Linux 运维趋势

2010 年 11 月 第二期

本期主题：可用性 | 关键字：集群，负载均衡，高可用，LVS

内容目录

Linux 运维趋势

杂志策划：51CTO 系统频道
本期主编：杨赛
Logo 制作：高鹏飞
封面制作：徐泽琼
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
投稿信箱：
yangsai@51cto.com

专访黄琨：运维工作中最大的挑战是什么？

采访/杨赛

人物简介：

黄琨，曾任知名外企 SP 公司运维经理，多年网络应用架构设计及运维管理经验。涉及技术包括：Linux/SUN 小型机/Windows 运维、互联网应用平台架构设计、Oracle/Mysql 数据库、开源分布式集群架构设计及调优、网络及安全设备架构及管理。现在任职于荣新 IT 培训中心，担任 IT 运维外包项目总监、企业人才外包项目总监。

运维是一个全面的工作，可以接触到各种领域的技术和人。运维是一种实操类的技能，其经验积累很大程度上来自于真实项目的积累。因此，对于运维领域的新人而言，如果他们工作的环境并没有提供一个良好的平台，就经常容易陷入困惑。

本期《趋势》当中，我们邀请到了黄琨老师到场，谈谈他自己的运维成长经历及挑战。

51CTO：您是什么时候开始做的运维？对工作一开始的几年有哪些深刻的记忆？

黄琨：我 2002 年之前主要是从事网络集成的项目的设计实施工作，之后进入石景山区信息中心负责全区各行政单位的网络、中心 IDC 的维护工作。那个时候的工作有苦有乐，最重要的是能够学到知识——有一个好的平台对我来说非常重要。当时正处于互联网业务发展的初期，有些企业的业务平台也陆续在中心 IDC 上线，为我的技术学习提供了良好的氛围和实验条件。

记忆最深刻的就是有一次中心机房要从教委迁移到区政府信息中心。那次迁移工作量相当大，包括：网络设备、服务器、新老应用割接、新设备上线、对网络和应用层做了链路冗余以及高可用等，让我有机会一次性的把之前做过的实验用到了真实的工作中。当时网络设备用的是 CISCO 的 6500 系列两台做的冗余，汇聚层和接入层也都是 Cisco 的产品 35 系列和 25 系列；服务器 400 台左右，安全方面有天融信的防火墙、还有 NIDS；规模大任务重。当时中心系统组负责人，也是现在我的好朋友张琦老师对我的帮助非常大，从原中心业务系统整体梳理、备份、链路及服务割接工作的计划设计、各别服务系统更新、重要服务应用高可用的设计、双因素认证系统等等工作帮助我整理的井井有条。工作非常顺利，当时还获得中心同事的表扬，至今记忆犹新。

51CTO：能介绍一下您现在的工作情况么？您的职责包括哪些方面？

黄琨：现在和白璐、杨晨等开源和网络方面的精英一起开办了一家专门培养运维人才的培训机构——荣新 IT 培训中心，经过这几年的努力，培训中心的规模已经扩大了 5 倍。

我现在任 CTO 的职位，一方面负责企业项目及运维外包服务的工作，为企业提供优良的技术服务之外将前沿技术引入到培训中来；另一方面负责培训学员到企事业单位的人才输送工作。

51CTO：能否大致的描述一下您每天的工作内容？

黄琨：本人现在主要负责：

1、IT 运维外包项目计划、项目方案设计监督、估算、管理、跟踪项目进度。

2、企业人才输送，组织技术指导，收集问题回馈，协助教学部形成教材。

3、Linux 等相关运维人才市场的动向监控。

运维一线现在已有 600 以上荣新学员，我也描述一下他们刚入行时候的工作内容吧：

1、快速分析整理公司业务及平台设计逻辑架构，缓存、应用、数据库、网络设备及其他设备的运作原理。

2、平台各层面监控，避免监控死角，实时了解平台各层应用的运转情况；处理突发问题，迅速做出问题响应，做好问题处理分析报告为后续自动化运维设计作补充。

3、平台代码更新，根据平台规模设计部署更新源资源下载服务、补丁批量更新机制。

4、配合运维经理设计实现运维支撑系统，包括系统监控、报警、管理功能；实现数据图形报表、整合手机短信、邮件、声音报警功能，根据监控排障反映上来的问题不断完善自动化运维机制。

5、配合运维经理对平台架构进行分析，不断提升整体应用的可靠性与健壮性、提高性能及安全性。

51CTO：您觉得您目前的运维生涯当中最大的挑战是什么？

黄琨：我认为挑战主要分为技术和沟通两方面，当然由于我现在从事培训和运维外包工作，所以另一个转型的挑战：

1、技术方面的挑战是运维工作的职责体现出来的，简单的说产品从需求收集、开发及网络系统架构设计、开发测试阶段、产品上线联调、问题反馈、正式商用后运维阶段等等，因篇幅有限我无法说得太详细，这些工作运维都需要跟下来。前几项工作中如果没有搞清楚产品的技术细节（比如：软/硬件资源评估确定硬件采购需求、平台性能的评估、服务性能调优安全加固、根据应用对服务器系统层的优化等等），将直接影响最后运维工作的正常开展。

从我看来，由于生产平台是企业的命脉所以运维工作上没有最大的挑战只有不断地挑战，例如平台上线后如果出现了瓶颈问题那么就需要快速锁定问题排查瓶颈，在最快的时间解决，尤其对于做互联网应用的企业，用户体验最重要，三天两头出问题，用户就会流失，企业利益就会受损。

2、沟通方面的挑战。一个合格的运维工程师不但工作要做好，与本职工作职责在一条链上的部门同事之间的沟通也至关重要，直接制约工作的效率与结果。比如平台运行中遭遇问题，经过排查也锁定了，但是之前与同事沟通不畅造成问题解决滞后，这个影响很大。

当然很多企业非常重视产品上线后的问题响应，从人力上设定了绩效，从技术上利用内部工单来配合解决，效果也是非常显著。不过制度始终是需要去遵守的，是死的，工作的人是活的，所以人与人之间的有效沟通也是非常重要的一项必修课，这对于运维工作人员来说是挑战，处理不好经常出现由于部门间工作性质不同带来的信息孤岛和沟通鸿沟。

3、最后一点，我希望将“Linux 高效运维”这项本领和更多的人分享，如何把最难理解的知识通过最平凡易懂的方式教授给学生，这才是当前工作的重中之重。

51CTO：您现在关注哪些技术领域呢？

黄琨：就运维所关注的技术领域来说，我只想用一句“多而杂”来形容。因为运维是保证企业业务平台稳定运行的基石：开发，测试，整个平台架构中的缓存、应用、中间件、数据库、网络方面数据传输效率、平台监控报警、硬件层面等等方面，都需要了解，并且深入。

根据我现在的工作性质，要不断关注最新的技术，最重要的就是如何能够提高运维团队的工作效率，以及组织学习兴趣小组总结运维工作中的技术难点，达到不断提高的目的。毕竟运维技术更新很快，但是学习资源相对比开发领域来说是有差距的。

51CTO：最后，能否大致谈谈您对于未来三年的个人发展计划？

黄琨：其实技术和业务是分不开的。关注互联网行业/3G 融合之后的杀手级业务相关技术是我的主攻目标。在这个范围内提高自己的技术，能够为未来的发展提供一个很好的路线引导。

未来 3 年我将继续做好 Linux 及其相关的开源运维培训工作，在 IT 培训领域做出一番成绩。

原文（本文有删节）：
http://os.51cto.com/art/201011/233317.htm

我认为 MySQL 完全被 NoSQL/数据库社区低估了，MySQL 的历史悠久，我的前同事们也在不断改进它，从 NDBAPI 可以看出 MySQL 有成为 NoSQL 的潜力。

NoSQL 小故事：单服务器如何应付每秒 75 万次查询

文/Yoshinori Matsunobu
译/黄永兵

大多数大规模 Web 应用程序都使用 MySQL + Memcached 架构，其中许多应用也同时使用了 NoSQL 数据库。也有一些人全部放弃 MySQL，转投 NoSQL 的怀抱。NoSQL 数据库在处理一些简单访问模式（如主键查找）时比 MySQL 的表现更好，而大多数 Web 应用程序的查询都很简单，因此这看上去是一个很合理的决定。

和许多其它大规模网站一样，我们 DeNA 多年来都存在类似的问题，但最终我们全部使用了 MySQL。我们一如既往地使用 Memcached 作为前端缓存，但没有使用 Memcached 缓存数据行；我们也没有使用 NoSQL，因为我们从 MySQL 获得的性能比其它 NoSQL 产品更好。在我们的基准测试中，我们在一台普通的 MySQL/InnoDB 5.1 服务器上获得了 750000+QPS 的成绩，在生产环境中的性能更优秀。

SQL 主键查询真得能很快吗？

```
[matsunobu@host ~]$ vmstat 1
 r  b  swpd   free   buff  cache      in     cs us sy id wa st
23  0     0 963004 224216 29937708 58242 163470 59 28 12  0  0
24  0     0 963312 224216 29937708 57725 164855 59 28 13  0  0
19  0     0 963232 224216 29937708 58127 164196 60 28 12  0  0
16  0     0 963260 224216 29937708 58021 165275 60 28 12  0  0
20  0     0 963308 224216 29937708 57865 165041 60 28 12  0  0
```

vmstat 输出

每秒你可以运行多少次主键查询？DeNA 的应用程序需要执行大量的主键查询，如通过用户 id 获取用户信息，通过日记 id 获取日志信息等。Memcached 和 NoSQL 都能很好地适应这种需求。当你运行简单的多线程“memcached get”基准测试时，每秒大约可以执行 400000+次 get 操作；即使 Memcached 客户端位于远程服务器上，当我使用最新的 libmemcached 和 memcached 测试时，在一台 2.5GHz 8 核 Nehalem 处理器，四个 Broadcom 千兆以太网卡的服务器上，测试成绩是每秒执行 420000 次 get 操作。

