# Providing Cluster Environments with High-Availability and Load-Balancing

M.A.R. Dantas[a], R.M. Alvim[b], F.L.L. Grossmann[b]

[a]Department of Informatics and Statistics (INE), Federal University of Santa Catarina (UFSC), 88040-900, Florianopolis, Brazil
[b]Department of Computer Science (CIC), University of Brasilia (UnB), 70919-970, Brasília, Brazil

[a]*mardantas@computer.org*

Cluster environments are usually provided at a low cost with an interesting performance when compared to parallel machines. However, some aspects of clusters configurations should be tackled to improve the environment to execute real applications. In this paper we present a framework in which we join functions of high-availability and virtual server to enhance services for application programmers using a cluster environment. Our results indicate that this approach can improve successfully this distributed system to execute real applications.

*Les grappes d'ordinateurs sont généralement peu dispendieuses, et donnent accès à une performance intéressante lorsqu'on les compare aux ordinateurs parallèles classiques. Mais, sous certains aspects, le processus de configuration de ces grappes doit être amélioré pour y permettre l'exécution d'applications courantes. Dans cet article, nous présentons un cadre qui combine des fonctions de haute disponibilité avec l'utilisation d'un serveur virtuel, dans le but d'améliorer les services accessibles aux programmeurs en environnement grappe. Les résultats que nous obtenons indiquent que cette façon de faire peut améliorer sensiblement la capacité d'une grappe à exécuter des applications courantes*

## 1. Introduction

Nowadays we cannot image any organization working without its computational systems even for a few minutes. A computational system not working for any fraction of time can represent an enormous lost for the organization. A redundancy approach it is necessary to avoid any risk to stop the computational system. The concern with this issue can be exemplified by IBM [1] and Microsoft [2,3], the two companies are already providing some features of reliability for their cluster solutions.

On the other hand, there are some researches on the open software arena. The idea is to demonstrate that open source solutions can provide interesting cost-efficient functions to cluster configurations. Examples of these environments are the *high-availability* (*HA*) [4,5] and *Linux virtual server projects* (*LVS*) [6,7].

The goal of the *high-availability project* is to provide a software package where a user can define one machine as a shadow of another computer. In other words, if any problem occurs with the first computer the second machine acts as a mirror, working in the same fashion as the original machine. This approach provides to programmers a more reliable environment to execute their applications.

The *virtual server* addresses the load balancing of a cluster configuration providing an even distribution of the workload among all machines of the environment. During an ordinary period of the cluster environment execution many requests are received to execute many tasks. Therefore, an element of the cluster could be in charge to redirect all the incoming tasks to the appropriate computers. An appropriate computer is a machine that has a low workload index. However, this approach has a drawback on the central element, which is responsible for redirects the tasks. This central function cannot work properly if the computer presents a failure.

In this article we present our research integrating functions of the *high-availability and virtual server projects.* Our target is to provide to application programmers the facility of virtual server with high-availability characteristics. This approach can enhance the vulnerability of the central element for distributing the workload, providing to programmers a reliable configuration. In section 2 we present some function characteristics of the *high-availability* and *virtual server projects.* Section 3 shows our cluster configurations and in section 4 we present some experimental results of the integration. Our conclusions and final comments of the research work are presented in section 5.

## 2. High-Availability and Virtual Server Projects

### 2.1 High-Availability (HA)

The high-availability project targets to provide reliability in distributed environment providing a shadow of one computer on a secondary machine. There are two important concepts when we consider especially the Linux High-Availability (LHA), these are the virtual address IP and the heartbeat [4]. The first concept (virtual address IP) means that the IP address is linked to a service and not to a certain host. During an interval of time, for example, services can be provided in computer **A**. In other interval of time, computer **B**, which is the backup service of computer **A**, can provide the same services because computer A presents hardware problems. In a cluster we can have many virtual IPs services, where each IP can be linked to one or more services [4]. The heartbeat concept is a process responsible for monitoring all services in a high-availability environment. The concept also provides the intranet services communications executing the message passing among servers [5]. Therefore, the heartbeat is the element that decides which server will be responsible for assuming a certain virtual IP address. The LHA works with deamons on both servers. After initializing the Linux, the heartbeat is initialized and checks if the servers are working. The virtual IP is created on the primary server and the machine exchange continuously messages. The procedure of message exchanging is used for check the availability of the servers.

