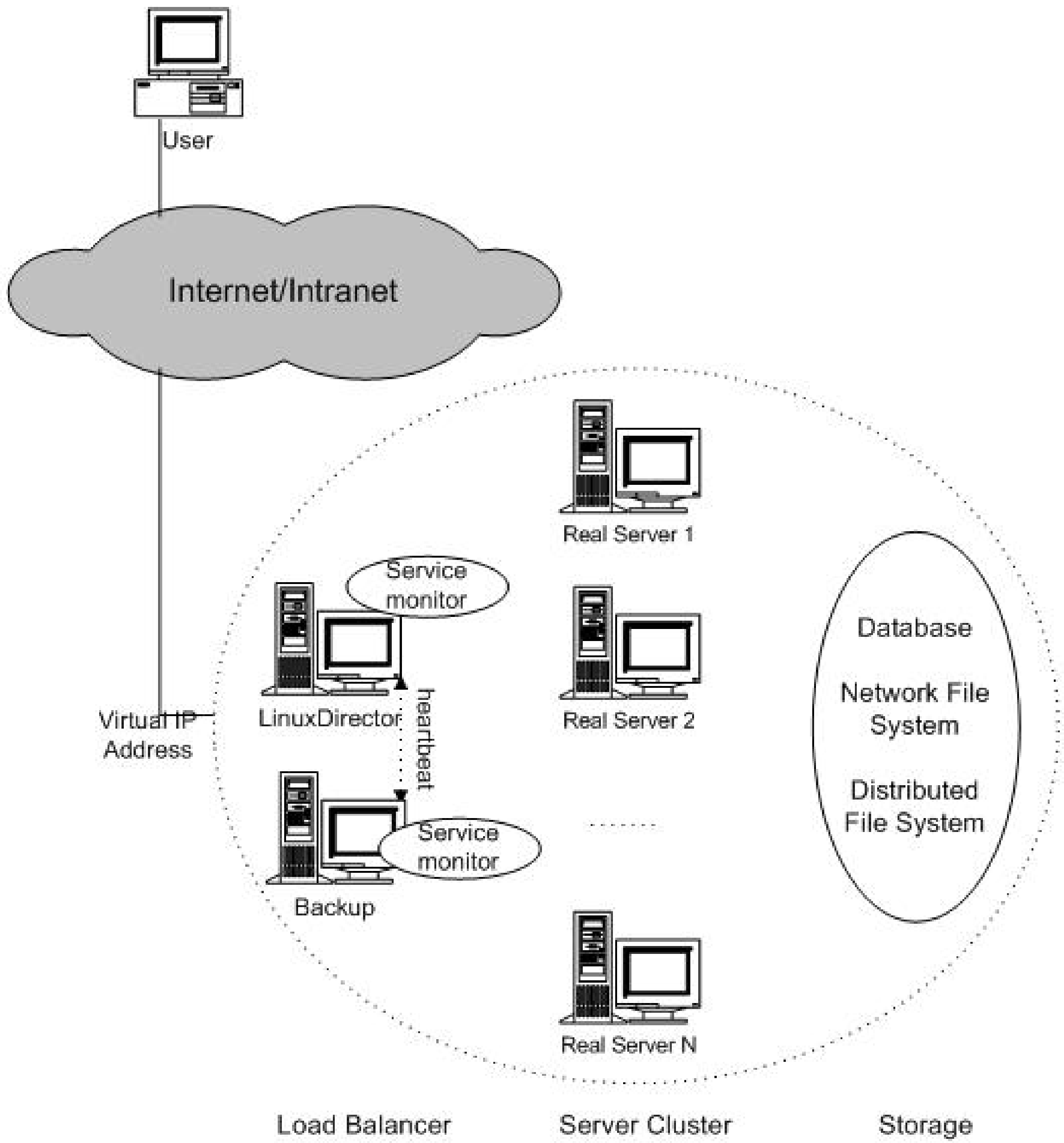


High Availability of Linux Virtual Server

Linux High Availability

In general, there are service monitor daemons running on the load balancer to check server health periodically, as illustrated in the figure of LVS high availability. If there is no response for service access request or ICMP ECHO_REQUEST from a server in a specified time, the service monitor will consider the server is dead and remove it from the available server list at the load balancer, thus no new requests will be sent to this dead server. When the service monitor detects the dead server has recovered to work, the service monitor will add the server back to the available server list. Therefore, the load balancer can automatically mask the failure of service daemons or servers. Furthermore, administrators can also use system tools to add new servers to increase the system throughput or remove servers for system maintenance, without bringing down the whole system service.

Linux High Availability

Now the load balancer might become a single failure point of the whole system. In order to prevent the whole system from being out of service because of the load balancer failure, we need setup a backup (or several backups) of the load balancer. Two heartbeat daemons run on the primary and the backup respectively, they heartbeat the message like "I'm alive" each other through serial lines and/or network interfaces periodically. When the heartbeat daemon of the backup cannot hear the heartbeat message from the primary in the specified time, it will take over the virtual IP address to provide the load-balancing service. When the failed load balancer comes back to work, there are two solutions, one is that it becomes the backup load balancer automatically, the other is the active load balancer releases the VIP address, and the recover one takes over the VIP address and becomes the primary load balancer again.