MySQL 执行主键查询需要多长时间？通过基准测试很容易找到答案，只需要从 sysbench，super-smack 和 mysqlslap 等运行并行查询即可。

每秒有 100000+次查询似乎还不错，但却远远低于 Memcached 的结果，MySQL 实际上做了些什么？从 vmstat 输出可以看出，%user 和%system 都很高。

在 SQL 解析阶段调用了 MYSQLparse()和 MYSQLlex()，在查询优化阶段调用了 make_join_statistics()和 JOIN::optimize()，这些都是 SQL 开销。很明显，性能下降主要是由 SQL 层，而不是 InnoDB（存储）层造成的。MySQL 做了很多 Memcached/NoSQL 不需要做的事情，如：

- 解析 SQL 语句
- 打开，锁住表
- 创建 SQL 执行计划
- 解锁
- 关闭表

MySQL 也做了许多并发控制，例如，在发送/接收网络数据包时多次调用了 fcntl()，全局互斥，如 LOCK_open，LOCK_thread_count 很频繁地创建/释放，这就是为什么 oprofile 输出中 my_pthread_fastmutex_lock()排名第二，%system 也不小的原因。

MySQL 开发团队和外部社区都知道并发问题，有些问题在 5.5 中已经得到解决。我很高兴地看到，大量的修复工作已经完成，但同样重要的是%user 也达到了 60%，互斥竞争导致%system 上升，而不是%user 上升。虽然 MySQL 中的所有互斥问题都得到了解决，但不要指望每秒超过 30 万次查询。

你可能听说过 HANDLER 语句，遗憾的是，HANDLER 语句对提高吞吐量并不会有太多帮助，因为查询解析，打开/关闭表等操作仍然是需要的。

CPU 效率对内存中的工作负载非常重要

如果内存中没有合适的活动数据，SQL 开销相对来说可以忽略不计，很简单，因为磁盘 I/O 成本是非常高的，在这种情况下，我们不需要太关心 SQL 成本。

但在我们的热点 MySQL 服务器上，几乎所有数据都装入到内存中了，它们完全变成 CPU 限制，分析结果和上面类似：SQL 层消耗了大部分资源。我们需要执行大量的主键查询（如 SELECT x FROM t WHERE id=?）或限制范围的扫描，即使 70-80%的查询是对相同表的简单主键查询（不同的只是 Where 子句中的值），每次 MySQL 都要解析/打开/锁住/解锁/关闭，这对我们来说效率是很低的。

你听说过 NDBAPI 吗？

在 MySQL 的 SQL 层有什么好的解决办法可以减少 CPU 资源争用吗？如果你使用的是 MySQL 集群，NDBAPI 可能是最好的解决方案。我在 MySQL/Sun/Oracle 担任顾问时，我看到许多客户对 SQL 节点+NDB 的性能表现很失望；当使用 NDBAPI 客户端性能提高 N 倍后，他们高兴极了。你可以在 MySQL 集群中同时使用 NDBAPI 和 SQL，建议频繁访问模式使用 NDBAPI，即席查询或非频繁模式使用 SQL+MySQL+NDB。

这正是我们想要的，我们希望更快速地访问 API，我们也希望对即席查询或复杂查询使用 SQL，但 DeNA 使用的是 InnoDB，和许多其它 Web 服务一样，切换到 NDB 不是小事，嵌入式 InnoDB 不支持 SQL 也不没有网络接口，因此它不适合我们。

HandlerSocket 插件，一个懂 NoSQL 网络协议的 MySQL 插件

我们认为最好的办法是在 MySQL 内部实现一个 NoSQL 网络服务器，也就是说，写一个网络服务器作为 MySQL 守护进程插件监听指定端口，接受 NoSQL 协议/API，然后使用 MySQL 内部存储引擎 API 直接访问 InnoDB。这个方法和 NDBAPI 类似，但它可以和 InnoDB 交互，这个概念最初是由 Kazuho Oku 去年在 Cybozu 实验室提出并创建了原型，他编写了使用 memcached 协议的 MyCached UDF，我的同事 Akira Higuchi 实现了另一个插件：HandlerSocket，下图显示了 HandlerSocket 可以做的事情。

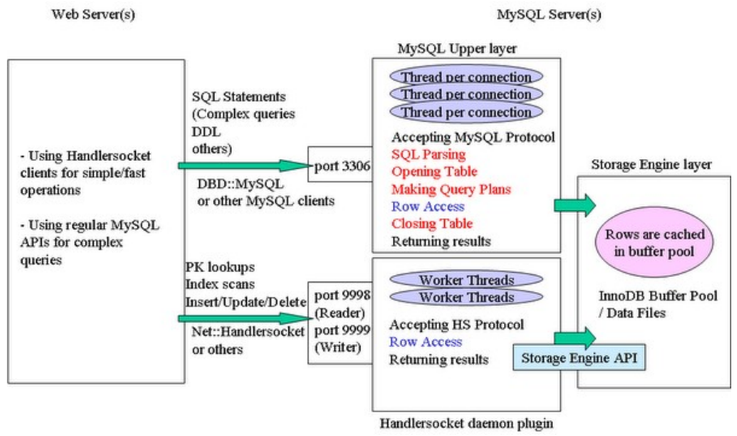

图 Handlersocket 是什么？

Handlersocket 是一个 MySQL 守护进程插件，它让应用程序可以将 MySQL 当 NoSQL 使。Handlersocket 的主要目的是与存储引擎，如 InnoDB 交互，而不需要 SQL 相关的开销。访问 MySQL 表时，Handlersocket 仍然需要打开和关

闭表，但不是每次访问都要求打开和关闭，因此减少了互斥争夺，极大地提高了系统性能，当流量变小时，Handlersocket 会关闭表，因此它永远不会阻止管理命令。

HandlerSocket 的特点和优势

HandlerSocket 有许多特点和优势，其中有一些是对我们真正有益的。

1、支持大量的查询模式

HandlerSocket 支持主键/唯一性查询，非唯一性索引查询，范围扫描，LIMIT 和 INSERT/UPDATE/DELETE，以及 multi_get 操作（类似 in(1,2,3)）。不支持未使用任何索引的操作。

2、可以处理大量并发连接

HandlerSocket 连接是轻量级的，因为 HandlerSocket 采用了 epoll()和工作线程/线程池架构，MySQL 内部线程的数量是有限的（可以由 my.cnf 中的 handlersocket_threads 参数控制），因此你可以建立上千或上万的网络连接，稳定性不会受到任何影响（消耗太多的内存，造成巨大的互斥竞争等，如 bug#26590，bug#33948，bug#49169）。

3、极好的性能

HandlerSocket 相对于其它 NoSQL 阵容性能表现一点也不逊色，事实上，我还没有看到哪个 NoSQL 产品在一台普通服务器上可以执行 750000+次查询。

HandlerSocket 不仅消除了 SQL 相关的函数调用，也优化了网络/并发相关的问题。

4、没有重复的缓存

当你使用 Memcached 缓存 MySQL/InnoDB 记录时，在 Memcached 和 InnoDB 缓冲池中均缓存了这些记录，因此效率非常低（内存仍然很贵！），由于 HandlerSocket 插件访问 InnoDB 存储引擎，记录可以缓存在 InnoDB 缓冲池中，这样其它 SQL 语句就可以重复使用它。

5、没有数据不一致的现象

由于数据只存储在一个地方（InnoDB 内），不需要在 Memcached 和 MySQL 之间检查数据一致性。

6、崩溃安全

后端存储是 InnoDB，它是事务性和崩溃安全的，即使你设置 innodb-flush-log-at-trx-commit!=1，在服务器崩溃时也只会丢掉<1 秒内的数据。

7、可以从 MySQL 客户端使用 SQL

在许多情况下，人们仍然希望使用 SQL（如生产摘要报告），这就是为什么我们不能使用嵌入式 InnoDB 的原因，大多数 NoSQL 产品都不支持 SQL 接口，HandlerSocket 仅仅是一个 MySQL 插件，你可以从 MySQL 客户端发送 SQL 语句，当你需要高吞吐量时最好使用 HandlerSocket 协议。

8、MySQL 所有操作都将受益

因为 HandlerSocket 在 MySQL 内部运行，因此所有 MySQL 操作，如 SQL，在线备份，复制，通过 Nagios/EnterpriseMonitor 监控等都是支持的，HandlerSocket 获得可以通过普通的 MySQL 命令监控，如 SHOW GLOBAL STAUTS，SHOW ENGINE INNODB STATUS 和 SHOW PROCESSLIST 等。

9、不需要修改/重建 MySQL

因为 HandlerSocket 是一个插件，它支持 MySQL 社区版和企业服务器版，无需对 MySQL 做出任何修改就可以使用。

10、独立于存储引擎

虽然我们只测试了 5.1 和 5.5 InnoDB 插件，但 HandlerSocket 可以和任何存储引擎交互。

本文为节选，全文阅读请参考：

英文原文：
http://yoshinorimatsunobu.blogspot.com/2010/10/using-mysql-as-nosql-story-for.html