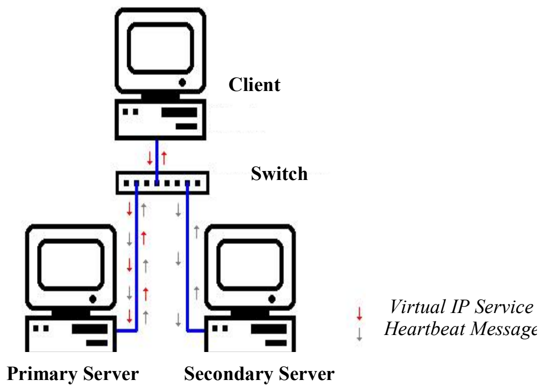


**Figure 1.** High-Availability working characteristics

Figure 1 shows the high-availability working characteristics, where three machines called *client, primary* and *secondary server* exists. The computer called *client* receives request of services for the cluster environment. The primary server receives the service of the virtual IP. On the other hand, the secondary server receives the messages from the heartbeat.

In the cluster environment we usually specify an interval of time that is valid for the messages exchange for the heartbeat function. When the interval is reached the high-availability software understand that the server of a specific service is out-of-order. Therefore, the secondary machine assumes all the tasks providing services, which were executing on the primary server. The approach of messages passing is not stopped because the primary server can run properly and assume all the services again. This is an example situation when the primary server returns to work.

## 2.2 Virtual Server (LVS)

The project call *Linux Virtual Server (LVS)* [6] is designed as an abstraction where inside a cluster environment only one computer can provides services to all incoming requests. This central host is called *virtual server*. An external host requesting services to this host *suppose* that all the tasks will be execute by this central node. However, the *virtual server* has two parts: one *load balancer* and others *n computers*. The function of the *load balancer* is to receive the external work requests and then to distribute among the others *n computers* of the *virtual server*. Reading the IP datagram, the *load balancer*, decides in which computer the task will execute. The external node does not know that another computer is executing the incoming request [7].

The main target of the *virtual server* approach is to prove high performance and high availability for distributed applications executing in the cluster. These two aspects are reached by the load balancing and redundancy functions. The first function executes a workload among the *n computers* of the configuration. The second feature it is provided by the *n computers* available in the cluster.

Figure 2 represents the *Linux Virtual Server* (LVS) working characteristics. This figure shows the *load balancer* and the others *n computers* (*Real Server*). We can consider an example in which a cluster is connected to the Internet. In this case incoming requests are received from the Internet through the *load balancer* and then it is distributed to each available computer of the cluster.

