

## Grid Computing and High-Availability

---

***Prof. Mario Dantas***  
***Department of Informatics and Statistics***  
***Federal University of Santa Catarina (UFSC)***  
***Florianopolis – Brazil***

E-mail : [mario@inf.ufsc.br](mailto:mario@inf.ufsc.br)

<http://www.inf.ufsc.br/~mario>

# ***An Evaluation of Globus and Legion Software Environments***

---

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

# **PART I**

## ***An Evaluation of Globus and Legion Software Environments***

# An Evaluation of Globus and Legion Software Environments

Dans cet article nous présentons une étude comparative des caractéristiques d'implémentation de deux environnements logiciels bien connus dans le monde du calcul distribué sur une "Grille" (GRID computing). Nous évaluons la performance de chacun de ces environnements pendant l'exécution en parallèle de tâches MPI distribuées. Une introduction des concepts entourant les calculs distribués sur une "Grille" est présentée, suivie de l'étude comparative des deux environnements logiciels, Globus et Legion, ces derniers étant les plus avancés dans le domaine. Nos résultats expérimentaux montrent que l'utilisation de la "Grille" peut s'avérer intéressante pour l'exécution d'applications parallèles MPI avec une certaine amélioration des performances.

# An Evaluation of Globus and Legion Software Environments

In this article we present a case study comparison of the implementation characteristics of two software environments which are well known in grid computing configurations. We evaluate the performance of these environments during the execution of parallel distributed MPI tasks. Therefore, first we consider some concepts of the grid paradigm and then we present a comparison between the two software environments. Our case study is based on the Globus and Legion environments, because these two research projects are in more developed stage when compared to other research initiatives. Our experimental results indicate that the grid computing approach can be interesting to execute parallel distributed MPI applications with a performance improvement.

# Agenda

- Computing Paradigms and Applications
- Users
- Grid Architecture
- Grid Computing Environments
- Experimental Results
- Conclusions and Future Work

# Agenda

- **Computing Paradigms and Applications**
- Users
- Grid Architecture
- Grid Computing Environments
- Experimental Results
- Conclusions and Future Work

# What is a Grid Computing ?



## **What is a Grid Computing ?**

A Grid is a computational high-performance environment which is characterized by resource sharing providing services for organizations geographically distributed.

# What is a Grid Computing ?

A Grid can also be view under the following physical aspects:

- Better utilization of bandwidth
- The use of a great computational power
- Fast access to data, software and remote facilities with QoS.
- Better utilization of remote CPUs, memories and disk spaces.

# What is a Grid Computing ?

***A Grid is parallel and distributed computational system***

***that enables***

***the sharing, selection and aggregation of geographically distributed autonomous resources dynamically at runtime***

***depending on their availability, capability, performance, cost, and providing users' applications with their requirements of quality-of-service.***

# What is a Grid Computing ?

We can also say :

*Grids target to exploit synergies*

*that result from  
cooperation-ability to share and aggregate  
distributed computational capabilities  
and deliver them as*

**services.**

# What is the difference between Cluster and Grid Computing ?

# What is the difference between Cluster and Grid Computing ?

A important difference between clusters and grids is mainly based in the way *resources are managed*.

In the clusters, the resource allocation is performed by a centralized resource manager and all nodes cooperatively work together as *a single unified resource*.

Inside the Grids, each node has its own resource manager and do *not target* for providing *a single system view*.

# Computing Paradigms and Applications

**The experimental research with the *I-WAY*, first large scale Grid effort, bring to us that there were five classes of applications using a specific computing paradigm.**

# Computing Paradigms and Applications

*Computing paradigms* and *applications* can be classified as following :

- **Distributed Supercomputing**
- **High-Throughput Computing**
- **On-Demand Computing**
- **Computing for Large Amount of Data**
- **Collaborative Computing**



# Computing Paradigms and Applications

## 1. Distributed Supercomputing

Applications that use this approach can be characterized by the fact that it is not possible to solve these applications in a single computational system.

The aggregation environment which we are referring to can be represented by all the supercomputers of a specific country or all the workstation inside of an organization.

# Distributed Supercomputing

Examples of applications using the *distributed supercomputing* approach are :

- *Distributed Interactive Simulation (DIS)* : this is a simulation technique used to model the behaviour and movement of hundred (or thousand) of entities which are usually employed for military planning and teaching.

# Distributed Supercomputing

- Simulation of complex models such as those in weather forecast and cosmology.

# Computing Paradigms and Applications

## 2. High-Throughput Computing

The main objective of this approach is to solve the problem of applications that require a transfer of a large amount of data.

The computational environment is used for scheduling a large number of loosely coupled tasks and enhance the utilization of machines with a low workload.

# High-Throughput Computing

Classical examples for *high-throughput computing* are :

- *Condor High-Throughput* – this software environment from the University of Wisconsin is used to manage *pools* of hundreds workstations in the university and other labs around the world.

# High-Throughput Computing

- The *Platform Computing software* - used by AMD during the projects of K6 e K7 processors. It is reported that the company has used all the desktops which were not in use by the engineers in a specific period of time.

# Computing Paradigms and Applications

## 3. On-Demand Computing

This class of applications usually can be characterized by the use of resources that can not be used in the local site, because it is not available.

The *resources* can be *computing, data streams, software, archives* and for examples *experimental results*.

# Computing Paradigms and Applications

## 3. On-Demand Computing

Difference between this approach and distributed Supercomputing is related to the cost of performance then the complete performance behaviour.



# Computing Paradigms and Applications

## 4. Computing for Large Amount of Data

This class of application and computing paradigm covers the requirement for processing large amount of data stored in a geographic distributed fashion.

Examples are large databases and digital libraries that are available for access in a distributed way.

The Digital Sky Survey and modern weather forecast Systems are applications examples.

# Computing Paradigms and Applications

## 5. Collaborative Computing

Examples for this class are those which are oriented to the improvement the relation between humans.

Many collaborative applications allow the share use of computational resources.

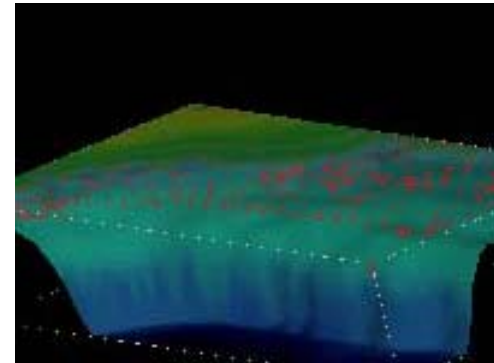
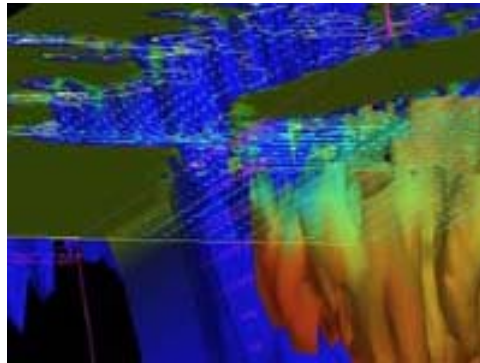
## Computing Paradigms and Applications

NICE is a collaborative learning environment for young children (approximately 6-8 years of age). The environment depicts a virtual island in which the children can tend a virtual garden and learn about environmental concepts.



# Computing Paradigms and Applications

## Cave5D



# Agenda

- Computing Paradigms and Applications
- **Users**
- Grid Architecture
- Grid Computing Environments
- Experimental Results
- Conclusions and Future Work

## Users

**Another approach used to understand what is a Grid, is to understand who is going to use.**

**A Grid is above of the mechanisms of resource sharing therefore we can image two questions :**

A - Which kind of entity is going to invest in the infrastructure for a Grid ?

B - Which kind of resources each community of the entity will be share ?

## **Users**

**Answers for the two questions should be based on costs and benefits for sharing resources.**

**Therefore it is usually presented in the academic and commercial reports efforts for the following groups of grid environments :**

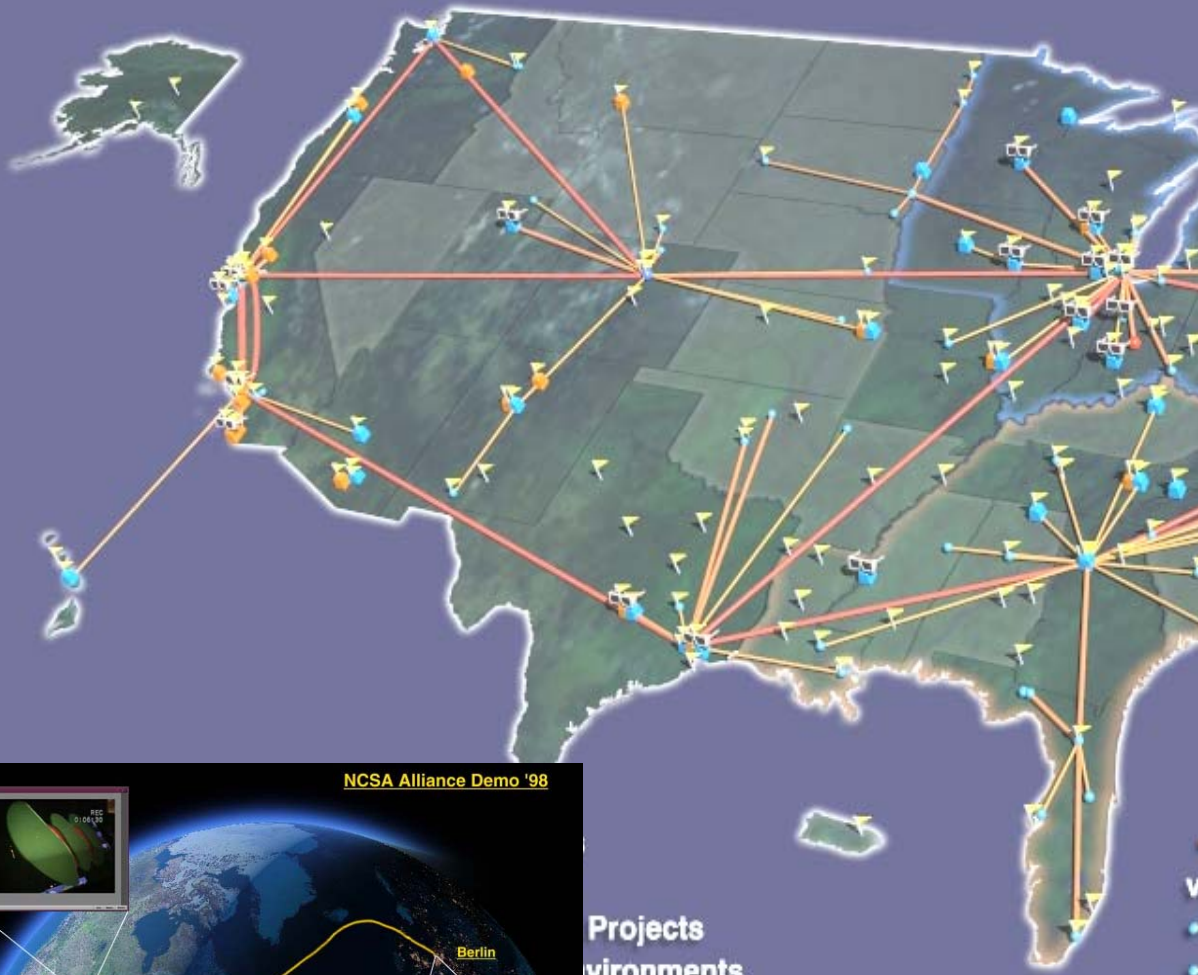
- **National Grid**
- **Private Grid**
- **Virtual Grid**
- **Public Grid**

## Users

- ***National Grid*** – the target of this group is to be a strategic computational resource and serve as a bridge between national sharing facilities.
- ***Private Grid*** – the health community it is an example of private grid organization. This group, was identified to benefit from grid configurations because of the strategic utilization of computational power.

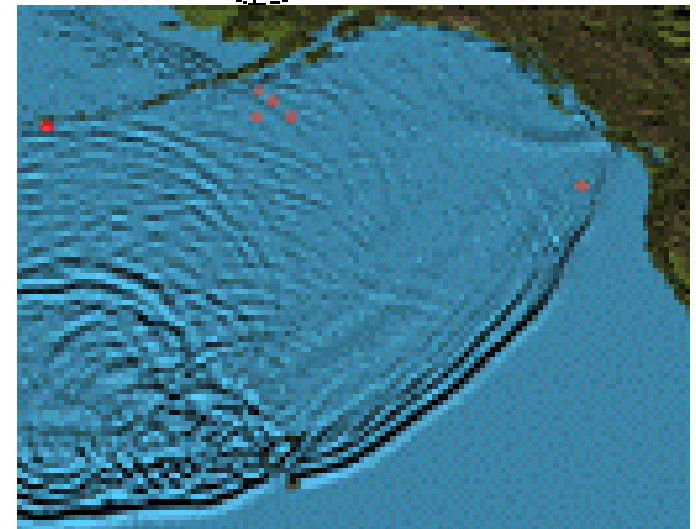
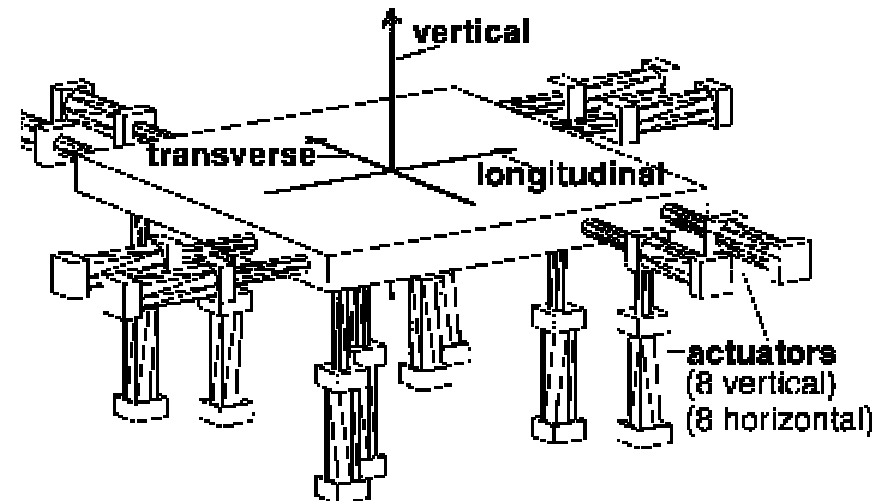


# NSF National Technology Grid



# Network for Earthquake Engineering Simulation

- NEESgrid: national infrastructure to couple earthquake engineers with experimental facilities, databases, computers, & each other
- On-demand access to experiments, data streams, computing, archives, collaboration

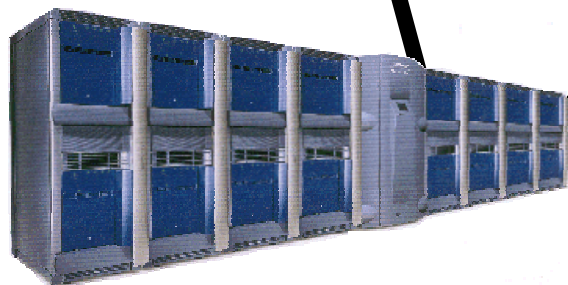


NEESgrid: Argonne, Michigan, NCSA, UIUC, USC

## Users

- *Virtual Grid* – this community is formed by researches and scientists which require the use of expensive equipments and a great computational power.
- *Public Grid* – this group is basically characterized by those which the main activity includes services using a great quantity of computational power.

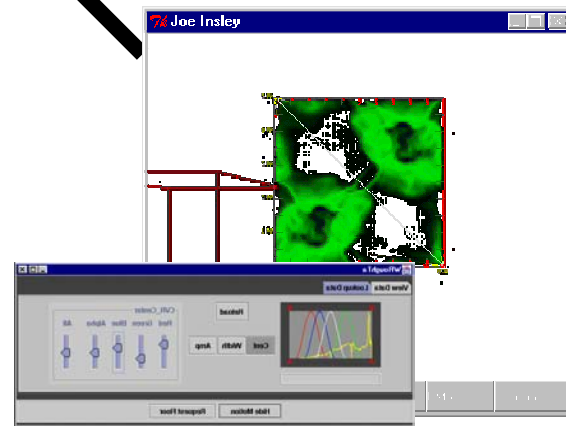
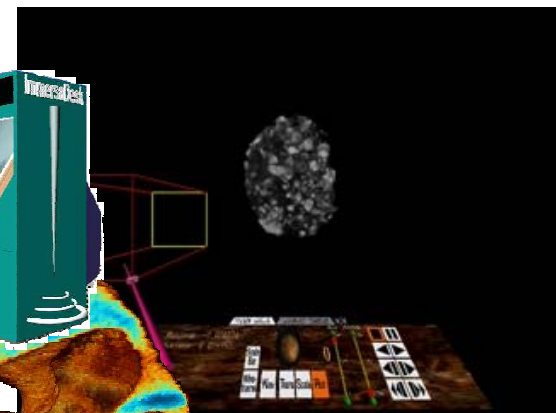
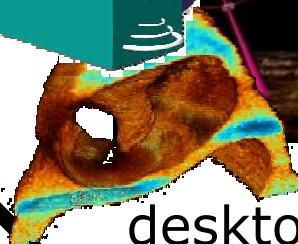
# Online Access to Scientific Instruments



wide-area dissemination

archival storage

desktop & VR clients with shared controls



DOE X-ray grand challenge: ANL, USC/ISI, NIST, U.Chicago

# Data Grids for High Energy Physics

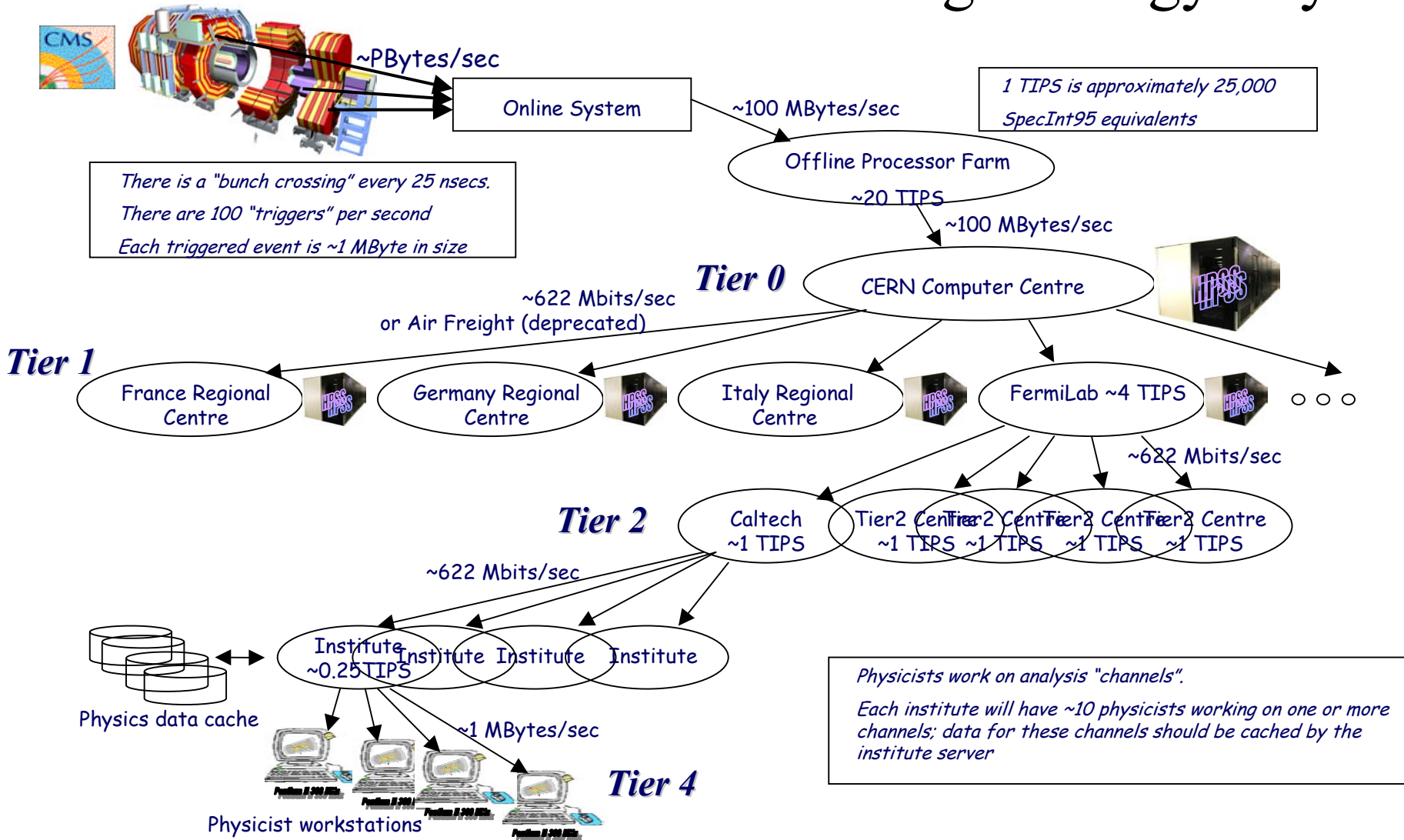


Image courtesy Harvey Newman, Caltech

# Agenda

## Computing Paradigms and Applications

- Users
- **Grid Architecture**
- Grid Computing Environments
- Experimental Results
- Conclusions and Future Work

# Grid Architecture

Before we start to study the Grid architecture it is interesting to know about *Virtual Organizations (VO)*.

