

# 网站集群架构

利用开源软件构建 高可用、高性能、  
可扩展 的集群系统



兰锋

[bluedata@gmail.com](mailto:bluedata@gmail.com)



## 摘要

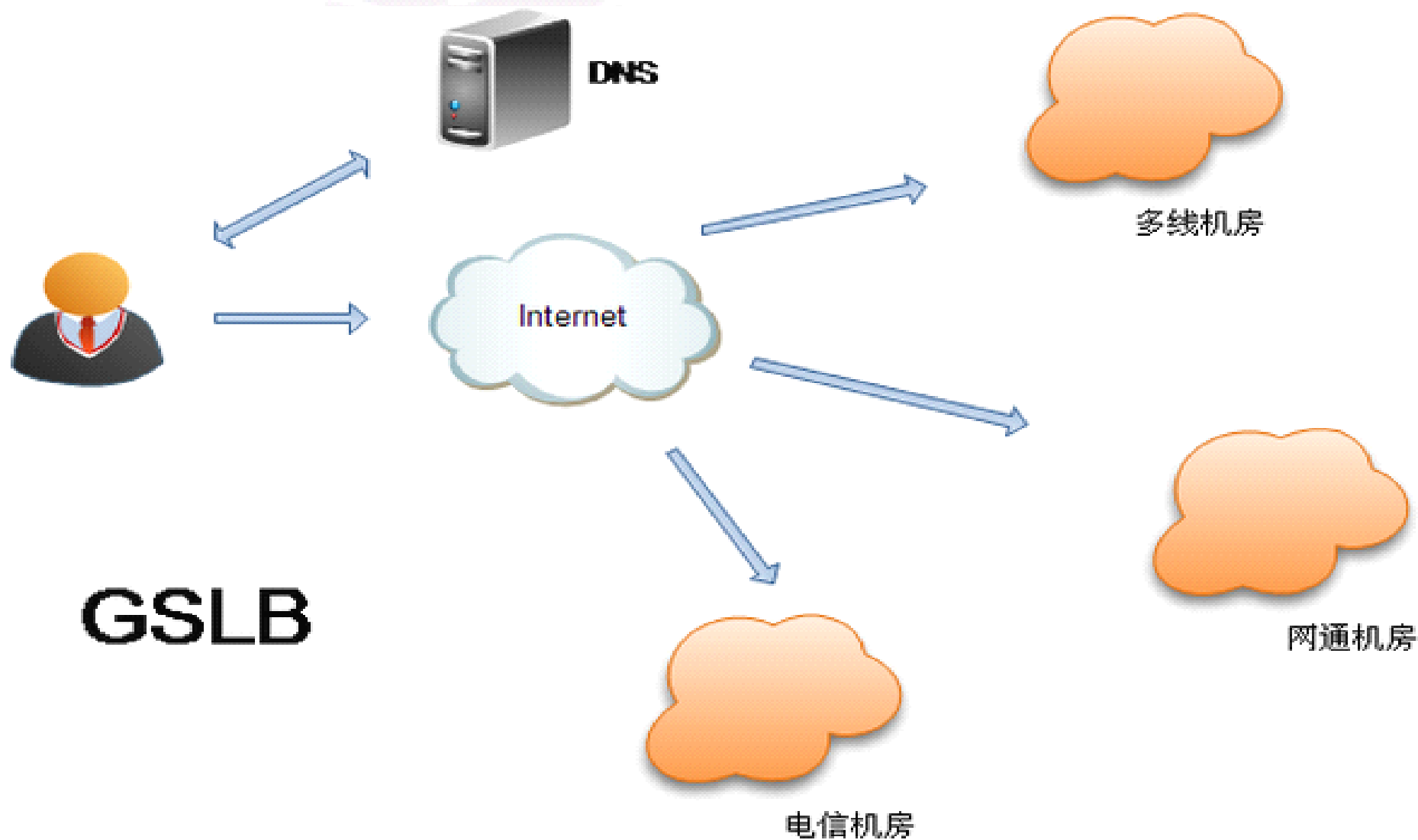
- 1. GSLB: 全局负载均衡
- 2. SLB: 服务器负载均衡
  - 四层交换 LVS
  - 七层交换 Nginx
- 3. Heartbeat 实现 HA
- 4. MySQL 数据库集群
- 5. 集群环境下的存储备份
- 6. 集群的监控及管理

# GSLB - Global Server Load Balance



- GSLB 是 Global Server Load Balance 的缩写，意思是全局负载均衡。
- 实现在广域网（包括互联网）上不同地域的服务器间的流量调配。
- 使用最佳的服务器（组）为最近的访问者提供服务，从而确保访问质量。