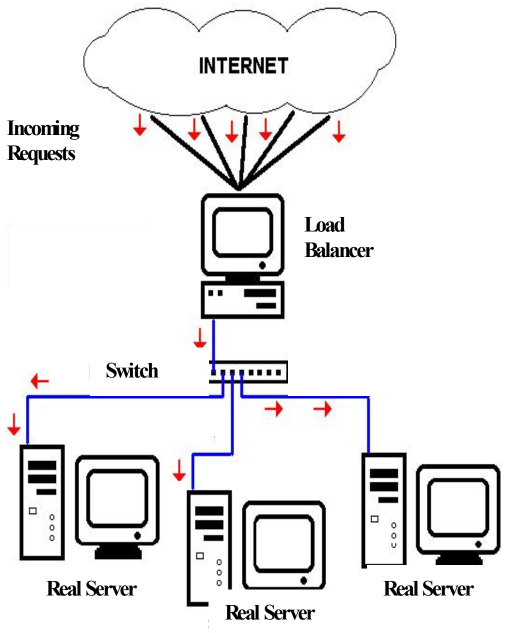


**Figure 2.** Linux Virtual Server working characteristics

In a virtual server environment the software responsible for monitoring and distribution of the tasks is called *ldirectord* [6,7]. Running into the *load balancer* this process uses the exchange of messages to identify computers to execute specific tasks and if all the *n computers* are working as expected. In the case of failure in a computer this machine will not receive any task before it shows that is working in a normal condition. After working as expected, the computer that had the failure is again inserted into the cluster to execute requests. The *ldirectord* can implement several scheduling policies for the load balancing. The usual, but not the unique, is the *round-robin* approach. Using this approach each machine of the cluster has a weight to execute a task (or service). In other words, this weight translates to the *load balancer* if a computer can execute *n* times more services than another computer.

Another important process in the *virtual server* configuration is the *IPVSADM*. This process execute in the *load balancer* providing vital information such as the cluster configuration and which tasks are executing in which computer.

## 3. Cluster Configuration

The cluster configuration used for our experiments is form by conventional IBM-PC machines executing the Linux operating system. This environment can be considered an interesting environment because it is an ordinary heterogeneous cluster configuration that is common in several laboratories.

The cluster configuration characteristics are presented in table I.

| | |
|---|---|
| **Processors** | **Pentium 100-100-233-233-233** |
| **Memory (Mbytes)** | **32-32-64-64-64** |
| **Operating system** | **Linux 6.0 kernel 2.2.17** |

**Table I.** Cluster configuration

## 4. Experimental Results

After understanding interesting features of the *high-availability* and *virtual server projects* we decided to extend the functionality of these environments, therefore we decide to integrate the two approaches. It is clear that the *load balancer* as a central element of the cluster receiving requests, it is a point that can occurs a failure. In this case, all the interesting functions of the cluster environment cannot be translated as a useful work.

In figure 3 we present our suggestion for integrating the HA and LVS functions for a cluster configuration. A secondary *load balancer* is connected to the cluster using the *high-availability* paradigm. This second *balancer* uses the *heartbeat* functions to monitor the primary *balancer.*

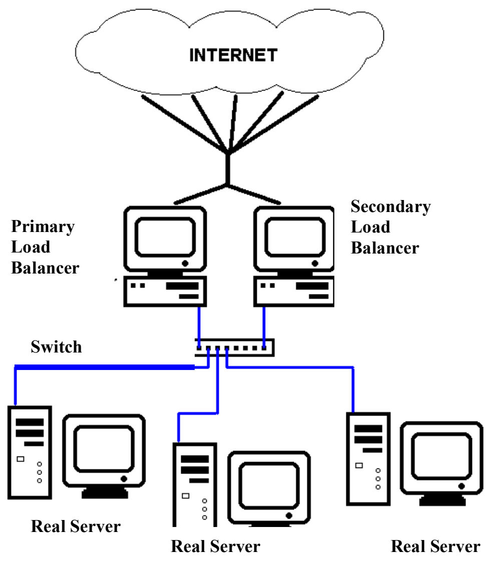


**Figure 3.** Integrating HA and LVS functions

All experimental results present in this section are based on our change of the cluster configuration. Therefore, we have created several study case situations to verify if our approach works as expected. Figure 4 shows the IP addresses and names used for our experiments. The integrated environment is form by the *load balancer*. The primary *balancer was* called *Laico Z* and the secondary as *Laico Y*. On the other hand, the *n computers* used were called as *Laico 2, Laico 3* and *Laico 4.*