*Virtual organizations* are the entities that share resources of the Grid under a specific policy .

Examples of **VO** are :

- Providers of applications, data storage and computational power.
- Research organizations
- Universities

# Virtual Organizations

*Virtual Organizations* are different from each other considering the following parameters :

- Main objective
- Geographic extension
- Size (or physical dimensions)
- Time to use the facilities
- Structure
- Community



# Grid Architecture

Similar to the experience with *Internet*, researches involved with the Grid established an architecture aiming the interoperability between *VOs*.

# Grid Architecture

Aspects such as :

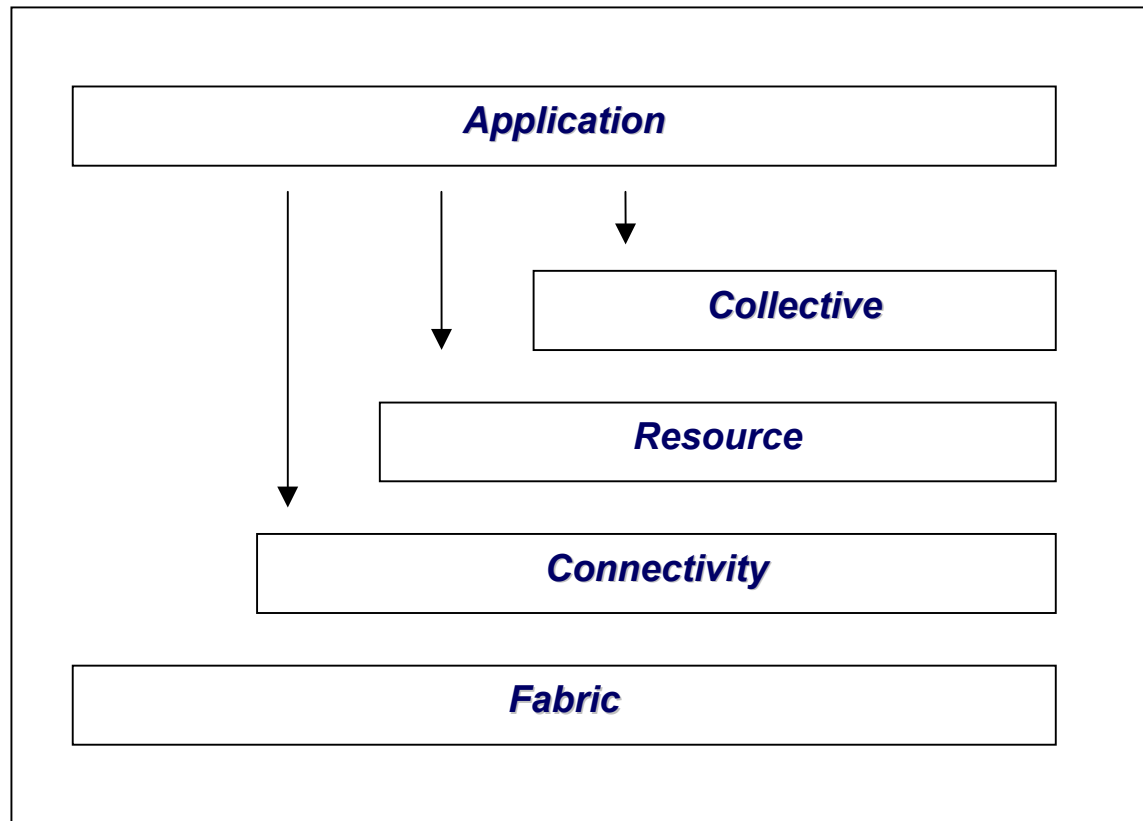
- authentication,
- authorization,
- mechanism of message passing,
- resource sharing,
- scheduling and
- load balancing of tasks

are some of issues which a Grid architecture should provide.

**A standard Grid architecture was proposed as :**

ÿ

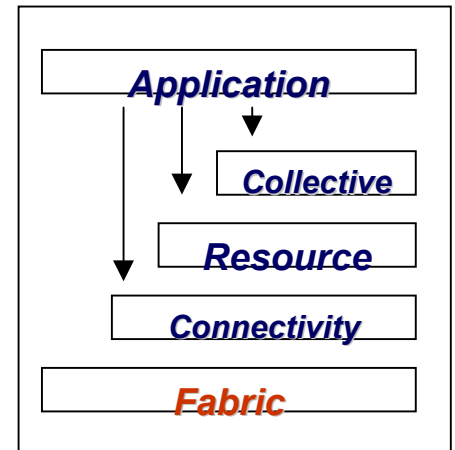
# Five Layers Grid Architecture



# Grid Architecture - LAYERS

## *Fabric* –

Components of this layer implement local operations which occurs in each resource mainly because of the sharing provided by the above layers.



# Grid Architecture - LAYERS

## *Fabric* –

Mechanisms are necessary to obtain information about the structure, state and available resources.

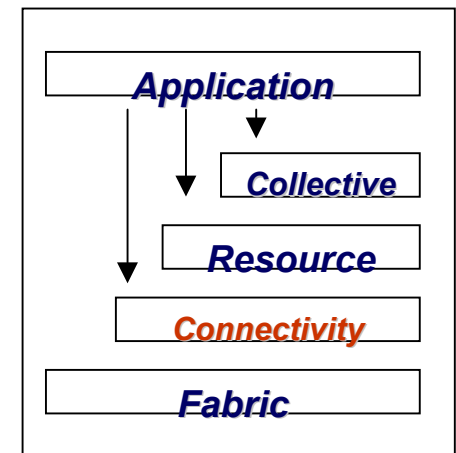
On the other hand, it is also important techniques to management the QoS (Quality of Service) for each query.

# Grid Architecture - LAYERS

## *Connectivity*

In this layer exists the definition of the basic protocols necessary for communication and authentication for a specific transaction of the Grid.

The communication protocols allow the data exchange between the *Fabric layers*. This service includes the transport, routing and name services.



# Grid Architecture - LAYERS

## *Connectivity*

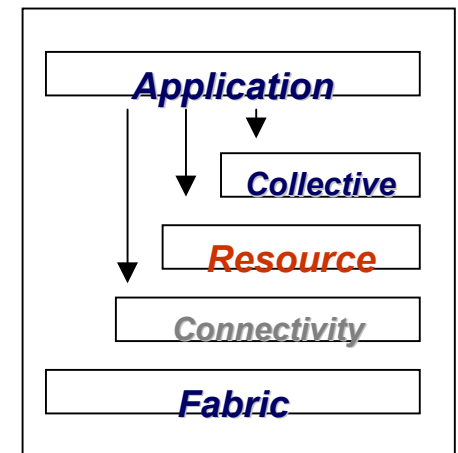
The authentication protocols are responsible for building the communication services which are way to prove secure mechanism to verify the identity of users and resources



# Grid Architecture - LAYERS

## *Resource*

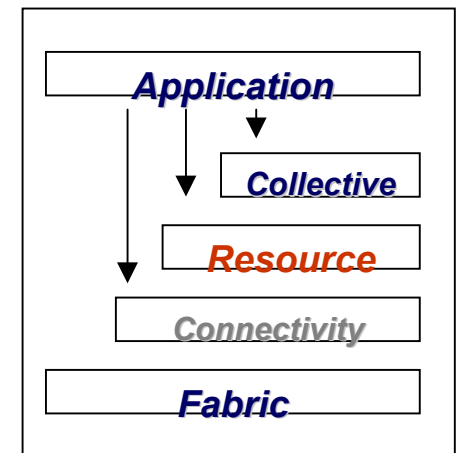
This layer uses the *connectivity* protocols (communication and authentication) to define protocols and APIs to provide security during the negotiation, starting, control, monitoring, creating reports and details involved during the individual resources operations.



# Grid Architecture - LAYERS

## *Resource*

Protocol implementations of this layer utilizes calls from the *Fabric* to access and control local resources.

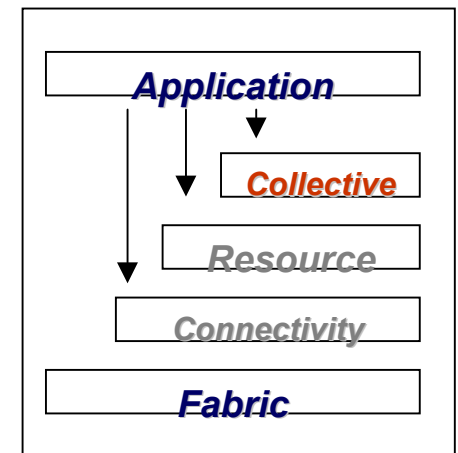


# Grid Architecture - LAYERS

## *Collective*

The *resource layer* treats the scope of individual resource operations.

On the other hand, in the *collective layer* components work with the interaction of resource collections.

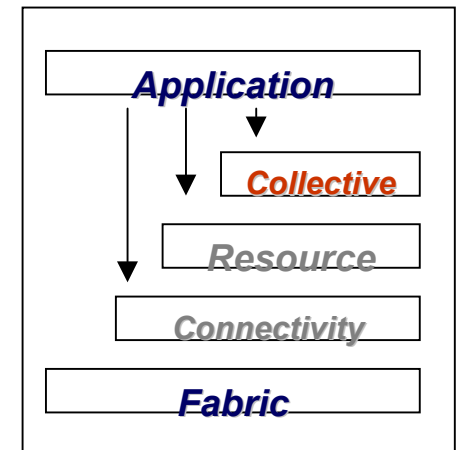


# Grid Architecture - LAYERS

## *Collective*

The elements from this layer use the *resource* and *application* layers to implement a variety of services, such as :

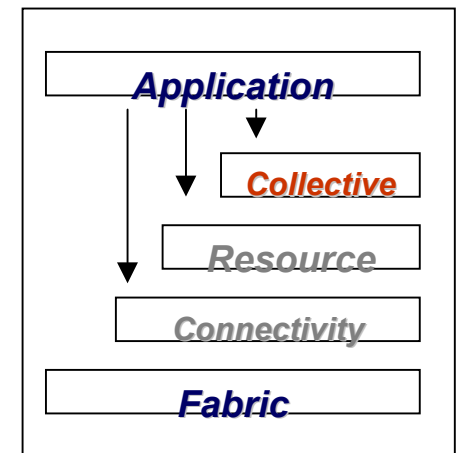
- Directory service : this facility allows members of *virtual organization* to discover which are the resources available .



# Grid Architecture - LAYERS

## *Collective*

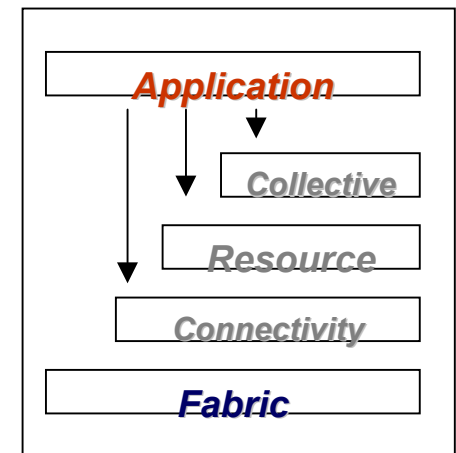
- Common Authorization Servers : this facility is also design to implement a better policy to access resources.



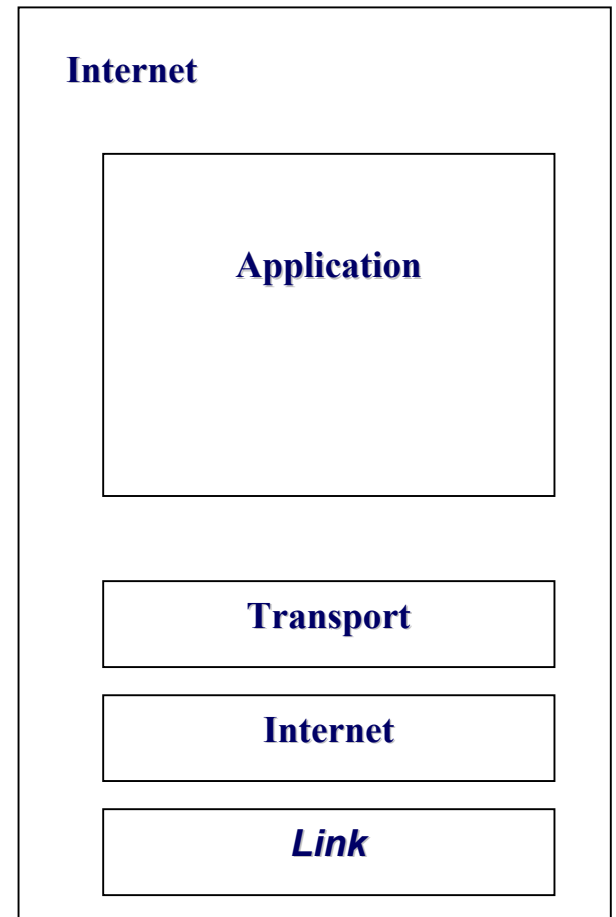
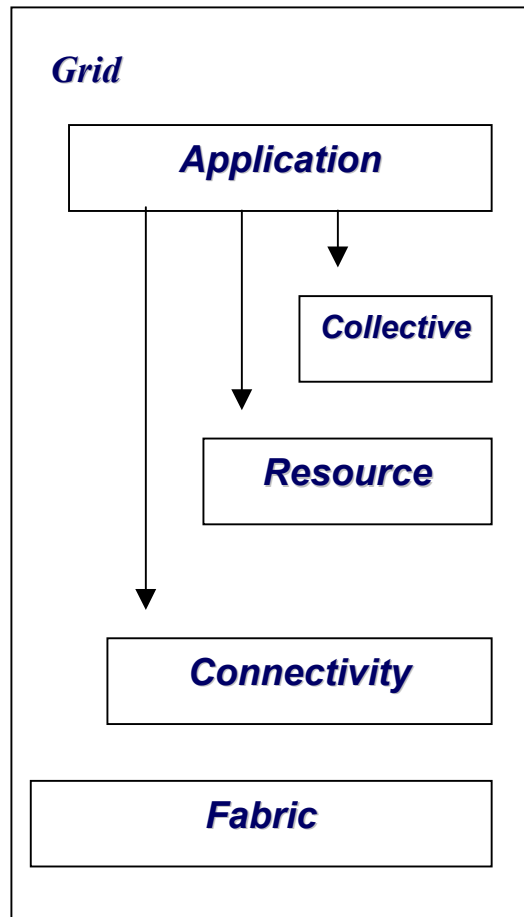
# Grid Architecture - LAYERS

## *Application*

This layer is related to the users' applications in their *virtual organizations*. The previous commented layers provide services for this layer.



# Equivalence between the Grid and Internet Models



# Agenda

- Applications and Computing Paradigms
- Users
- Grid Architecture
- **Grid Computing Environments**
- Experimental Results
- Conclusions and Future Work



## Grid Consortiums and Open Forums

- **C3CA**
- ***Global Grid Forum***
- ***Australian Grid Forum***
- ***Peer-to-Peer (P2P) Working Group***
- ***eGrid: European Grid Computing Initiative***
- ***Asia Pacific Grid***

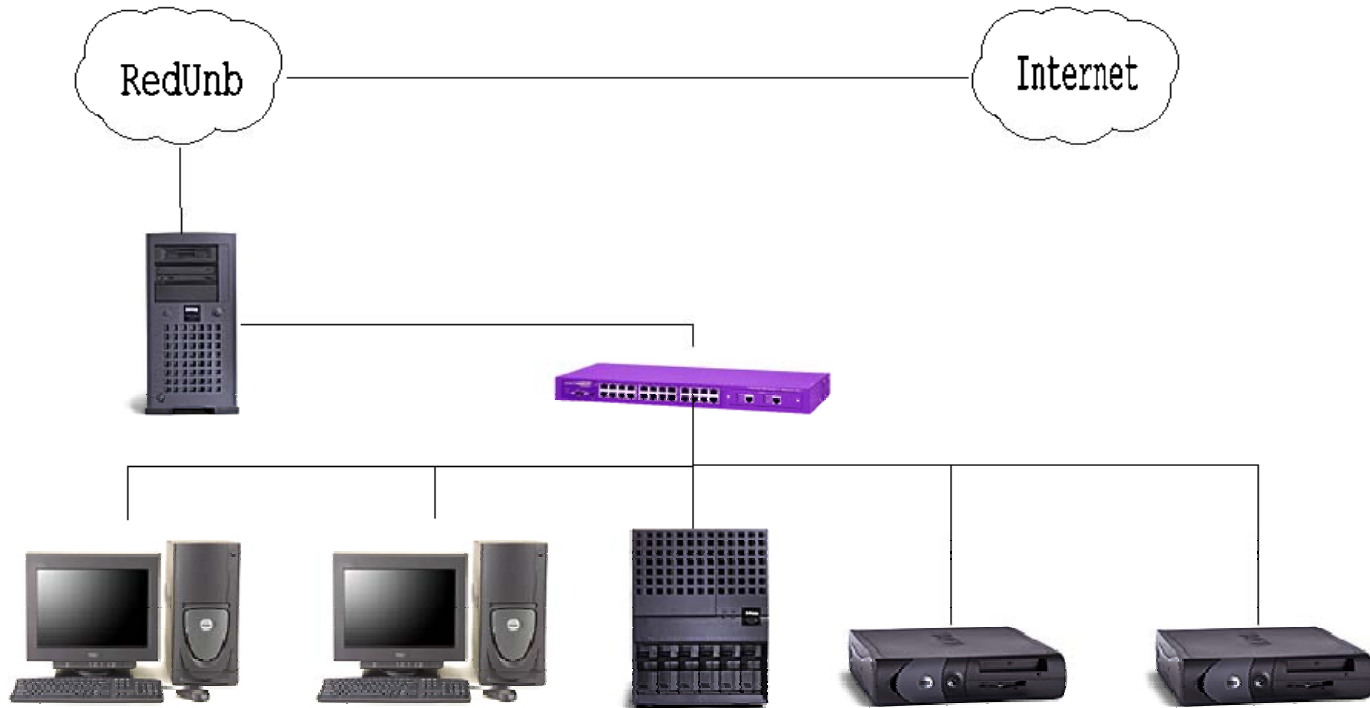
# Grid Computing Environments

## Grid Consortia and Open Forums

- ***GridForum Korea***
- ***EuroTools SIG on Metacomputing***
- ***IEEE Task Force on Cluster Computing***
- ***New Productivity Initiative (NPI)***
- ***The Distributed Coalition***
- ***Content Alliance: About Content Peering***
- ***The Brazilian ....***

