one-liners-from-commandlinefu-explained

译文：
http://www.isspy.com/most_useful_linux_commands_2/

招募启事

《Linux 运维趋势》的建设需要您的加入！

您可以通过如下方式参与我们杂志的建设：

1、推荐文章

无论是您在互联网上看到的好文章，还是您自己总结/整理的资料；无论是英文还是中文；无论是入门的还是高端的，都欢迎推荐！推荐方式包括：

a）在技术圈中分享：http://g.51cto.com/linuxops

b）在邮件群中分享：linuxops-cn@googlegroups.com

c）发邮件给编辑：yangsai@51cto.com

2、投稿

如果您认为自己在 Linux 方面具有专家级别的能力，并且有与大家分享您技术经验的热诚，同时也有兴趣挣点稿费花花，那么欢迎您的投稿！

如果您在 IT 技术方面的翻译有很高的能力，能够快速、高质量的完成译文，并且也经常浏览到一些 Linux 方面的优秀外文，那么也欢迎您的投稿！

投稿信箱：yangsai@51cto.com

3、推广与意见

如果您喜欢我们的杂志，认为这本杂志对于您的工作有所帮助，请向您的 Linux 好友、同事们推荐它！

如果您觉得这份杂志还有什么地方需要改进或补充，也希望您能够提出您的宝贵意见！

联系人：yangsai@51cto.com

下期预告

下期主题为：内网与外网的运维管理。敬请期待！

本刊为月刊，预定每月发布日期为：

每个月的第二个星期五

您可以通过如下方式检查是否有新刊发布：

1、加入电子邮件群组：linuxops-cn@googlegroups.com

获得邮件提醒

2、经常光顾 51CTO Linux 频道：http://os.51cto.com/linux/