译文全文：
http://database.51cto.com/art/201010/231410.htm

在 2010 年 10 月-11 月之间，发生了下面这些事……

八卦，趣闻与数字 2010.10 - 2010.11

收集整理/51CTO 系统频道

【Ubuntu & Amazon】继 Ubuntu 10.04 中尝到了云计算的甜头以后，Ubuntu 10.10 中更是加大了云服务，也加大了与美国云计算厂商亚马逊的合作。据 Canonical 透露，Ubuntu 现在是 Amazon EC2 上最流行的服务器操作系统。

http://os.51cto.com/art/201010/230329.htm

【OOo】不管是在 Oracle 还是在文档基金会的管理下，OpenOffice.org 的大部分代码都会得到保留，不是因为它们写得好，而是因为它们是免费的。

http://os.51cto.com/art/201010/230444.htm

【系统管理员】一个做什么事情都亲自动手的系统管理员，他浪费的不仅仅是他自己的时间，也是你的时间。

http://os.51cto.com/art/201010/230920.htm

【服务器】在今后的五年内，79.4%的受访者表示有计划增加 Linux 服务器，而对于其他操作系统，将近 40%表示会增加微软的 Windows 服务器，22%表示会增加 Unix 服务器。

http://os.51cto.com/art/201010/230533.htm

【红帽 CEO】人们说他们对云感兴趣，但是他们真正赞成的是对现有的 IT 商业模式的批判。

http://os.51cto.com/art/201010/231041.htm

【安全】在安全方面，Unix 和 Windows 有很多相同点，而微软似乎更想将 Unix 的安全特性全部纳入其 Windows 系统。但微软有一点没有做好的是，其安全性设计都是驾临于操作系统之上，而不是从系统架构出发考虑的。

http://os.51cto.com/art/201010/231075.htm

【Linux 内核】最新的 Linux 内核很值得注意，因为它没有带来体积的增长，却包含了成百上千个进展，将会对通常都察觉不到 Linux 内核升级所带来的改变的最终用户带来显著的体验提升。

http://os.51cto.com/art/201010/231163.htm

【Unity & Gnome】Mark Shuttleworth 宣布 Unity 将取代 GNOME Shell 成为 Ubuntu 11.04 的默认操作界面。

http://os.51cto.com/art/201010/231232.htm

【Linux Mint】基于 Ubuntu 10.10 的 Linux Mint 的下一个版本，Linux Mint 10 可能会在可用性方面处于领先地位。

http://os.51cto.com/art/201010/231412.htm

【漏洞】Linux 操作系统最近被找出两个漏洞，一个是与甲骨文所贡献的 RDS 协定有关，另一个则与 GNU C 的数据库有关，均可能让入侵的黑客取得 Linux 系统的最高管理人员权限。

http://os.51cto.com/art/201010/231420.htm

【伦敦证交所】伦敦证券交易所宣布，一个基于 Linux 的新交易系统创造了 126 微秒交易速度的世界纪录。

http://os.51cto.com/art/201010/231570.htm

【开源】当你完成一次呼吸时，世界上也许就多了一个开源项目。

http://os.51cto.com/art/201010/231680.htm

【国家的 Linux】俄罗斯政府已经证实，他们正在开发一个全国性的软件系统，该平台基于 Linux 代码构建，这一举动背后的理由似乎是试图改善安全性，降低成本，减少该国对微软的依赖。

http://os.51cto.com/art/201010/231817.htm

【中国雅虎】“雅虎是我见过的最尊重 Linux 的一家公司”。

http://os.51cto.com/art/201011/232354.htm

【Fedora 14】Fedora 14 的开发代号 Laughlin 取自诺贝尔物理学奖获得者 Robert B. Laughlin，这位物理学家提出了“整体大于部分之和”的概念，代表 Fedora 不仅仅是开源软件的简单组合。

http://os.51cto.com/art/201011/232528.htm

【Ubuntu 11.04】Ubuntu 11.04 的第一个 Alpha 将于今年 11 月 4 日发布，总共会发布 5 个 Alpha 版， Beta 版于 2011 年 3 月 24 日发布，最终正式版发布时间为 2011 年 4 月 28 日。

http://os.51cto.com/art/201011/232974.htm

本期专题：可用性

所谓网站可用性(availability)也即网站正常运行时间的百分比。
这是每个运营团队最主要的 KPI 。
对于 Web 站点来说，传统的那个 24x7 的说法已经不是很适用了。
现在业界更倾向用 N 个 9 来量化可用性。
最常说的就是类似 "4 个 9(也就是 99.99%)" 的可用性。
- via dbanotes

现阶段使用环境中，基本没有真正的100%的在线环境，或者说，如果达到100%的在线能力，要付出非常大的代价。

什么是高可用性

文/陈吉平

高可用（HA）有两种不同的含义，在广义环境中，是指整个系统的高可用（High Availability）性，在狭义方面，一般指主机的冗余接管，如主机HA。在高可用的解释方面，可以分为如下一些方面：

（1）系统失败或崩溃（System faults and crashes）

（2）应用层或者中间层错误（Application and middleware failures）

（3）网络失败（Network failures）

（4）介质失败，一般指存放数据的媒体介质故障（Media failures）

（5）人为失误（Human Error）

（6）分级与容灾（Disasters and extended outages）

（7）计划宕机与维护（Planned downtime, maintenance and management tasks）

高可用的计算方法一般以年在线率来计算。高可用性的在线率（可用级别）与停机时间可以参考如图1所示的对照表。

图1 高可用级别对照表

现阶段使用环境中，基本没有真正的100%的在线环境，或者说，如果达到100%的在线能力，要付出非常大的代价，所以一般能达到99.9%以上的可用性的环境，一般都可以认为是比较高的可用环境了。

对于高可用性在线效率的计算，可以参考如图2所示的方法。

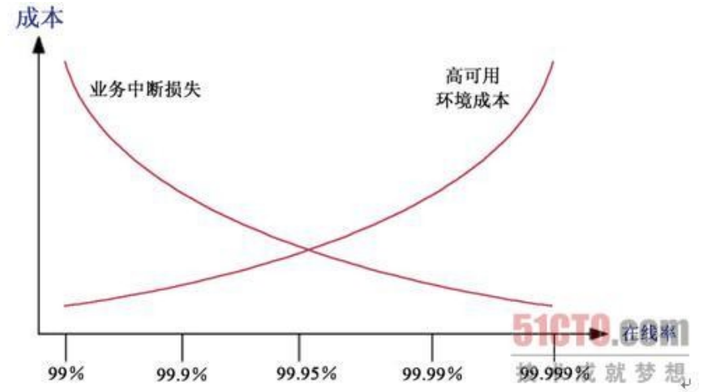

图2 收益与成本

在公司收益与投入成本计算方面取得一个平衡，则是最终所希望的在线效率，但是收益与成本的计算方法则是决策者与实施者需要着重考虑的问题了。本书的很多地方，其实也希望能提供一种思想，那就是怎样搭建最适合自己的高可用环境，而不是盲目地去追逐最高可用性。

本文节选自《构建Oracle高可用环境：企业级高可用数据库架构、实战与经验总结》第一章。章节试读地址：

http://book.51cto.com/art/200812/102072.htm

一般而言，硬件负载均衡在功能、性能上优于软件方式，不过成本昂贵。实际上如果几台服务器，用 F5 之类的绝对是杀鸡用牛刀，而用软件就要合算得多

手把手让你了解 Linux 集群 - 原理篇

资料整理/张冠宇

集群并不是一个全新的概念，其实早在七十年代计算机厂商和研究机构就开始了对集群系统的研究和开发。由于主要用于科学工程计算，所以这些系统并不为大家所熟知。直到 Linux 集群的出现，集群的概念才得以广为传播。

对集群的研究起源于集群系统的良好的性能可扩展性(scalability)。提高 CPU 主频和总线带宽是最初提供计算机性能的主要手段，但是这一手段对系统性能的提供是有限的。接着人们通过增加 CPU 个数和内存容量来提高性能，于是出现了向量机，对称多处理机(SMP)等。但是当 CPU 的个数超过某一阈值，象 SMP 这些多处理机系统的可扩展性就变的极差。主要瓶颈在于 CPU 访问内存的带宽并不能随着 CPU 个数的增加而有效增长。与 SMP 相反，集群系统的性能随着 CPU 个数的增加几乎是线性变化的。

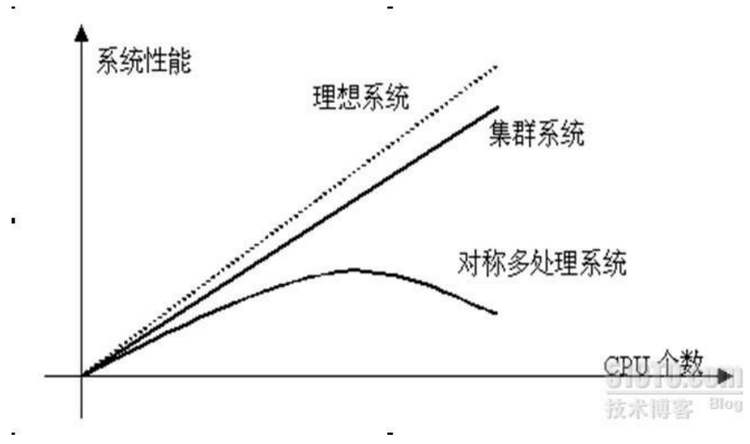

图 1 几种计算机系统的可扩展性

集群系统的优点并不仅在于此。下面列举了集群系统的主要优点：

1. 高可扩展性：如上所述。
2. 高可用性：集群中的一个节点失效，它的任务可以传递给其他节点。可以有效防止单点失效。
3. 高性能：负载平衡集群允许系统同时接入更多的用户。
4. 高性价比：可以采用廉价的符合工业标准的硬件构造高性能的系统。

