



BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Universidad Nacional De La Plata

Facultad De Informática



Laboratorio de Procesamiento Paralelo Multi-Cluster accesible vía WEB


Tesina de Licenciatura en Sistemas

Autor: Adrián Pousa

Director: Ing. Armando E. De Giusti

Codirector: Dr. Marcelo Naiouf

La Plata Junio de 2007

| | |
|---|---|
| <p>TES 07/11 DIF-03096 SALA</p> |  <p>UNIVERSIDAD NACIONAL DE LA PLATA FACULTAD DE INFORMÁTICA Biblioteca 50 y 120 La Plata catalogo.info.unlp.edu.ar biblioteca@info.unlp.edu.ar</p>  <p>DIF-03096</p> |
|---|---|



BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

DONACION. FACULTAD.....

\$.....

Fecha. 12-3-08.....

Inv. E.....Inv. B. 003096.....

| |
|-------|
| TES |
| 07/11 |
| |

Esta tesina ha sido aceptada como demo educativo y exposición en TE&ET'07 (II Congreso de tecnología en educación y educación en tecnología), La Plata, Junio de 2007.

Agradecimientos

Quisiera agradecer:

a mi país por haberme dado la posibilidad de realizar mis estudios en instituciones públicas y gratuitas;

a la Universidad Nacional de La Plata y en particular a la Facultad de Informática por mi formación profesional;

a los tres laboratorios de la Facultad de Informática, LIFIA, LINTI y III-LIDI por las posibilidades que me dieron;

a todos los profesores que me educaron en esta profesión;

a Armando De Giusti y Marcelo Naiouf no solo por la dirección de mi tesis sino también por la idea y la motivación en su realización;

a todos aquellos que de forma desinteresada siempre estuvieron dispuestos a darme una mano;

a mis compañeros y compañeras de trabajo, conocidos, grandes amigos y amigas por la paciencia, la comprensión, el apoyo y el inmenso cariño;

a mi familia, mis padres, mi hermano y mi recordada abuela pilares fundamentales en mi vida y a los que les estoy eternamente agradecido.

Gracias a todos, gracias a Dios.



Índice

| | |
|-----------------------------|---|
| Objetivo - Motivación | 1 |
| Resumen | 5 |

Capítulo 1 - Procesamiento sobre cluster

| | |
|--|----|
| 1.1 Historia | 11 |
| 1.2 Definición | 12 |
| 1.3 Bibliotecas y lenguajes para ejecución sobre un cluster..... | 13 |
| 1.4 Configuración y administración | 14 |

Capítulo 2 - Acceso remoto

| | |
|---------------------------------------|----|
| 2.1 Introducción..... | 17 |
| 2.2 Tecnologías de acceso remoto..... | 18 |
| 2.3 Acceso remoto al cluster..... | 20 |

Capítulo 3 - Ejecución remota de programas

| | |
|---|----|
| 3.1 Introducción..... | 23 |
| 3.2 Elección de un lenguaje apropiado | 24 |
| 3.3 Ejecución de comandos de sistema y programas desde JAVA | 27 |
| 3.3.1 La clase Runtime | 27 |
| 3.3.2 Ejecución de comandos o programas | 27 |
| 3.3.3 El objeto Process | 28 |
| 3.3.4 Ejecución sincrónica y asincrónica | 28 |
| 3.3.5 Código de salida | 28 |
| 3.3.6 Obteniendo la salida de la ejecución | 29 |
| 3.3.7 Algunos riesgos de utilizar la clase Runtime | 31 |
| 3.4 JAVA Threads y memoria compartida utilizando monitores..... | 32 |
| 3.4.1 Threads | 32 |
| 3.4.1.1 Como subclase de la clase Thread | 33 |
| 3.4.1.2 Implementando la interface Runnable | 33 |
| 3.4.2 Memoria compartida utilizando monitores | 34 |
| 3.4.2.1 Monitores con métodos synchronized..... | 34 |
| 3.4.2.2 Monitores con locks y variables condición..... | 35 |
| 3.4.2.3 Comentarios sobre el uso de monitores en JAVA | 37 |
| 3.5 Acceso remoto vía Web a una aplicación JAVA | 38 |
| 3.6 Aplicación al cluster | 39 |

Capítulo 4 - Aplicación Web desarrollada

| | |
|---|----|
| 4.1 Introducción..... | 43 |
| 4.2 Aplicaciones RIA, uso de AJAX..... | 44 |
| 4.3 Manejo de usuarios..... | 46 |
| 4.3.1 Usuarios a nivel de aplicación..... | 46 |
| 4.3.2 Usuarios a nivel de sistema | 47 |
| 4.4 Acceso compartido y exclusivo..... | 49 |
| 4.5 áreas de la aplicación..... | 51 |
| 4.5.1 Area de ejecución | 52 |
| 4.5.1.1 Espacio de trabajo | 52 |

| | |
|--|----|
| 4.5.1.2 Compilación de código fuente | 52 |
| 4.5.1.3 Ejecución de programas (modo sincrónico, asincrónico e interactivo) | 53 |
| 4.5.2 Area de programas en ejecución | 55 |
| 4.5.3 Area de cambio de contraseña | 55 |
| 4.5.4 Area de manejos de usuario | 55 |
| 4.5.5 Area de manejo de clusters | 55 |
| 4.5.6 Area de ambientes de ejecución..... | 55 |
| 4.5.7 Area de asignación de accesos exclusivos | 56 |
| 4.5.8 Area de registros de ejecución | 56 |
| 4.6 Aspectos de seguridad | 57 |

Capítulo 5 - Configuración de la arquitectura

| | |
|---|----|
| 5.1 Introducción..... | 61 |
| 5.2 Configuración del servidor | 62 |
| 5.2.1 Sistema operativo..... | 62 |
| 5.2.2 Espacio de trabajo de los usuarios de la aplicación | 63 |
| 5.2.3 Usuario de sistema..... | 64 |
| 5.2.4 Configuración JAVA | 65 |
| 5.2.5 Servidor Web..... | 66 |
| 5.2.6 Ambientes de ejecución MPI..... | 68 |
| 5.2.6.1 LAM - MPI | 69 |
| 5.2.6.2 MPICH2 - MPI | 71 |
| 5.2.7 Configuración SSH, RSH, acceso sin autenticación | 73 |
| 5.2.8 Variables de ambiente locales a un usuario | 74 |
| 5.2.9 NFS (Network File System) | 75 |
| 5.2.10 MySQL | 76 |
| 5.3 Configuración de una máquina perteneciente al cluster..... | 77 |
| 5.3.1 Sistema operativo..... | 77 |
| 5.3.2 Usuario de sistema | 77 |
| 5.3.3 Configuración JAVA | 78 |
| 5.3.4 Ambientes de ejecución MPI..... | 78 |
| 5.3.5 Configuración SSH, RSH, acceso sin autenticación | 78 |
| 5.3.6 Variables de ambiente locales a un usuario | 78 |
| 5.3.7 NFS (Network File System) | 78 |

Capítulo 6 - Conclusiones y líneas de investigación y desarrollo abiertas

| | |
|--|----|
| 6.1 Conclusiones..... | 83 |
| 6.2 Líneas de investigación y desarrollo abiertas..... | 84 |
| 6.2.1 Aspectos de seguridad en una arquitectura multicluster..... | 85 |
| 6.2.2 Schedulers y manejador de recursos..... | 86 |
| 6.2.3 Fragmentación y escalabilidad con clusters en redes remotas..... | 87 |
| 6.2.4 Generalizaciones a Grid..... | 88 |

A. Manual de usuario

| | |
|--|-----|
| A1. Pantalla inicial – autenticación de usuario..... | 93 |
| A2. Pantalla principal | 94 |
| A2.1 Modo Administrador..... | 94 |
| A2.2 Modo Usuario | 95 |
| A3. Área de ejecución | 96 |
| A3.1 Espacio de trabajo | 97 |
| A3.2 Menú de ambientes de ejecución | 98 |
| A3.3 Zona de ejecución | 98 |
| A3.3.1 Compilación..... | 98 |
| A3.3.2 Ejecución..... | 99 |
| A3.3.3 Visualización..... | 99 |
| A4. Área de manejo de usuarios..... | 100 |
| A5. Área de manejo de clusters..... | 101 |
| A6. Área de procesos en ejecución..... | 102 |
| A7. Área de ambientes de ejecución MPI | 103 |
| A8. Área de asignaciones exclusivas..... | 104 |
| A9. Área de registro de ejecuciones | 105 |
| A10. Área de cambio de contraseña | 106 |

B. Documentación y diagramas

| | |
|--|-----|
| B1. Diagrama de clases | 109 |
| B2. Descripción de los objetos | 110 |
| B3. Descripción de casos de uso | 111 |
| B3.1 Descripción básica de casos de usos | 111 |
| B3.2 Descripción detallada de casos de usos particulares | 113 |
| B4. Diagramas de interacción..... | 116 |
| B5. Diagrama de base de datos..... | 122 |
| B6. Tablas resultantes..... | 123 |

C. Herramientas utilizadas durante el desarrollo

| | |
|--|-----|
| C1. Introducción..... | 127 |
| C2. Sistema operativo..... | 127 |
| C3. Lenguaje de programación..... | 127 |
| C4. Servidor Web y contenedor de aplicaciones..... | 128 |
| C5. Ambiente de desarrollo | 128 |
| C6. Ambientes de ejecución en MPI | 129 |
| C7. Motor de base de datos | 130 |

| | |
|-------------------|-----|
| Bibliografía..... | 131 |
|-------------------|-----|

Objetivo - Motivación

Objetivo

Desarrollar una aplicación WEB que permita la ejecución remota de aplicaciones en una arquitectura multicluster heterogénea (en principio los clusters existentes en la Facultad) así como la administración de dicha arquitectura.