# Grid Computing Environments

## *Our Brazilian ....*



# Grid Computing Environments

## Grid Middleware

- **Cosm P2P Toolkit**
- ***Globus***
- **GRACE: GRid Architecture for Computational Economy**
- **Gridbus**

# Grid Computing Environments

## Grid Middleware

- **Grid Datafarm**
- **GridSim: Toolkit for Grid Resource Modeling and Scheduling Simulation**
- **Simgrid**
- **Jxta Peer to Peer Network**
- ***Legion: A Worldwide Virtual Computer***

# Grid Computing Environments

## DataGrid Initiatives

- Virtual Laboratory: Tools for Data Intensive Science on Grid
- EU DataGrid
- DIDC Data Grid work
- GriPhyN (Grid Physics Network)
- HEPGrid (High Energy Physics and Grid Networks)
- Particle Physics Data Grid (PPDG)
- Datacentric Grid

# Grid Computing Environments

## Grid Systems

- Compute Power Market
- Global Operating Systems
- XtremWeb
- JAVELIN: Java-Based Global Computing
- MILAN: Metacomputing In Large Asynchronous Networks
- Harness Parallel Virtual Machine Project
- Management System for Heterogeneous Networks
- PUNCH - Network Computing Hub
- MOBIDICK
- MetaNEOS

# Grid Computing Environments

## Grid Systems

- Amica
- MultiCluster
- Poland Metacomputing
- Echelon: Agent Based Grid Computing
- Bayanihan
- NeuroGrid
- GridLab
- DAMIEN
- CrossGrid
- DIET



## Computational Economy

- GRACE: GRid Architecture for Computational Economy
- Compute Power Market (CPM)
- G-Commerce
- Mariposa: A New Approach to Distributed Data
- The Information Economy
- FORTH Information Economies
- Share Meta
- D'Agent
- Program for Research on the Information Economy

## Computational Economy

- Xenoservers - Accountable Execution of Untrusted Programs
- Electricity Trading Over the Internet Begins in Six New England States
- POPCORN
- CSAR: Resource Tokens and Trading Pool
- OCEAN - The Open Computation Exchange & Arbitration Network
- Spawn: A Distributed Computational Economy
- Market-Based Computing
- Multiagent systems

# Grid Computing Environments

## Comutational Economy

- W3C effort: Common Markup for micropayment per-fee-links
- Agent-Based Computational Economics
- Electronic Brokerage
- Society for Computational Economics
- Internet Ecologies

# Grid Computing Environments

## Grid Schedulers

- Nimrod/G Grid Resource Broker
- AppLeS
- SILVER Metascheduler
- ST-ORM
- Condor/G
- NetSolve
- DISCWorld
- Computing Centre Software (CCS)

# Grid Computing Environments

## Grid Portals

- ActiveSheets
- UNICORE - Uniform Interface to Computing Resources
- SDSC GridPort Toolkit
- Enginframe
- Lecce GRB Portal
- Grid Enabled Desktop Environments
- Interactive Control and Debugging of Distribution-IC2D
- NLANR Grid Portal Development Kit

# Grid Computing Environments

## Grid Programming Environments

- Nimrod - A tool for distributed parametric modeling
- Ninf
- Cactus Code
- MetaMPI - Flexible Coupling of Heterogeneous MPI Systems
- Virtual Distributed Computing Environment
- GrADS: Grid Application Development Software Project
- Java-based CoG Kit
- GAF3J - Grid Application Framework for Java
- ProActive PDC
- REDISE - Remote and Distributed Software Engineering
- Albatross: Wide Area Cluster Computing

# Grid Computing Environments

## Grid Performance Monitoring and Forecasting

- Network Weather Service
- NetLogger
- Remos

## Grid Testbeds and Developments

- World Wide Grid (WWG)
- Polder Metacomputer
- NASA Information Power Grid (IPG)
- NPACI: Metasystems
- Asia Pacific Bioinformatics Network
- The Distributed ASCI Supercomputer (DAS)
- G-WAAT
- Micro Grid
- Alliance Grid Technologies
- The Alliance Virtual Machine Room
- EuroGrid



## Grid Testbeds and Developments

- Internet Movie Project
- Nordic Grid
- ThaiGrid
- TeraGrid
- Irish Computational Grid (ICG)
- GrangeNet
- LHC Grid
- I-Grid

## Grid Applications

- Molecular Modelling for Drug Design
- Neuro Science - Brain Activity Analysis
- Cellular Microphysiology
- HEPGrid: High Energy Physics and the Grid Network
- Access Grid
- Globus Applications
- The International Grid (iGrid)
- UK Grid Apps Working Group
- NLANR Distributed Applications
- DataGRID - WP9: Earth Observation Science Application

## Grid Applications

- Particle Physics Data Grid
- DREAM project: Evolutionary Computing and Agents Applications
- Knowledge Grid
- Fusion Collaboratory
- APEC Cooperation for Earthquake Simulation
- Australian Computational Earth Systems Simulator
- EarthSystemGrid

# Grid Computing Environments

## Grid Applications

- Australian Virtual Observatory
- US Virtual Observatory
- Distributed Proofreaders
- NEESgrid: Earthquake Engineering Virtual Collaboratory
- Geodise: Aerospace Design Optimisation
- Japanese BioGrid

# Agenda

- Applications and Computing Paradigms
- Users
- Grid Architecture
- Grid Computing Environments
- **Experimental Results**
- Conclusions and Future Work

# An Evaluation of Globus and Legion Software Environments

## Globus

The *Globus* software environment is a project developed by *Argonne National Laboratory* (ANL) and *University of Southern California*. In our work we use the version 1.1.4 of the Globus software package because this release provides support to MPI applications.

The Globus environment is composed by a set of components implementing basic services to resource allocation, communication, security, process management and access to remote data .

# An Evaluation of Globus and Legion Software Environments

The resource allocation component of the Globus environment (GRAM - *Globus Resource Allocation Manager*) is the element that acts as an interface between global and local services.

Application programmers use the GRAM element, through the *gatekeeper* software portion which is responsible for the user authentication and association with a local computer account.

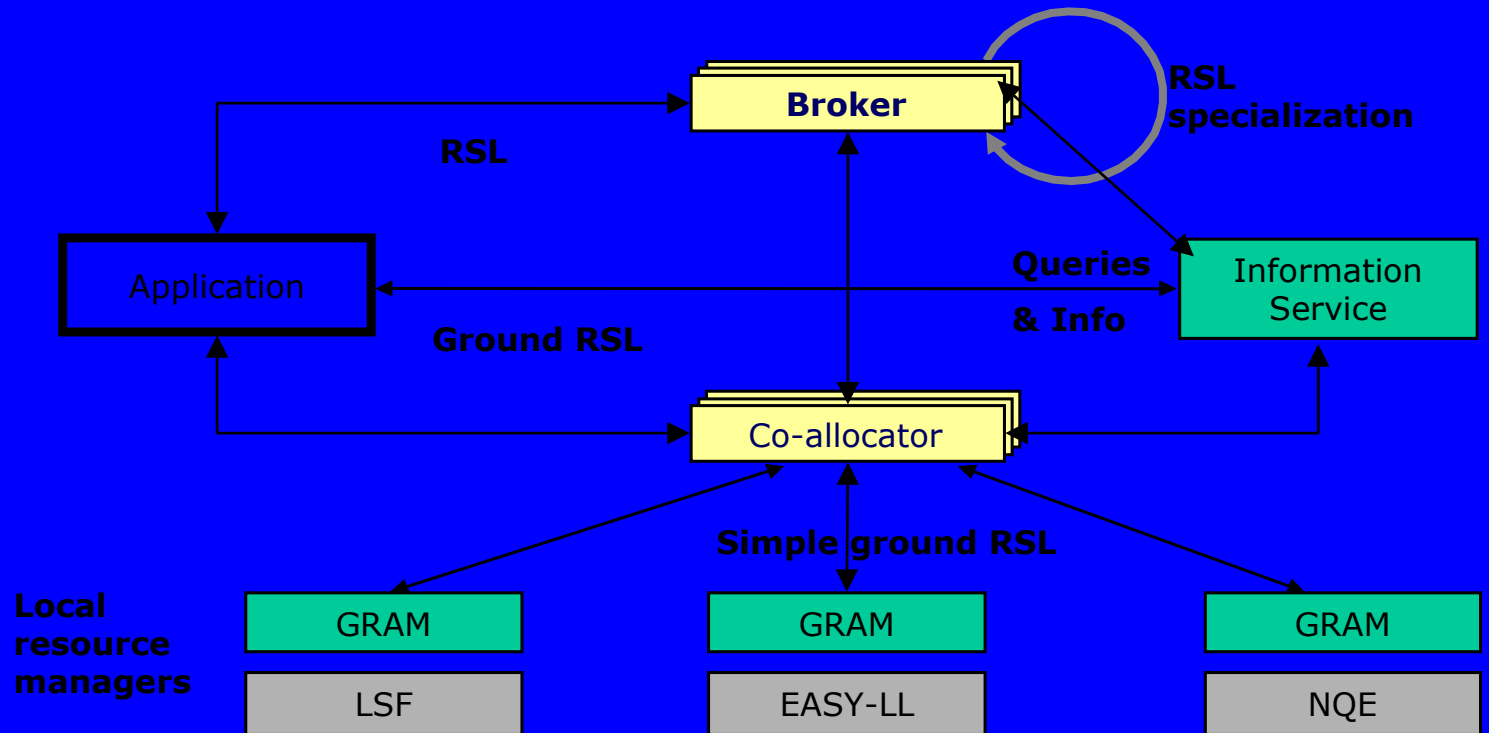
# An Evaluation of Globus and Legion Software Environments

The mechanism to identify users of the *grid* is based on a file called *map-file*. In this file exists information about authorized users of the *grid configuration*.

Any requirement for resource should be translated to the *Resource Specification Language (RSL)*.



# Tutorial 3

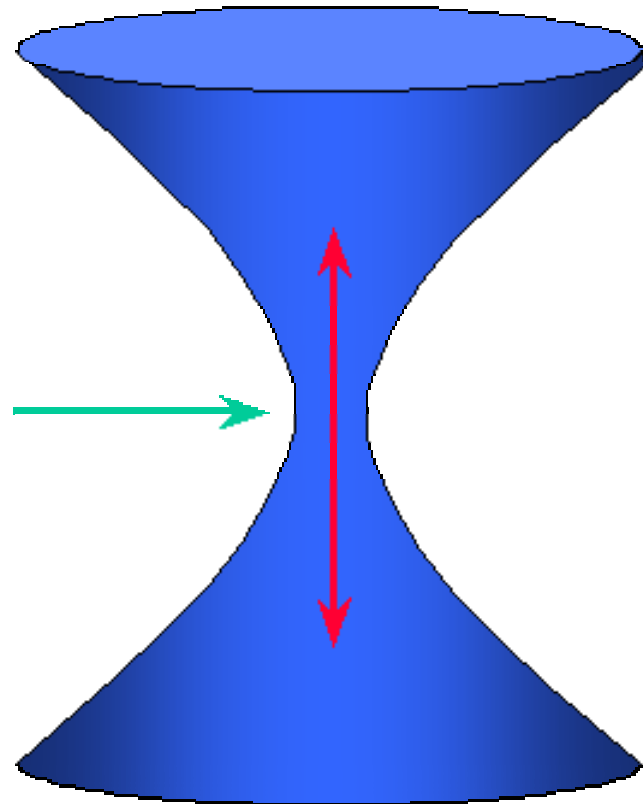


# Tutorial 3

## Applications

Diverse global services

Core  
Globus  
Services



Local OS

# An Evaluation of Globus and Legion Software Environments

Communication in the Globus environment is performed using a communication library called *Nexus*.

This component defines low a level API to support high level programming paradigms.

Examples of high level programming paradigms supported are message passing, remote procedure call and remote I/O procedures.

The information about the system and the *grid configuration* are management by a component called *Metacomputing Directory Service (MDS)*.

# **An Evaluation of Globus and Legion Software Environments**

An important aspect of the Globus software environment is the security.

This software tool employs the certificate approach, which is carried by a CA (Certificate Authority) using the protocol *Secure Socket Layer* (SSL)

# An Evaluation of Globus and Legion Software Environments

## Legion

The *Legion* software environment is a system object oriented which is being developed since 1993 at University of Virginia.

This environment has an architecture concept of grid computing providing a unique virtual machine for users' applications.

The approach of the *Legion* is to have some important concepts of a grid configuration (e.g. scalability, easy to program, fault tolerance and security) transparent to final users.

# **An Evaluation of Globus and Legion Software Environments**

In the *Legion*, every entity such as processing power, RAM memory and storage capacity is represented as objects. Objects communicate with each other using services calls to a remote mechanism.

# An Evaluation of Globus and Legion Software Environments

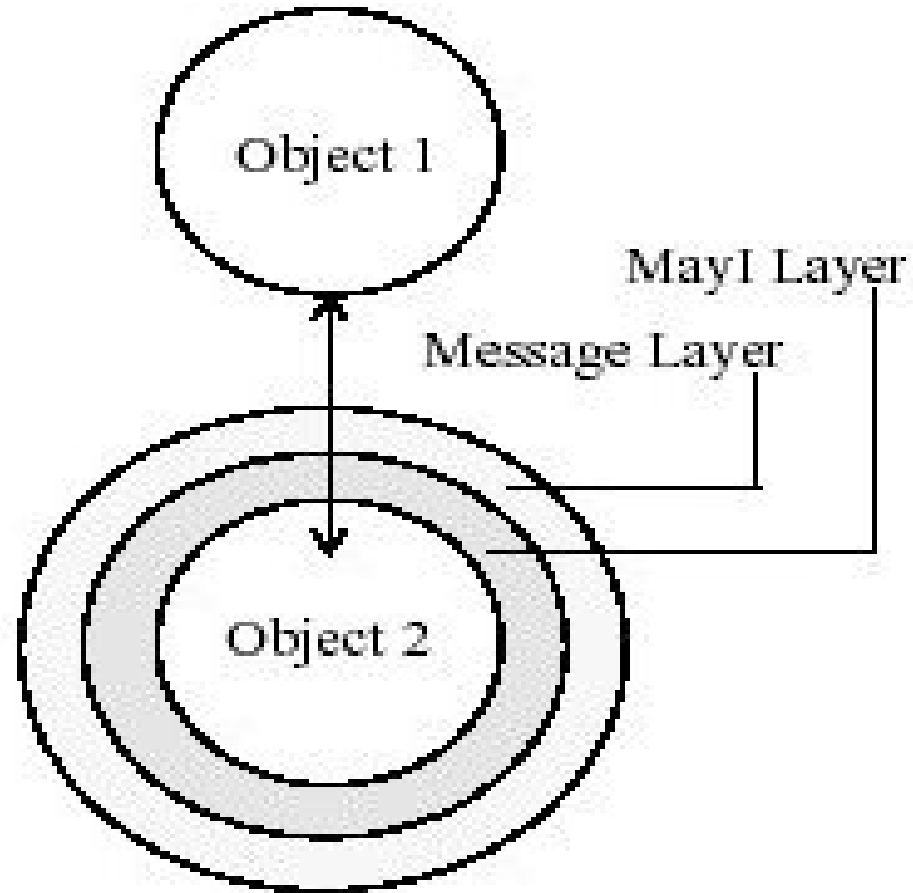
The security component of the *Legion*, as the others elements of this software, is based on an object.

The application programmer specifies the security related to an object, where it is defined which type of mechanism is allowed.

In addition, the *Legion* provides some extra basic mechanism to ensure more security.

The *May I* method is an example. Every class should define the method *May I*, which check for a called object the related allowed access.

# Tutorial 3





# An Evaluation of Globus and Legion Software Environments

The traditional system file is emulated in the *Legion* environment through the combination of persistent objects with the global information of object identification.

This approach simplifies the manipulation of files to application programmers. In addition, it is allow to users to add fault tolerance characteristics to applications using rollback and recovery mechanisms

# An Evaluation of Globus and Legion Software Environments

<i>Grid Environment</i>	<i>Legion</i>	<i>Globus</i>
<b>Software requirement</b> - -	<b>OpenSSL 0.9.5</b> - <b>bin/ksh</b>	<b>SSLeay 0.9.0</b> - <b>OpenLDAP 1.2.7</b>
<b>Minimum Disk space</b>	<b>De 250MB a 300 MB</b>	<b>200 MB</b>
<b>Minimum Memory RAM</b>	<b>256 MB</b>	<b>Not specified</b>

# **An Evaluation of Globus and Legion Software Environments**

## **A CASE STUDY**

# An Evaluation of Globus and Legion Software Environments



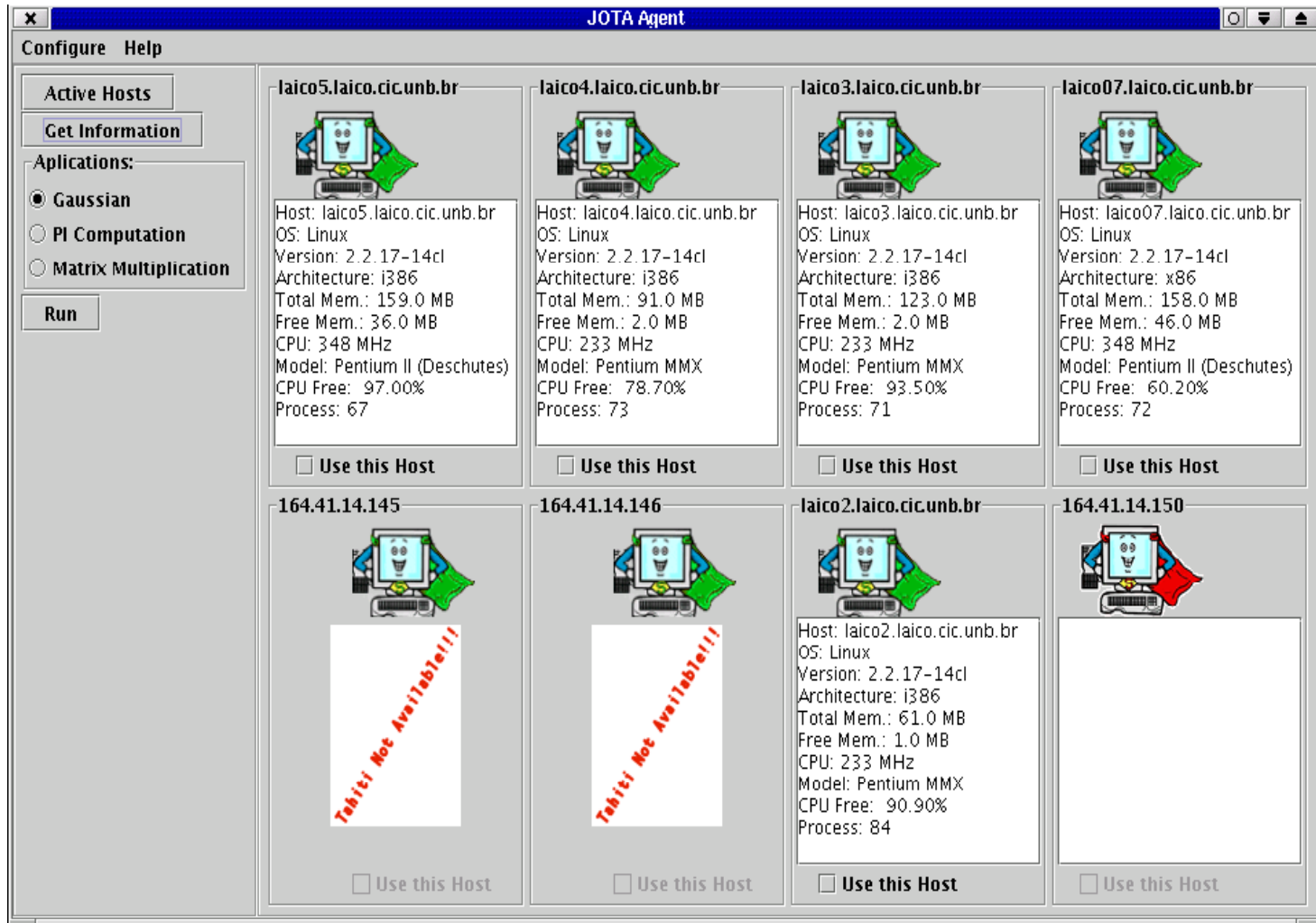
# An Evaluation of Globus and Legion Software Environments