# GSLB 架构图





## 多 IDC 与单 IDC 的对比

- 好处
- 可用：不会因某一 IDC 机房由于“不可抗力”造成的网络中断而影响访问，可用性高。
- 容灾：可避免灾难性事件（比如地震）造成无可挽回的数据丢失。
- 速度：机房离访问者更近，访问质量更高。
- 坏处
- 实现复杂：流量分配问题，数据同步问题。
- 管理复杂：跨地区、距离远，维护麻烦。
- 成本较高：要在多个 IDC 租用机柜或机位。

# 利用 Bind9 的视图功能实现 GSLB 1



- cat named.conf
- ...
- include "acl\_chinanet.conf";
- view "chinanet" {
- match-clients { "chinanet"; };
- ...
- include "acl\_cnc.conf";
- view "cnc" {
- match-clients { "cnc"; };
- ...
- view "other" {
- match-clients { "any"; };
- ...

# 利用 Bind9 的视图功能实现 GSLB 2



- `cat acl_chinanet.conf`
- `acl "chinanet" {`
- `58.32.0.0/13;`
- `58.40.0.0/15;`
- `...`
- `222.222.0.0/15;`
- `222.240.0.0/13;`
- `};`
- `acl_cnc.conf` 则为网通的 IP 段。
- 然后分别定义各视图 zone 文件中域名所对应的 IP 地址。
- 详细原理可参考 GSLB using xBayDNS: <http://you.video.sina.com.cn/b/9144571-1435882772.html>



# SLB - Server Load Balancing

- SLB 是 Server Load Balancing 的缩写，意思是服务器负载均衡。可实现多个服务器之间的负载均衡。
- 当客户端向虚拟服务器（虚拟服务器是多个真实服务器的群集）发起连接时，通过均衡算法转发到真实服务器。