Motivación

La utilización de redes de computadoras como máquinas paralelas virtuales se ha generalizado y el desarrollo de software de administración de recursos físicos, comunicaciones y software disponible en clusters, multiclusters y grids es tema actual de investigación y desarrollo.

En particular en la facultad existen varios subsistemas en red que son utilizados por alumnos, docentes e investigadores como máquinas paralelas virtuales: Cluster de 16 máquinas (CL1), Cluster Sala IBM (CL2) y Cluster Sala 101 (CL3). Además se realizan investigaciones de procesamiento paralelo empleando todas las máquinas de CL1, CL2 y CL3.

La capa de software de administración de estos recursos y en particular su acceso remoto por alumnos e investigadores requiere un esfuerzo de desarrollo importante, para ofrecer una vista “transparente” de los recursos. Asimismo es de particular interés la posibilidad de acceso desde otras Universidades.

El acceso de los distintos usuarios a los recursos del cluster requiere de un análisis sobre herramientas para el control de acceso a los mismos, anterior a la aplicación Web desarrollada el acceso era local o remoto pero de una forma tal vez poco transparente y no existía un control sobre la arquitectura de cluster.

Es por eso que resulta de importancia el desarrollo de una aplicación Web, utilizando las actuales tecnologías de desarrollo y herramientas de software libre, que permitiera controlar y administrar los recursos de la arquitectura de cluster además de permitir el uso de dicha arquitectura a alumnos, docentes e investigadores de una manera controlada.

Resumen

Resumen

En la Facultad de Informática existen varios clusters que son accedidos por alumnos, docentes e investigadores como máquinas paralelas virtuales. El acceso de los distintos usuarios a los recursos del cluster era local o remoto pero de una forma poco transparente y no existía un control sobre la arquitectura de cluster. Para solucionar estos inconvenientes se decidió hacer una aplicación Web que permitiera el uso de los distintos usuarios como también la administración de la arquitectura de cluster.

El desarrollo de una aplicación Web para este tipo de arquitectura no solo incluye la implementación de la aplicación sino que es necesario tener en cuenta varios aspectos que requieren de investigación y configuración además del desarrollo.

Fue necesaria la investigación sobre arquitecturas de cluster y multicluster, administración y ejecución de programas en las mismas basadas en bibliotecas de pasajes de mensajes como son MPI o PVM.

También se investigó sobre formas de acceso remoto como las basadas en terminales SSH o telnet, las basadas en escritorio remoto en ambientes como GNOME, KDE o Windows, y por último las aplicaciones Web, haciendo comparaciones entre todas y determinando que la aplicación Web es la forma de acceso que mejor se adapta a la arquitectura de cluster de acuerdo a los requerimientos en la Facultad.

Fue necesario investigar sobre qué lenguaje de programación correría la aplicación Web teniendo en cuenta PHP y JAVA, considerando JAVA más adecuado por adaptarse mejor a los requerimientos de la aplicación Web en relación a ejecutar programas y comandos de sistemas, manejar threads, exclusión mutua y establecer sincronización.

Se aprovecharon para el desarrollo, las nuevas tecnologías aplicadas a aplicaciones Web como AJAX lo que la convierte en una aplicación Web denominada RIA (**R**ich **I**nternet **A**pplication).

A nivel de la aplicación fue necesario establecer los perfiles de usuarios, y las diferencias entre perfiles determinan las actividades que pueden realizar sobre el cluster. Estas actividades van desde el uso mismo de la arquitectura hasta su administración y están separadas en áreas.

Se definió que cada usuario tenga un espacio de trabajo dentro de la aplicación Web a manera de sistema de archivos donde se permiten distintas operaciones tradicionales sobre archivos o directorios, además de compilar y ejecutar archivos en ambientes basados en librerías MPI.

Para que la aplicación Web funcione fue necesario montar y configurar un servidor Web, un motor de base de datos, así como también montar y configurar toda la arquitectura de cluster para adaptarla a la aplicación Web.

Los aspectos de seguridad no son menores y se definieron algunos controles mínimos para evitar problemas.

Fue importante la utilización de software de distribución gratuita que van desde el sistema operativo, herramientas de desarrollo, lenguajes y ambientes de ejecución.

Se prevé seguir evolucionando con la aplicación Web hacia características que tienen que ver con dar mayor seguridad, incluir schedulers y gestores de colas de trabajos, incluir clusters que no estén en la misma red y adaptar la aplicación a entornos de Grid.

Breve descripción de los capítulos

Este documento se divide en varios capítulos que en detalle hacen referencia a los distintos aspectos antes mencionados.

El objetivo del primer capítulo es hacer una introducción a la noción de cluster de computadoras además de describir bibliotecas de procesamiento distribuido como MPI y PVM, las cuales son utilizadas para la ejecución en el cluster.

El segundo capítulo trata sobre las formas de acceso remoto que se podrían dar en una arquitectura de cluster considerando la importancia de un acceso mediante una aplicación Web de acuerdo a los requerimientos que se han dado.

Uno de los problemas que se plantea es como ejecutar programas en un cluster a partir de una aplicación Web escrita en un determinado lenguaje. El siguiente capítulo explica por qué JAVA es un lenguaje apropiado y cómo hace para ejecutar programas externos, además explica herramientas a utilizar en la construcción de la aplicación como son los threads y monitores en JAVA, comenta como es accedida una aplicación JAVA vía Web y como este conjunto de elementos se pueden aplicar a la ejecución y administración del cluster.

El cuarto capítulo detalla el funcionamiento de la aplicación Web implementada, se da una introducción sobre las nuevas tecnologías de desarrollo Web utilizadas como es el caso de AJAX, se explica además el manejo y políticas de usuarios, el acceso compartido y exclusivo al cluster y las distintas áreas de la aplicación Web.

El siguiente capítulo explica cómo fue configurada toda la arquitectura, desde el servidor para montar la aplicación Web hasta cada máquina del cluster para que sea accesible desde la aplicación.

El capítulo seis esta reservado a las conclusiones y a aquellas líneas de investigación y desarrollo que quedan abiertas como por ejemplo los aspectos de seguridad, la ejecución sobre el cluster utilizando manejadores de recursos y schedulers, la fragmentación y escalabilidad de una arquitectura multicluster y la generalización a una infraestructura de Grid.

Se incluyen tres anexos, el primero es un manual de usuario de la aplicación Web, el segundo es documentación de desarrollo y el tercero describe las herramientas utilizadas durante el desarrollo destacando la importancia de ser software gratuito y disponible en la Web.

Capítulo 1

Procesamiento sobre clusters

1.1 Historia

A medida que la tecnología, tanto de hardware como de software de computadoras, fue avanzando, se incrementó la necesidad de resolver problemas, más complejos aún. La solución de estos problemas requerían de una cantidad determinada de recursos y abrían caminos a nuevos problemas, más complejos aun, que requerían más recursos.

Esta demanda nace del avance de la tecnología y crea un círculo, cuando se tienen mas recursos se resuelven los problemas planteados y entonces se piensa en resolver problemas más complejos y, nuevamente, se llega a la conclusión de que se necesitan más recursos.

Existen problemas complejos dentro de áreas como:

- Modelos de sistemas biológicos.
- Reacciones químicas.
- Física de partículas.
- Predicción meteorológica.

Estos problemas llegaron a necesitar de las llamadas supercomputadoras, sistemas con capacidades de procesamiento paralelo y de gran potencia, capaces de realizar cientos de millones de operaciones en coma flotante por segundo.

Algunas de estas supercomputadoras fabricadas son las siguientes:

- **BlueGene/L:** fabricada por IBM, con una potencia de 281 Teraflops (281 billones de operaciones de cálculo por segundo). Construida para el departamento de energía de EE.UU. ubicada en el Laboratorio Nacional Lawrence Livermore. Está constituida por 131.072 procesadores distribuidos en 64 estantes.
- **Earth Simulator Japonais:** fabricada por la empresa japonesa NEC, está formada por 5120 procesadores distribuidos en 320 estantes, que ocupan 3250 metros cuadrados, y su poder es de 36 Teraflops (36 billones de operaciones de cálculo por segundo).
- **Tera-10:** fabricada por la empresa francesa Bull con una potencia superior a los 60 teraflops (60 billones de operaciones por segundo).

Estas supercomputadoras son de un altísimo costo (millones de dólares) tanto en la inversión monetaria como en su mantenimiento y en su actualización, no están al alcance de cualquiera, inclusive solo se han fabricado unas pocas supercomputadoras, en algunos casos un solo equipo. De todos modos el aporte a la ciencia de estas maquinas han justificado su costo.

Un tipo particular, y más económico, de “supercomputadoras” se logra mediante la agrupación de muchas computadoras en lo que se denomina cluster.

1.2 Definición

Una definición básica de un cluster podría ser:

“Un conjunto de computadoras conectadas a través de una red y que se comportan como si fuesen una única computadora., trabajando en un gran problema de computo que ha sido dividido en varios subproblemas pequeños.”

De la misma forma en que se agrupan las maquinas para formar un cluster, los cluster se pueden agrupar para formar lo que se denomina una arquitectura multicluster.