集群系统基本可分为三种类型：

负载均衡集群 (LB:load balancing)

负载均衡集群为企业需求提供了更实用的系统。如名称所暗示的，该系统使负载可以在计算机集群中尽可能平均地分摊处理。

高可用性集群 (HA:High Availability)

高可用性集群的出现是为了使集群的整体服务尽可能可用，以便考虑计算硬件和软件的易错性。如果高可用性集群中的主节点发生了故障，那么这段时间内将由次节点代替它。

高性能集群 (HP :High Performance)

使用商业系统，并且在公共消息传递层上进行通信以运行并行应用程序，往往涉及为集群开发并行编程应用程序，以解决复杂的科学问题。其处理能力与真的超级计算机相等，通常一套象样的集群配置开销要超过 $100,000。

在集群的这三种基本类型之间，经常会发生混合与交杂。于是，可以发现高可用性集群也可以在其节点之间均衡用户负载，同时仍试图维持高可用性程度。同样，可以从要编入应用程序的集群中找到一个并行集群，它可以在节点之间执行负载均衡。尽管集群系统本身独立于它在使用的软件或硬件，但要有效运行系统时，硬件连接将

起关键作用。

下面笔者着重介绍前两种集群方式，也是企业中最常用的。

一、负载均衡集群（LB:load balancing）

笔者用一幅图为例，简单介绍下负载均衡原理：

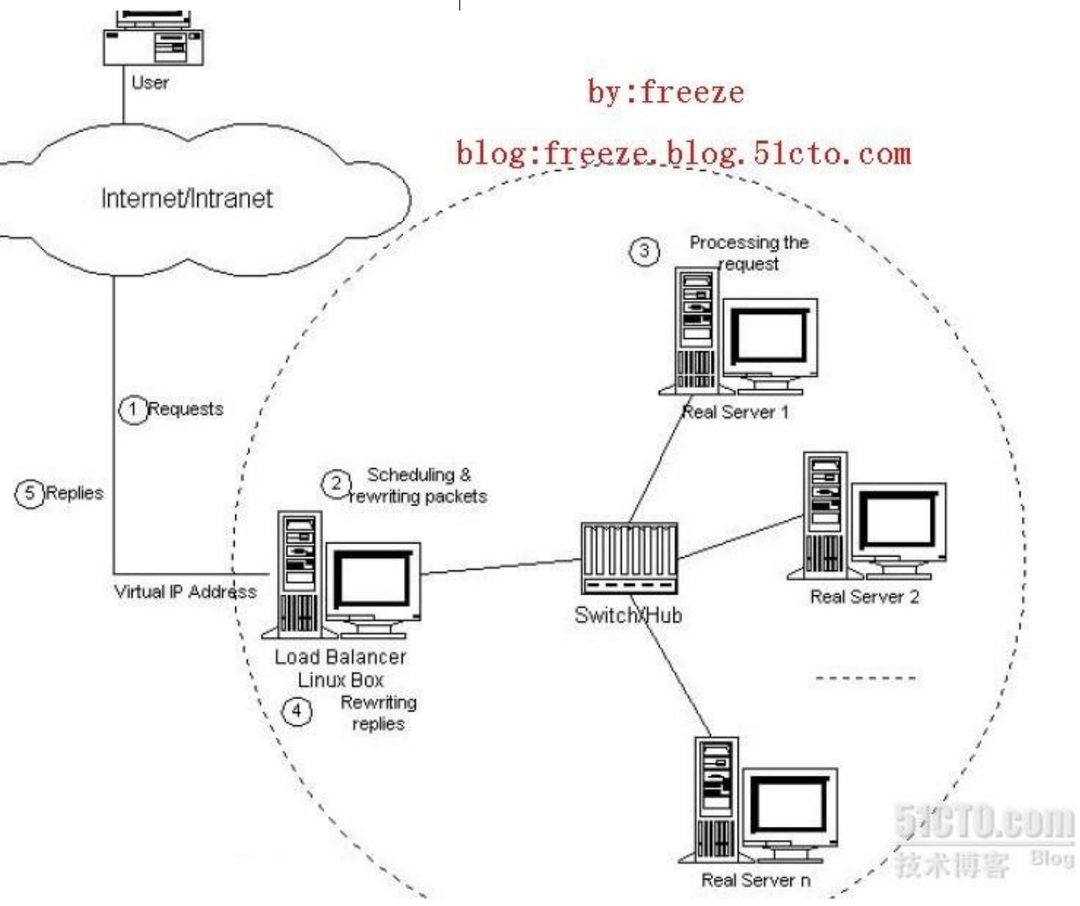

图 2：负载均衡原理

用户通过互联网访问到某个网站时，前端的 Load Balancer（类似负载均衡器）根据不同的算法或某种特定的方式，将请求转发到后端真正的服务器（节点），后台多台服务器共同分担整个网站的压力。后台的某个节点如果有宕机，其他节点也可以提供服务，从而维持整个网站的正常运行。

实现负载均衡有两种方式：

1、硬件：

如果我们搜一搜"负载均衡"，会发现大量关于 F5 等负载均衡设备的内容。基于硬件的方式能够直接通过智能交换机实现,处理能力更强，而且与系统无关，这就是其存在的理由。但其缺点也很明显：

首先是贵。这个贵不仅是体现在一台设备上，而且体现在冗余配置上。很难想象后面服务器做一个集群，但最关键的负载均衡设备却是单点配置，一旦出了问题就全趴了。

第二是对服务器及应用状态的掌握：硬件负载均衡一般都不管实际系统与应用的状态，而只是从网络层来判断。所以有时候系统处理能力已经不行了，但网络可能还来得及反应（这种情况非常典型，比如应用服务器后面内存已经占用很多，但还没有彻底不行，如果网络传输量不大就未必在网络层能反映出来）。

所以硬件方式更适用于一大堆设备、大访问量、简单应用。

一般而言，硬件负载均衡在功能、性能上优于软件方式，不过成本昂贵。实际上如果几台服务器，用 F5 之类的绝对是杀鸡用牛刀，而用软件就要合算得多，因为服务器同时还可以跑应用，呵呵。下面来看看可以在 linux 平台实现负载均衡的软件。

2、软件：

软件负载均衡解决方案是指在一台或多台服务器相应的操作系统上安装一个或多个附加软件来

实现负载均衡，它的优点是基于特定环境，配置简单，使用灵活，成本低廉，可以满足一般的负载均衡需求。

著名项目：开源软件方面，最著名的是 LVS。

★★★LVS 系统结构与特点

1. Linux Virtual Server：简称 LVS。是由中国 Linux 程序员章文嵩博士发起和领导的，基于 Linux 系统的服务器集群解决方案，其实现目标是创建一个具有良好的扩展性、高可靠性、高性能和高可用性的体系。许多商业的集群产品，比如 RedHat 的 Piranha、Turbo Linux 公司的 Turbo Cluster 等，都是基于 LVS 的核心代码的。

2. 体系结构：使用 LVS 架设的服务器集群系统从体系结构上看是透明的，最终用户只感觉到一个虚拟服务器。物理服务器之间可以通过高速的 LAN 或分布在各地的 WAN 相连。最前端是负载均衡器，它负责将各种服务请求分发给后面的物理服务器，让整个集群表现得像一个服务于同一 IP 地址的虚拟服务器。

3. LVS 的三种模式工作原理和优缺点：Linux Virtual Server 主要是在负载均衡器上实现的，负载均衡器是一台加了 LVS Patch 的 2.2.x 版内核的 Linux 系统。LVS Patch 可以通过重新编译内核的方法加入内核，也可以当作一个动态的模块插入现在的内核中。

二、高可用性集群（HA:High Availability）

1、什么是高可用集群

高可用集群，英文原文为 High Availability Cluster，简称 HA Cluster，是指以减少服务中断（如因服务器宕机等引起的服务中断）时间为目的的服务器集群技术。简单的说，集群（cluster）就是一组计算机，它们作为一个整体向用户提供一组网络资源。这些单个的计算机系统就是集群的节点（node）。

高可用集群的出现是为了使集群的整体服务尽可能可用，从而减少由计算机硬件和软件易错性所带来的损失。它通过保护用户的业务程序对外不间断提供的服务，把因软件/硬件/人为造成的故障对业务的影响降低到最小程度。如果某个节点失效，它的备援节点将在几秒钟的时间内接管它的职责。因此，对于用户而言，集群永远不会停机。高可用集群软件的主要作用就是实现故障检查和业务切换的自动化。

只有两个节点的高可用集群又称为双机热备，即使用两台服务器互相备份。当一台服务器出现故障时，可由另一台服务器承担服务任务，从而在不需要人工干预的情况下，自动保证系统能持续对外提供服务。双机热备只是高可用集群的一种，高可用集群系统更可以支持两个以上的节点，提供比双机热备更多、更高级的功能，更能满足用户不断出现的需求变化。

2、怎样实现高可用集群？

在 linux 上实现高可用的解决方案主要有：

1. 开放源代码的 HA 项目（http://www.linux-ha.org/）

2. RedHat 公司的开放源代码 RedHat Cluster Suite，简称 RHCS .

3. Novell 公司的开放源代码高可用集群 HA 套件

4. Novell 公司的 Novell Cluster Service

5. Steeleye Lifekeeper for linux

6. HP MC/Service Guard for linux

7. Turbolinux 高可用集群系统

原文（本文为节选）：
http://freeze.blog.51cto.com/1846439/388957

尽管 windows 占据了绝大部分的桌面市场，但在服务器领域，其份额还是很少的：象 google、yahoo、baidu 等拥有上万台服务器应用的机构都不约而同的选择 linux/unix 做为运营平台来支撑巨大的业务访问。