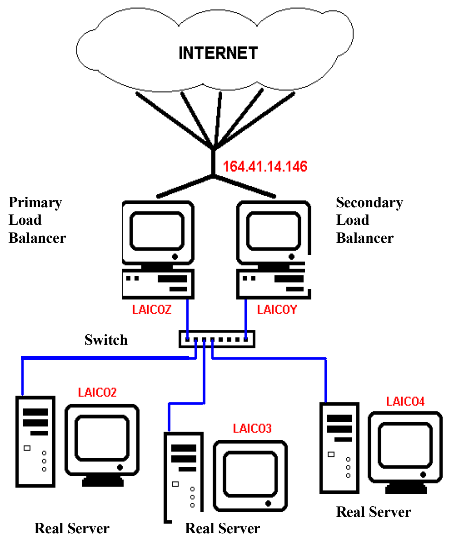
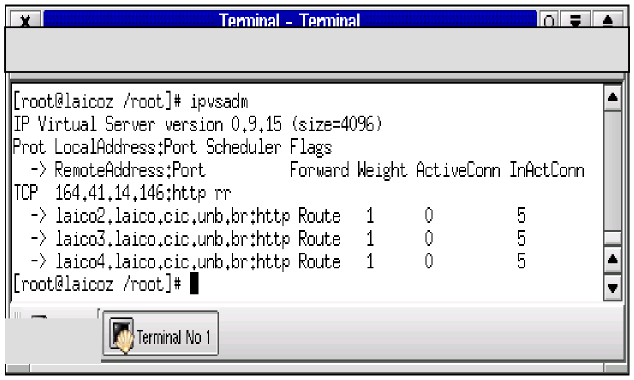


**Figure 4.** The integrated environment

In figure 5 we present a monitoring view of the cluster (using the IPVSADM) in which shows that all the *n computers* are working with the same weight and executing 5 processes.



```
[root@laicoz /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  164.41.14.146:http rr
  -> laico2.laico.cic.unb.br:http Route   1      0          5
  -> laico3.laico.cic.unb.br:http Route   1      0          5
  -> laico4.laico.cic.unb.br:http Route   1      0          5
[root@laicoz /root]#
```

**Figure 5.** Cluster with all servers working

As a first study case example we have simulated a problem in the Laico 2 computer shutting down this machine, as illustrated in figure 6. Just after shutting down the computer we have used the IPVSADM monitoring tool to evaluate the configuration status. In the figure 7 it is possible to verify that the Laico 2 is not available to provide services to the cluster configuration.

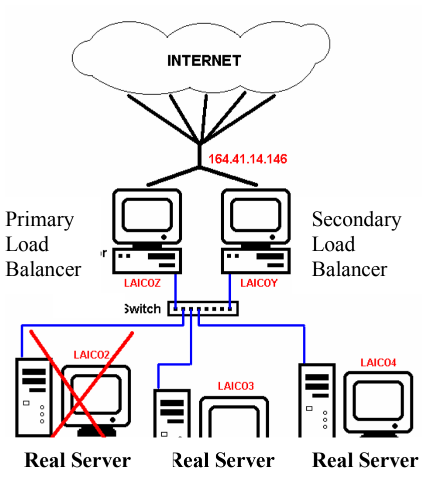
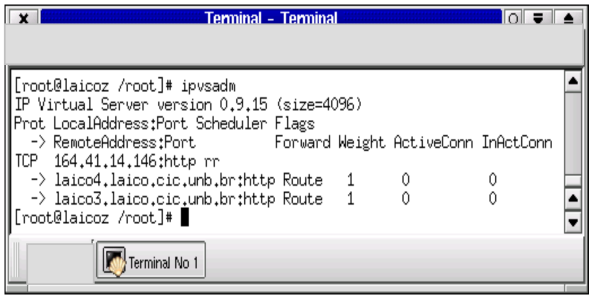


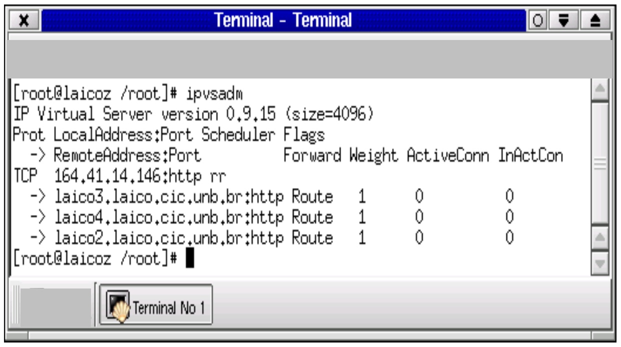
**Figure 6.** A study case without a machine Laico 2



```
[root@laicoz /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  164.41.14.146:http rr
  -> laico4.laico.cic.unb.br:http Route   1      0          0
  -> laico3.laico.cic.unb.br:http Route   1      0          0
[root@laicoz /root]#
```