De un cluster se espera que presente combinaciones de los siguientes servicios:

- Alto rendimiento (High Performance).
- Alta disponibilidad (High Availability).
- Equilibrio de carga (Load Balancing).
- Escalabilidad (Scalability).

Existen diferentes configuraciones de hardware y software disponible para trabajar en clusters, dividiendo los problemas y aprovechando al máximo los recursos disponibles. De esto surgen los paradigmas de resolución de problemas que tienen sus bases en el “computo paralelo”.

El paralelismo consiste en poder dividir una tarea en partes que trabajen independientemente, posee dos componentes bien diferenciados:

- Paralelismo a nivel hardware

Puede realizarse a nivel del mismo procesador y simultáneamente a nivel del sistema en su conjunto.

- El paralelismo a nivel del procesador consiste en la capacidad que tienen los nuevos procesadores para realizar más de una cosa a la vez.
- El paralelismo a nivel del sistema es donde se pueden ubicar los clusters, junto a las computadoras de múltiples procesadores. Se necesitará una forma de compartir los recursos de memoria.

- Paralelismo a nivel software

Dado un problema en particular, el objetivo es encontrar áreas bien definidas que pueden resolverse independientemente de las demás.

Los beneficios que puedan obtenerse están relacionados con la capacidad de descomponer un programa en módulos que puedan resolverse de forma independiente. Muchos problemas tienen por naturaleza una solución paralela, otros problemas, sin embargo, sólo pueden resolverse de forma secuencial, pero pueden reformularse para aprovechar las ventajas de un sistema de este tipo.

1.3 Bibliotecas y lenguajes para ejecución sobre un cluster

Existen diversos tipos de bibliotecas y lenguajes que permiten la ejecución sobre un cluster.

Los lenguajes que básicamente se usan son:

- C
- C++
- JAVA
- FORTRAN
- PERL
- PYTHON
- ADA

Cada uno de estos lenguajes requiere de una configuración particular en el cluster para poder funcionar.

Las librerías más conocidas para ejecución en el cluster son las de programación distribuida (utilizando pasaje de mensajes) como:

- MPI (*Message Passing Interface*)

Es un protocolo de comunicación entre computadoras, estándar para la comunicación entre los nodos que ejecutan un programa en un sistema de memoria distribuida. Consiste en un conjunto de bibliotecas de rutinas que pueden ser utilizadas en programas escritos en los lenguajes de programación C, C++, Fortran y Ada. Los programas que utilizan la biblioteca son portables (dado que MPI ha sido implementado para casi toda arquitectura de memoria distribuida), y rápidos, (porque cada implementación de la librería ha sido optimizada para el hardware en la cual se ejecuta).

Existen distintas distribuciones de esta librería, como por ejemplo LAMMPI, MPICH2 y JMPI (implementación de la librería para JAVA). Todas respetan el estándar, y en consecuencia el mismo código puede ser compilado en cualquiera de estas distribuciones (obviamente utilizando el mismo lenguaje de programación), salvo que un programa compilado en una distribución no puede correr en otra.

- PVM (*Parallel Virtual Machine*)

Es una librería para el cómputo paralelo en un sistema distribuido de computadoras. Está diseñado para permitir que una red de computadoras heterogénea comparta sus recursos de cómputo (como el procesador y la memoria RAM) con el fin de aprovechar esto para disminuir el tiempo de ejecución de un programa al distribuir la carga de trabajo en varias computadoras.

La librería funciona en lenguajes como C, C++ y Fortran, aunque existe una versión para JAVA llamada JPVM.

MPI ha ganado más popularidad que PVM por cuestiones de performance.

1.4 Configuración y administración

La configuración y administración de una arquitectura de cluster no es algo sencillo y requiere de varias tareas, cada una con mayor o menor complejidad. Dentro de estas se encuentran:

- Armado de la red.
- Instalación y configuración de sistemas operativos.
- Instalación y configuración de herramientas y servicios de sistema, como SSH, RSH, NFS, Lenguajes de programación, Compiladores etc.
- Configuración de red de los equipos.
- Instalación y configuración de librerías para procesamiento distribuido como MPI o PVM.
- Control y administración del acceso de usuarios a la arquitectura para ejecución.

Muchas de estas tareas no pueden ser automatizadas mediante alguna aplicación, en cambio otras podrían integrarse en un tipo de aplicación que permita controlarlas y administrarlas inclusive remotamente, como puede ser una aplicación Web.

Capitulo 2

Acceso remoto

2.1 Introducción

La Facultad de Informática cuenta con varios clusters que son utilizados por docentes, investigadores y alumnos. Hasta el momento el acceso se realiza de dos formas, local y remotamente mediante el protocolo SSH a manera de consola de texto.

El acceso local requiere que el usuario se acerque al lugar físico donde esta el cluster, este tipo de acceso no es bueno para los alumnos debido a que sólo pueden trabajar cuando tienen la oportunidad de usar el cluster.

El acceso remoto mediante SSH es más útil porque permite utilizar el cluster en cualquier momento o al menos cuando se permite el acceso. El inconveniente es que no es una forma transparente y además requiere controles a nivel firewall, permite al usuario ingresar al sistema y tal vez ver información que no debería.

Últimamente se ha incrementado el uso de las arquitecturas de cluster por lo que se requiere una forma de acceso diferente, en particular un acceso de forma remota en lo posible segura y transparente al punto que los usuarios solo vean lo que tienen que ver y puedan interactuar con el sistema de forma amigable.

Esta demanda requiere del estudio de accesos remotos y del método más adecuado para arquitecturas de este tipo.

2.2 Tecnologías de acceso remoto

La idea de acceder remotamente a una arquitectura de cluster para ejecución requiere de la investigación de tecnologías de acceso remoto, entre las cuales se encuentran:

- Telnet

Telnet es el nombre de un protocolo, que tal vez hoy no se utilice tanto, y que sirve para acceder mediante una red a otra máquina para manejarla como si estuviéramos sentados delante de ella.

Sólo sirve para acceder en modo terminal, es decir, sin gráficos.

Su mayor problema es de seguridad, debido a que todos los nombres de usuario y contraseñas necesarias para entrar en las máquinas viajan por la red como texto plano.

- SSH (*Secure SHell*)

Es muy común, en particular para administradores de red y sobre plataformas Unix/Linux, el acceso remoto mediante el protocolo SSH.

Este protocolo permite manejar por completo un equipo mediante un intérprete de comandos, de la misma forma que telnet, y además puede redirigir el tráfico de X (entorno gráfico) para poder ejecutar programas gráficos si tenemos un Servidor X corriendo.

Las diferencias con telnet residen en que no sólo pueda ejecutar aplicaciones gráficas, sino que además SSH usa técnicas de cifrado que hacen que la información que viaja por el medio de comunicación vaya de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión.

- Acceso a escritorio remoto

Hoy en día los sistemas operativos que admiten gráficos como Windows, Linux en los ambientes gráficos más conocidos KDE y GNOME, Solaris en sus ambientes gráficos CDE y Java Desktop System, incluyen protocolos que permiten a los usuarios el acceso remoto a lo que estos sistemas llaman escritorio remoto.

Los ambientes gráficos de estos sistemas, localmente dan a cada usuario un espacio de trabajo denominado escritorio que sería una herramienta de interacción entre el usuario y el equipo, generalmente sencilla y amigable.

Los sistemas traen incorporadas formas de poder acceder a estos escritorios remotamente y dar la sensación al usuario de estar en la misma máquina. El avance de estas tecnologías ha dado paso al estudio de clientes livianos, equipos que prácticamente no tienen recursos y que sólo les alcanza con una conexión de red para poder conectarse a un servidor que les provee del sistema, en particular mostrándole al usuario su escritorio remoto.

El protocolo de Windows es RDP (*Remote Desktop Protocol*), mientras que los otros sistemas como Solaris o Linux implementan XDMCP (*X Display Manager Control Protocol*) lamentablemente los protocolos RDP y XDMCP no son compatibles aunque se han hecho programas, como rdesktop para Solaris y Linux, que permiten acceder por RDP a Windows.

De todos modos existen otros protocolos como VNC o NX que permiten ser configurados en cada sistema operativo y de esta forma acceder a los escritorios de Windows a Linux o Solaris y de estos dos últimos a Windows.

- Aplicaciones WEB

En los primeros tiempos de la computación cliente-servidor se utilizaban aplicaciones desktop que requerían implementar un servidor y un cliente que debía ser instalado y configurado en cada equipo que quisiera acceder, y si había algún cambio, se debían hacer modificaciones en cada equipo además, hasta la aparición de lenguajes como JAVA, no solían ser multiplataforma.

El crecimiento de Internet trajo entre otros protocolos a HTML y el desarrollo de otras tecnologías que hicieron surgir las aplicaciones Web. Una aplicación Web es una aplicación informática que los usuarios utilizan accediendo a un servidor Web a través de Internet mediante un navegador Web. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en varios clientes es una razón de su popularidad, lo único que deberían actualizar los clientes sería el navegador Web.

Los servidores Web para aplicaciones Web traen un contenedor para alojar aplicaciones escritas en lenguajes generalmente interpretados, entre estos, JAVA o PHP, encargadas de dar las respuestas a los pedidos de los clientes y que serán visualizadas por los clientes en un navegador Web.