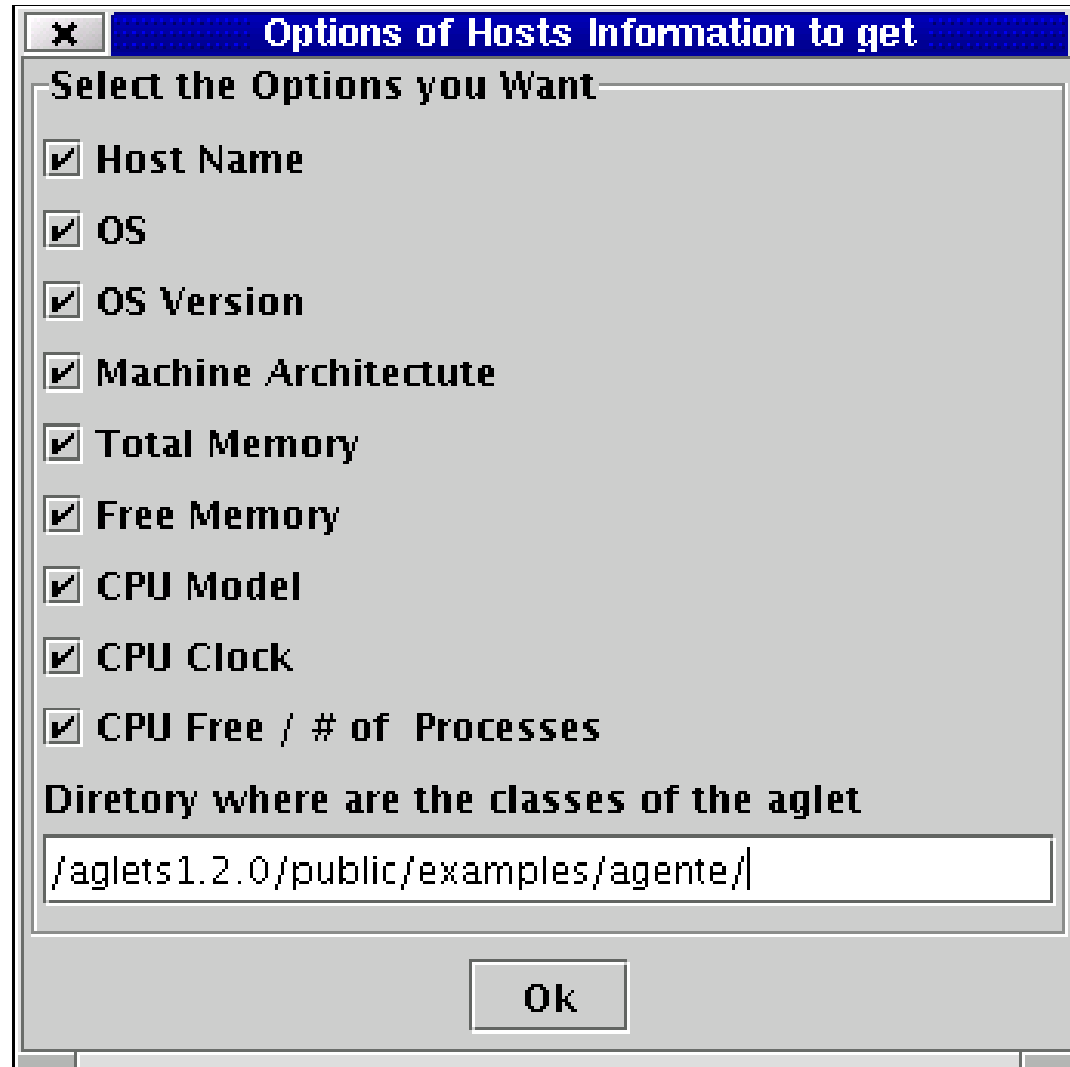
# **An Evaluation of Globus and Legion Software Environments**

## **A Friendly Interface**

# An Evaluation of Globus and Legion Software Environments



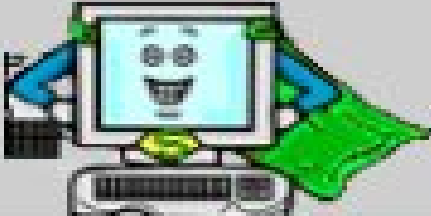
# An Evaluation of Globus and Legion Software Environments





# An Evaluation of Globus and Legion Software Environments

laico5.laico.cic.unb.br



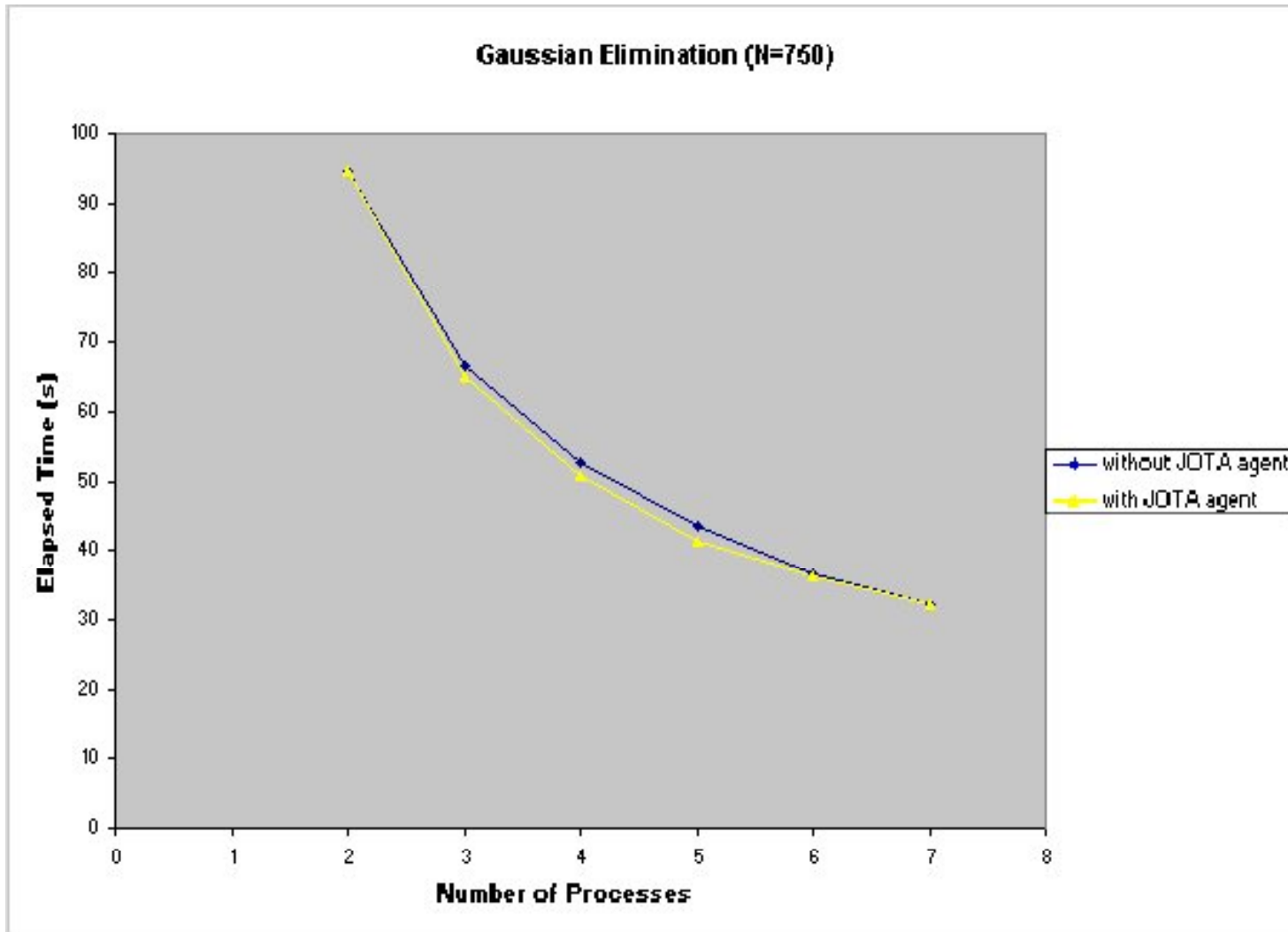
Host: laico5.laico.cic.unb.br  
OS: Linux  
Version: 2.2.17-14cl  
Architecture: i386  
Total Mem.: 159.0 MB  
Free Mem.: 36.0 MB  
CPU: 348 MHz  
Model: Pentium II (Deschutes)  
CPU Free: 97.00%  
Process: 67

Use this Host

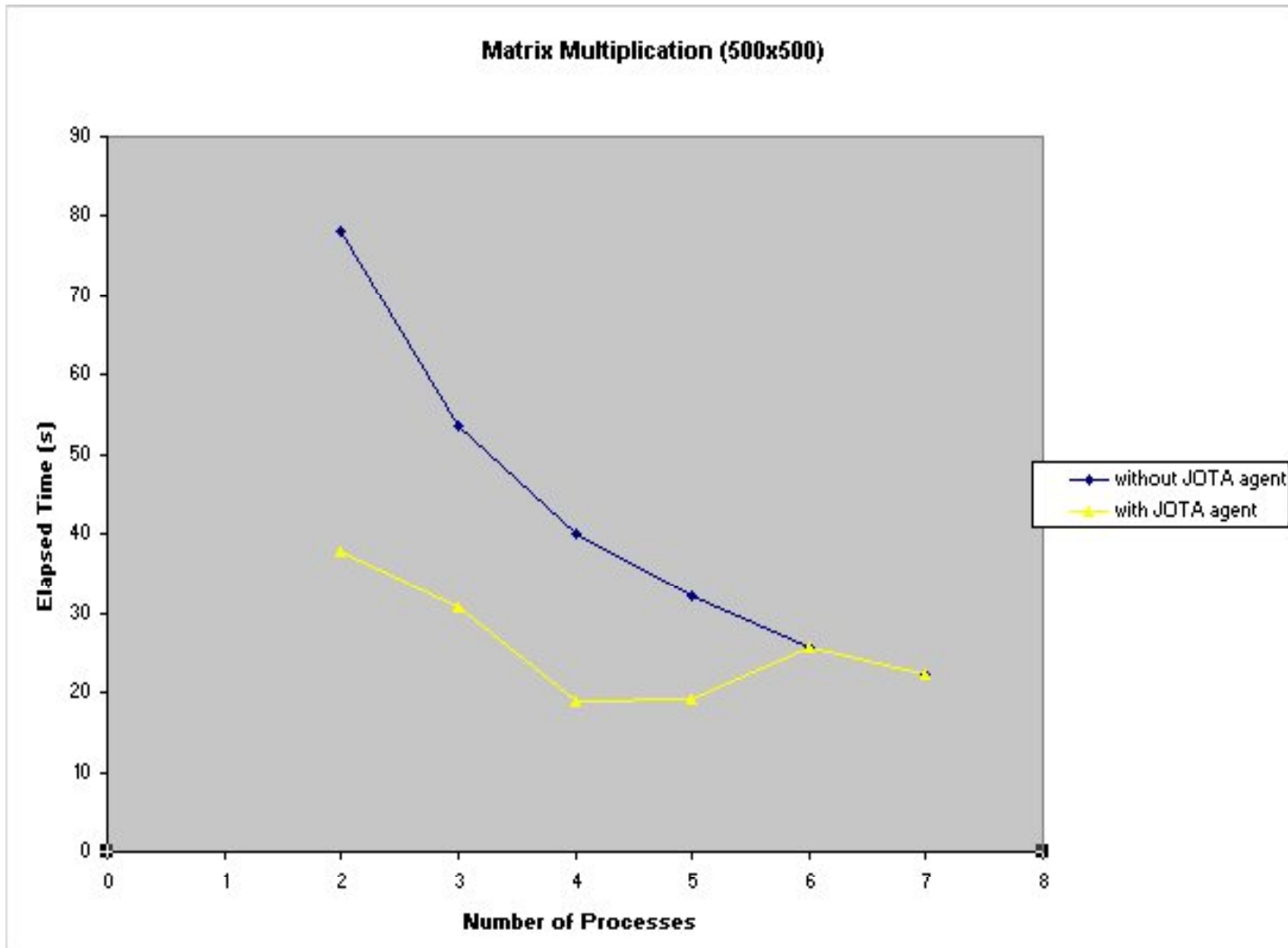
# **An Evaluation of Globus and Legion Software Environments**

## **Experimental Results**

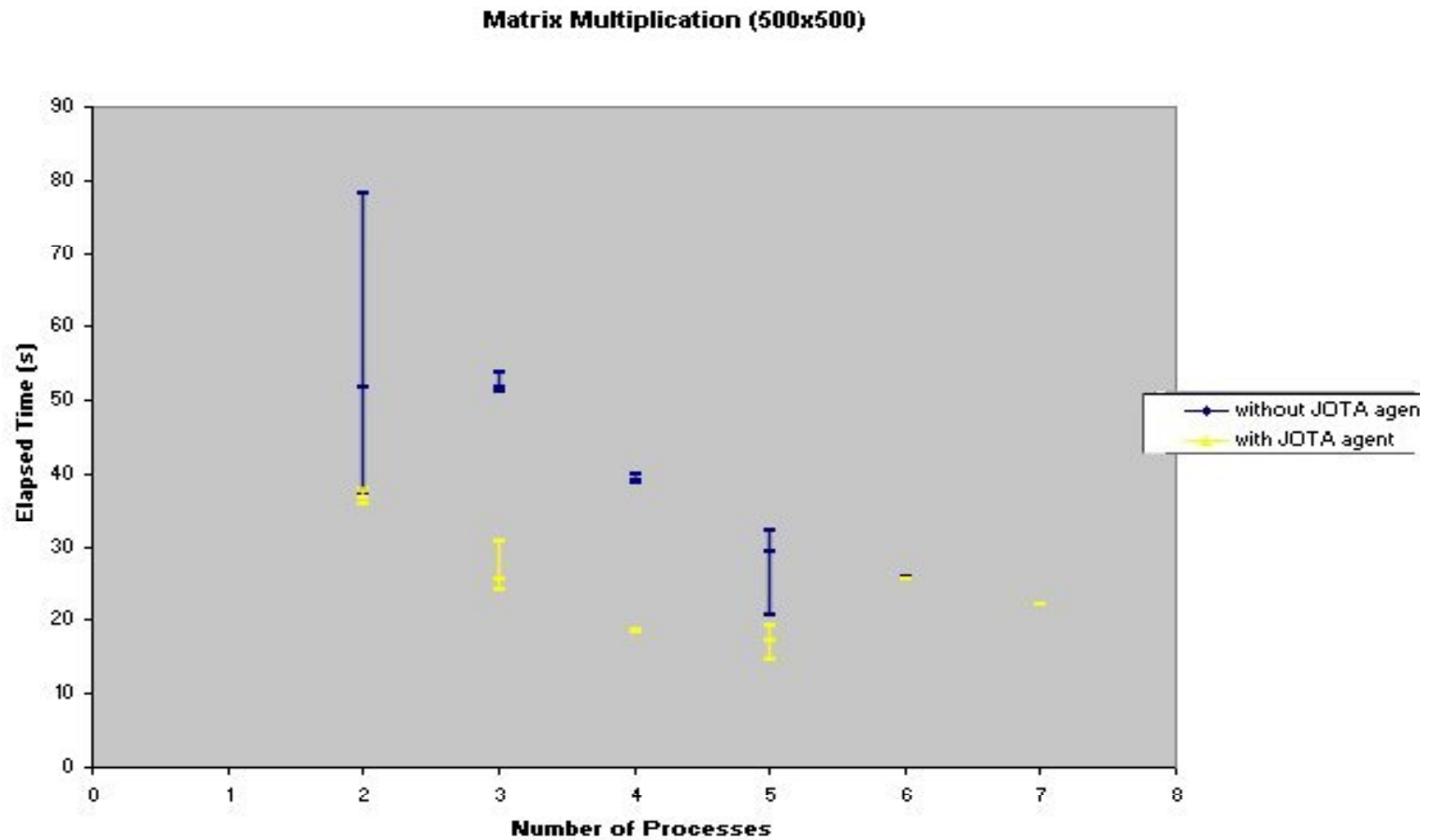
# An Evaluation of Globus and Legion Software Environments



# An Evaluation of Globus and Legion Software Environments



# An Evaluation of Globus and Legion Software Environments



# **An Evaluation of Globus and Legion Software Environments**

## **Hardware and Software Environment**

# An Evaluation of Globus and Legion Software Environments

After providing some characteristics of the Globus and Legion software tools, in this section we present our grid configuration environment.

It is important to mention that all the machines were in the same laboratory. However, using a *Ethernet Layer 3 Switch* we were able to have the abstraction of a WAN (Wide Area Network) inside this box.

In other words, this equipment could prove the abstraction of a distributed resource environment for our experiments.

# An Evaluation of Globus and Legion Software Environments

Computer Name	AIX 1	AIX 2	AIX 3	AIX 4
Operating System	AIX 4.3	AIX 4.3	AIX 4.3	AIX 4.3
Processor	PowerPC_604 233 MHz	PowerPC_604 233 MHz	PowerPC_604 233 MHz	PowerPC_604 233 MHz
Memory RAM	256 MB	128 MB	128 MB	512 MB
Hard disk	Two disks of 9 GB	Two disks of 4 GB	Two disks of 4 GB and one 2 GB disk	Two disks of 4 GB and one 2 GB disk
Software Environment	<i>Legion</i>	<i>Globus</i>	<i>Globus</i>	Legion

Table I: The grid environment configuration



# An Evaluation of Globus and Legion Software Environments

The *Legion* software provides a homogeneous view of the *grid* to the application programmer.

The environment uses its own tools to create the homogeneity. The procedure to install the software does not represent any problem, because the application programmer needs only to uncompress binary files and execute some script files. However, for the AIX environment it is necessary more information than those available from the software documents.

# An Evaluation of Globus and Legion Software Environments

We fixed some problems using our background on AIX and exchanging several e-mails with other AIX systems managers. The *Legion* concept of file system represents an advantage of the environment.

The *Legion* file system presents a unique identifier for each object.

# An Evaluation of Globus and Legion Software Environments

This approach provides application programmers the facility to access files widely distributed only using their names.

In other words, the users only use the name of the file, which can be storage in a local or remote machine.

On the other hand, we have verified some problems with the package. As a first problem, we can mention the necessary installation of the entire environment when the *bootstrap host* has a power failure.

# An Evaluation of Globus and Legion Software Environments

The *bootstrap host* is responsible for the domain control. Another drawback of the environment is the low communication rate between objects.

The paradigm of the *Legion* is to be a *framework environment*, where users can develop their own tools, such as security and fault tolerance facilities.

This freedom can represent some flexibility to any developers group. However, it does not allow the use external tools.

# An Evaluation of Globus and Legion Software Environments

The *Globus* approach allows users to use existing system available tools and have a uniform interface to the grid environment. Interesting features of the *Globus* environment are related to the security and to the autonomy of the configuration.