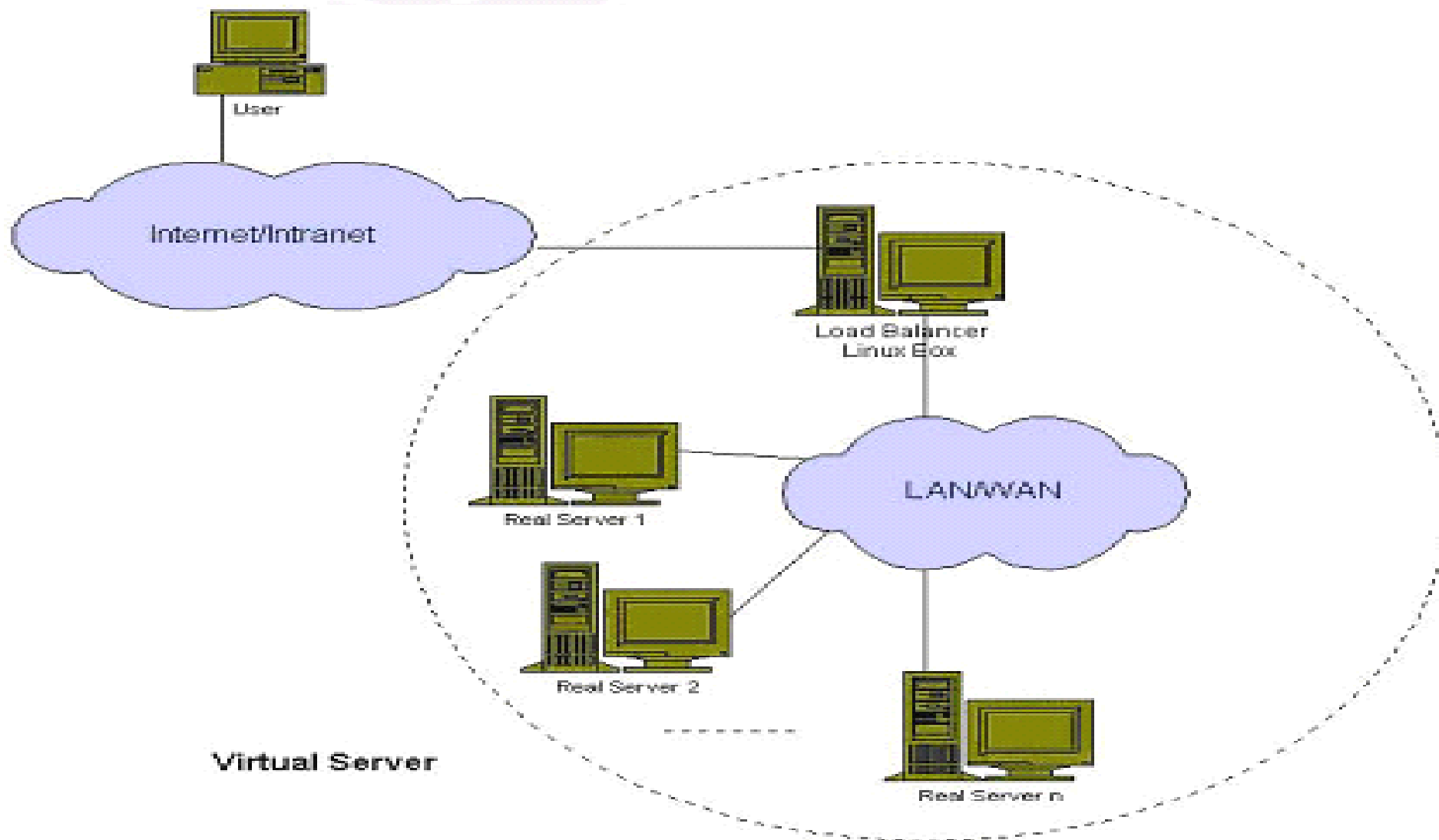




# LVS - Linux Virtual Server

- LVS 是在 Linux 内核中做四层交换。
- LVS 只用 128 个字节记录一个连接信息，512M 可用内存即可支持四百万条连接数，性能很高。
- Linux 2.6 内核已经包含了 ipvs 模块，只需安装 ipvsadm 。
- 真实服务器需要解决 ARP 问题（ ARP 缓存造成的 ARP “欺骗”）
- `arp_ignore = 1`      ## 只要这台机器上面任何一个设备有这个 IP，就响应 ARP 请求，并发送 MAC 地址应答。
- `arp_announce = 2`      ## 发送 ARP 请求时只使用本设备的 IP 地址和 MAC 地址，避免造成 ARP 欺骗。
- LVS 的有关中文文档：  
<http://www.linuxvirtualserver.org/zh/index.html>

# LVS 架构图





# Nginx

- Nginx ("engine x") 是一个高性能的 HTTP 和 反向代理 服务器，也是一个 IMAP/POP3/SMTP 代理服务器。
- 七层交换，大约能支持五万个并发连接。
- 可以通过定义 upstream 实现后端应用服务器（如运行 php-cgi 的机器）的负载均衡。
- ```
upstream php-fcgi {
```
- ```