2.3 Acceso remoto al cluster

Las tecnologías como telnet, SSH y de escritorio remoto son muy útiles pero dan al usuario un control total(o casi total) sobre el equipo remoto, el usuario podría ver la estructura de directorios de esa máquina, servicios en ejecución y tal vez poder leer información que no sea de su interés. Estas son cosas que, a excepción de un administrador del cluster, estarían demás para usuarios que sólo quisieran acceder a un cluster para ejecutar sus programas, que a lo sumo se le podría brindar un espacio de trabajo donde guardar sus archivos para ejecutar o resultados de las ejecuciones y que sea transparente la ubicación del mismo dentro del equipo remoto.

Es por eso que desarrollar una aplicación Web, que brinde al usuario una vista transparente y un espacio de trabajo para que este sólo haga lo que tiene que hacer (simplemente obtener resultados de ejecuciones de programas) sería lo más adecuado para este caso.

Otra ventaja mas que tienen para este tipo de acceso remoto las aplicaciones Web es que al funcionar por la Web, no se exponen las arquitecturas de redes a riesgos de seguridad al abrir otros puertos; los servidores Web trabajan sobre el puerto 80, mientras que telnet lo hace sobre el puerto 23, SSH sobre el 22, RDP sobre el 389 etc.; tener abiertos esos puertos implica tener un control más en los firewalls sobre el acceso no deseado a estos servicios. Para cada usuario nuevo que se registre para utilizar el cluster se debe permitir el acceso en alguno de estos protocolos y se deben hacer distintas configuraciones sobre los firewalls.

Capítulo 3

Ejecución remota de programas

3.1 Introducción

Una aplicación escrita para utilizar y administrar el cluster debe permitir que los usuarios ejecuten sus programas y obtengan los resultados de los mismos, como también que un administrador del cluster pueda ejecutar ciertos comandos de sistema para distintas tareas de administración sobre la arquitectura de cluster.

La aplicación, además, debe permitir ser accedida remotamente vía Web. Como ya se mencionó anteriormente una aplicación Web es una forma de acceso adecuada para este caso y es en lo que consistió el desarrollo.

Por lo tanto se debió elegir un lenguaje apropiado para escribir esta aplicación tal que permita las siguientes características:

- La ejecución de programas y comandos de sistemas.
- El acceso remoto vía Web.

3.2 Elección de un lenguaje apropiado

La elección se realizó entre dos lenguajes, PHP y JAVA, que se describen a continuación:

PHP (*Hypertext Pre-processor*)

Es un lenguaje de programación muy sencillo, basado en la sintaxis de C, usado generalmente para la creación de contenido para sitios Web. Se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. Últimamente también se utiliza para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando la biblioteca GTK+.

Su interpretación y ejecución tiene lugar en el servidor, en el cual se encuentra almacenado el script, y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página Web, generada por un script PHP, el servidor ejecuta el intérprete de PHP, el cual procesa el script solicitado que generará el contenido de manera dinámica (pudiendo modificar el contenido a enviar) y regresa el resultado al servidor, el cual se encarga de regresárselo al cliente.

JAVA

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado por una máquina virtual JAVA lo que le permite ser independiente de la plataforma.

JAVA presenta además lo que se conoce como J2EE que es una plataforma de programación (parte de la Plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación JAVA con arquitectura distribuida de n niveles.

Una herramienta muy conocida dentro de la plataforma J2EE es el uso de **SERVLETS** ("programas" que se ejecutan en un servidor), el servidor puede ser un servidor Web con la particularidad de tener un contenedor de aplicaciones (Ej.: Tomcat) o bien puede ser un servidor de aplicaciones (Ej.: OC4J Oracle, Sun Java Sistema Application Server).

Otra herramienta muy usada en esta plataforma es la tecnología **JSP** (Java Server Pages).

JSP es una tecnología Java que permite a los programadores generar contenido dinámico para Web, en forma de documentos HTML, XML, o de otro tipo. Las JSP's permite al código Java y a algunas acciones predefinidas ser incluidas en el contenido estático del documento Web.

En las **JSP** se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente.

De todos modos, incluir código JAVA dentro de las JSP no es recomendable por razones de independencia entre modelo y vista.

Como ya se mencionó anteriormente, la elección de un lenguaje apropiado para desarrollar una aplicación Web que permita administrar la arquitectura de cluster esta relacionado con dos características:

- La ejecución de programas y comandos de sistemas.
- El acceso remoto vía Web.

En el caso de la primera característica, tanto PHP como JAVA tienen la capacidad para ejecutar programas o comandos de sistemas desde un programa escrito en alguno de estos lenguajes.

JAVA permite mejor y mayor control sobre la ejecución de programas y comandos de sistemas, y la combinación con threads y otras herramientas le dan una gran potencia, permitiendo obtener los resultados mas precisos sobre la ejecución y códigos de salida de los mismos.

Con respecto a la segunda característica, ambos lenguajes permiten el acceso remoto vía Web.

Además de estos dos aspectos, se consideraron algunos parámetros más que fueron evaluados para la elección del lenguaje de programación para el desarrollo de la aplicación Web.

Un parámetro fue la capacidad de estos lenguajes de utilizar Threads, dado que la aplicación esta relacionada con aplicaciones en programación concurrente/paralela los threads suelen ser útiles.

Para el caso de PHP, no se encontró información que mencionara la capacidad de este para manejar threads; por el contrario, JAVA tiene una gran capacidad para el manejo de threads, además de otro tipo de herramientas para desarrollos en memoria compartida y distribuida.

Relacionado con el manejo de threads está el uso de memoria compartida; varios threads pueden querer acceder a un mismo recurso exclusivo o compartido de forma sincronizada, y por lo tanto, pueden existir recursos en el sistema ya sea de hardware o software que necesiten de exclusión mutua o de algún tipo de sincronización para ser utilizados por uno o mas threads en ejecución

(un ejemplo de un recurso exclusivo es la arquitectura de cluster misma). Las herramientas mas utilizadas para este tipo de requerimiento son los semáforos y monitores.

Para PHP no se da una referencia clara de cómo manejar estas herramientas; pareciera que existen al menos semáforos pero se hace referencia a wrappers de semáforos de sistemas Unix/Linux, lo cual indicaría que no tiene semáforos a nivel del lenguaje.

En el caso de JAVA se tienen semáforos a nivel del lenguaje, y dos formas de manejar monitores mediante métodos sincronizados y bloqueos; esto permite excluir mutuamente clases y sincronizar threads en ejecución.

Otro parámetro considerado fue que implementar en este lenguaje permita utilizar fácilmente el patrón MVC (*Model View Controller*).

La idea de este patrón es que la funcionalidad de la aplicación (modelo) debe ser independiente de la interfaz con el usuario (vista), esto es, cualquier cambio que se produzca en la vista no debería afectar al modelo; de esta forma una misma aplicación puede tener distintas interfaces para los diferentes tipos de usuarios, pudiendo ser tanto una interfaz de una aplicación de escritorio, una página Web o un acceso por terminal de texto.

En medio, entre la vista y el modelo, se encuentra el controlador encargado de que ambos interactúen.

En el caso de PHP se ha logrado separar en varios casos el modelo de la vista que el usuario tiene del mismo, pero no siempre es sencillo; seguramente dependerá de versiones posteriores de este lenguaje que aún está en pleno desarrollo.

En el caso de JAVA, esto tiene mayor desarrollo; se ha trabajado mucho en el uso de servlets y JSP surgiendo un framework preparado para el uso adecuado del patrón MVC. Este framework se llama STRUTS.

Por último y teniendo en cuenta el patrón MVC, existe la idea que a futuro, la aplicación a desarrollar, pueda usarse en una arquitectura grid, lejos de una aplicación Web con servlets, JSP y struts, y mucho más lejos aún de PHP.

Un ejemplo es el middleware Globus toolkit que posee un contenedor de aplicaciones pero que funcionan bajo el concepto de "Servicio Web" sobre un protocolo con estado, a diferencia de una aplicación Web tradicional la cual lo hace bajo HTTP sin estado.

El soporte está dado para lenguajes como C, C++ y JAVA, pero no sobre PHP.

Desde una arquitectura de grid la aplicación debería poder funcionar de la misma forma, sin que el modelo se vea alterado por el cambio en la vista que tiene el usuario o en la forma de ser accedido, esta es una razón más para que el patrón MVC sea considerado importante.

Por estos motivos se decidió realizar un modelo escrito en lenguaje JAVA y al ser una aplicación Web, fue hecho combinándolo con servlets y JSP utilizando el framework STRUTS.

A continuación se detalla cómo se pueden ejecutar programas y comandos de sistemas a partir de una aplicación escrita en este lenguaje y cómo es el acceso vía Web a esta aplicación.

3.3 Ejecución de comandos de sistema y programas desde JAVA

El lenguaje JAVA permite ejecutar comandos de sistema u otros programas; por ejemplo, se podría listar el contenido de un directorio o abrir un editor de textos desde una aplicación JAVA, o para el caso de una arquitectura de cluster, ejecutar una aplicación escrita con la librería MPI (para lograr esto existe la clase Runtime).

3.3.1 La clase Runtime

La clase Runtime no sólo permite ejecutar comandos o programas sino también controlar la maquina virtual, pero a los propósitos de la aplicación WEB solo fue de interés la ejecución de comandos o programas.

Solo existe una instancia de la clase Runtime que se obtiene de la siguiente forma:

```
Runtime r = Runtime.getRuntime();
```

3.3.2 Ejecución de comandos o programas

El objeto obtenido en r permite ejecutar un comando o programa a través del siguiente mensaje al objeto:

```
r.exec("comando o programa a ejecutar");
```

El parámetro "comando o programa a ejecutar" puede ser cualquier comando o programa incluyendo, en la mayoría de los casos, la ruta completa. En el caso de ejecutar comandos las características del parámetro dependen del sistema operativo:

- Windows

La forma del parámetro para la ejecución de un comando es "cmd /c comando" y para la ejecución de un programa "cmd /c start programa". En la mayoría de los casos se puede evitar el uso de cmd si se conoce la ruta completa del programa.