**Figure 7.** Cluster without Laico 2 services

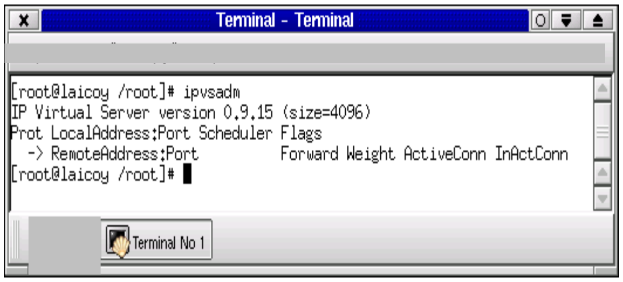
A few minutes later we decide to start up again the Laico 2 computer and verify how the cluster configurations would be presented. Figure 8 represents an output of the IPVSADM a few moments after Laico 2 works.



```
[root@laicoz /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  164.41.14.146:http rr
  -> laico3.laico.cic.unb.br:http Route   1      0          0
  -> laico4.laico.cic.unb.br:http Route   1      0          0
  -> laico2.laico.cic.unb.br:http Route   1      0          0
[root@laicoz /root]#
```

**Figure 8.** Cluster with Laico 2 services working

Another case situation that we have created was to simulate a failure in the primary *load balancer* (called as Laico Z). Before the simulation we have verified the status of the secondary *load balancer (*called as Laico Y). Figure 9 shows the status of the Laico Y computer before the failure on the primary server. This figure shows that no work has being assigned to this machine. The *heartbeat* is executing as expected. In other words, the computer is active but not executing any external task.



```
[root@laicoy /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
[root@laicoy /root]#
```

**Figure 9.** IPVSADM executing in the second load balancer

Figure 10 represents a failure in the primary *load balancer* (Laico Z). The expected situation is that the secondary *load balancer* (Laico Y) should take action to execute all the external tasks, which were running in the primary load balancer.

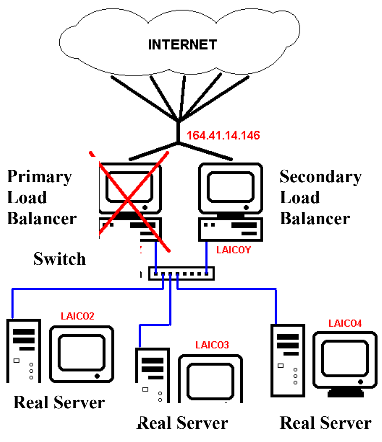
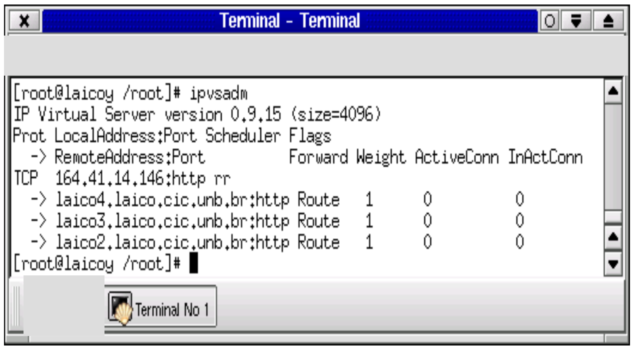