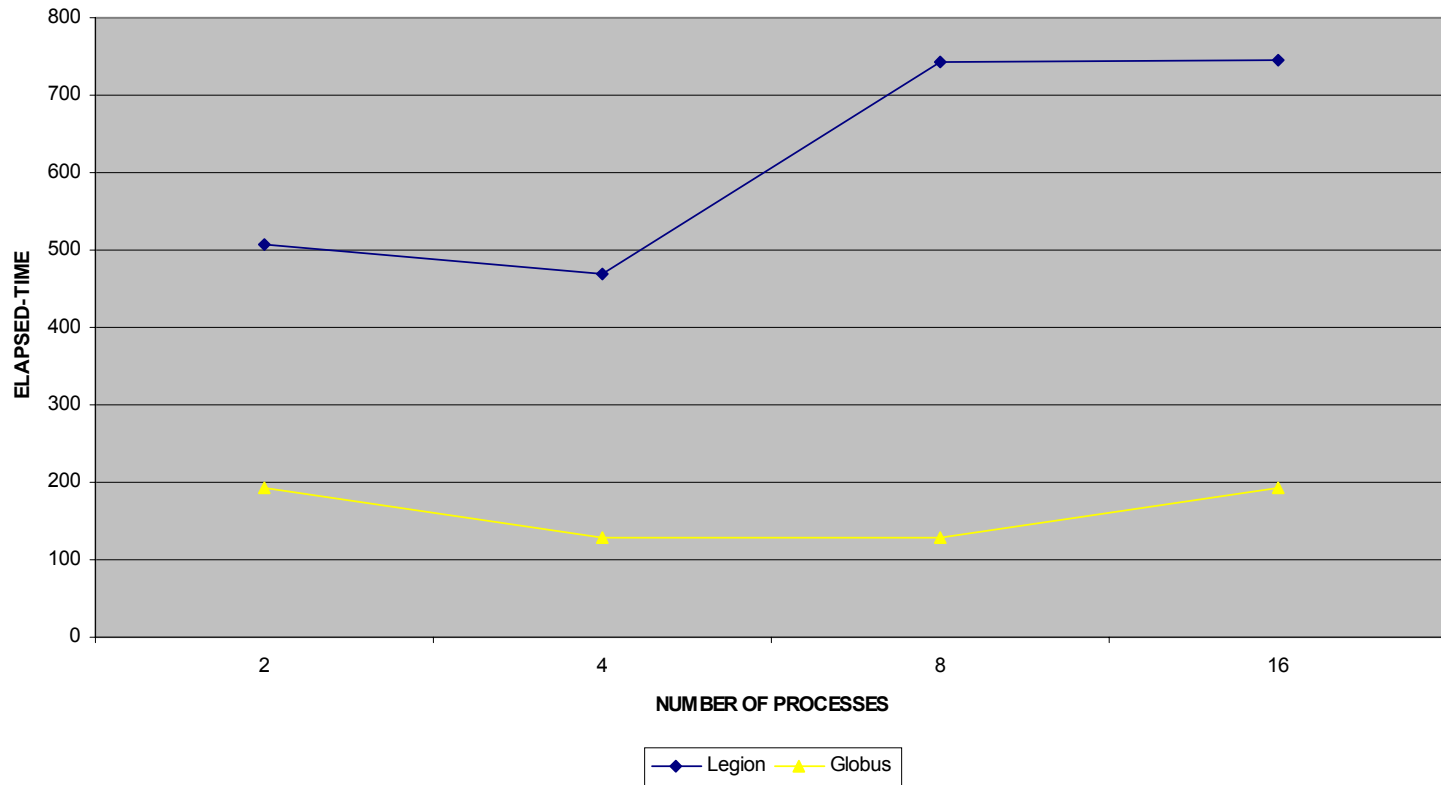
# **An Evaluation of Globus and Legion Software Environments**

The system has an infrastructure based on X509 certificate and the use the mutual authentication. On the other hand, one drawback of the software is the scalability, which can be understood as the capability to add new resources and new sites.

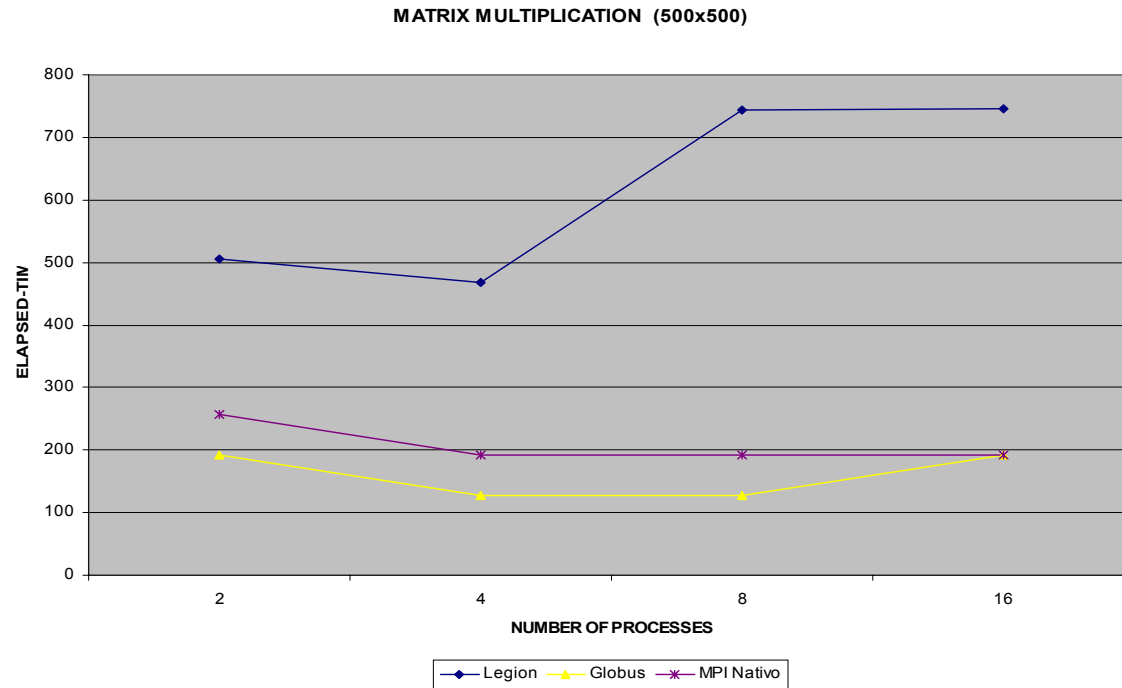
When considering new facilities application programmers are required to have account into all new hosts.

# An Evaluation of Globus and Legion Software Environments

MATRIX MULTIPLICATION (500x500)



# An Evaluation of Globus and Legion Software Environments

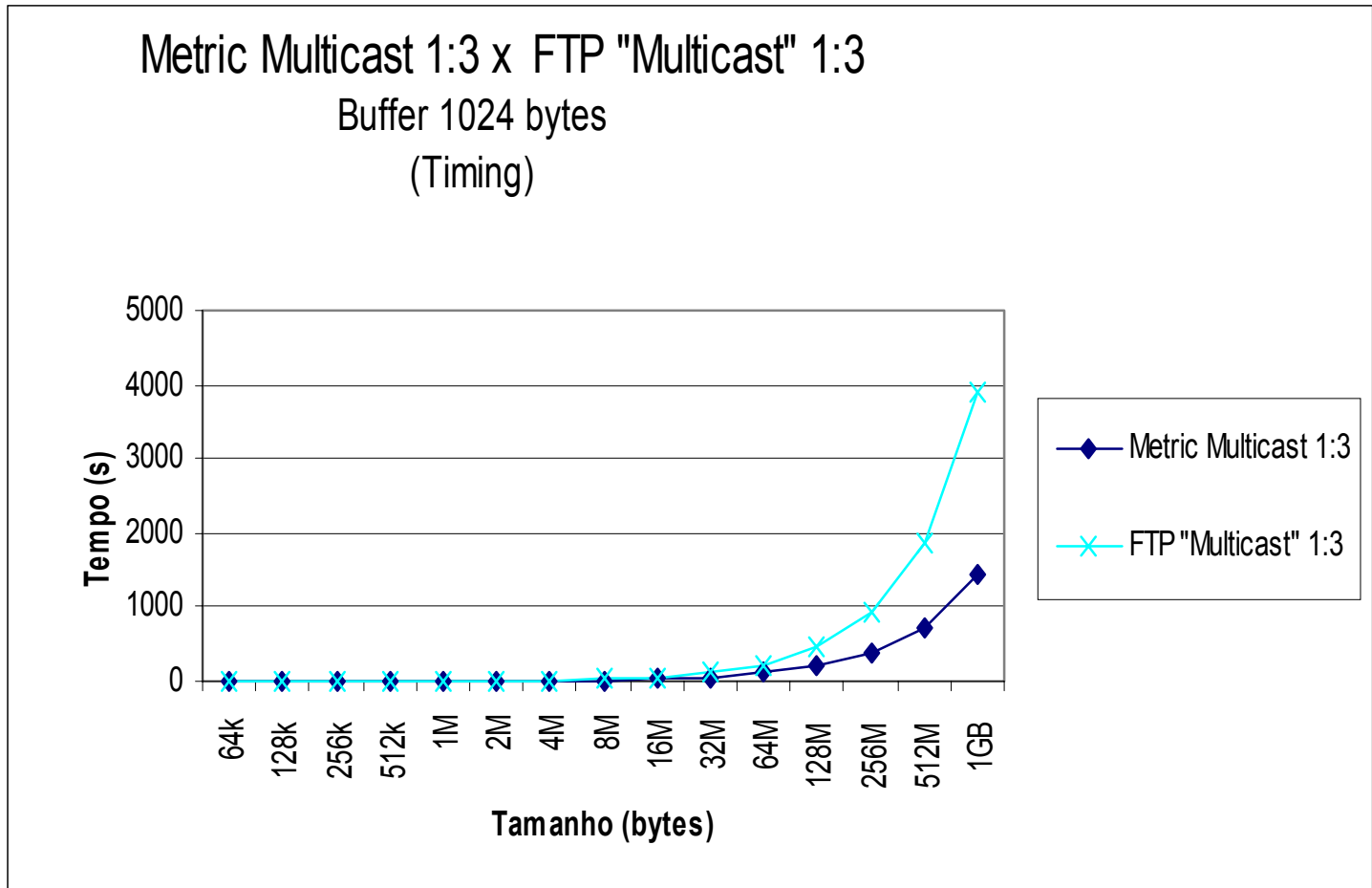




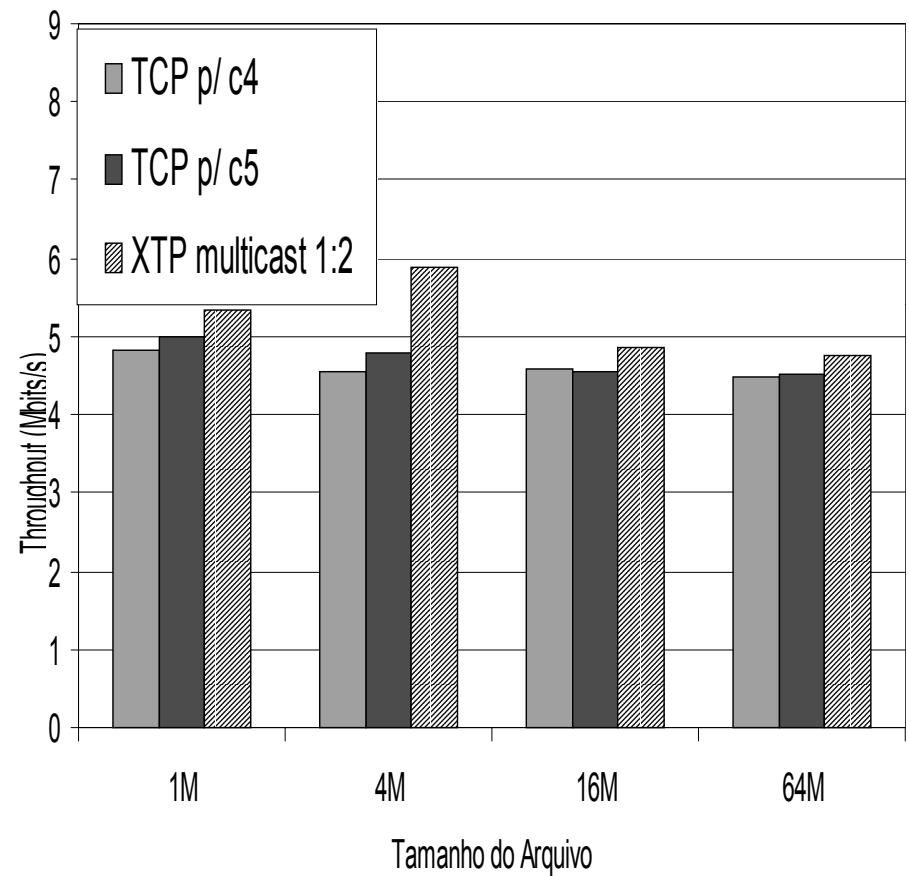
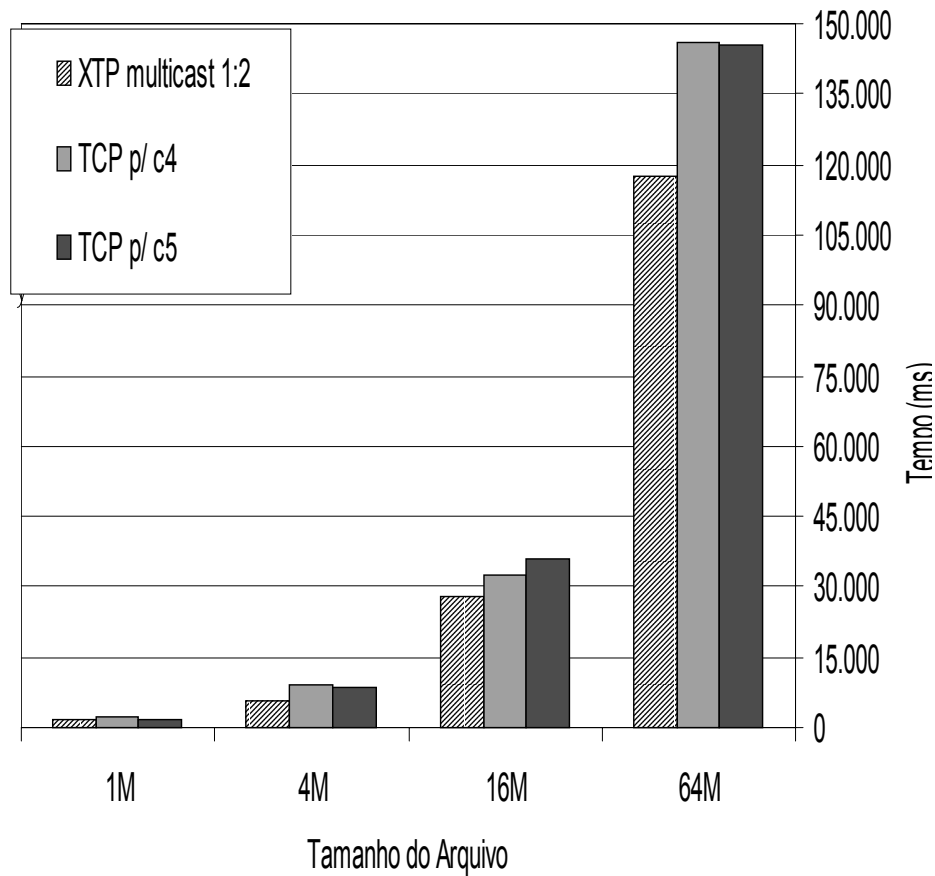
# **An Evaluation of Globus and Legion Software Environments**

## **Lightweight Protocols**

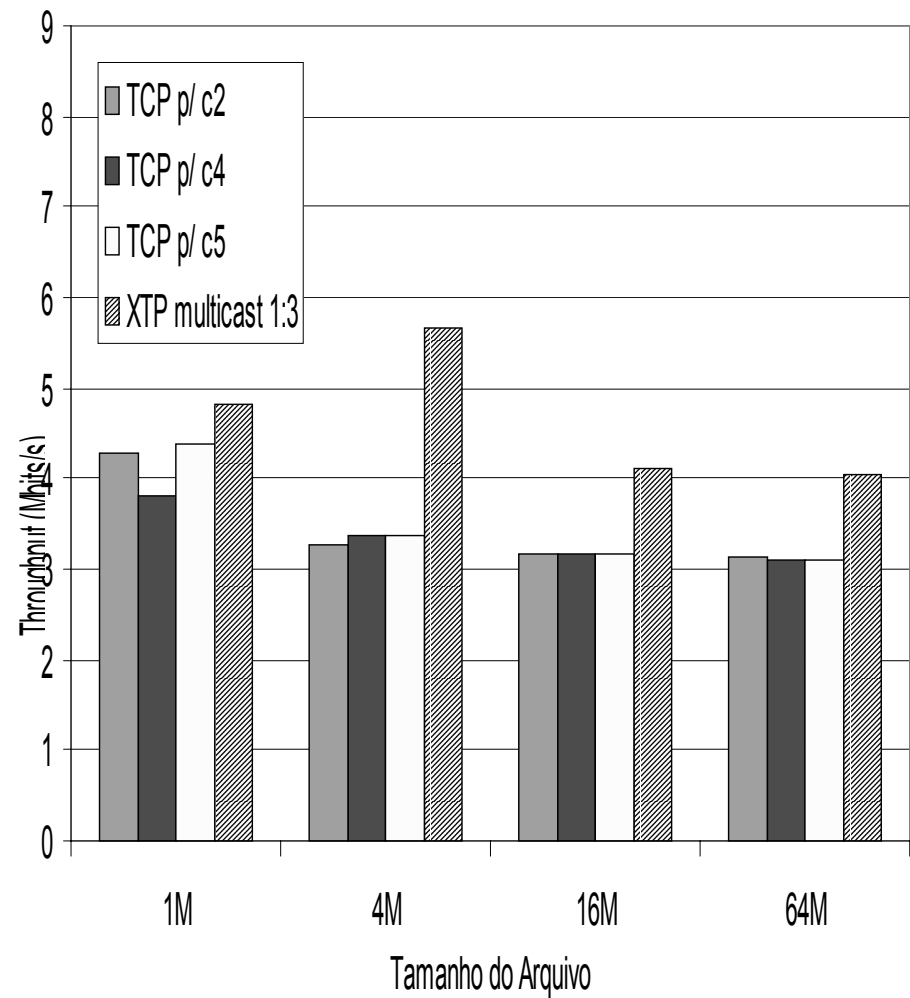
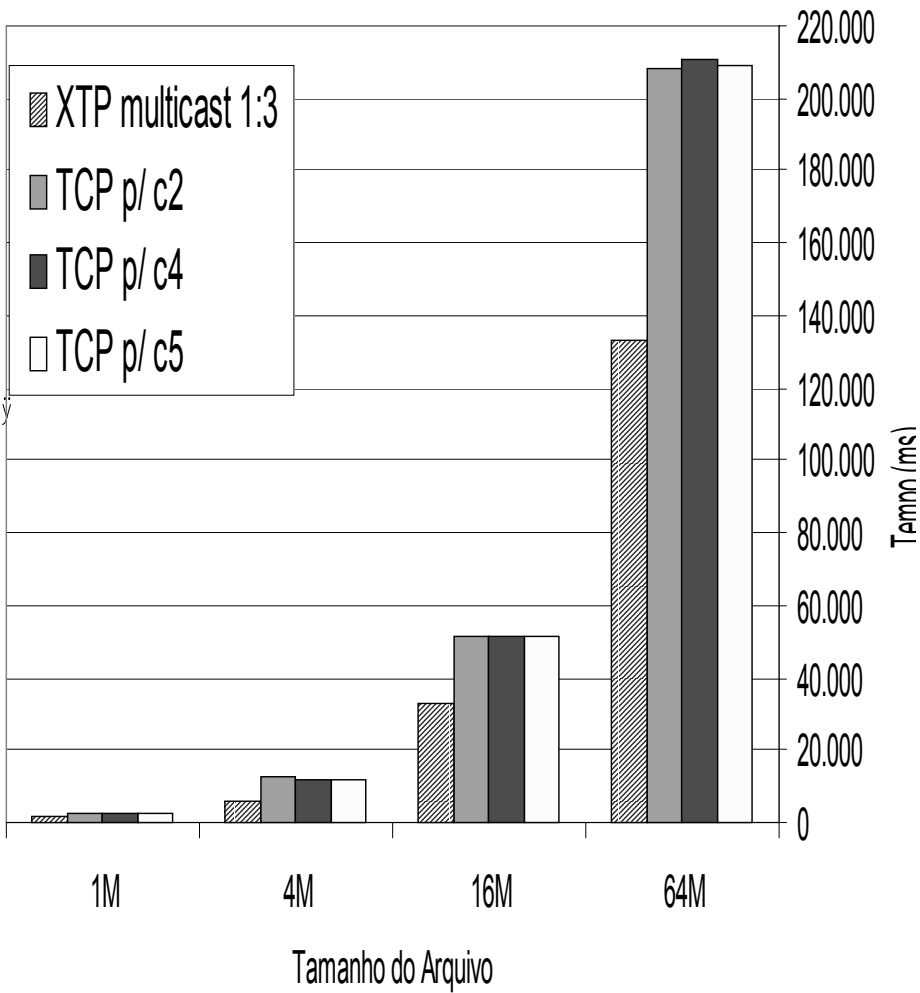
# An Evaluation of Globus and Legion Software Environments