Como Windows también asocia a las extensiones de archivos determinadas aplicaciones con los cuales pueden ser abiertos, se podría abrir un archivo Word con la línea "cmd /c start archivo.doc" o una pagina Web con la línea "cmd /c start http://www.google.com.ar".

- Unix/Linux

La forma del parámetro para la ejecución en este caso es mucho mas sencilla, solo se debe escribir el comando o programa con la ruta completa y asegurarse que se tenga permiso para la ejecución del mismo.

3.3.3 El objeto Process

El mensaje `exec` retorna un objeto `Process` que es muy útil para saber los resultados y características de la ejecución, este se puede obtener de la siguiente forma:

```
Process p = r.exec("comando a ejecutar");
```

3.3.4 Ejecución sincrónica y asincrónica

El llamado a la ejecución no es sincrónico, esto es, una vez que se ejecuta la línea `r.exec()`; el programa JAVA hace el llamado al sistema operativo delegándole la responsabilidad de la ejecución del comando o programa, luego continua la ejecución en la siguiente línea de programa sin esperar que la ejecución del comando o programa termine.

Existe una forma de hacer que el programa JAVA espere luego de enviar el pedido de ejecución, utilizando el objeto `Process` retornado de la siguiente forma:

```
p.waitFor();
```

El caso en que el programa espera la ejecución del comando o programa lo denominamos sincrónico y el caso en que no se produce esta espera lo llamo asincrónico.

Es útil en estos tipos de ejecución obtener el código de salida así como la salida misma de la ejecución, más aún, si se da el caso de comandos que retornan líneas por consola como salida.

3.3.5 Código de salida

La mayoría de los comandos o programas retornan un valor como código de salida (generalmente se suele retornar un 0 si la ejecución fue exitosa o un valor distinto si la ejecución termino con algún error) en particular el número retornado será identificado con algún tipo de error.

Para capturar el código de salida que retorna la ejecución de un comando o programa desde JAVA se hace con el objeto `Process` de las siguientes formas:

```
Int codigoDeSalida = p.waitFor();
```

O bien

```
p.waitFor();  
int codigoDeSalida = p.exitValue();
```

3.3.6 Obteniendo la salida de la ejecución

La salida de la ejecución puede ser de dos tipos, dependiendo de si fue con éxito o con errores.

Para obtener cualquiera de ellas, al objeto Process se le pueden enviar cualquiera de los dos mensajes:

- `getErrorStream()`

Retorna un Stream donde se describe la salida que retornó la ejecución del comando o programa si es que terminó de forma errónea.

- `getInputStream()`

Retorna un Stream donde se describe la salida que retornó la ejecución del comando o programa si es que terminó de forma exitosa.

Luego estos Streams son procesados para obtener las líneas de salida devueltas por el comando o programa ejecutado y enviarlas en este caso al navegador WEB del usuario.

El tipo de stream que se debe leer depende del código de salida, en caso de ser 0 se supone que la ejecución del comando o programa terminó con éxito y se leería el `inputStream`, en caso de terminar con otro código de salida se supone que se finalizó erróneamente y por lo tanto se leería el `errorStream`.

Los streams son generados y utilizados luego de invocar a `exec()`; independientemente de que el programa JAVA se quede en espera haciendo `waitFor()`; las líneas de la salida de la ejecución van escribiéndose una a una en el stream que corresponda aunque la ejecución del comando o programa no haya terminado su ejecución, a medida que la ejecución del comando o programa retorne una línea, ésta se escribirá en el stream.

Si consideramos el caso de esperar usando un `waitFor()`, se puede pensar en esperar cuál es el código de salida de la ejecución y dependiendo de este leer el stream determinado. Otra forma es mandar a leer ambos streams independientemente del código de salida, y así, cuando el programa JAVA continúe después del `waitFor()` ya tendrá las líneas procesadas.

Para esto se pueden utilizar dos Threads, uno para procesar el stream por error y otro para procesar el stream por una salida exitosa, y así hacer una ejecución simultánea que bien se puede aprovechar con las nuevas tecnologías en el mercado de procesadores duales.

A continuación se muestra un ejemplo completo de cómo hacer este tipo de procesamiento el cual puede tener lugar dentro de cualquier método implementado en alguna clase JAVA.

Ejemplo:

```
public class PruebaEjecucion{

    public resultado ejecutar(String sentencia){

        //Obtener la instancia del objeto Runtime
        Runtime runtime = Runtime.getRuntime();

        //Ejecutar la sentencia
        Process proc = runtime.exec(sentencia);

        //Obtiene el stream por error
        InputStream streamPorError = proc.getErrorStream();

        //Crea el Thread para procesar la salida por error
        ThreadPorError threadPorError = new ThreadPorError(streamPorError);

        //Obtiene el stream por éxito
        InputStream streamPorExito = proc.getInputStream();

        //Crea el Thread para procesar la salida por éxito
        ThreadPorExito threadPorExito = new ThreadPorExito(streamPorExito);

        //Corre ambos threads
        threadPorError.start();
        threadPorExito.start();

        //Luego espera el final de la ejecución y obtiene el código de salida
        int codigoDeSalida = proc.waitFor();

        //Dependiendo del código de error lee la salida correspondiente ya
        procesada
        if(codigoDeError == 0){
            return threadPorExito.salida();
        }else{
            return threadPorError.salida();
        }
    }
}
```

Los streams, ya sea por error o por éxito, son de la misma clase `InputStream`; además los `Threads` pueden ser exactamente los mismos, ambos realizan la misma tarea de procesamiento, la única diferencia es que uno es por error y el otro por una salida exitosa. De todos modos queda a criterio del programador y de las características del sistema definir una misma clase o dos clases diferentes.

El constructor de los `Threads` requiere que se le pase como parámetro el stream a procesar, que luego será usado en el correspondiente método `run()`.

El procesamiento de un stream de este tipo, dentro del `Thread` se haría de una forma similar a la siguiente:

```
InputStreamReader isr = new InputStreamReader(streamDeSalida);
BufferedReader br = new BufferedReader(isr);
String linea=null;
while ( (linea = br.readLine()) != null){
    //procesar línea
}
```

Cada `Thread` implementara algo similar a esto en el método `run()`, el parámetro `streamDeSalida` hace referencia al `streamError` o al `streamExito` pasado como parámetro en el constructor del `Thread`.

3.3.7 Algunos riesgos de utilizar la clase `Runtime`

- Portabilidad

Como es conocido todo código `JAVA` tiene la particularidad de ser portable, por lo cual debería evitarse utilizar en el parámetro del método `exec()`; características específicas de un sistema operativo concreto para mantener dicha portabilidad.

No constituye un riesgo importante pero vale la pena remarcarlo.

- Seguridad en la ejecución

Invocar un programa desde otro programa `java`, puede constituir un riesgo si no se sabe que comando o programa se va a ejecutar; debería asegurarse que el programa `JAVA` al menos corra bajo ciertas restricciones y sin permisos para ejecutar comandos de sistemas que sean críticos.

3.4 JAVA Threads y memoria compartida utilizando monitores

3.4.1 Threads

Anteriormente se mencionó el uso de threads. Un thread es un conjunto de sentencias que se puede ejecutar simultáneamente con otros threads, y cada uno tiene su espacio de trabajo.

JAVA permite definir threads de dos formas:

- Creando una clase subclase de la clase Thread
- Creando una clase que implemente la interface Runnable, en caso de que la clase a crear ya tenga una superclase definida; esto es debido a que JAVA tiene herencia simple.

Cualquiera sea la forma en que se implemente el modelo es el mismo, ambas deben incluir un método run() que implementa el comportamiento del Thread.

Para hacer correr el Thread no se debe invocar al método run() sino que se llama a un método start() que se encarga de crear el contexto para que el thread se ejecute y que además llama al método run().

3.4.1.1 Como subclase de la clase Thread

La definición de un thread como subclase de la Thread es la siguiente:

```
class MiThread extends Thread {  
  
    //Definición de variables y métodos  
  
    public void run() {  
        //Implementación del comportamiento del Thread  
    }  
}
```

Para instanciar el thread y ejecutarlo se hace:

```
//Lo instancia  
Thread miInstanciaDeThread = new MiThread();  
//Lo ejecuta  
miInstanciaDeThread.start();
```

3.4.1.2 Implementando la interface Runnable

La definición de un thread implementando la interface Runnable es la siguiente:

```
class MiThread implements Runnable {  
  
    //Definición de variables y métodos  
  
    public void run() {  
        //Implementación del comportamiento del Thread  
    }  
}
```

Para instanciar el thread y ejecutarlo se hace:

```
//Lo instancia  
Runnable miInstanciaDeThread = new MiThread();  
//Lo ejecuta  
new Thread(miInstanciaDeThread).start();
```

3.4.2 Memoria compartida utilizando monitores

Pueden existir recursos exclusivos que son accedidos por uno o varios usuarios, programas o threads de forma simultánea y que requieren exclusión mutua o algún tipo de sincronización para quienes deseen usarlos. Tal es el caso del cluster mismo, donde en la mayoría de los casos solo se debe estar ejecutando un programa a la vez.