    server 192.168.0.101:9000 weight=5;
```
- ```
    server 192.168.0.102:9000 weight=4;
```
- ```
    server 192.168.0.103:9000 weight=3;
```
- ```
}
```
- ```
location ~ .*\.php?$ {
```
- ```
    fastcgi_pass php-fcgi;
```
- ```
    include fastcgi_params;
```
- ```
}
```

# ngx\_http\_upstream\_hash\_module



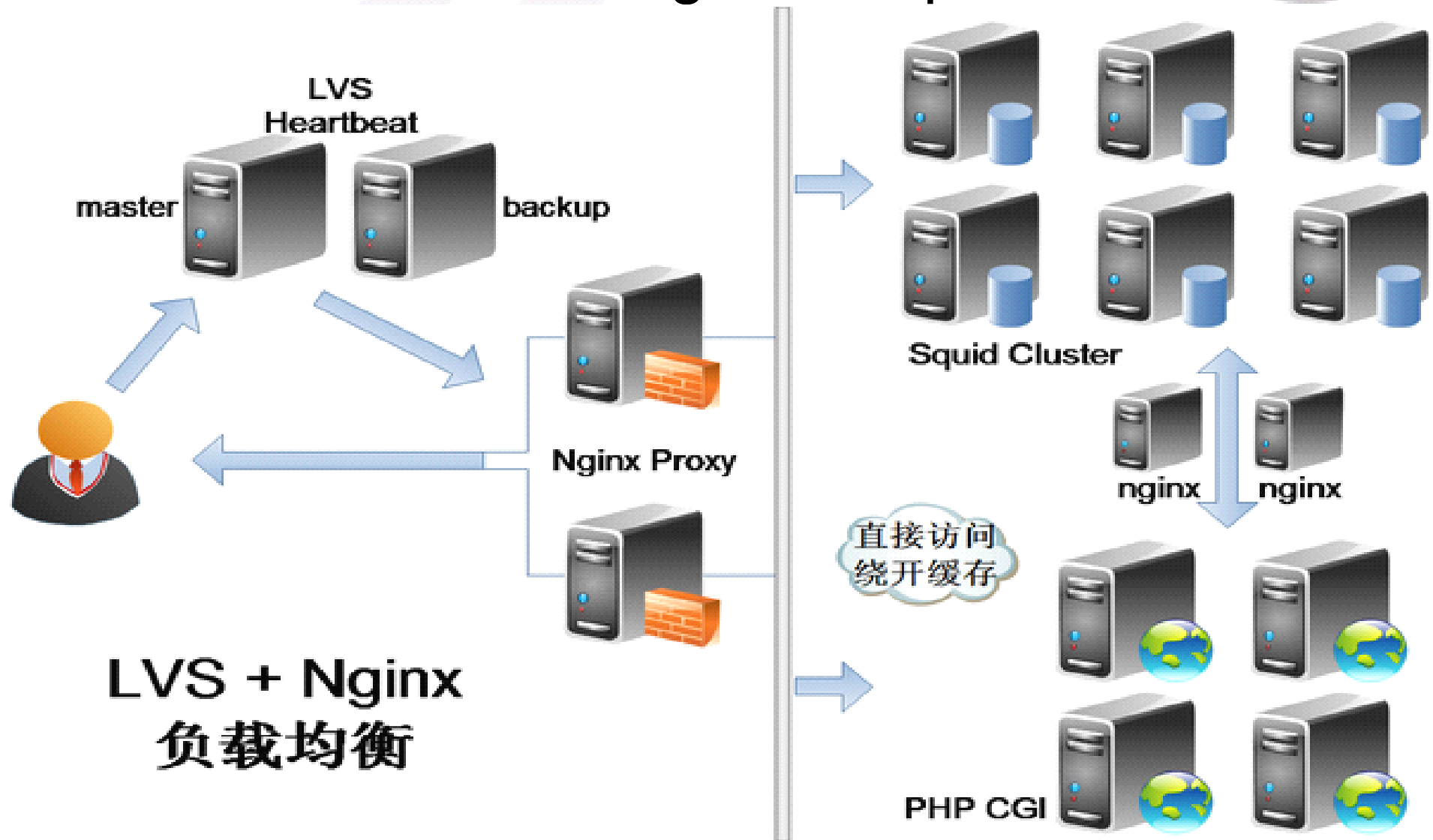
- url hash 是用于提高 squid 群集命中率的一种算法。
- 通过 DNS 或 LVS 可以把流量分配到多个 squid 节点，但每台 squid 的缓存数据是重复的。
- upstream squid {
  - server squid1:3128;
  - server squid2:3128;
  - hash \$request\_uri;
  - hash\_again 1;
- }
- 定义 hash\_again ， 在有 squid 节点失效时，会重新 hash 。
- Nginx 的中文维基: <http://wiki.nginx.org/NginxChs>