可扩展、高可用服务网络设计方案

文/田逸

对于基于 LVS 的互联网应用，配置不是重点，关键是思路，怎样才能是一个可以使用的完整应用。下面本文介绍的是一个可扩展、高可用服务的网络设计方案，具体的技术实现细节可参考另一篇文章。

一、现状

1、系统多数是 windows，可靠性和稳定性都非常的差。在历次的网络安全事故中，windows 都是最大的受害者。尽管 windows 占据了绝大部分的桌面市场，但在服务器领域，其份额还是很少的：象 google、yahoo、baidu 等拥有上万台服务器应用的机构都不约而同的选择 linux/unix 做为运营平台来支撑巨大的业务访问。

2、存在单点故障。每个业务都运行在一个系统/机器上，一旦系统/机器发生故障，业务将不可避免的停止服务。拿网站做例子，web 服务 apache 或数据库(mysql)只要任意一个服务出故障，整个网站的访问将变成不可能。

3、缺乏集中的，可靠性高的存储机制。现有的配置文件、程序、数据库等数据都是单独存放在各自运行的系统上，维护成本非常高，而且很容易丢失。

4、不具备可扩展性和高可用性。任何一个服务器出故障，运行在上面的业务将不再问用户提供有效服务。

5、缺乏有效的流量监控设施。现在总的访问流量是未知数，因此对总带宽的使用率没有评估的依据。租了 20M 的带宽，实际使用了多少，不得而知。

二、改进措施

1、尽可能的把应用移植到 linux 平台。

2、采用 NAS 存储解决方案。

3、部署同一个业务到不同的服务器，然后使用 LVS-DR 做负载均衡，同时避免了单点故障。

4、后台数据库 mysql 采用主从方式的复制机制保证 database 的高可用性。

三、基本原理：

1、LVS-DR：这是一个开源的产品，已经成为 linux 内核的一部分。用户的访问首先被转向到 LVS-DR,然后根据业务的类别被重新定向到真实的服务器，由于 LVS-DR 只是转发，一旦客户短与提供服务的真实服务器连接成功，就不再使用 LVS-DR 的资源。

2、多服务器运行同一个应用。既克服单点故障，又能增加系统的容量。

3、NAS 存储。提供集中可靠的存储机制。

4、Mysql 复制。避免数据库单点故障；如果将来访问量增大到一定程度的时候，可以改变到 mysql 集群的方式

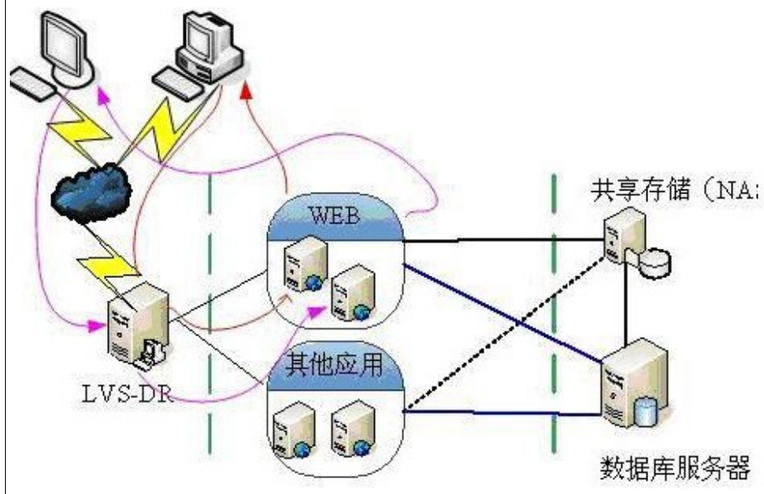

图 1: 正常情况

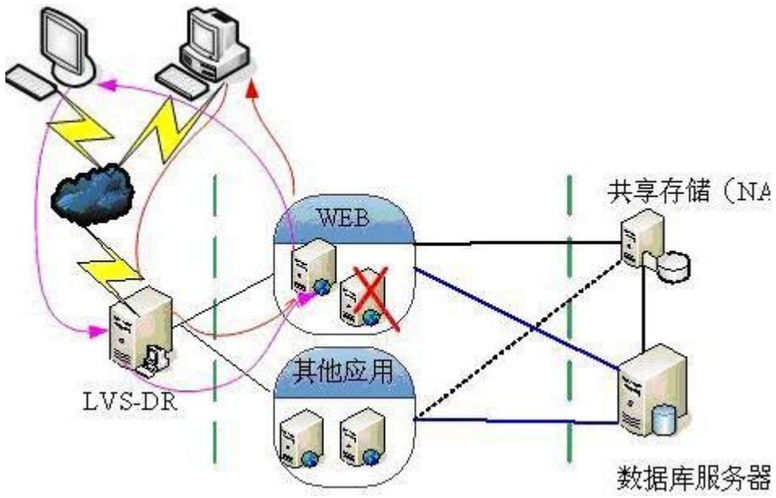

图 2: 出现故障

三、实施步骤

1、移植应用到 linux 平台

2、配置 LVS-DR 负载均衡控制器部分。

3、部署相同的应用(web 等)到两个不同的服务器

4、部署 NAS

5、测试

6、正式运营。

四、设备分配

1、LVS-DR 1 个服务器

2、Web 服务器 2 个

3、Mysql 服务器 2 个

4、其他的几个服务器暂时不变

5、网管交换机一个(cisco 2950)

6、NAS 一套

五、进度安排

名称	花费时间	备注
LVS-DR 控制器配置	1 天（以后逐步增加转发条目）	以配置 ipvs 转发规则和防火墙规则
平台移植（windows 到 linux）	5 天	已经移植了 web 和 bbs
把服务器加入 lvs 集群	2 天	已经加了一个 bbs 和 mms
新建一个邮件服务器	1 天	测试中
部署流量监控	1 天	
Nas 上线及配置	2 天	
其他	7 天	

原文：

http://sery.blog.51cto.com/10037/40660

技术实现：

http://sery.blog.51cto.com/10037/54645

推荐阅读：

企业级 WEB 的负载均衡高可用之 LVS+Keepalived

http://network.51cto.com/art/201006/206831.htm

生产环境下的高可用 NFS 文件服务器

http://network.51cto.com/art/201010/230237.htm

揭秘企业级 web 负载均衡完美架构

http://network.51cto.com/art/201007/209823.htm

LVS 项目从成立到现在为止，受到不少关注，LVS 集群系统已被应用于很多重负载的站点。

Linux 集群服务 LVS 概述与安装配置详解

文/李洋

LVS 项目从成立到现在为止，受到不少关注，LVS 集群系统已被应用于很多重负载的站点。到目前为止，该系统已在美、英、德、澳等国的几十个站点上正式使用。目前，一些大型 LVS 应用实例如下：

·SourceForge 在全球范围内为开发源码项目提供 WEB、FTP、Mailing List 和 CVS 等服务，他们也使用 LVS 将负载调度到十几台机器上。

·世界上最大的 PC 制造商之一采用了两个 LVS 集群系统，一个在美洲，一个在欧洲，用于网上直销系统。

·Real 公司使用由 20 台服务器组成的 LVS 集群，为其全球用户提供音频视频服务。在 2000 年 3 月时，整个集群系统已收到平均每秒 20,000 个连接的请求流。

·NetWalk 用多台服务器构造 LVS 系统，提供 1024 个虚拟服务，其中包括本项目的一个美国镜像站点。

安装 LVS

安装 LVS 和配置 LVS 的工作比较繁杂，读者在配置的过程中需要非常细心和耐心。

1. 获取支持 LVS 的内核源代码

如果读者需要使用 LVS，需要下载 2.4.23 以后版本的内核源代码。下载地址为 http://www.kerner.org。目前主流的 Linux 内核已经支持 LVS，只需要直接使用，不需要进行内核的下载和更新工作。

2. 用户配置工具 ipvsadm

该软件的下载地址为：

http://www.linuxvirtualserver.org/software/ipvs.html。

3. 调整内核配置选项

读者在内核配置时应该对下列必选项（用*号表示）进行检查，如果某些选项的设置不正确，将有可能影响 LVS 的正常工作和使用。在查看这些选项之前，用户需要使用 make menuconfig 命令，进入 Networking options 选项进行查看：

```
Networking options  --->
<*> Packet socket
<*> Netlink device emulation
[*] TCP/IP networking
[*] IP: advanced router
[*] Network packet filtering (replaces
 ipchains)
IP: Netfilter Configuration  --->
<*> Connection tracking (required for
 masq/NAT)
<*> IP tables support (required for
 filtering/masq/NAT)
<*> Full NAT
<*> MASQUERADE target support
IP: Virtual Server Configuration  --->
<*> virtual server support (EXPERIMENTAL)
<M>   IPVS connection table size (the Nth
 power of 2)--- IPVS scheduler
<M>   round-robin scheduling
<M>   weighted round-robin scheduling
<M>   least-connection scheduling
<M>   weighted least-connection
 scheduling
<M>   locality-based least-connection
 scheduling
<M>   locality-based least-connection
 with replication scheduling
<M>   destination hashing scheduling
<M>   shortest expected delay scheduling
<M>   never queue scheduling
```