Dentro del lenguaje de programación se puede definir alguna estructura que represente al cluster y que permita controlar el acceso exclusivo.

El hecho de tener un elemento de software que necesite de exclusión mutua o sincronización para el acceso de uno o varios threads es lo que se conoce como memoria compartida, y las herramientas mas conocidas para trabajar en esto son los semáforos y monitores.

JAVA provee semáforos y dos formas de trabajar con monitores, estos últimos fueron los utilizados dentro de la aplicación Web y a continuación son detallados.

3.4.2.1 Monitores con métodos synchronized

Cuando dos o más threads necesitan acceder de manera simultánea a un recurso de datos compartido necesitan asegurarse de que sólo uno de ellos accede a este a la vez. JAVA permite definir a una clase de manera que los métodos implementados en ella sean accedidos como en un monitor.

Sólo uno de los threads puede ser el propietario de un monitor en un instante dado. Los restantes threads que estuviesen intentando acceder al monitor bloqueado quedan en suspenso hasta que el thread propietario del objeto monitor en ese momento salga del mismo.

La manera de acceder a un objeto monitor es llamando a un método marcado con la palabra clave synchronized. Durante todo el tiempo en que un thread permanezca en un método sincronizado, los demás threads que intenten llamar a un método sincronizado sobre la misma instancia tendrán que esperar. Una vez que termina la ejecución del método sincronizado el objeto monitor se libera dejando ingresar a otro thread.

La forma de implementar un objeto monitor es la siguiente:

```
class ObjetoMonitor {  
  
    synchronized void metodo1() {  
        //Implementación  
    }  
  
    synchronized void metodo2() {  
        //Implementación  
    }  
}
```

De esta forma se permite una exclusión mutua sobre una clase que fácilmente puede representar por ejemplo un cluster al que se quiere controlar el acceso.

El uso de métodos synchronized es muy útil para realizar exclusión mutua, pero no lo es tanto si se quisiera hacer sincronización como es el caso de los monitores en su definición tradicional utilizando variables de condición; más de una sincronización entre threads con métodos sincronizados se torna muy complejo.

3.4.2.2 Monitores con locks y variables condición

Esta implementación de monitores es más útil si se quieren realizar sincronizaciones dentro de monitores como lo es en la definición tradicional de un monitor, utilizando variables de condición.

De la misma forma que en el caso anterior, define a una clase de manera que los métodos implementados en ella sean accedidos como en un monitor.

El mecanismo de exclusión mutua ahora no se logra utilizando los métodos sincronizados sino usando lo que se conoce como locks.

Esto es introducido en JAVA a partir de la versión 1.5 del JDK, por lo tanto para versiones anteriores lo único que se puede utilizar es la exclusión mutua y sincronización por medio de métodos sincronizados.

En una clase que represente a un objeto monitor se debe definir como variable de instancia un lock que permitirá realizar la exclusión mutua en cada método implementado en la clase. La consecuencia de esto es que, a diferencia de los métodos sincronizados y de la definición de monitores tradicionales, la exclusión mutua en este caso es explícita, es decir, se utiliza la variable lock para bloquear o desbloquear cada método y por lo tanto hay que tener mucho cuidado en el uso de los métodos asociados a esta variable.

Según la definición de los monitores tradicionales, dentro del monitor pueden existir distintos tipos de sincronizaciones logradas mediante lo que se conoce como variables de condición que permiten dormir o despertar procesos dentro del monitor, las operaciones más conocidas sobre ellas son:

Wait(C): duerme un proceso dentro del monitor en una cola de espera asociada a la variable condición C y libera el monitor para que otros procesos ingresen.

Signal(condition): despierta un proceso dormido en la cola de espera asociada a la variable condición C para que continúe su ejecución.

SignalAll(condition): despierta todos los procesos dormidos en la cola de espera asociada a la variable condición C para que continúen su ejecución, solo uno seguirá.

La sincronización mediante variables de condición es muy útil y permite sincronizar uno o varios threads de distintas formas.

Con JAVA se puede lograr tener variables de condición para sincronizar threads, cada variable de condición esta asociada a un lock, en este caso al lock que se define para la exclusión mutua y permite realizar operaciones como las definidas para los monitores tradicionales.

La forma de implementar un objeto monitor usando locks es la siguiente:

```
class ObjetoMonitor{

    //Define la variable lock para exclusión mutua
    private final ReentrantLock lock = new ReentrantLock();

    //Define variables de condición asociadas a la variable lock
    final Condition variableCondicion1 = lock.newCondition();
    final Condition variableCondicion2 = lock.newCondition();

    //Definición de métodos
    public void metodo1(){
        lock.lock();
        try{
            //Implementación del método
        } finally {lock.unlock();}
    }

    public void metodo2){
        lock.lock();
        try{
            //Implementación del método
        } finally {lock.unlock();}
    }
}
```

Notar que la definición de la variable lock hace referencia a una clase ReentrantLock. Esto se debe a que un thread que esté ejecutando dentro del objeto monitor puede querer llamar a otro método dentro del mismo, y al hacer la invocación tendría que salir del monitor y competir por el ingreso con otros threads; definir la variable como ReentrantLock impide esta situación y permite que el thread no pierda el control del monitor.

En cada método la variable lock se utiliza para bloquear y liberar el acceso al método, como se dijo anteriormente, la exclusión mutua es explícita.

Las variables de condición definidas están asociadas al lock y permiten ser utilizadas para sincronizar. Estas variables no son variables propiamente dichas sino que son objetos a los que se les puede enviar mensajes que son semejantes a las operaciones de los monitores tradicionales, por ejemplo:

- objetoVariableCondicion.await(): semejante al wait tradicional.
- objetoVariableCondicion.signall(): semejante al signal tradicional.
- objetoVariableCondicion.signalAll(): semejante al signalAll tradicional.

Las variantes entre variables locks y variables de condición son muchas y permiten por ejemplo una exclusión mutua parcial, donde en un objeto se puede acceder a dos grupos de métodos en forma simultánea pero no a dos o más métodos en un mismo grupo.

También se permiten dormir procesos por un determinado tiempo o despertarlos en un determinado orden.

La esta estructura es muy flexible pero hay que tener muchas precauciones.

3.4.2.3 Comentarios sobre el uso de monitores en JAVA

Las dos implementaciones de monitores son válidas y pueden coexistir en una misma implementación de un programa JAVA, solo basta con reconocer cual de ellas es más adecuada que la otra dependiendo del uso que se le quiera dar.

Si sólo se necesita un objeto que requiere de exclusión mutua sería más adecuado el uso de métodos sincronizados que no necesitan definiciones de variables adicionales; si en cambio se necesita una fuerte sincronización entre procesos dentro del objeto monitor, sería más adecuada una implementación utilizando la variable lock y variables de condición.

También es importante tener en cuenta que el objeto monitor debiera ser único en el sistema, no se deberían tener varias instancias, sino no tendría sentido la exclusión mutua; para lograr esto se utiliza un patrón denominado Singleton.

De todos modos existen implementaciones que requieren varias instancias de un mismo monitor por lo que hay que tener mucho cuidado en el uso de las mismas.

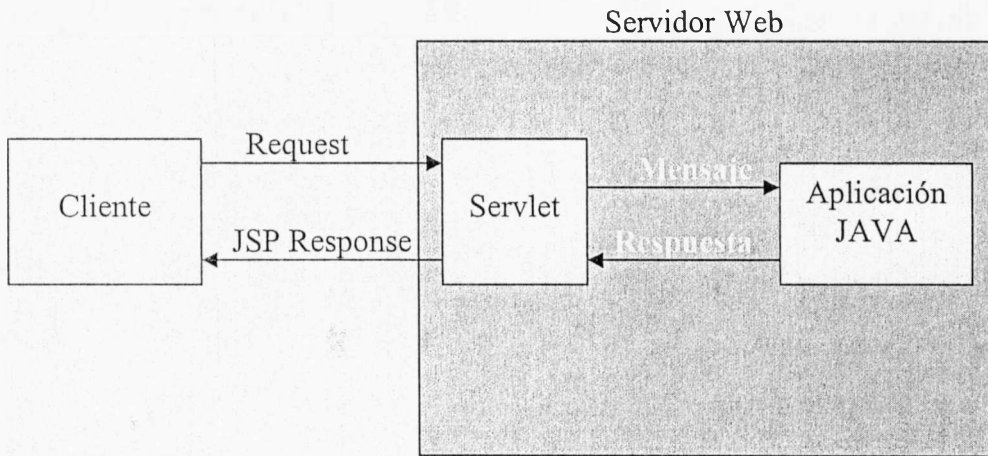
3.5 Acceso remoto vía Web a una aplicación JAVA

Una aplicación JAVA que pretenda ser accedida remotamente vía Web debe correr dentro de un servidor Web con contenedor de aplicaciones (Tomcat), o sobre un servidor de aplicaciones (Sun Java System application server, Oracle oc4j). En este caso se eligió utilizar Tomcat que resulta más sencillo de configurar.

Además del servidor Web hay toda una estructura de programación necesaria para que una aplicación JAVA pueda ser accedida vía Web, y está relacionada con los servlets y las JSP.

La forma adecuada de leer esta estructura es ver al servlet como un programa JAVA que se ejecuta en un servidor; el servlet es invocado por el usuario desde un navegador Web, este delega la invocación a la aplicación JAVA la cual le retorna determinado resultado, el servlet entonces retorna una pagina JSP que procesa el resultado para mostrarlo y es enviada nuevamente al usuario.