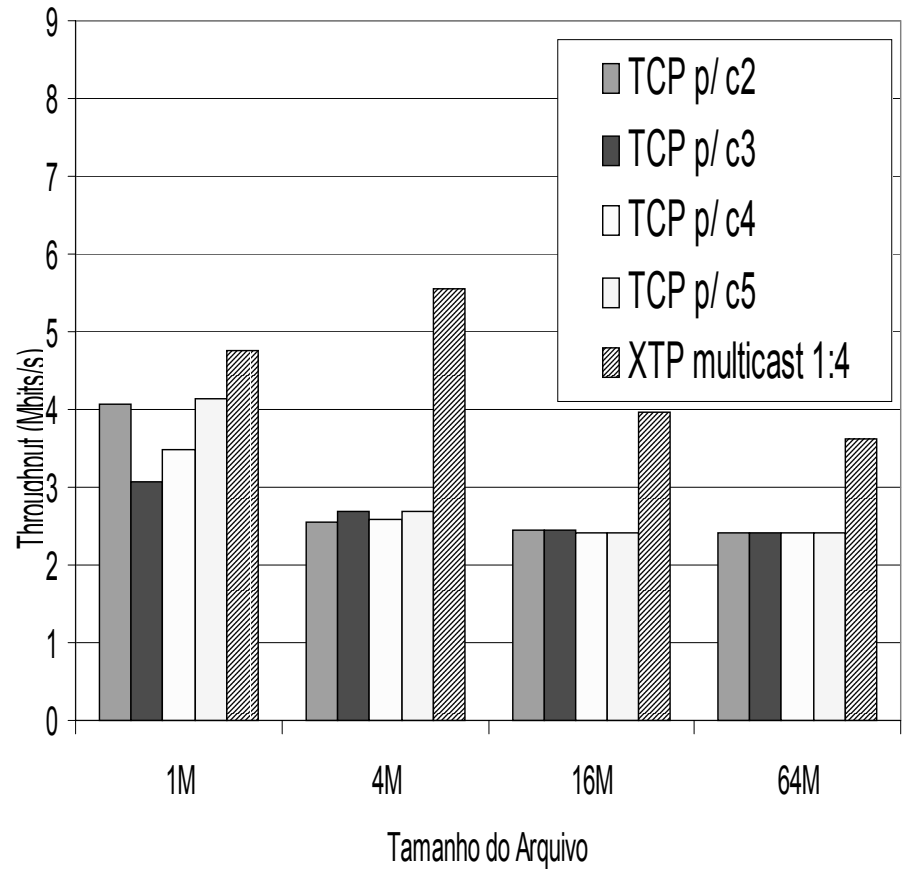
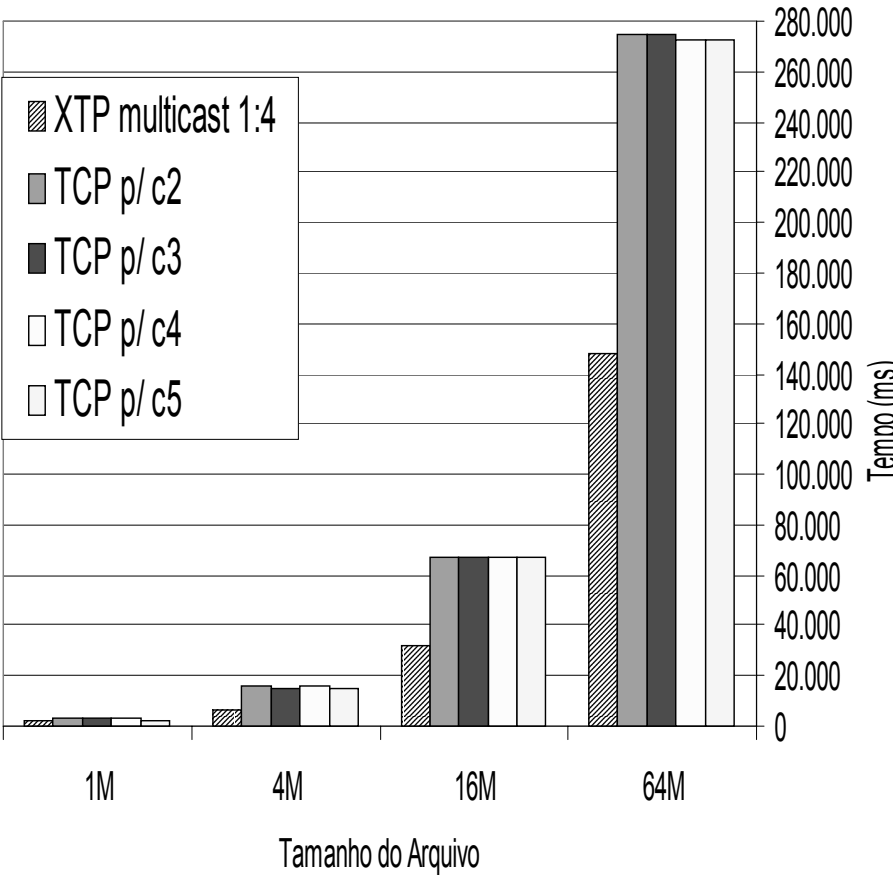
# An Evaluation of Globus and Legion Software Environments



# An Evaluation of Globus and Legion Software Environments



# An Evaluation of Globus and Legion Software Environments



# Agenda

- Applications and Computing Paradigms
- Users
- Grid Architecture
- Grid Computing Environments
- Experimental Results
- **Conclusions and Future Work**

# An Evaluation of Globus and Legion Software Environments

In this research work we have presented the implementation characteristics of the *Globus* and *Legion* software environments.

In addition, we have evaluated these two environments when executing a parallel distributed MPI application.

# An Evaluation of Globus and Legion Software Environments

*Globus* and *Legion* are important tools to configure grid configurations. The *Globus* environment has presented a more robust features, because the software includes security and fault monitoring mechanisms together with many others services.



# An Evaluation of Globus and Legion Software Environments

On the other hand, because it is an object-oriented package the *Legion* environment is more efficient to present the grid abstraction. This software is a *framework* and it is not a finished tool. However, we believe that *Legion* can address those users who are expecting a grid configuration that can be customized for their organisations.

# ***An Evaluation of Globus and Legion Software Environments***



**Questions ?**

# PART II

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

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

Les grappes d'ordinateurs sont généralement peu coûteuses, 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 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.

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

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.

# Agenda

- **Introduction to the problem**
- **High-Availability**
- **Virtual Server**
- **Integrating High-Availability and Virtual Server**
- **Experimental Results**
- **Conclusions and Future Work**

# Agenda

- **Introduction to the problem**
- **High-Availability**
- **Virtual Server**
- **Integrating High-Availability and Virtual Server**
- **Experimental Results**
- **Conclusions and Future Work**

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

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 and Microsoft, the two companies are already providing some features of reliability for their cluster solutions.



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

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.

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

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.

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

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.

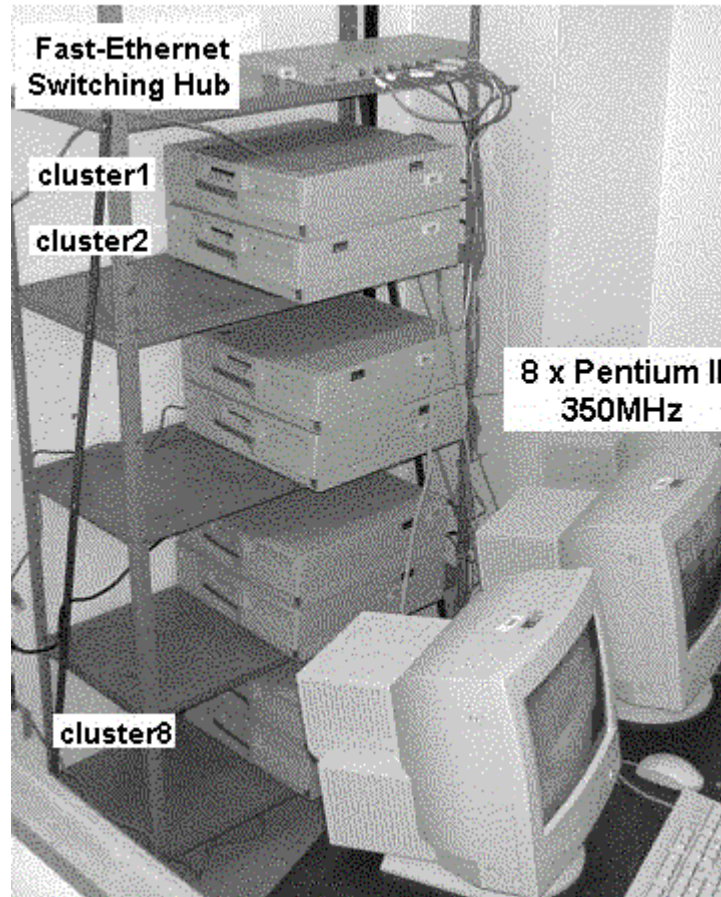
## ***Providing Cluster Environments with High-Availability and Load-Balancing***

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.

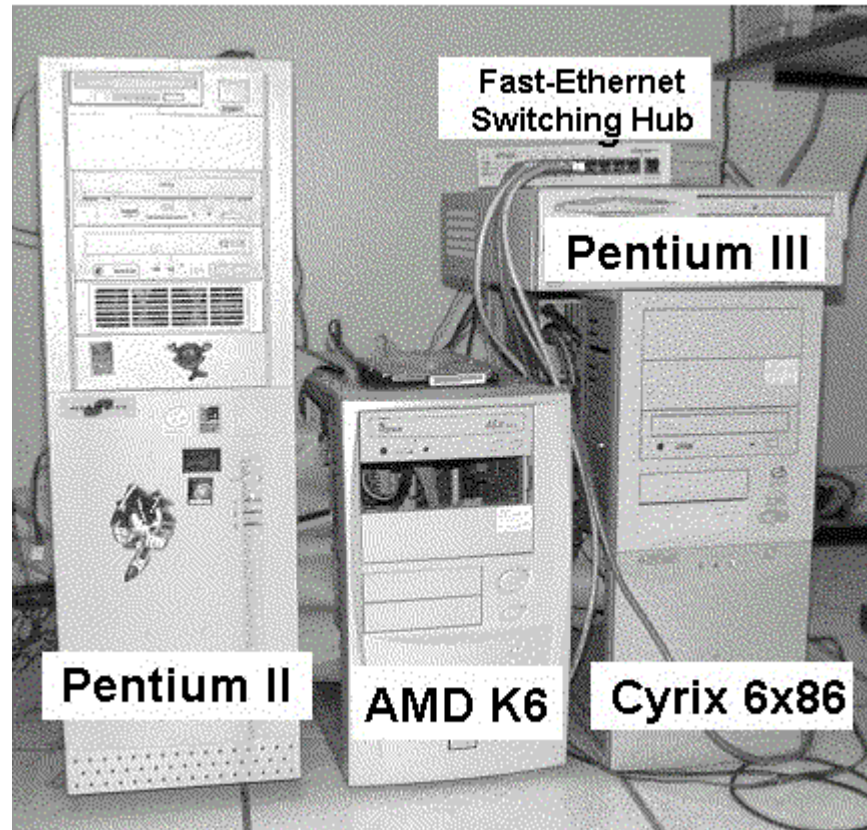
In other words, if any problem occurs with the first computer the second machine acts as a mirror, working in the same fashion as the original machine.

This approach provides to programmers a more reliable environment to execute their applications.

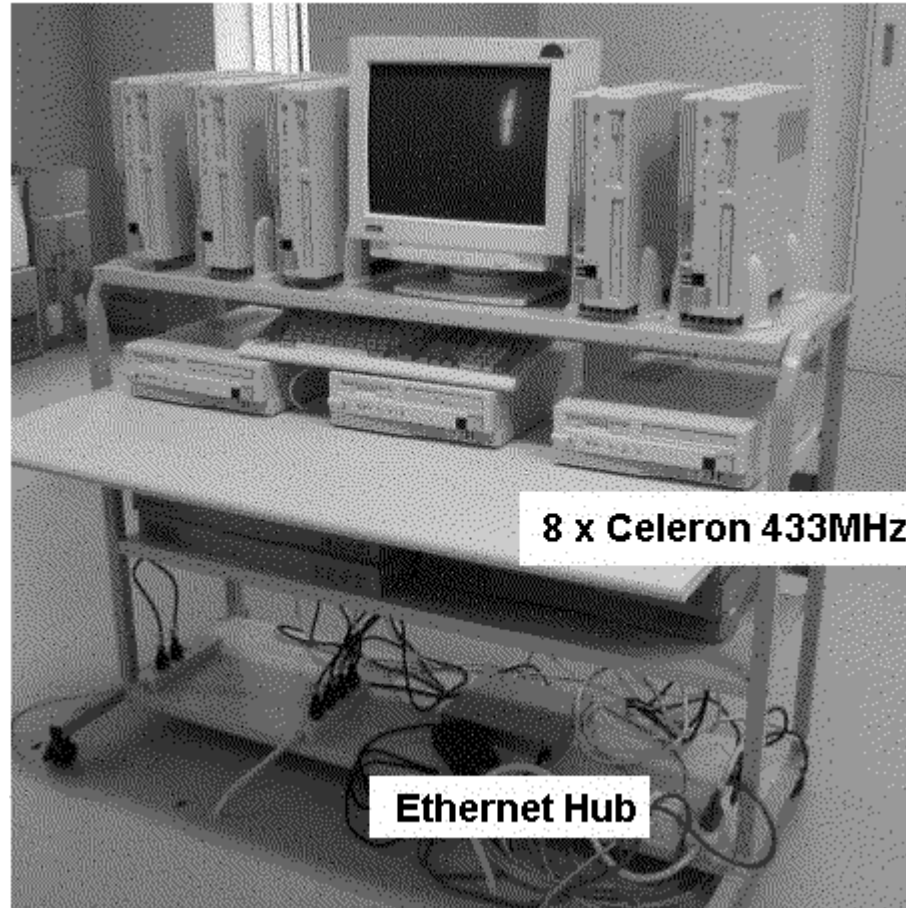
# ***Providing Cluster Environments with High-Availability and Load-Balancing***



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



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



8 x Celeron 433MHz

Ethernet Hub

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

In this second part of the tutorial 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.



# Agenda

- Introduction to the problem
- **High-Availability**
- **Virtual Server**
- **Integrating High-Availability and Virtual Server**
- **Experimental Results**
- **Conclusions and Future Work**

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

*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.*

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

*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 services because computer A presents hardware problems.*

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

*In a cluster we can have many virtual IPs services, where each IP can be linked to one or more services.*

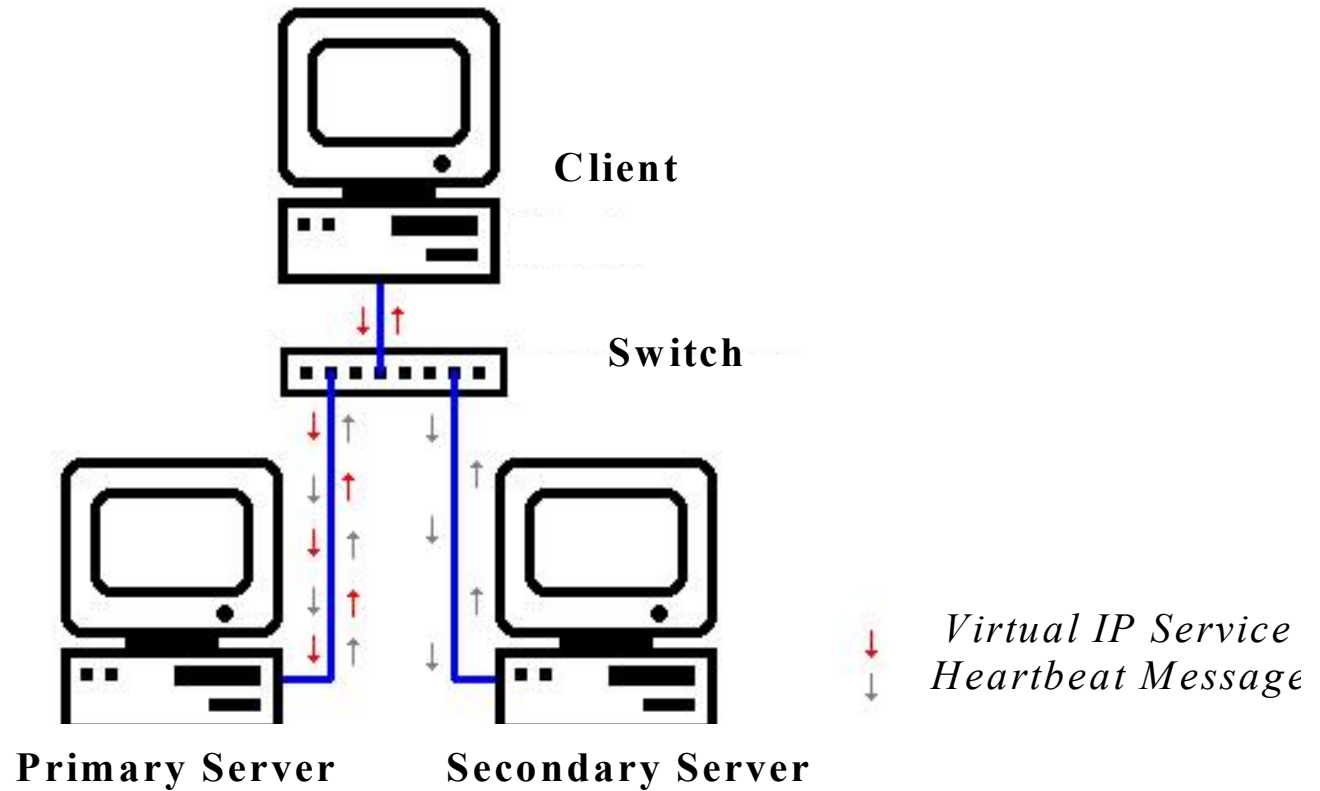
# ***Providing Cluster Environments with High-Availability and Load-Balancing***

*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 .*

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

*Therefore, the heartbeat is the element that decides which server will be responsible for assuming a certain virtual IP address.*

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



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

*The LHA works with daemons 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.*



# Agenda

- Introduction to the problem
- High-Availability
- **Virtual Server**
- **Integrating High-Availability and Virtual Server**
- **Experimental Results**
- **Conclusions and Future Work**

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

The project call *Linux Virtual Server (LVS)* is designed as an abstraction in which 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*.