配置和使用 LVS

在安装好 LVS 之后，就可以配置和使用 LVS 了，在本节我们将以一个具体的例子来对其进行讲解。下图为一个采用 LVS 系统的实际网络拓扑图。它基于 NAT 机制，具体的配置如下：

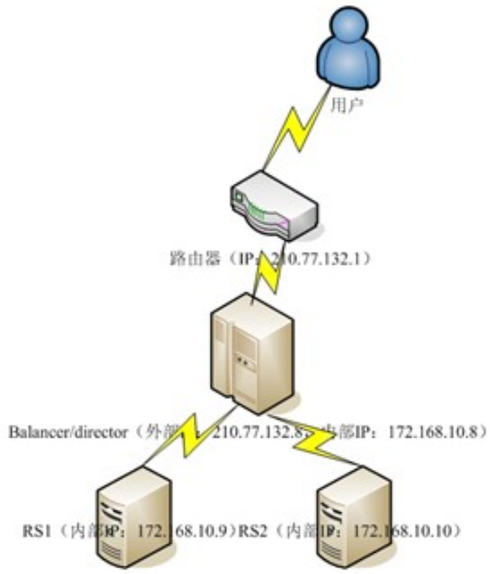

图 1 采用 LVS 系统的实际网络拓扑图

一台对外服务的超级服务器：它部署了 LVS，也称为 balancer 或者 director，主要功能为负载均衡和任务调度，其外部 IP 地址为：210.77.132.8，内部 IP 地址为：172.168.10.8。外部用户可以通过路由器（IP 地址为：210.77.132.1）访问它；

两台内部的服务器：它们为实际的工作机器，通过前述的服务器对其进行调度。一台为 RS1，其内部 IP 地址为：172.168.10.9；另一台为 RS2，其内部 IP 地址为：172.168.10.10。

根据上述的网路配置和拓扑，对 LVS 的配置如下：

1. 配置 LVS 的超级服务器（称为 load balancer 或者 director）

运行如下命令：

```
//配置重定向
#echo "1" >/proc/sys/net/ipv4/ip_forward
#echo "0"
 >/proc/sys/net/ipv4/conf/all/send_redire
 cts
#echo "0"
 >/proc/sys/net/ipv4/conf/default/send_re
 directs
#echo "0"
 >/proc/sys/net/ipv4/conf/eth0/send_redir
 ects
#echo "0"
 >/proc/sys/net/ipv4/conf/eth1/send_redir
 ects
//清除 ipvsadm 表
#/sbin/ipvsadm -C
//使用 ipvsadm 安装 LVS 服务
#add http to VIP with rr scheduling
#/sbin/ipvsadm -A -t 210.77.132.8:80 -s
 rr
//增加第一台内部服务器 RS1
#forward http to realserver 172.168.10.9
 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t 210.77.132.8:80 -r
 172.168.10.9:80 -m -w 1
//增加第二台内部服务器 RS2
#forward http to realserver 172.168.10.10
 using LVS-NAT (-m), with weight=1
/sbin/ipvsadm -a -t 210.77.132.8:80 -r
 172.168.10.10:80 -m -w 1
```

2. 配置 LVS 中的内部服务器

在 172.168.10.9（RS1）和 172.168.10.9（RS2）上分别将其网关设置为 172.168.10.8，并分别启动 apache 服务。在客户端使用浏览器多次访问：http://210.77.132.8，然后再 210.77.132.8 上运行 ipvsadm 命令，应该有类似下面的输出：

```
IP Virtual Server version 1.0.12
 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward
 Weight ActiveConn InActConn
TCP  210.77.132.8:http rr
  -> 172.168.10.9:http            Masq    1
 0          33
  -> 172.168.10.10:http           Masq
 1      0          33
```

从上面的结果可以看出，我们配置的 LVS 服务器已经成功运行。

原文：

http://os.51cto.com/art/201011/233146.htm

其实在服务器性能优异，内存足够的情况下，Apache 的抗并发能力并不弱，整个网站的瓶颈应该还是在数据库方面。

19 个心得 明明白白说 Linux 下的负载均衡

文/抚琴煮酒

作为一名 Linux/unix 系统工程师，这几年一直在涉及到对外项目，经手过许多小中型网站的架构，F5、LVS 及 Nginx 接触的都比较多，我想以一种比较通俗易懂的语气跟大家说明下何谓负载均衡，何谓 Linux 集群，帮助大家走出这个误区，真正意义上来理解它们。

一、

目前网站架构一般分成负载均衡层、web 层和数据库层，我其实一般还会多加一层，即文件服务器层，因为现在随着网站的 PV 越来越多，文件服务器的压力也越来越大；不过随着 moosefs、DRDB+Heartbeat+NFS 的日趋成熟，这问题也不大了。网站最前端的负载均衡层称之为 Director，它起的是分摊请求的作用，最常见的就是轮询。

二、

F5 是通过硬件的方式来实现负载均衡，它较多应用于 CDN 系统，用于 squid 反向加速集群的负载均衡，是专业的硬件负载均衡设备，尤其适用于每秒新建连接数和并发连接数要求高的场景；LVS 和 Nginx 是通过软件的方式来实现的，但稳定性也相当强悍，在处理高并发的情况也有相当不俗的表现。

三、

Nginx 对网络的依赖较小，理论上只要 ping 得通，网页访问正常，nginx 就能连得通，nginx 同时还能区分内外网，如果是同时拥有内外网的节点，就相当于单机拥有了备份线路；lvs 就比较依赖于网络环境，目前来看服务器在同一网段内并且 lvs 使用 direct 方式分流，效果较能得到保证。

四、

目前较成熟的负载均衡高可用技术有 LVS+Keepalived、Nginx+Keepalived，以前 Nginx 没有成熟的双机备份方案，但通过 shell 脚本监控是可以实现的，有兴趣的可具体参考我在 51cto 上的项目实施方案；另外，如果考虑 Nginx 的负载均衡高可用，也可以通过 DNS 轮询的方式来实现，有兴趣的可以参考张宴的相关文章。

五、

集群是指负载均衡后面的 web 集群或 tomcat 集群等，但现在的集群意义泛指了整个系统架构，它包括了负载均衡器以及后端的应用服务器集群等，现在许多人都喜欢把 Linux 集群指为 LVS，但我觉得严格意义上应该区分开。

六、

负载均衡高可用中的高可用指的是实现负载均衡器的 HA，即一台负载均衡器坏掉后另一台可以在<1s 秒内切换，最常用的软件就是 Keepalived 和 Heatbeat，成熟的生产环境下的负载均衡器方案有 Lvs+Keepalived、Nginx+Keepalived。

七、

LVS 的优势非常多：①抗负载能力强；②工作稳定(因为有成熟的 HA 方案)；③无流量；④基本上能支持所有的应用，基于以上的优点，LVS 拥有不少的粉丝；但世事无绝对，LVS 对网络的依赖性太大了，在网络环境相对复杂的应用场景中，我不得不放弃它而选用 Nginx。

八、

Nginx 对网络的依赖性小，而且它的正则强大而灵活，强悍的特点吸引了不少人，而且配置也

是相当的方便和简约，小中型项目实施中我基本是考虑它的；当然，如果资金充足，F5 是不二的选择。

九、

大型网站架构中其实可以结合使用 F5、LVS 或 Nginx，选择它们中的二种或三种全部选择；如果因为预算的原因不选择 F5，那么网站最前端的指向应该是 LVS，也就是 DNS 的指向应为 lvs 均衡器，lvs 的优点令它非常适合做这个任务。重要的 ip 地址，最好交由 lvs 托管，比如数据库的 ip、webservice 服务器的 ip 等等，这些 ip 地址随着时间推移，使用面会越来越大，如果更换 ip 则故障会接踵而至。所以将这些重要 ip 交给 lvs 托管是最为稳妥的。

十、

VIP 地址是 Keepalived 虚拟的一个 IP，它是一个对外的公开 IP，也是 DNS 指向的 IP；所以在设计网站架构时，你必须向你的 IDC 多申请一个对外 IP。

十一、

在实际项目实施过程中发现，LVS 和 Nginx 对 https 的支持都非常好，尤其是 LVS，相对而言处理起来更为简便。

十二、

在 LVS + Keepalived 及 Nginx + Keepalived 的故障处理中，这二者都是很方便的；如果发生了系统故障或服务器相关故障，即可将 DNS 指向由它们后端的某台真实 web，达到短期处理故障的效果，毕竟广告网站和电子商务网站的 PV 就是金钱，这也是为什么要将负载均衡高可用设计于此的原因；大型的广告网站我就建议直接上 CDN 系统了。

十三、

现在 Linux 集群都被大家神话了，其实这个也没多少复杂；关键看你的应用场景，哪种适用就选用哪种，Nginx 和 LVS、F5 都不是神话，哪种方便哪种适用就选用哪种。

十四、

另外关于 session 共享的问题，这也是一个老生长谈的问题了；Nginx 可以用 ip_hash 机制来解决 session 的问题，而 F5 和 LVS 都有会话保持机制来解决这个问题，此外，还可以将 session 写进数据库，这也是一个解决 session 共享的好办法，当然这个也会加重数据库的负担，这个看系统架构师的取舍了。

十五、

我现在目前维护的电子商务网站并发大约是 1000 左右，以前的证券资讯类网站是 100 左右，大型网上广告大约是 3000，我感觉 web 层的并发越来越不是一个问题；现在由于服务器的强悍，再加上 Nginx 作 web 的高抗并发性，web 层的并发并不是什么大问题；相反而言，文件服务器层和数据库层的压力是越来越大了，单 NFS 不可能胜任目前的工作，现在好的方案是 moosefs 和 DRDB+Heartbeat+NFS；而我喜欢的 Mysql 服务器，成熟的应用方案还是主从，如果压力过大，我不得不选择 oracle 的 RAC 双机方案。