JSP es una tecnología Java que permite a los programadores generar contenido dinámico para Web, en forma de documentos HTML, XML, o de otro tipo.



Esta es una forma adecuada de trabajar conservando el patrón MVC, donde el modelo es la aplicación JAVA, la vista es la JSP y el controlador en este caso el servlet

De esta forma se crea un servlet por cada requerimiento de usuario.

Esta estructura requiere de muchas tareas de desarrollo repetitivas y que llevan mucho tiempo; para estructurar aún más esto y hacerlo algo más sencillo existe un framework llamado Struts.

Struts brinda una forma adecuada de trabajar bajo el patrón MVC con JAVA, servlets y JSPs.

En vez de crear muchos servlets, Struts crea uno solo y varias acciones; el servlet recibe el requerimiento de usuario y lo delega a la acción correspondiente.

Agrega además un conjunto de tags para trabajar dentro de las JSP sin necesidad de escribir código JAVA para procesar información.

3.6 Aplicación al cluster

Un usuario que quiera ejecutar un programa escrito en C usando la librería MPI deberá seguir los siguientes pasos:

- Conectarse remotamente vía Web a la aplicación Web escrita en JAVA
- Enviar su programa
- Pedir la ejecución del programa a la aplicación Web

Gracias a la capacidad de JAVA de ejecutar desde una aplicación escrita en este lenguaje otras aplicaciones o comandos, el usuario recibirá la salida de la ejecución de su programa C/MPI.

Esta capacidad no solo le permite a un usuario ejecutar programas sino también poder compilar códigos fuentes; de esta forma el usuario sólo debe escribir el programa y puede ser compilado en el cluster mismo si es que el usuario no tiene la posibilidad de poder compilarlo fuera de el.

Pensando en lo académico, los alumnos muchas veces no tienen o les es difícil configurar herramientas para compilar y correr en un cluster (no suele ser fácil configurar un conjunto de librerías MPI), por este motivo el único momento que tienen para trabajar es viniendo a la facultad en el horario que disponen de las maquinas.

Con la aplicación Web tienen la posibilidad de utilizar el cluster remotamente y dedicarle mas tiempo a aquello que las cátedras esperan de ellos y no perder el tiempo en “pelearse” con las configuraciones o esperando una semana para usar el cluster.

Capítulo 4

Aplicación Web desarrollada

4.1 Introducción

La aplicación Web fue desarrollada en lenguaje JAVA utilizando el framework Struts ejecutándose sobre un servidor Tomcat.

Además de estas herramientas fue importante incluir el uso de AJAX, una tecnología que permite mayor interacción y transparencia en el uso de la aplicación Web lo que la convierte en lo que se denomina una aplicación RIA (*Rich Internet Application*).

Se tuvieron en cuenta los tipos de usuarios tanto a nivel de aplicación como a nivel de sistema, y de esta forma se determinaron los distintos perfiles y la forma de interacción de la aplicación con el sistema para dar cierto grado de seguridad.

Se realizó un análisis de los distintos tipos de accesos al cluster de acuerdo a la necesidad de los diferentes tipos de usuarios que lo utilizan ya sean alumnos, docentes o profesores.

La aplicación Web presenta distintas áreas, y cada una de ellas estará disponible o no de acuerdo el perfil de los usuarios.

4.2 Aplicaciones RIA, uso de AJAX

RIA (*Rich Internet Application*) es un nuevo tipo de aplicación Web con más ventajas que las tradicionales aplicaciones Web.

Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces, a recargar la misma página con un mínimo cambio.

En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos, recargando solo los datos necesarios.

Hay muchas herramientas para la creación de entornos RIA. Entre estas podemos mencionar las plataformas Flash y Flex de Macromedia, AJAX y Open Laszlo (herramientas Open Source).

AJAX fue la técnica utilizada en esta aplicación.

AJAX (*Asynchronous JavaScript And XML*) es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente (en el navegador del usuario) y mantiene comunicación asincrónica con el servidor en segundo plano.

De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tecnologías ya existentes:

- XHTML (HTML) y hojas de estilo en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones JavaScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor Web. En algunos frameworks y en determinadas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado y texto plano.

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

El uso de esta tecnología permitió mayor interacción en la comunicación entre la aplicación Web y el usuario. Un ejemplo de las ventajas se observa por ejemplo cuando un usuario envía a ejecutar un programa al cluster esperando el resultado; sin AJAX la página debía ser recargada (esta carga incluye componentes de la página que no es necesario recargar como por ejemplo menús, inclusive la página podría quedarse en blanco si es que el programa no termina su ejecución).

Con AJAX la página no se recarga, solo se envía un requerimiento al servidor para que ejecute el programa, y una vez ejecutado retorna los resultados que son mostrados en la

misma pagina; si el programa se llegara a demorar o trabar, el usuario seguirá teniendo la pagina cargada pudiendo realizar otras operaciones, entre ellas pedirle a la aplicación que termine una ejecución “colgada”.

Otra ventaja del uso de esta tecnología dentro de la aplicación Web se ve en el área de trabajo de cada usuario: en un navegador de archivos tradicional, sin AJAX, habría que recargar la pagina completa por cada cambio en la estructura de archivos, cada vez que se cree un archivo o directorio o cada vez que se cambie de directorio.

4.3 Manejo de usuarios

El manejo de usuarios se plantea en dos niveles:

- Nivel de aplicación

El nivel de aplicación hace referencia a qué tipos de usuarios va a tener la aplicación Web. De acuerdo a los tipos de usuarios aparecen distintos perfiles, y cada uno podrá o no acceder a las distintas áreas del sistema dependiendo de los permisos de ese perfil sobre un determinado tipo de área.

- Nivel de sistema

Dado que la aplicación Web va a ejecutar programas sobre un sistema operativo además de leer y escribir archivos sobre algunas áreas de la estructura de archivos de este, se necesita tener uno o varios usuarios de sistema que puedan tener permisos para tales operaciones. Es por eso que debe estudiarse una correspondencia entre los usuarios a nivel aplicación y a nivel sistema, de manera que cada operación realizada por un usuario desde la aplicación que requiera una ejecución de programa, escritura o lectura sobre el sistema de archivos, pueda realizarse en el sistema operativo con un usuario a nivel sistema que tenga permisos para hacerlo.

4.3.1 Usuarios a nivel de aplicación

Dentro de la aplicación Web existen dos perfiles de usuarios, y para cada uno de ellos la aplicación Web habilita o deshabilita distintas áreas. Los perfiles son los siguientes:

- Usuarios comunes (perfil usuario)

La aplicación Web les permite tener un espacio de trabajo, compilar, ejecutar y detener programas sobre el cluster.

- Administradores (perfil administrador)

La aplicación Web les permite las mismas características que los usuarios comunes y agrega la posibilidad de administrar usuarios, ejecuciones de programas, clusters y arrancar o detener ambientes para ejecución basados en MPI.

4.3.2 Usuarios a nivel de sistema

Se consideraron dos alternativas para hacer la correspondencia entre usuarios de la aplicación y usuarios de sistema:

- Crear un usuario de sistemas para cada usuario de aplicación

Esto permite un mayor grado de seguridad impidiendo que un usuario no pueda modificar la estructura de archivos de otro al no tener permisos a nivel de sistema operativo, pero hace difícil la tarea de agregar un usuario nuevo a la aplicación.

Cada vez que se agregue un usuario a la aplicación, se debe agregar en el sistema, se debe configurar la estructura de archivos, variables de entorno y servicios como SSH/RSH para ese usuario; esto demanda tiempo, algo que se hace muy complicado en caso de tener que incluir usuarios en forma masiva.

Otro aspecto es que cada máquina del cluster debe conocer ese usuario, no solo debe tener el mismo nombre de usuario sino el mismo grupo e identificador (UID) en cada equipo para que sea reconocido por cada uno de estos como el mismo usuario.

La solución a esto último se puede obtener creando el usuario en cada máquina con las mismas características o bien utilizando un sistema para autenticación de usuarios centralizada como puede ser NIS, NIS+ o LDAP.

Otro aspecto importante es que las aplicaciones que ejecutan sobre el servidor Web corren bajo un único usuario de sistema, por lo tanto cada vez que un usuario de aplicación quiera hacer algo a nivel sistema deberá hacerse un cambio de usuario (desde el usuario del servidor Web al usuario de sistema que representa al usuario de aplicación). Estos cambios de usuario requieren de autenticaciones, para pasar de un usuario a otro es necesario ingresar una contraseña y esto es algo que no es deseable porque funciona internamente y el usuario no tiene forma de ingresarla y el sistema tampoco; para evitar esta situación se debe configurar el comando SUDO para que evite pedir contraseña al hacer un cambio de usuario, y esto es algo que debe hacerse para cada usuario nuevo que se agregue.

- Crear un solo usuario de sistemas para todos los usuarios de aplicación

Esto trae un problema de seguridad donde un usuario podría llegar a modificar la estructura de archivos de otro al tener los mismos permisos a nivel de sistema operativo. De todos modos, por la forma en que la aplicación Web fue desarrollada esto solo se daría de forma mal intencionada, no puede darse casualmente, porque los usuarios no pueden ver la estructura de otros usuarios.

Se hace más sencilla la tarea de agregar un usuario nuevo; solo se agrega a la aplicación y se configura el espacio de trabajo que inclusive puede ser automático, sin mayores complicaciones y sin consumir tiempo.