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

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*.

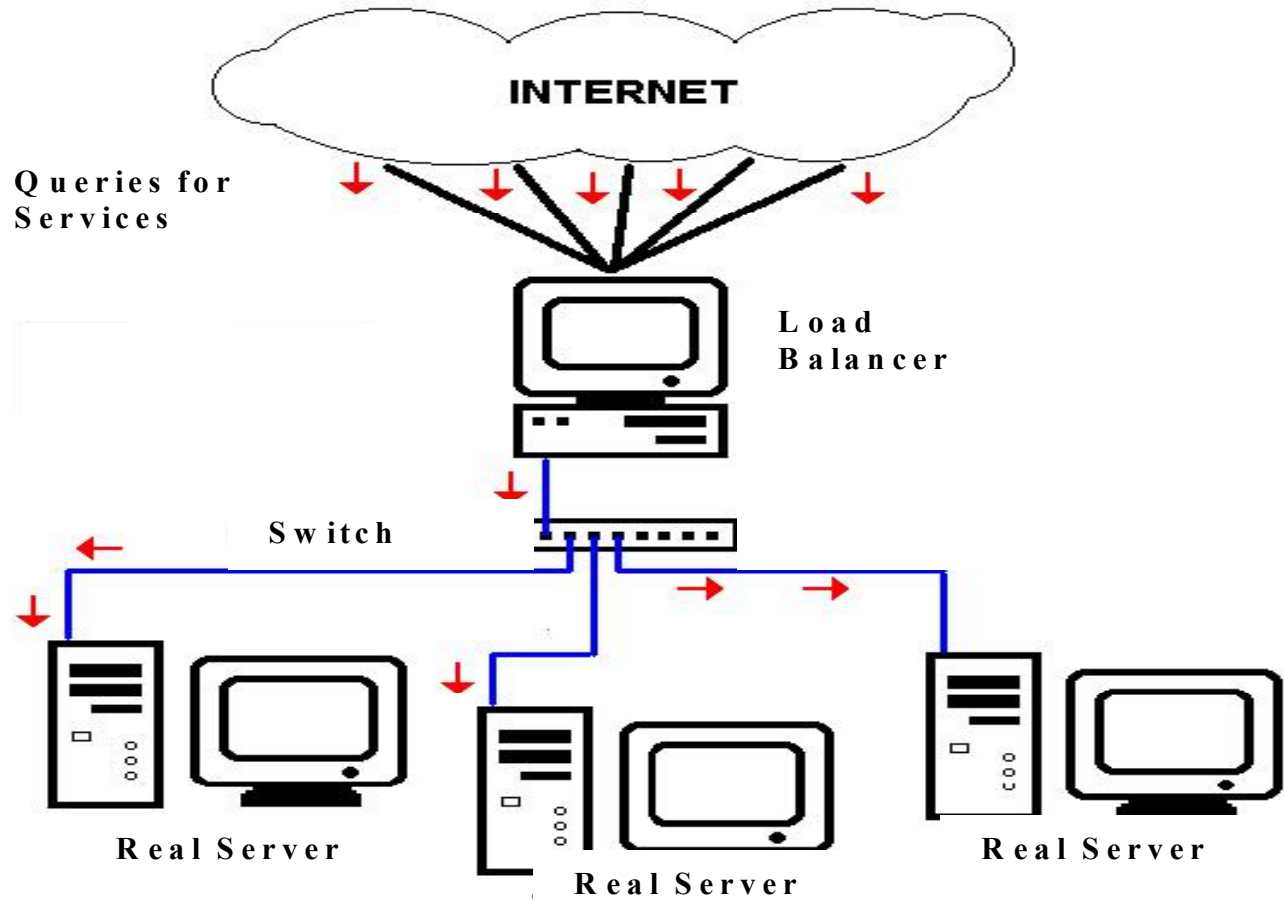
Reading the IP datagram, the *load balancer*, decides in which computer the task will execute. The external node does not know that another computer is executing the incoming request .

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

The main target of the *virtual server* approach is to provide 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.

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



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

<b>Processors</b>	<b>Pentium 100-100-233-233-233</b>
<b>Memory (Mbytes)</b>	<b>32-32-64-64-64</b>
<b>Operating system</b>	<b>Linux 6.0 kernel 2.2.17</b>

**Table I: Cluster configuration**

# Agenda

- Introduction to the problem
- High-Availability
- Virtual Server
- **Integrating High-Availability and Virtual Server**
- **Experimental Results**
- **Conclusions and Future Work**

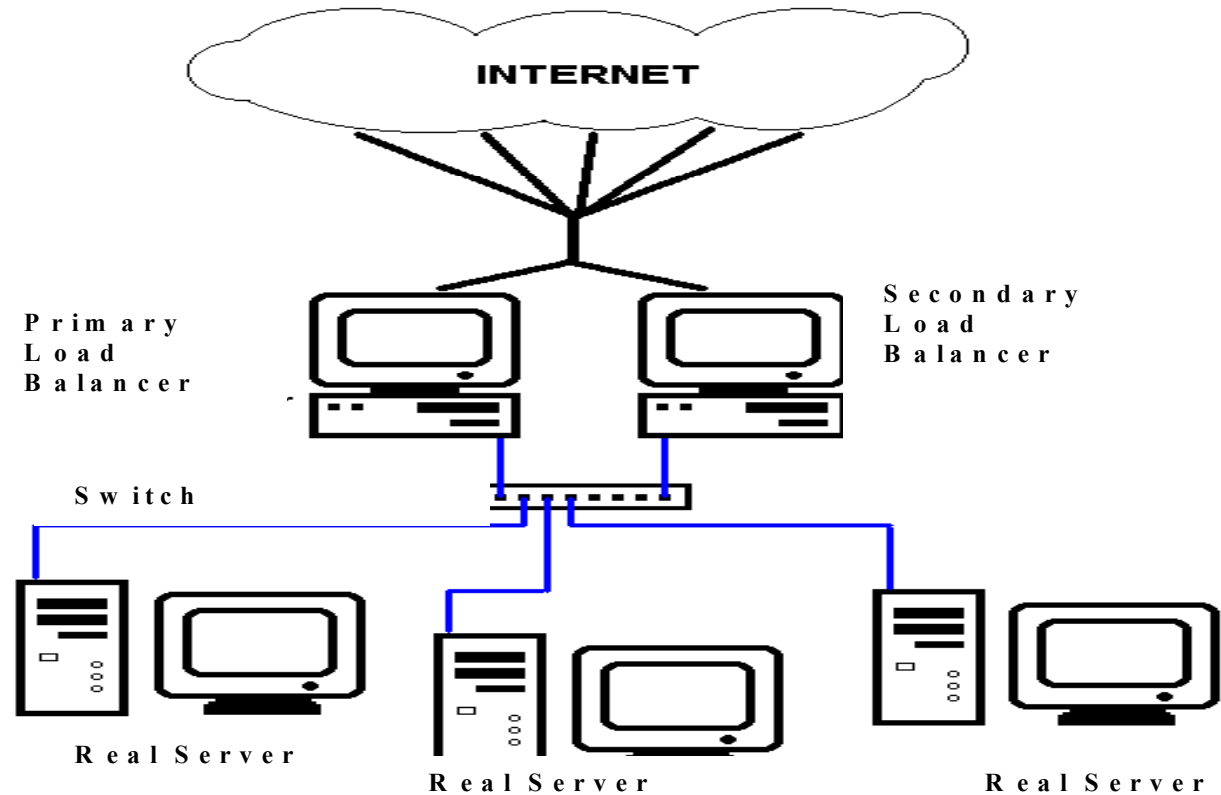
## ***Providing Cluster Environments with High-Availability and Load-Balancing***

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.



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



# Agenda

- Introduction to the problem
- High-Availability
- Virtual Server
- Integrating High-Availability and Virtual Server
- **Experimental Results**
- **Conclusions and Future Work**

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

All experimental results present in this section are based on our change of the cluster configuration.

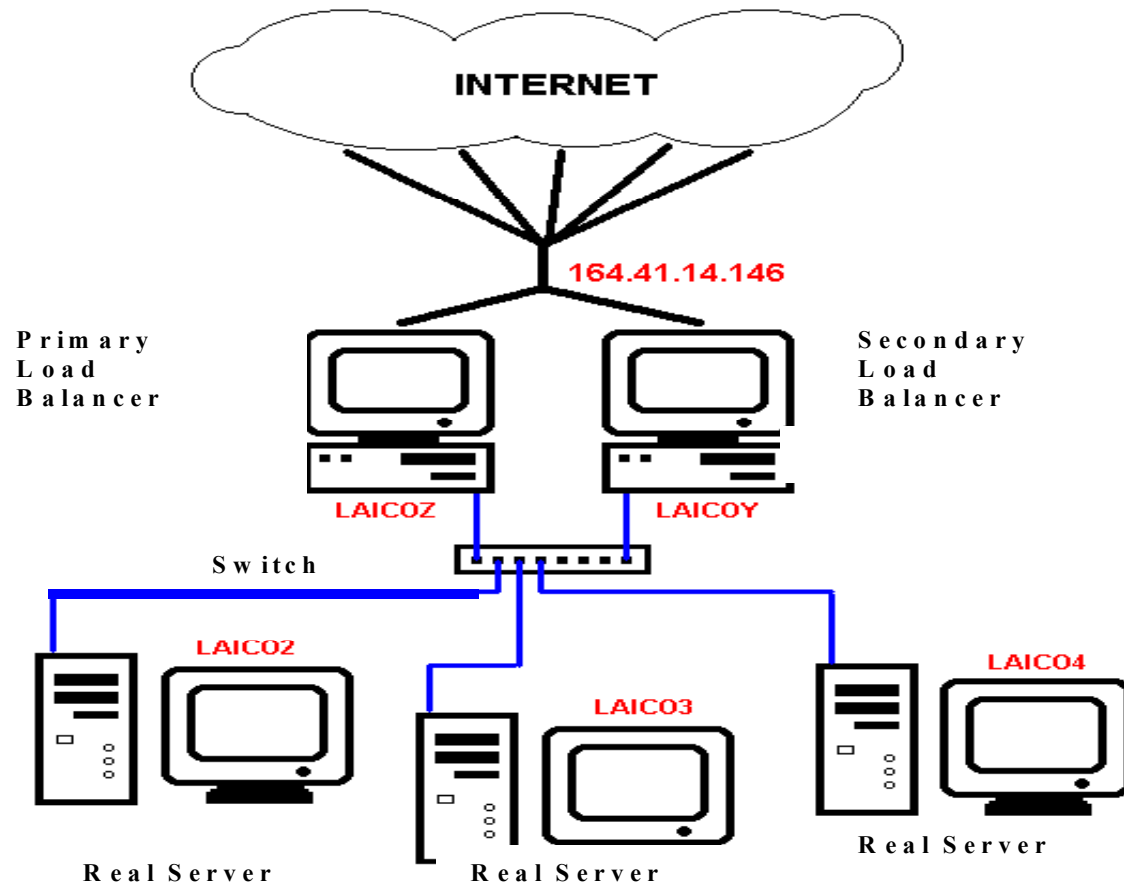
Therefore, we have created several study case situations to verify if our approach works as expected.

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

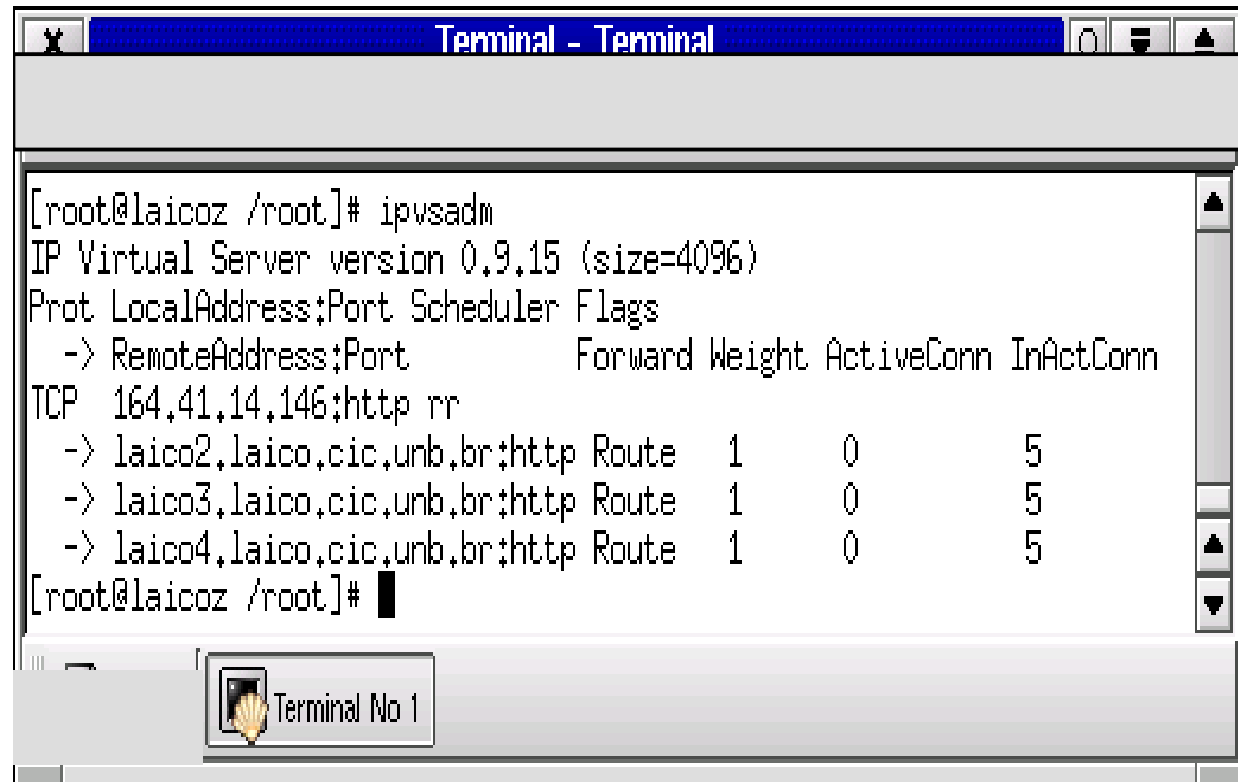
The next figure shows the IP addresses and names used for our experiments.

The integrated environment is formed 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*.

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

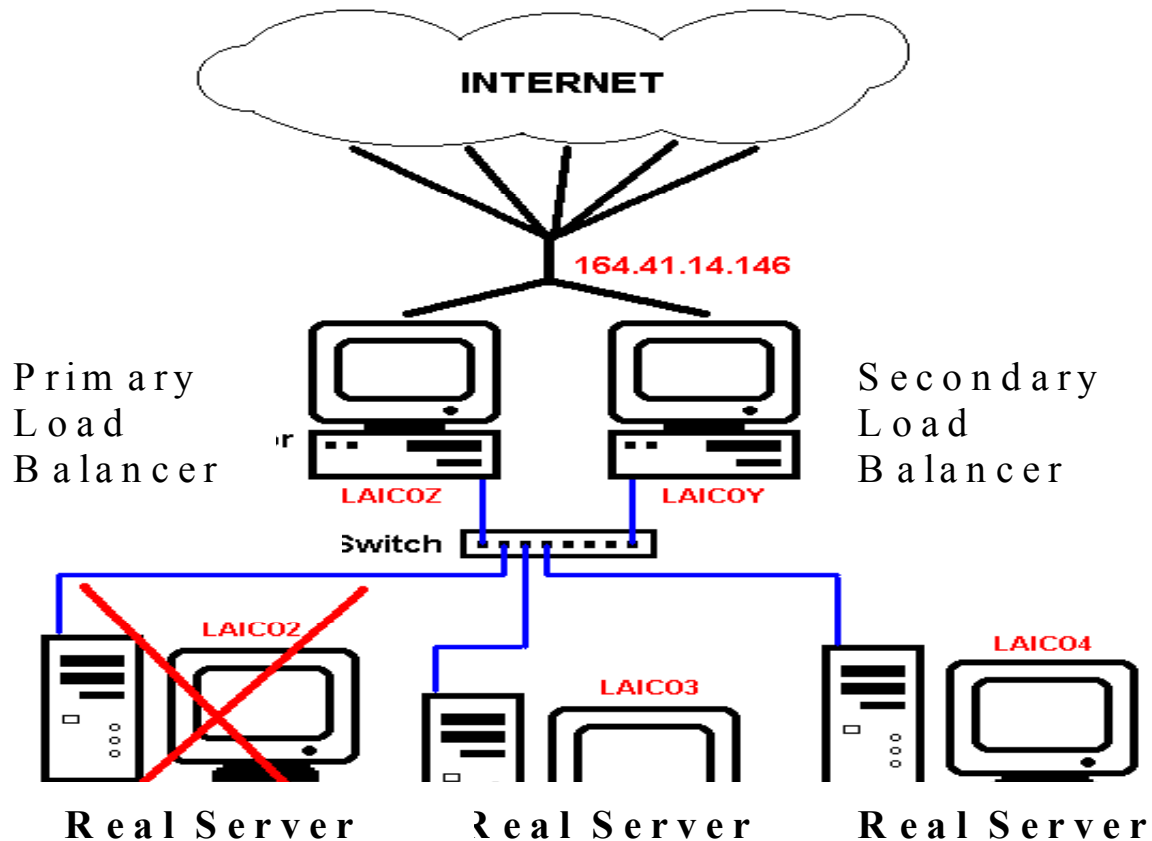


## *Providing Cluster Environments with High-Availability and Load-Balancing*

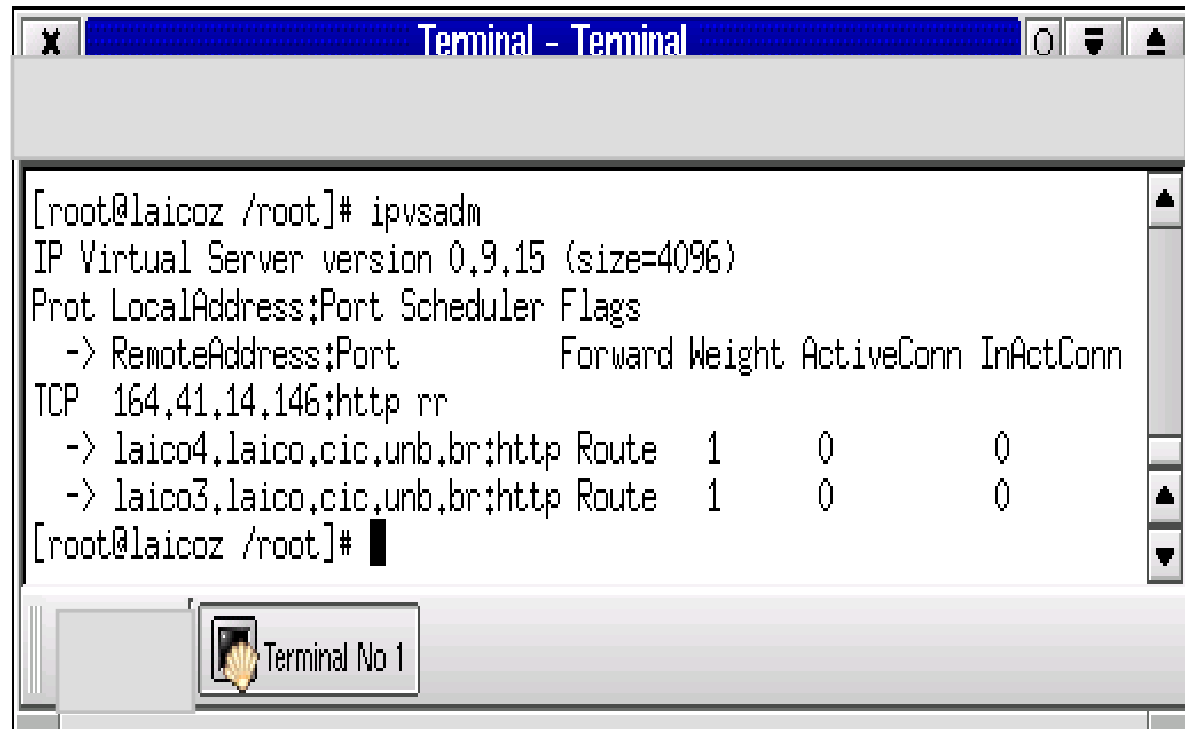
A terminal window titled "Terminal - Terminal" showing the output of the ipvsadm command. The output displays the IP Virtual Server version (0.9.15) and a table of virtual servers. The table has columns for Protocol, LocalAddress:Port, Scheduler, Flags, RemoteAddress:Port, Forward, Weight, ActiveConn, and InActConn. Three virtual servers are listed, all using the Round Robin (rr) scheduler and having a weight of 1 and 5 active connections.

```
[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]#
```