# Heartbeat

- Heartbeat 最核心的两个部分：心跳监测部分和资源接管部分。
- 心跳监测可以通过网络或串口进行，而且支持冗余链路，节点之间相互发送报文来告诉对方自己当前的状态。如果一个节点在指定的时间内未收到对方发送的报文，那么就认为对方失效。
- 这时需要启动资源接管模块来接管运行在对方主机上的资源或服务。
- 直接安装 heartbeat 包即可，CentOS 下还可安装 heartbeat-ldirectord 监测和管理 LVS 集群中的真实服务器（Debian 下是 ldirectord）。
- 使用网络监测要注意同一网络中存在多个 HA 应用时的广播 (bcast) 造成的干扰，这种情况下应该使用单播 (ucast)。
- 危险的脑裂 (split-brain)。
- Heartbeat 应用在操作存储设备的情况下是有一定的风险的。

# Heartbeat + LVS + Nginx + Squid + PHP





# MySQL 数据库集群

- 1. MySQL Cluster
- 2. MySQL 复制 之 一主多从 模式
- 3. MySQL 复制 之 多主模式

# MySQL Cluster 架构图



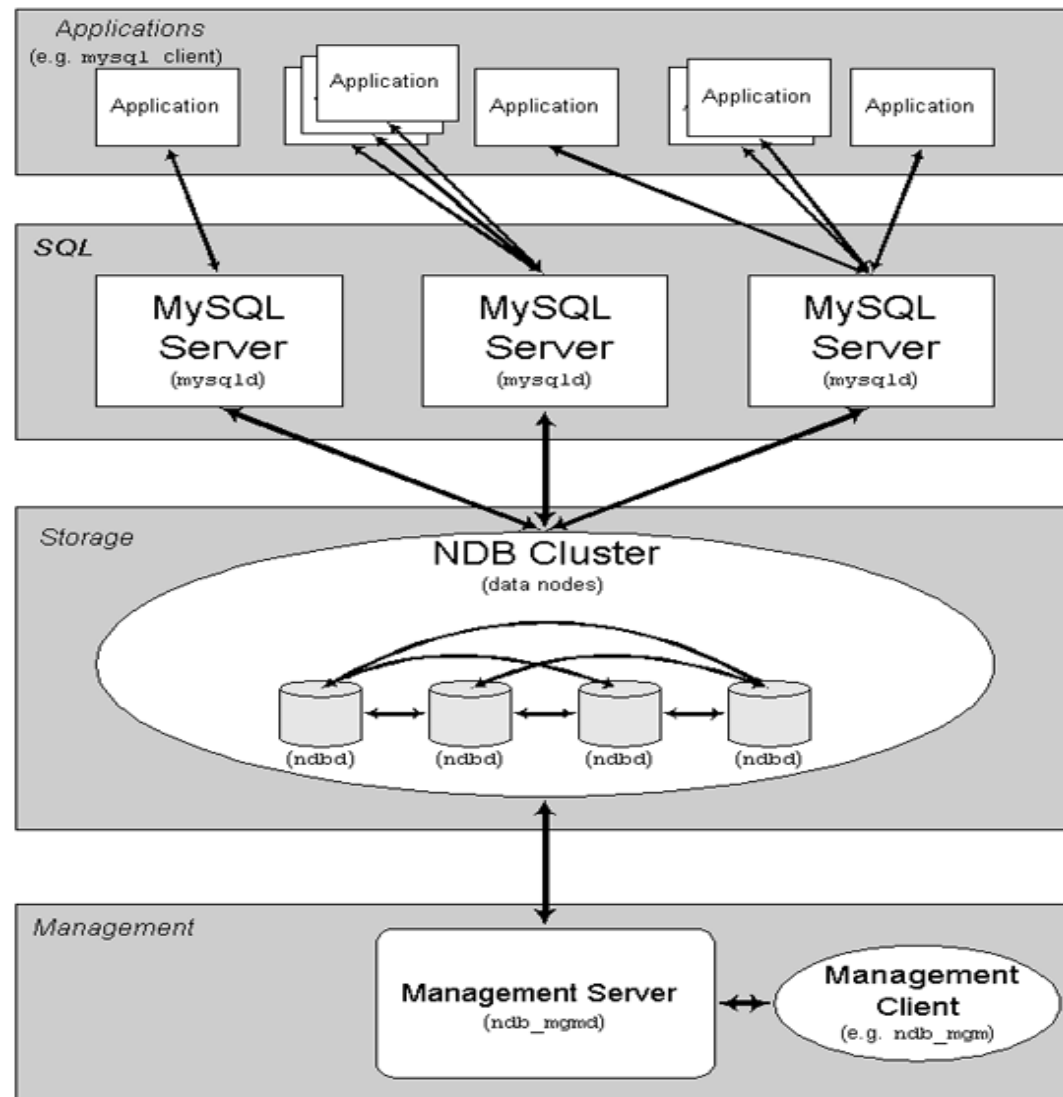
不用考虑读写分离问题。

SQL 节点需要使用 LVS 做流量的分配。

NDB 的性能较低（新发布的 MySQL Cluster 7.0 有所改善）。

稳定性尚有待提高。

生产环境下的应用实例太少。

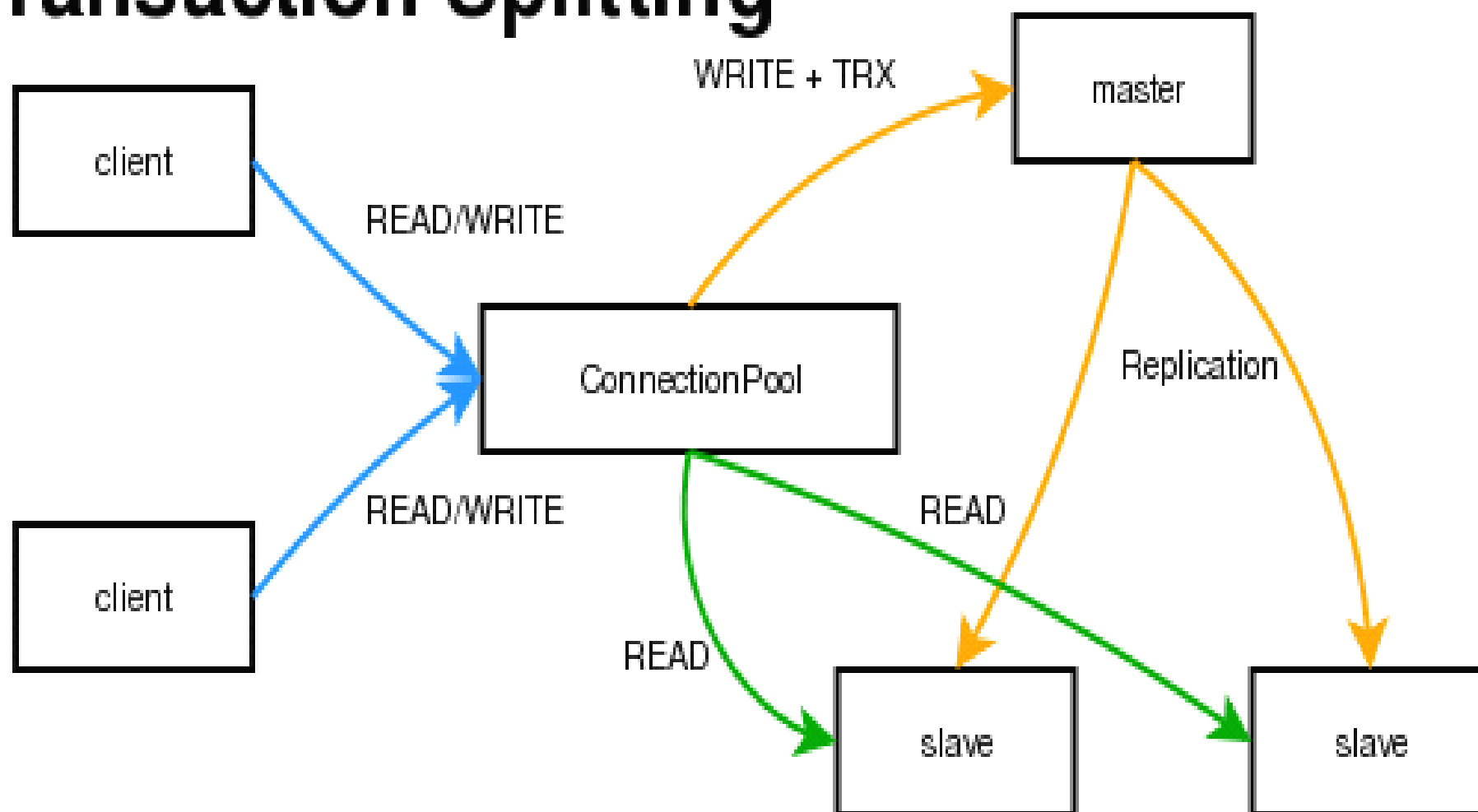






## MySQL 一主多从模式架构图

# Transaction Splitting

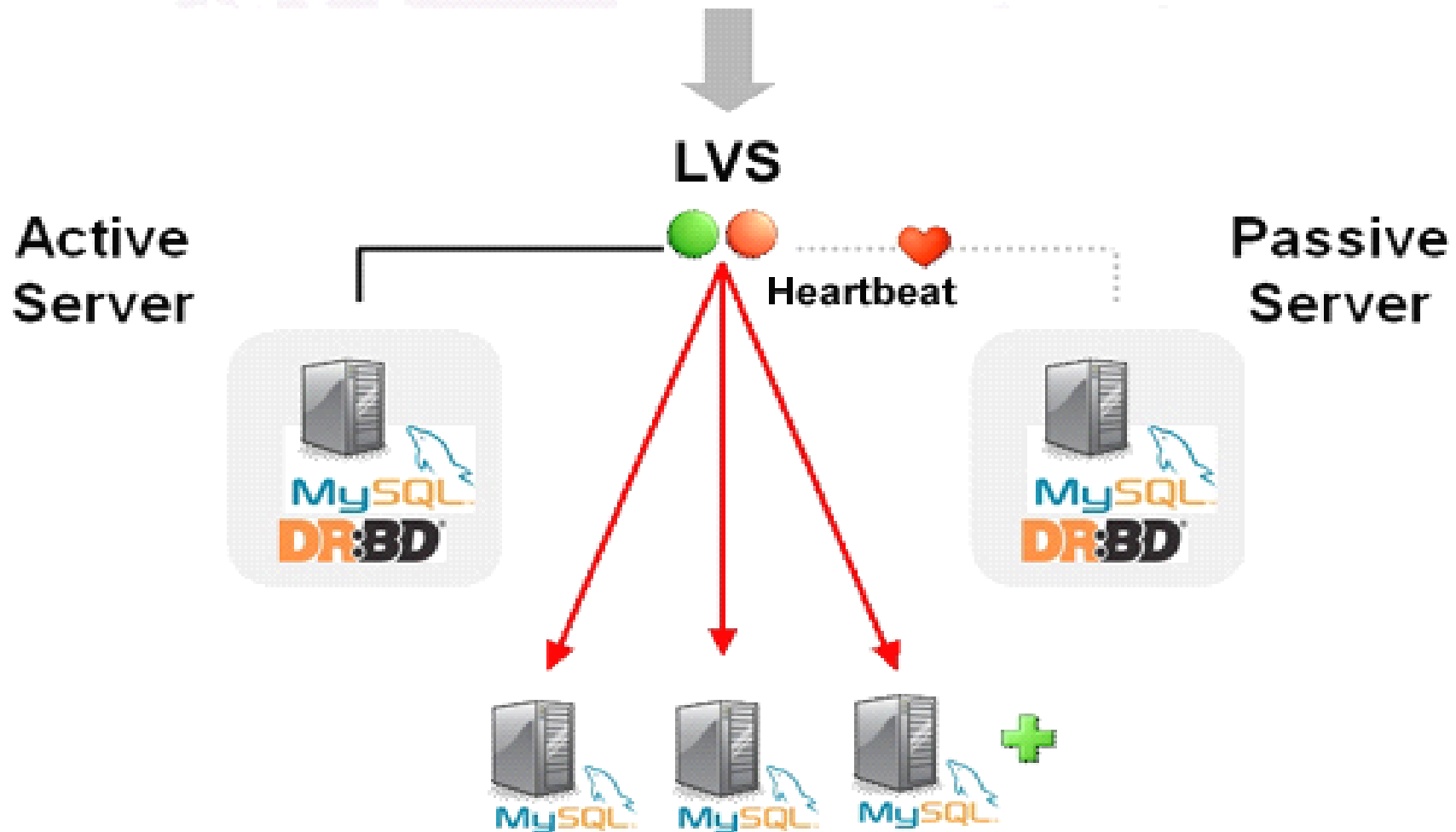