**Figure 10.** The primary load balancer presents problem

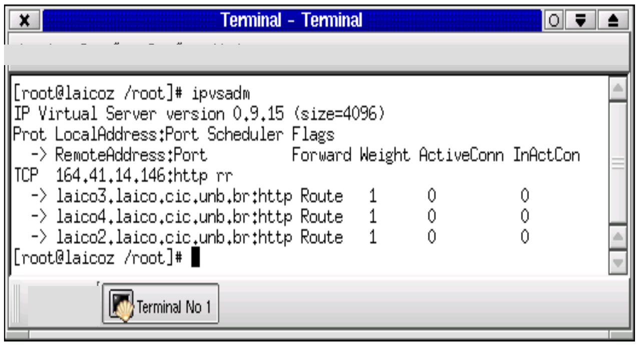
In the figure 11, we present the result of the IPVSADM monitoring tool, where it is possible to see that all the n computers of the configuration are under the secondary load balancer (Laico y) domain.



```
[root@laicoy /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  164.41.14.146:http rr
  -> laico4.laico.cic.unb.br:http Route   1      0          0
  -> laico3.laico.cic.unb.br:http Route   1      0          0
  -> laico2.laico.cic.unb.br:http Route   1      0          0
[root@laicoy /root]#
```

**Figure 11.** IPVSADM executing in the second load balancer

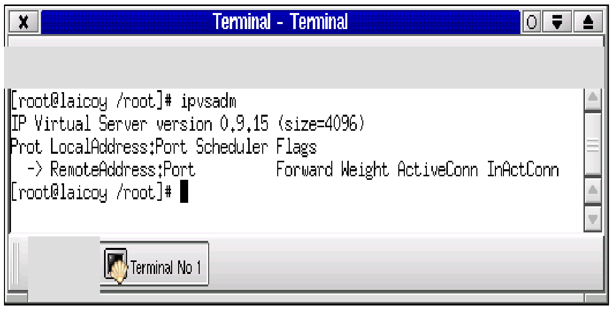
After a few minutes we have returned the normal situation, where the Laico Z is working without failure and this machine is the primary *load balancer*. Our case study is presented in figure 12 where the domain of the *n computers* is now under the Laico Z responsibility.



```
[root@laicoz /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActCon
TCP  164.41.14.146:http rr
  -> laico3.laico.cic.unb.br:http Route   1      0          0
  -> laico4.laico.cic.unb.br:http Route   1      0          0
  -> laico2.laico.cic.unb.br:http Route   1      0          0
[root@laicoz /root]#
```

**Figure 12.** IPVSADM executing in the primary load balancer

In parallel with the primary *load balancer* activity, figure 13 shows that the secondary *load balancer* (Laico Y) is not more responsible for the domain.



```
[root@laicoy /root]# ipvsadm
IP Virtual Server version 0.9.15 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
[root@laicoy /root]#
```

**Figure 13.** IPVSADM executing in the second load balancer

## 5. Conclusions and Future Work

The several experiments illustrated in our study cases indicate the integration success of HA and LVS. Integrating the *high-availability* and the *virtual server functions* for a distributed cluster can represent to this environment an interesting reliable configuration to execute several applications.

As a future work we are planning to submit the cluster environment to execute hundred thousand requests at the same time. Our goal is to verify how the load balancer works when millions of incoming requests are coming. In addition, it is interesting to use the integrated approach in a grid configuration where many failure problems occur and the number of requests is enormous.

## Acknowledgments

The first author's work was partially funded by the Brazilian National Research Council- CNPq (300874/006). The Computer environment used was from the LAICO laboratory and it was supported by the PROJECT INEG/CNPq-UnB.

## References

[1]IBM,http://www.ibm.com/servers/eserver/pseries/solutions/ha/hans.html, 2002.

[2] Microsoft, http://www.microsoft.com/windows2000/technologies/clustering/default.asp, 2002.

[3] Martin, Peter (p.martin@ies.uk.com) e Cronin, John (jsc3@havoc.gtf.org), 05/07/2001, http://www.microsoft.com/SERVICEPROVIDERS/solutions/haoview.asp, 2001.

[4] High Availability Project, http://www.Linux-HA.org, 2002.

[5] Milz, H., "HA-HOWTO" http://www.ibiblio.org/pub/Linux/ALPHA/linux-ha/High-Availability-HOWTO.html

[6] Linux Virtual Server Project, http://www.Linuxvirtualserver.org, 2002.

[7] Mack, J., "LVS-HOWTO", http://www.linuxvirtualserver.org/Documents.html#manuals, 2001.