Linux High Availability

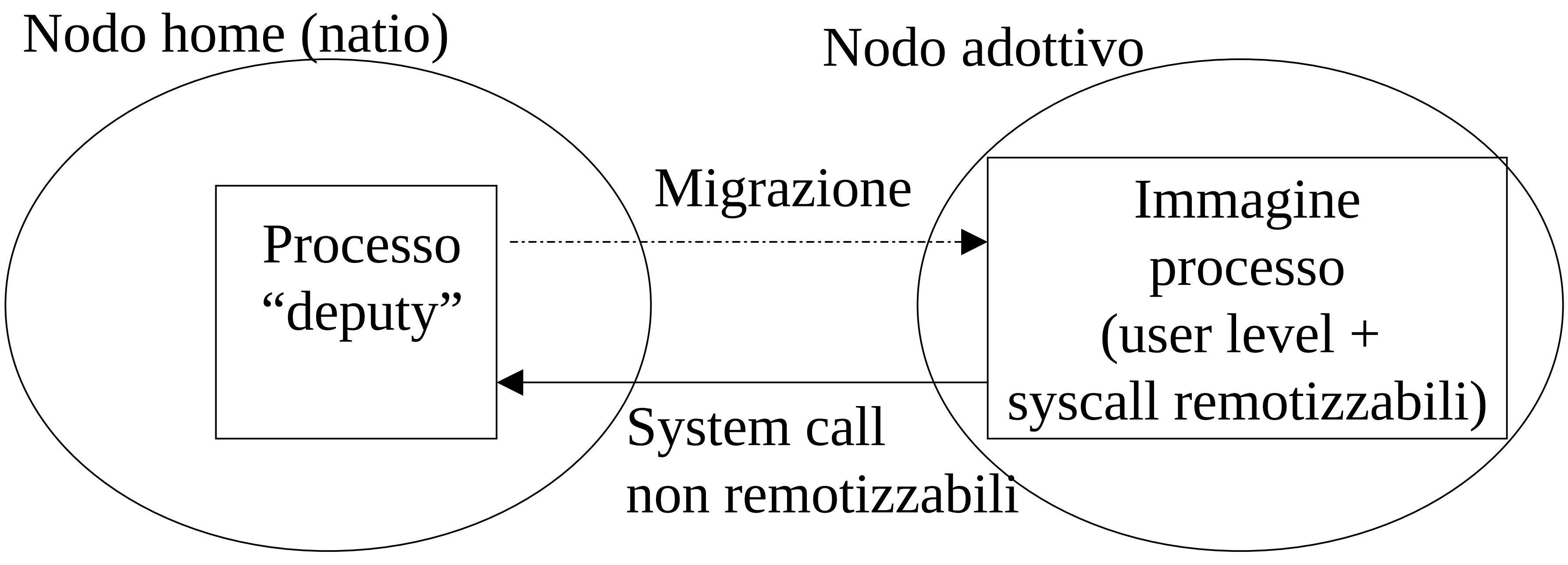
The primary load balancer has state of connections, i.e. which server the connection is forwarded to. If the backup load balancer takes over without those connections information, the clients have to send their requests again to access service. In order to make load balancer failover transparent to client applications, we implement connection synchronization in IPVS, the primary IPVS load balancer synchronizes connection information to the backup load balancers through UDP multicast. When the backup load balancer takes over after the primary one fails, the backup load balancer will have the state of most connections, so that almost all connections can continue to access the service through the backup load balancer.

Open Mosix

OpenMosix adds clustering abilities to the Linux* kernel that allow *any* standard Linux process to take advantage of a cluster's resources. By using adaptive load-balancing techniques, processes running on one node in the cluster can transparently "migrate" to another node where they can execute faster. Because openMosix is completely transparent to all running programs, the process that has been migrated doesn't even *know* (or *need* to know) that it's running on a remote system. As far as that remote process and other processes running on the original node (called the "home node") are concerned, the process is running locally.
This transparency of openMosix means that no special programming is required to take advantage of openMosix's load-balancing technology. In fact, a default openMosix installation will migrate processes to the "best" node automatically.

Open Mosix

OpenMosix, like an SMP system, cannot execute a single process on multiple physical CPUs at the same time. This means that openMosix won't be able to speed up a single process such as Mozilla, except to migrate it to a node where it can execute most efficiently. In addition, openMosix doesn't currently offer support for allowing multiple cooperating threads to be separated from one another.



Windows 2003

Microsoft Windows® Server 2003 Enterprise Edition now supports 8-node clusters (was two), and Windows Server 2003 Datacenter Edition now supports 8-node clusters (was four).

Windows Server 2003 provides no mechanism to mirror or replicate user data across the nodes of an MNS cluster, so while it is possible to build clusters with no shared disks at all, it is an application specific issue to make the application data highly available and redundant across machines.

Better Failover – in Windows 2000, if Node A owned the quorum disk and lost all network interfaces (i.e. public and heartbeat), it would retain control of the cluster, despite the fact that no one could communicate with it, and that another node may have had a working public interface. Windows Server 2003 cluster nodes now take the state of their public interfaces into account prior to arbitrating for control of the cluster.