# MySQL 一主多从模式的要点

- 读写分离会使应用层变的更复杂。
- 可以用 MySQL-Proxy 实现读写分离，而多个 slave 可以由 LVS 调度。
- MySQL-Proxy 同样存在稳定性问题，多了一层结构当然也就多了一个故障点。
- 修改 数据库连接类 实现读写分离是较为常见的做法。
- 不能在 slave 上做实时查询，需要实时查询的应用只能走 master，这需要在应用层做区分。

# MySQL 一主多从模式的集群方案 Heartbeat + DRBD + LVS



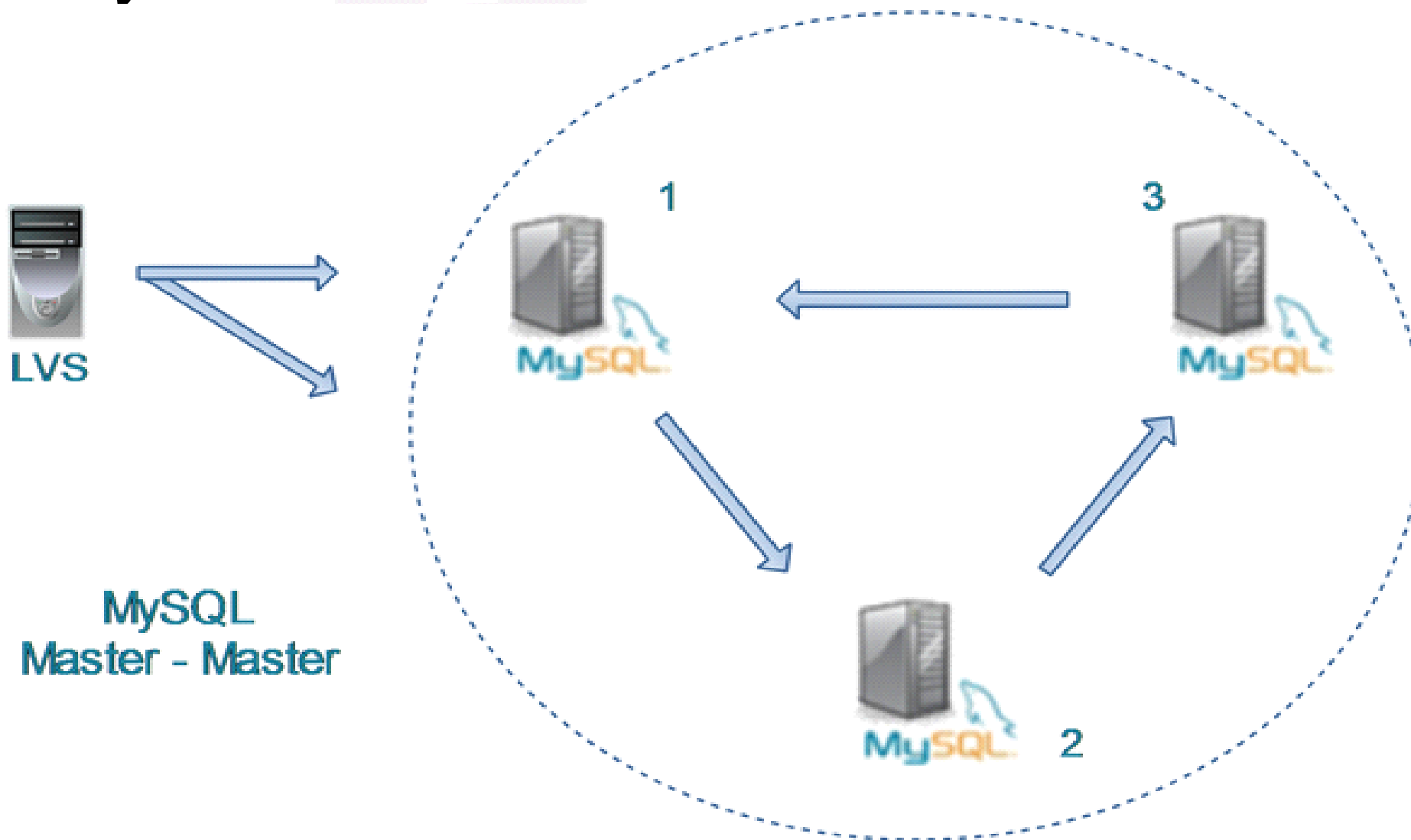
MySQL Replication Slaves – Read Scalability - Asynchronous