十六、

现在受张宴的影响，大家都去玩 Nginx 了(尤其是作 web)，其实在服务器性能优异，内存足够的情况下，Apache 的抗并发能力并不弱，整个网站的瓶颈应该还是在数据库方面；我建议可以双方面了解 Apache 和 Nginx，前端用 Nginx 作负载均衡，后端用 Apache 作 web，效果也是相当的好。

十七、

Heartbeat 的脑裂问题没有想象中那么严重，在线上环境可以考虑使用；DRDB + Heartbeat 算是成熟的应用了，建议掌握。我在相当多的场合用此组合来替代 EMC 共享存储，毕竟 30 万的价格并不是每个客户都愿意接受的。

十八、

无论设计的方案是多么的成熟，还是建议要配置 Nagios 监控机来实时监控我们的服务器情况；邮件和短信报警都可以开启，毕竟手机可以随身携带嘛;有条件的还可以购买专门的商业扫描网站服务，它会每隔一分钟扫描你的网站，如果发现没有 alive 会向你的邮件发警告信息或直接电话联系。

十九、

至少网站的安全性问题，我建议用硬件防火墙，比较推荐的是华赛三层防火墙+天泰 web 防火墙，

DDOS 的安全防护一定要到位；Linux 服务器本身的 iptables 和 SElinux 均可关闭，当然，端口开放越少越好。

补充说明：

测试网站的响应时间是用 http://tools.pingdom.com,发现上了 LVS + Keepalived、Nginx+Keepalived 后并不影响速度,这一点大家就不要多虑了，Nginx 现在作反向加速也日趋成熟了。

原文：

http://network.51cto.com/art/201008/217262.htm

推荐阅读：

负载均衡实例：如何解放昼夜不眠的网管？

http://network.51cto.com/art/201005/202727.htm

vi 编辑器极其强大，特性和功能非常丰富。即使在多年使用 vi 之后，您仍然可能会发现有不知道的新命令。

几个 vi 技巧和诀窍分享

文/Martin Wicks
译/IBM Developerworks 中国

在使用 vi 编辑器时，无论是初次使用的用户还是有经验的用户，大多数人往往只掌握核心命令集。这些命令可以执行最常用的功能：导航或保存文件；插入、更新、删除或搜索数据；退出但不保存修改。

但是，vi 编辑器极其强大，特性和功能非常丰富。即使在多年使用 vi 之后，您仍然可能会发现有不知道的新命令。本文讨论的命令就属于不太为人所知的命令，但是它们可以简化您目前采用的操作方法，让您的工作方式更高效，或者让您能够完成原来不知道可以用 vi 完成的操作。

打开和关闭行号

vi 编辑器的许多选项可以控制编辑会话的外观和感觉。使用 :set 命令修改 vi 中的会话设置。按 Escape 键进入命令模式之后，可以使用 :set all 命令显示选项和设置的列表。

可以设置的选项之一是 number，它的作用是打开和关闭行号（见 清单 1）。

清单 1. 打开行号之前

```
#
# Internet host table
#
::1 localhost
127.0.0.1 localhost loghost
192.168.0.6 centos5
192.168.0.10 appserv
192.168.0.11 webserv
192.168.0.12 test
192.168.0.5 solaris10 # Added by DHCP
~
~
~
:set number
```

这个命令让 vi 在当前编辑的文件中的每个记录上显示行号。让 vi 进入命令模式之后，可以输入 :set number 并按回车来打开行号（见 清单 2）。

清单 2. 打开行号之后

```
1 #
2 # Internet host table
3 #
4 ::1 localhost
5 127.0.0.1 localhost loghost
6 192.168.0.6 centos5
7 192.168.0.10 appserv
8 192.168.0.11 webserv
9 192.168.0.12 test
10 192.168.0.5 solaris10 # Added by DHCP
~
~
~
:set number
```

可以使用 :set nonumber 命令关闭行号。还可以使用这个命令和 :set number 命令的简写，即 :set nu 和 :set nonu。

如果需要快速计算要用 vi 函数处理的行数，显示行号会非常有帮助。当行数很多，可能跨多个屏幕时，行号尤其有用。另外，有时候您知道要处理的行范围，但是需要查明要在 vi 命令中使用的初始和结束行号。

如果希望每次进入 vi 会话时都显示行号，那么在主目录中的 .exrc 文件中添加 set number 行。

自动缩进

在用某些编程语言编写代码时，缩进是样式的重要部分，可以确保代码的可读性更好。如果需要，可以在 vi 编辑器中根据编程语言的样式设置自动缩进。使用 autoindent 打开或关闭自动缩进（见 清单 3）。

清单 3. 打开自动缩进

```
#!/bin/ksh
#
#
for file in /etc/*
do
if [[ -f ${file} ]] ; then
echo "${file} is a file"
~
~
~
~
~
:set autoindent
```

在此之后，如果在一行的开头输入空格或制表符，那么后续的新行将会缩进到相同的位置。在命令模式下，输入 :set autoindent，然后按回车打开自动缩进。通过设置 shiftwidth 确定缩进级别。例如，:set shiftwidth=4 把每级缩进设置为四个空格（见 清单 4）。

清单 4. 设置缩进级别

```
#!/bin/ksh
#
#
for file in /etc/*
do
if [[ -f ${file} ]] ; then
echo "${file} is a file"
elif [[ -d ${file} ]] ; then
echo "${file} is a directory"
fi
done
~
```

```
~
:set shiftwidth=4
```

在命令模式下，可以使用 >> 命令让现有的一行增加一级缩进，使用 (命令减少一级缩进。在这些命令前面加上一个整数，即可让多行增加或减少一级缩进。例如，把游标放在 清单 4 中第 6 行的开头，进入命令模式之后，输入 5) 就会让下面五行增加一级缩进。清单 5 显示结果。

清单 5. 缩进代码块

```
#!/bin/ksh
#
#
for file in /etc/*
do
if [[ -f ${file} ]] ; then
echo "${file} is a file"
elif [[ -d ${file} ]] ; then
echo "${file} is a directory"
fi
done
~
~
```

可以使用 :set noautoindent 命令关闭自动缩进。还可以使用这个命令和 autoindent 命令的简写，即 :set ai 和 :set noai。还可以使用 :set ai sw=4 在一个命令中打开缩进并设置缩进级别。

如果希望每次启动 vi 会话时都启用自动缩进并把缩进级别设置为四个空格，那么在主目录中的 .exrc 文件中添加 set ai sw=4 行。

在搜索时不区分大小写

如您所知，在 UNIX 中执行搜索时，模式匹配是区分大小写的。但是，如果希望 vi 不区分大小写，那么可以使用 :set ignorecase 命令。使用 :set noignorecase 恢复区分大小写。还可以使用简写（:set ic 和 :set noic）。

如果希望每次进入 vi 会话时都启用不区分大小写的搜索，那么在主目录中的 .exrc 文件中添加 set ignorecase 行。

复合搜索

在 vi 中，可以使用 / 命令搜索字符串，这需要以字面字符串或正则表达式的形式指定要匹配的模式。例如，要想在文件中搜索单词 echo，只需进入命令模式，输入 /echo，然后按回车。这个命令会找到 清单 6 所示文件的第 3 行的第一个单词。

本文为节选，全文请参阅原文：

http://www.ibm.com/developerworks/cn/aix/library/au-vitips.html

与如何备份系统一样，你需要不时进行一下“消防演习”，对恢复流程进行试验。

全新的备份利器推荐：Duplicity 使用评测

文/Tom Limoncelli
译/司马牵牛

曾经，我需要将一个服务器备份到远程硬盘上，有几种方式，而且我也可以自己动手写一个方法。不过，我找到了 Duplicty，现在我向你强烈推荐：

http://duplicity.nongnu.org

Duplicity 使用 librsync 生产一个非常小的额外备份。它能够生成递增备份，然后使用 GPG 进行加密，能够适用常用的方法发送至另一个服务器上，比如：scp、ftp、sftp、rsync 等等。可以从任何目录开始备份，不限于加载点（mountpoint），并且可以指定你想要排除的文件。

安装

如果你之前没有做过，那最难的部分就是设置 GPG 密钥。你需要保护好你的密钥。比如，你的机器着火丢失了所有数据，而且没有做密钥备份，那将无法进行恢复。我把密钥刻录了几张 CD 作为多个备份。

我要备份的机器是 colo 上的虚拟机。他们没有提供备份服务，所以不得不自己动手。机器使用的是 FreeBSD 8.0-RELEASE-p4 并且运行正常。代码也是非常便携式的：Python、GPG、librsync 等等。没有涉及内核或原始设备等类似问题。

我写了一个简单脚本，扫描所有我想要备份的目录，然后运行：

```
duplicity --full-if-older-than 5W
 --encrypt-key="${PGPKEYID}" $DIRECTORY
 scp://myarchives@mybackuphost/
 $BACKUPSET$dir
```

“--full-if-older-than 5W”表示进行递增备份，每隔 35 天进行一次完全的备份。我用 5W 而不是 4W，因为我想确保完全备份不会低于一个月。我每个月支付宽带费用，我不想在一个月遇到两件麻烦事，这会让人崩溃。