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



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

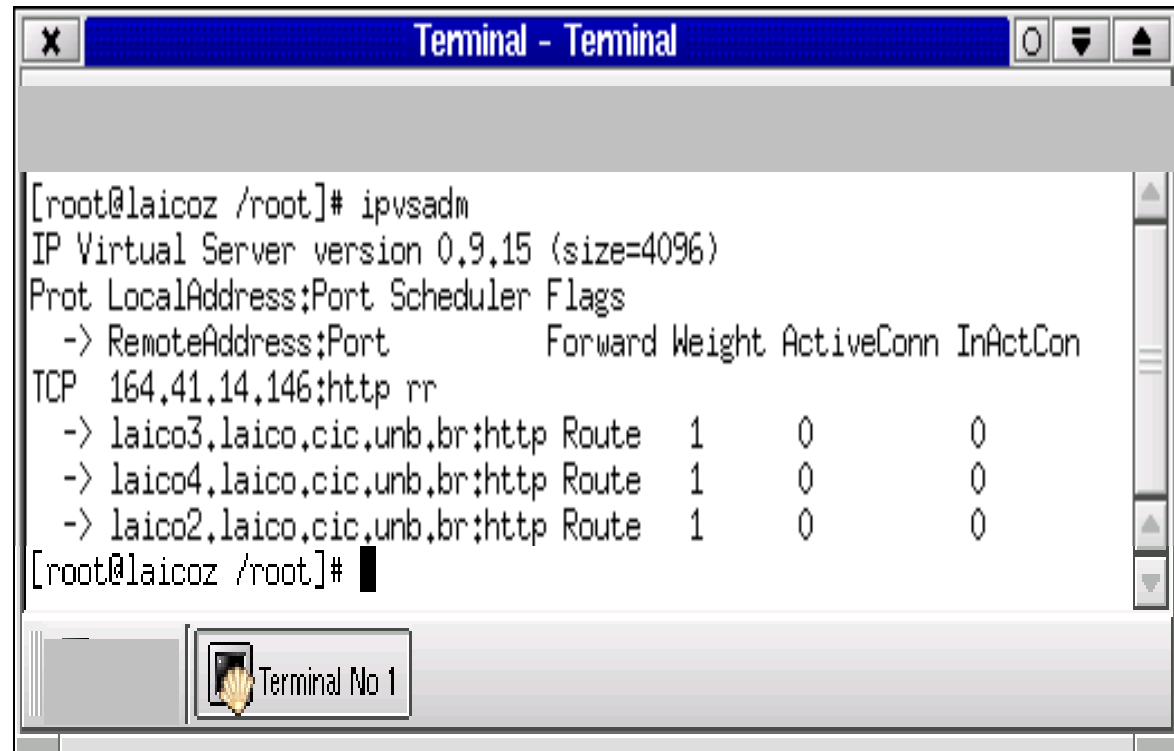


A terminal window titled "Terminal - Terminal" showing the output of the ipvsadm command. The output displays the IP Virtual Server version (0,9,15) and a table of virtual servers. The table has columns for Protocol, LocalAddress:Port, Scheduler, Flags, RemoteAddress:Port, Forward, Weight, ActiveConn, and InActConn. Two virtual servers are listed: one for TCP on 164.41.14.146:80 using the rr scheduler, and two for TCP on 164.41.14.146:80 using the Route scheduler, one pointing to laico4 and the other to laico3.

```
[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]#
```



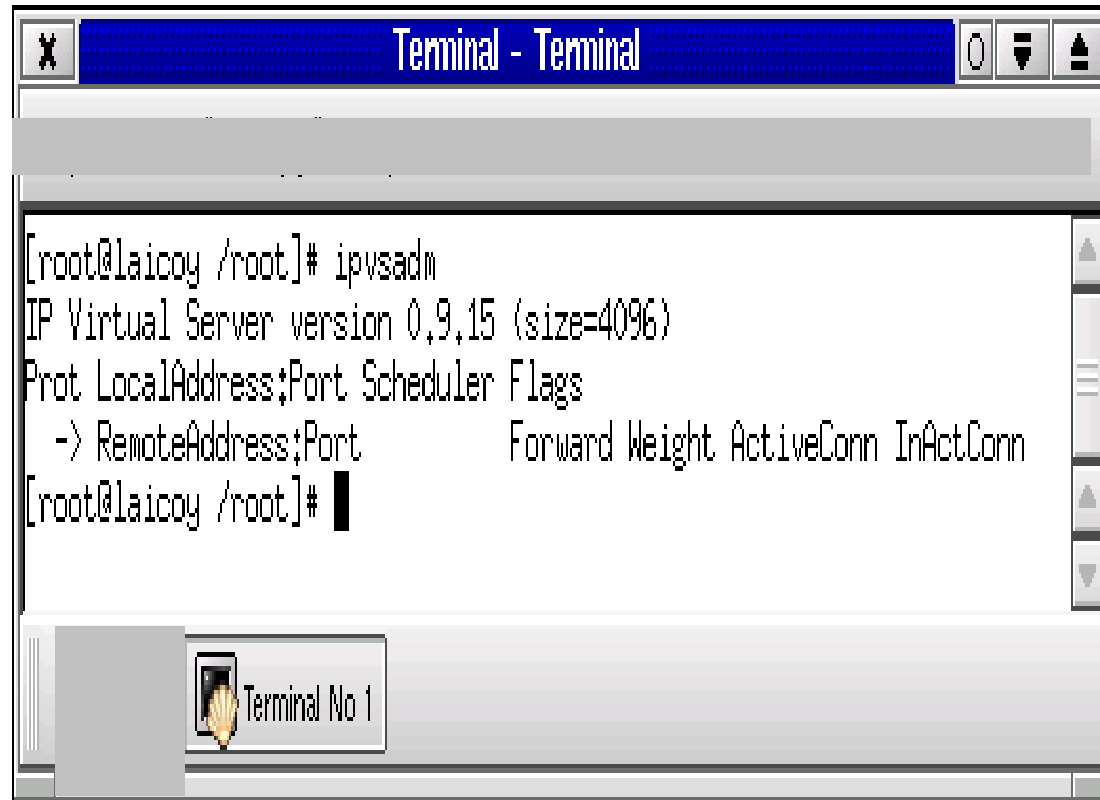
# *Providing Cluster Environments with High-Availability and Load-Balancing*



A terminal window titled "Terminal - Terminal" showing the output of the ipvsadm command. The output displays the IP Virtual Server version (0.9.15) and a table of virtual servers. The table has columns for Protocol, LocalAddress;Port, Scheduler, Flags, RemoteAddress;Port, Forward, Weight, ActiveConn, and InActCon. Three virtual servers are listed, all using the rr scheduler and Route flags, with a weight of 1 and 0 active connections.

```
[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]#
```

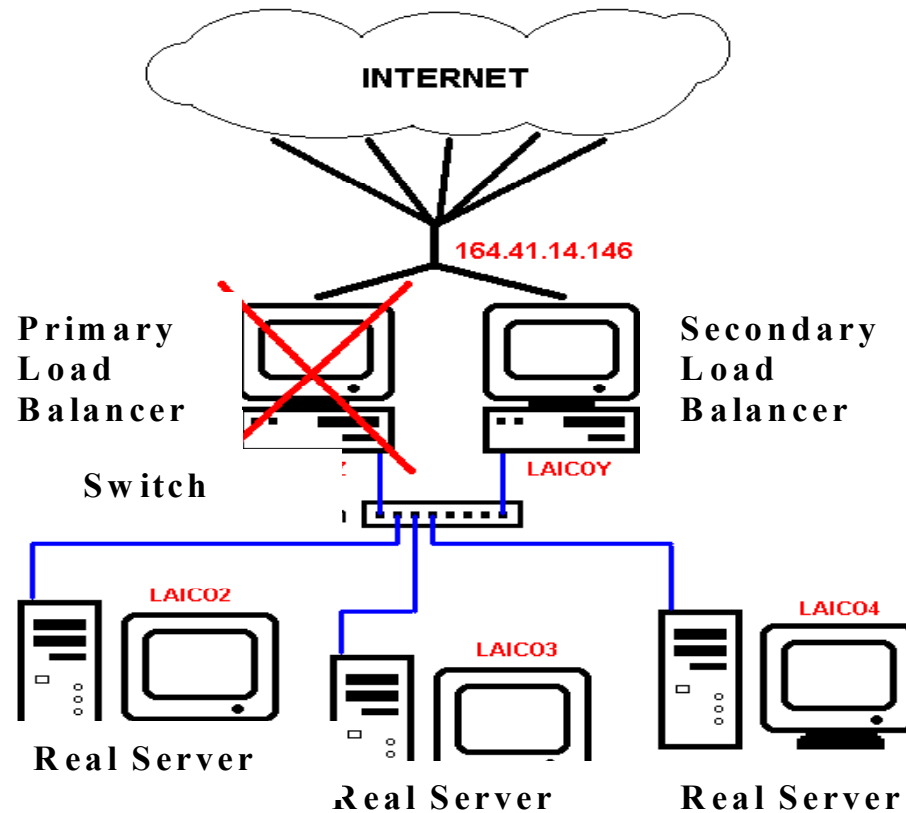
# *Providing Cluster Environments with High-Availability and Load-Balancing*



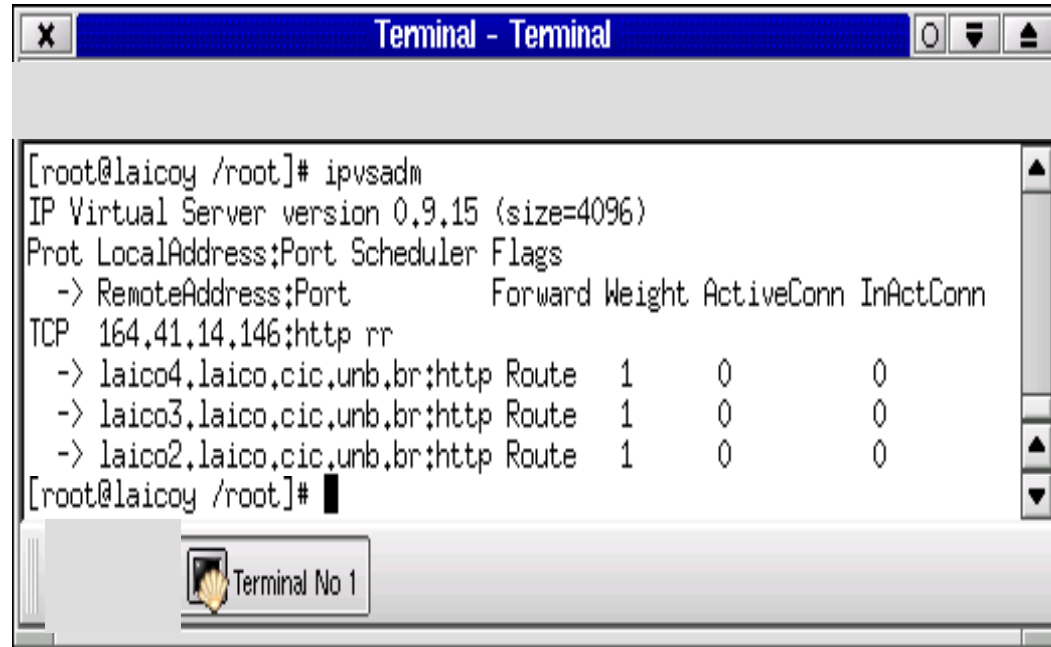
A terminal window titled "Terminal - Terminal" showing the execution of the `ipvsadm` command. The output displays the IP Virtual Server version (0.9.15) and a table header for the configuration. The table header includes columns for Protocol, Local Address:Port, Scheduler, Flags, Remote Address:Port, Forward, Weight, Active Connections, and Inactive Connections. The terminal prompt is `[root@laicoy /root]#`.

```
[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]#
```

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



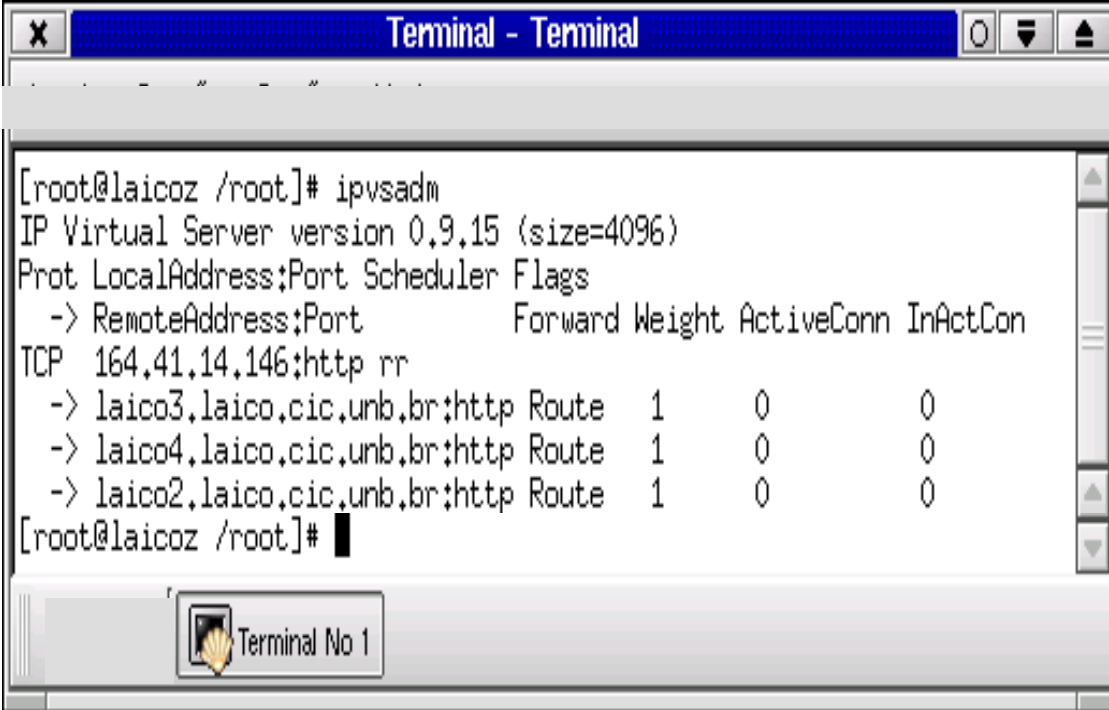
## *Providing Cluster Environments with High-Availability and Load-Balancing*



A terminal window titled "Terminal - Terminal" showing the output of the `ipvsadm` command. The output displays the IP Virtual Server version (0.9.15) and a table of virtual servers. The table has columns for Protocol, LocalAddress:Port, Scheduler, Flags, RemoteAddress:Port, Forward, Weight, ActiveConn, and InActConn. Three virtual servers are listed, all using the Round Robin (rr) scheduler and routing traffic to three different nodes (laico4, laico3, and laico2).

```
[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]#
```

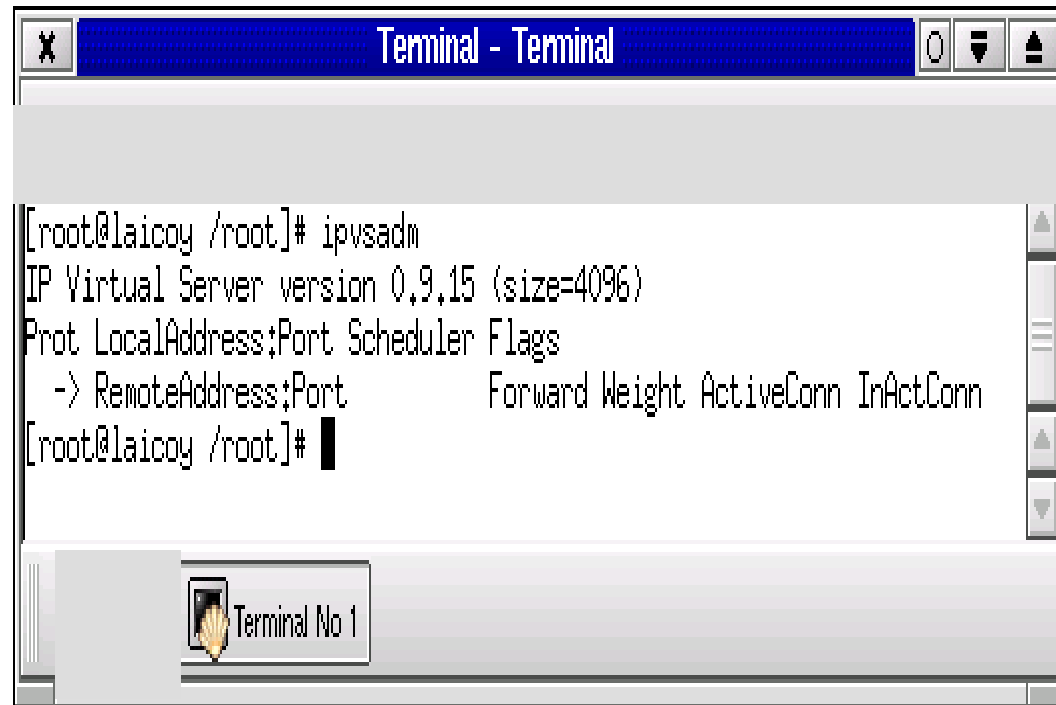
## *Providing Cluster Environments with High-Availability and Load-Balancing*



A terminal window titled "Terminal - Terminal" showing the output of the ipvsadm command. The output displays the IP Virtual Server version and a table of virtual servers. The table has columns for Protocol, LocalAddress:Port, Scheduler, Flags, RemoteAddress:Port, Forward, Weight, ActiveConn, and InActCon. Three virtual servers are listed, all using the rr scheduler and Route flags, with a weight of 1 and 0 active connections.

```
[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        0
  -> laico4.laico.cic.unb.br:http Route    1      0        0        0
  -> laico2.laico.cic.unb.br:http Route    1      0        0        0
[root@laicoz /root]#
```

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



A terminal window titled "Terminal - Terminal" showing the execution of the 'ipvsadm' command. The output displays the version of the IP Virtual Server (0.9.15) and a table header for the command's output. The table header includes columns for Protocol, Local Address and Port, Scheduler, Flags, Remote Address and Port, Forward, Weight, Active Connections, and Inactive Connections. The terminal prompt is [root@laicoy /root]#.

```
[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]#
```

# Agenda

- Introduction to the problem
- High-Availability
- Virtual Server
- Integrating High-Availability and Virtual Server
- Experimental Results
- **Conclusions and Future Work**

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

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.





**Questions ?**