# MySQL + Heartbeat + DRBD + LVS



- MySQL + DRBD 是 MySQL 官方网站上推荐的一个方案。解决了 master 的单点问题。
- 用 Idirectord 管理 slave 可以有效的将失效节点移出集群。
- 由于 MySQL、文件系统和磁盘均有缓存功能，在 master 掉电的情况下，binlog 文件最后的 pos 要落后 slave 已经复制的 pos 很多。
- 这样就算备份节点接管了资源，仍然会造成复制的中断。需要手动恢复复制进程。
- 因 split-brain 造成资源争抢进而因错误写入造成存储的失效。恢复时间较长。

# MySQL 多主模式 架构图





# MySQL 多主模式的要点

- 不存在读写分离问题。
- 可扩展性较差，扩展较为麻烦。
- 自增 ID 的重复问题。
- 可以通过设定 `auto_increment_increment` 和 `auto_increment_offset` 来解决。
- 例如把 `auto_increment_increment` 设置为 3，而把 `master1`, `master2`, `master3` 的 `auto_increment_offset` 分别设置为 1, 2, 3，自增 ID 就分别为：
  - 1 4 7 10 13 ...
  - 2 5 8 11 14 ...
  - 3 6 9 12 15 ...

## MySQL 一主多从和多主模式的混合使用



- 多个 IDC 间由 master-master 模式同步数据。
- 同一 IDC 的 slave 从本 IDC 的 master 同步数据。
- 确保 master 之间的 binlog 一致，需要用到 google-mysql-tools 中的 MirroredBinlogs 。
- MirroredBinlogs 可以在 slave 保存一份和 master 完全相同的 binlog 。



# 集群环境下的存储备份

- NAS - Network Attached Storage （网络附属存储）
- 基本上可以直接叫做 Storage Server ， NFS 就是最常见的应用。
- 特点是简单易用，但性能有限。集群规模较小的时候可以使用。
- SAN - Storage Area Network （存储区域网络）
- 价格昂贵，在中小型网站中很少应用。
- ISCSI 价格相对较低，配合 RHCS + GFS 应用也比较常见。





## 网站应用存储文件的分类及特点

- 程序文件：更新少，数量少，频繁读。
- 用户文件：数量巨大，读写较多，读取分散。
- 数据库文件：文件大，读写操作极为频繁，连续写。



# 程序文件存储的解决方案

- NFS 是一个还算不错的选择，但有如下弱点：
  - 很严重的单点故障隐患。
  - 扩展性较差，支持的节点数量有限。
  - 大量不必要的网络传输。
- 根据程序文件读写的特点，可以考虑把文件放在本机。在文件更改时同步到各个节点。
- Rsync: 多节点的同步不方便；扫描目录树的改变比较慢。
- 可用以下两个工具解决上述两个问题：
- csync2(<http://oss.linbit.com/csync2/>): 集群同步工具。
- inotify-tools(<http://inotify-tools.sourceforge.net/>): 是为 Linux 下 inotify 文件监控工具提供的一套 C 的开发接口库函数，同时还提供了一系列的命令行工具，这些工具可以用来监控文件系统的事件。

# csync2 + inotify 实现集群文件即时同步 1



- Debian 下 csync2 的安装:
- apt-get install openssh-inetd csync2
- cat /etc/csync2.cfg
- nssl \* \*;
- group cluster1
- {
- host web1 web2 web3 web4;
- key /etc/csync2\_ssl\_cert.key;
- include /home;
- exclude /home/tmp;
- exclude \*~ .\*;
- auto younger;
- }
- 如果进行远程同步，为了安全，应该使用 ssl 连接，这需要安装 openssl 。

## csync2 + inotify 实现集群文件即时同步 2



- Debian 下 inotify-tools 的安装:
- `apt-get install inotify-tools`
- 可以在需要同步的节点上运行下面这个脚本进行同步了:
- `cat /usr/local/sbin/csync2.sh`
- `#!/bin/sh`
- `src=/home`
- `inotifywait -mrq --timefmt '%d/%m/%y %H:%M' --format '%T %w%f' \`
- `--exclude "\.swp$" \`
- `-e close_write,modify,delete,create,attrib \`
- `${src} \`
- `| while read file`
- `do`
- `csync2 -x > /dev/null 2>&1`
- `done`
- 用作即时备份也是个不错的选择。