我的方法：使用 scp 将文件备份到另一个机器上，这是一个很便宜的 USB2.0 硬盘，容量 1T。这样设置，以便我能够用 ssh 从源机器访问目标机器而无需密码。上例中的“myarchives”是我做备份用的用户名，“mybackuphost”是主机。事实上，我指定了主机名，使用 .ssh/confi 将默认用户名设置为“myarchives”。这样，我可以在其他 shell 脚本中指定“mybackuphost”。

恢复

当然，人们真实关心的不是备份，而是恢复。在恢复文件时，duplicity 找到哪些递增和完整备份需要进行恢复和解密。你只需指定日期（默认是“the lastest”，即最新），然后它会自动完成所有工作。我所需的工作如此至少，这点让我印象非常深刻。

系统运行一段时间之后，就需要进行恢复了，以确保一切正常。

恢复语法有点让人困惑，文档中也没有提供太多示例。最常见的情况并非是对所有备份组进行恢复，而是：“某个文件有问题，或者我觉得某个文件有问题，所有我需要恢复旧版本（从某个特定日期来看）到 /tmp，来看看这个文件原本是什么样的。”

我感到困惑的是：

1）指定了文件或目录的路径，但你没有列出指向备份加载点或目录的路径。事后来看，这是显而易见的事情，当时这的确让我很纳闷。让我明白过来的是，当我列出文件时，文件会显示出来，而没有加载点

2）指定备份放置的地方时，你必须非常小心。你可以在命令行中指定，并在“--file-to-restore”选项中指定恢复的文件。你不能在命令行中指定所有文件，然后让 duplicity 去猜从哪里分割——它做不到。

为了不用在意外删除了重要文件之后的紧急情况下重新温习那些命令，我做了一个如何进行恢复的记录。为了服务新手们，我又进行了一些改善如下：

第一步：

列出所有备份到 “home/ta”区的文件：

```
duplicity list-current-files
 scp://mybackuphost/directoryname/home/ta
 l
```

要理出它们在某个特定日期的情况，添加：--restore-time "2002-01-25"

第二步：

从该列表恢复文件（不是恢复到最初位置）：

```
duplicity restore --encrypt-key=XXXXXXXX
 --file-to-
 restore=path/you/saw/in/listing
 scp://mybackuphost/directoryname/home/ta
 l /tmp/restore
```

假设旧文件在“/home/tal/path/to/file”，备份在“/home/tal”，你需要指定--file-to-restore 作为“”，而不是“/home/tal/path/to/file”。你可以列出一个目录来获取所有文件。/tmp/restore 应是一个已经存在的目录。

恢复某个特定日期的文件，添加：

```
--restore-time "2002-01-25"
```

结论：

Duplicity 非常棒，而且速度也很快。这是因为善用 libsync 让备份变得很小，同时也因为他们保持了备份文件的索引，这样为了获得文件列表无需对整个备份进行读取。备份文件很小，划分为多个小文件，这样源机器就不需太多的临时空间。用起来也非常简单：他们能够完成所有递增和完整备份的工作，从而让你能够把关注焦点放在对什么进行备份和恢复上。

提示：与如何备份系统一样，你需要不时进行一下“消防演习”，对恢复流程进行试验。建议你将备份流程打包在一个 shell 脚本中，这样每次都可以使用同一种方式。

原文：

http://everythingsysadmin.com/2010/10/review-the-duplicity-backup-sy.html

译文：

http://os.51cto.com/art/201010/229609.htm

开源自动化配置管理工具 Puppet 入门教程

文/stone

系统管理员经常陷入一系列的重复任务中：如升级软件包、管理配置文件、系统服务、cron 任务以及添加新的配置、修复错误等。这些任务通常是重复低效的，解决这类任务的第一反应是让他们自动化，于是出现了定制脚本。由于环境复杂，定制脚本和应用程序一再被重复开发，并且很难适合多种平台，灵活性和功能也很难保证，于是像 Puppet 这样的自动化配置管理工具便出现了。

Puppet 的语法允许你创建一个单独脚本，用来在你所有的目标主机上建立一个用户。所有的目标主机会依次使用适用于本地系统的语法解释和执行这个模块。举例：如果这个配置是在 Red Hat 服务器上执行，建立用户使用 useradd 命令；如果这个配置是在 FreeBSD 主机上执行，使用的是 adduser 命令。

Puppet 另一个卓越的地方是它的灵活性。源于开源软件的天性，你可以自由的获得 Puppet 的源码，如果你遇到问题并且有能力的话，你可以修改或者加强 Puppet 的代码去适用于你的环境。另外，社区开发者和捐献者还在不断增强 Puppet 的功能。一个大的开发者和用户社区也致力于提供 Puppet 的文档和技术支持。

Puppet 也是易于扩展的。定制软件包的支持功能和特殊的系统环境配置能够快速简单的添加进 Puppet 的安装程序中。

安装配置：

1．Puppet 在 RedHat/CentOS 系统上安装

Puppet 是基于 Ruby 写成的，所以安装前要准备好 Ruby 环境。在中心的 Server 上安装 puppet-server 包，并运行 puppetmasterd 进程；在被管理机上安装 puppet 包，并运行 puppetd 进程。另外，在每台主机上配置好自己的 hostname，之后每台机器要以 hostname 区分。

1）．安装 ruby 环境：

```
yum install ruby ruby-rdoc
```

2）．安装 puppet

Server 端安装：

```
rpm -Uvh
 http://download.fedora.redhat.com/pub/ep
 el/5/i386/epel-release-5-3.noarch.rpm
yum install puppet-server
chkconfig –level 2345 puppetmaster on
```

修改 hosts，添加下面行：

```
Vi /etc/hosts
172.16.228.30   puppet.sina.com.cn puppet
172.16.228.29   web1.sina.com.cn web1
```

客户端安装：

```
rpm -Uvh
 http://download.fedora.redhat.com/pub/ep
 el/5/i386/epel-release-5-3.noarch.rpm
yum install puppet
chkconfig –level 2345 puppet on
```

修改 hosts，添加下面行：

```
Vi /etc/hosts
172.16.228.30   puppet.sina.com.cn puppet
172.16.228.29   web1.sina.com.cn web1
```

3）．启动 puppet

Server 端首次运行前，编辑/etc/puppet/manifests/site.pp 文件，内容可以用最基本的：

```
# Create “/tmp/testfile” if it doesn’t
 exist.
class test_class {
file { “/tmp/testfile”:
ensure => present,
mode => 644,
owner => root,
group => root
```

```
}
}
# tell puppet on which client to run the
 class
node web1.sina.com.cn {
include test_class
}
```

启动 Server 端：

```
service puppetmaster start
```

启动客户端：

```
/etc/init.d/puppet once -v
```

这时客户机会去连 server，但是由于连接是在 ssl 上的，而 Server 还没有 sign 过客户端的 cert，客户机被断开。

到 Server 端执行：puppetca -list，会显示等待签名的客户端的主机名，执行：puppetca -sign <客户端主机名> 即可为其签名。

在 Server 端为 web1.sian.com.cn 授权：

```
puppetca -list
web1.sian.com.cn
puppetca -sign web1.sian.com.cn
```

这时再到客户机上启动 puppetd，即可看到客户在正常地连接 server，并且应用 Server 上为客户端定制的配置策略。

启动客户端：

```
/etc/init.d/puppet once -v
```

4). 测试：

也可以将日志直接打印到终端上进行测试。

Server 端：

```
puppetmasterd -d -no-daemonize -v -trace
```

客户端：

```
puppetd -test -trace -debug
```

2. puppet 命令集

1). puppet 用于执行用户所写独立的 mainfests 文件

```
# puppet -l /tmp/manifest.log manifest.pp
```

2). puppetd 运行在被管理主机上的客户端程序

```
# puppetd -server puppet.leju.com
```

3). puppetmasterd 运行在管理机上的服务器程序

```
# puppetmasterd
```

4). puppetca puppet 认证程序

```
# puppetca -l
pclient.leju.com
# puppetca -s pclient.leju.com
```

5). puppetrun 用于连接客户端，强制运行本地配置文件

```
# puppetrun -p 10 -host host1 -host host2
 -t remotefile -t webserver
```

6). filebucket 客户端用于发送文件到 puppet file bucket 的工具

```
# filebucket -b /tmp/filebucket /my/file
```

7). ralsh 转换配置信息到 puppet 配置代码

```
# ralsh user luke
user { ‘luke’:
home => ‘/home/luke’,
uid => ‘100′,
ensure => ‘present’,
comment => ‘Luke Kanies,,,’,
gid => ‘1000′,
shell => ‘/bin/bash’,
groups =>
 ['sysadmin','audio','video','puppet']
}
```

8). puppetdoc 打印 puppet 参考文档

```
# puppetdoc -r type >
 /tmp/type_reference.rst
# puppetdoc -outputdir /tmp/rdoc -mode
 rdoc /path/to/manifests
# puppetdoc /etc/puppet/manifests/site.pp
```

原文：

http://www.yoyotown.com/?p=574

推荐阅读：

系统管理员最需要自动化的十大任务

http://os.51cto.com/art/201010/230920.htm

Linux 装机利器 Cobbler 简述

http://os.51cto.com/art/201008/218652.htm

Linux 批量安装 五大开源软件挨个看

http://os.51cto.com/art/201008/218645.htm

超好懂的 PXE+Kickstart 批量安装 RHEL5 教程

http://os.51cto.com/art/201009/223294.htm

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：运维人与开发者眼中的 Linux。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/

《Linux 运维趋势》是由 51CTO 系统频道策划、针对 Linux/Unix 系统运维人员的一份电子杂志，内容从基础的技巧心得、实际操作案例到中、高端的运维技术趋势与理念等均有覆盖。