El servidor Web debe estar configurado bajo este usuario de sistema, de esta forma se evitan los cambios de usuarios entre el usuario del servidor

Web y el usuario de sistema, y por lo tanto no se necesitan hacer configuraciones adicionales.

Para el caso de la aplicación Web se utilizó la segunda alternativa, por las siguientes razones:

- Altas masivas de usuarios:

El uso académico de la aplicación Web hace probable que se den altas de muchos usuarios por años para cada cátedra que quiera utilizarlo; por cada usuario hay que hacer una importante cantidad de configuraciones si se considera la primer alternativa.

- Estadísticas de ejecuciones:

Las ejecuciones están controladas, se sabe qué usuario ejecutó un programa y cuando fue ejecutado; dado que cada usuario es bien conocido se pueden controlar las “malas intenciones”.

- Transparencia e independencia del espacio de trabajo

El espacio de trabajo a los usuarios les es transparente, y además es independiente del sistema de archivos del sistema operativo. De esta forma el usuario no sabe dónde están sus archivos como tampoco los de los demás usuarios.

Que un usuario intente obtener información extra sobre esto estaría hablando de mala intención y no de algo casual.

A los aspectos sobre la seguridad se le dedicara posteriormente un párrafo.

4.4 Acceso compartido y exclusivo

Un cluster se debe ver como un recurso exclusivo, es decir, solo debería haber un proceso ejecutándose. Esto es porque los programas que se utilizan en el cluster están orientados a ganar en tiempo teniendo a su disposición la mayor cantidad de recursos para lograrlo.

Si se esta buscando hacer un estudio sobre performance en un determinado programa no tiene sentido que haya otro programa corriendo en el cluster porque le estaría quitando recursos que este necesita y por lo tanto el estudio no sería válido.

La mayoría de las veces lo que se hace es asignar un determinado tiempo a un usuario para que ejecute sus programas en el cluster, y mientras este usuario haga uso de la arquitectura ningún otro usuario tendrá acceso. Es posible que haya algún tiempo en que el cluster este ocioso y aunque el usuario tenga la exclusividad, podría dejar utilizar el cluster a otro usuario pero todo dependerá del tipo de programa y el tiempo en que demore la ejecución de este último.

Para el caso de la aplicación Web se consideró el tipo de usuario que va a ejecutar sobre la arquitectura. Se tienen en cuenta:

- Investigadores y profesores

Este tipo de usuarios generalmente necesita el cluster en forma exclusiva por un periodo de tiempo importante, los programas que se ejecutan están orientados a obtener mediciones y resultados en tiempos superiores a una hora.

Estos usuarios además toman mediciones comparando distintos algoritmos para ver cual de todos ellos les da mejores resultados lo que hace pensar que necesitan mucho tiempo para correr uno y otro algoritmo.

- Alumnos

Por el momento solo se tienen en cuenta alumnos de dos cátedras:

- Programación paralela: donde se hacen mediciones semejantes a las hechas por investigadores y profesores pero no necesitan de demasiado tiempo, solo les alcanzaría con algunos minutos.
- Programación concurrente: en general la idea no es hacer mediciones, y sólo se utiliza el cluster para mostrar resultados de programas utilizando memoria distribuida y comparar con algoritmos que representan la solución al mismo problema pero utilizando algoritmos secuenciales y de memoria compartida, además de darles a los alumnos una introducción a la noción de cluster.

Los programas que se corren en esta cátedra suelen ser programas simples que no superan los cinco minutos.

Debe tenerse en cuenta que en este caso el que sean usuarios investigadores, profesores o alumnos de cualquiera de las cátedras mencionadas, no está relacionado con el perfil de usuario, ya que todos tendrán el mismo perfil de usuarios comunes.

De acuerdo a los tipos de usuarios mencionados anteriormente, la aplicación Web, permite establecer dos modos de acceso a la arquitectura de cluster:

- Modo exclusivo

Orientado a investigadores, profesores y, tal vez, a alumnos de la cátedra de Programación paralela.

La idea de este modo es asignar a un usuario una franja horaria para que pueda trabajar impidiendo que otros usuarios puedan ingresar al sistema mientras se este haciendo uso de la arquitectura. No se les permitirá ningún tipo de acceso a otros usuarios ni siquiera cuando el cluster este ocioso, no les permitirá autenticarse, salvo administradores en caso que se deba solucionar algún problema para que el usuario que tiene asignado el uso exclusivo del cluster pueda usarlo sin inconvenientes.

- Modo compartido

A pesar del nombre, el modo compartido, **NO** significa que se permita el uso del cluster a dos procesos de distintos usuarios al mismo tiempo, sino que se sigue manteniendo la exclusividad en la ejecución sobre la arquitectura de cluster.

Este modo está mucho más orientado a usuarios de las cátedras de Programación concurrente y Programación paralela. Permite autenticar usuarios dejándolos trabajar sobre su espacio de trabajo pero sólo les deja ejecutar programas si el cluster no esta corriendo programas de otro usuario; de esta forma se aprovechan los tiempos ociosos de la arquitectura.

En caso que un usuario desee ejecutar un programa sobre el cluster y el mismo esté ocupado, la aplicación reintentará la ejecución hasta que logre el acceso para ejecutar o bien hasta que el usuario cancele el pedido de ejecución.

4.5 áreas de la aplicación

La aplicación Web tiene distintas áreas dependiendo del perfil de usuario:

Para usuarios comunes:

- Area de ejecución: permite a los usuarios utilizar su espacio de trabajo, compilar y ejecutar programas.
- Area de programas en ejecución: permite ver que programas propios se están ejecutando sobre el cluster con la posibilidad de detenerlos.

Para administradores:

- Area de ejecución: igual al área en usuarios comunes.
- Area de manejos de usuarios: permite administrar los usuarios del sistema.
- Area de manejo de clusters: permite administrar los cluster y sus equipos.
- Area de programas en ejecución: igual al área en usuarios comunes pero con la diferencia que permite administrar ejecuciones de otros usuarios.
- Area de ambientes de ejecución: permite arrancar o detener los ambientes de ejecución MPI.
- Area de asignación de accesos exclusivos.
- Area de registros de ejecución: permite ver las ejecuciones que se realizaron en el cluster.

Cada una de estas áreas será explicada en detalle a continuación.

4.5.1 Area de ejecución

El área de ejecución le da a cada usuario un espacio de trabajo en el servidor y le permite compilar y ejecutar programas así como también obtener los resultados de estas ejecuciones.

4.5.1.1 Espacio de trabajo

El espacio de trabajo no es más que una estructura de directorios y archivos que el usuario puede manejar. Esta estructura permite:

- Crear directorios.
- Subir archivos.
- Eliminar archivos o directorios.
- Hacer movimientos de archivos o directorios dentro de la estructura de archivos.
- Descomprimir archivos .zip
- Visualizar archivos (sólo aquellos con extensión .c o .txt).
- Editar archivos (sólo aquellos con extensión .c .txt).

Esta estructura de archivos es independiente de la estructura de archivos del servidor, por lo tanto es transparente al usuario.

A partir de esta estructura el usuario tendrá disponible sus archivos tanto para editarlos, compilarlos y ejecutarlos, y además podrá visualizar resultados de ejecuciones que fueron dejadas en archivos de salida.

El hecho de descomprimir archivos comprimidos en formato zip es una ventaja cuando se quieren hacer “uploads” de varios archivos; la aplicación Web solo permite hacerlo de un archivo a la vez lo que no es muy practico si se tienen varios archivos, y por lo tanto al comprimir los archivos en un solo archivo zip, se podrá hacer un “upload” de este y descomprimirlo en el espacio de trabajo dejando todos los archivos a disposición.

4.5.1.2 Compilación de código fuente

El sistema da la posibilidad al usuario de poder compilar sus archivos fuentes, en este caso implementados en lenguaje C usando las librerías de MPI. Esto es algo muy útil para aquellos que no tengan una implementación de MPI instalada.

El archivo a compilar debe ser un archivo con extensión .c y la salida de la compilación será un archivo ejecutable con una forma particular:

usuario_programa_ambiente.out

usuario: el nombre del usuario logueado.

programa: el nombre del programa compilado.

ambiente: el nombre del ambiente en que fue compilado, puede ser lammpi o mpich2mpi.

Si el usuario **MiUsuario** compila un programa llamado **programa.c** utilizando la implementación de la librería **MPI Lam**, el archivo de salida será:

MiUsuario_programa.c_lammpi.out

este archivo es un programa ejecutable que puede ser ejecutado solamente en Lam mpi.

4.5.1.3 Ejecución de programas (modo sincrónico, asincrónico e interactivo)

Los programas a ser ejecutados en el cluster utilizan la librería MPI en dos implementaciones LAM y MPICH2 (previamente fueron compilados con alguna de estas implementaciones utilizando el lenguaje de programación C).

La ejecución permite tres modos:

- Sincrónico

Se envía el pedido de ejecución al servidor y se espera una respuesta en lo inmediato, por lo tanto la salida de la ejecución es retornada en la misma página WEB.

- Asincrónico:

Se envía el pedido de ejecución al servidor y no se espera una respuesta en lo inmediato; es para aplicaciones que tienen una demora importante, por lo tanto la salida de la ejecución es retornada en un archivo dentro del espacio de trabajo del usuario, y el contenido de este archivo podrá visualizarse posteriormente.

- Interactivo:

Da al usuario la misma sensación que tiene cuando ejecuta una aplicación de este tipo en una consola, donde la salida de la ejecución se va mostrando línea a línea. El usuario envía a ejecutar el programa y el servidor va entregando de a línea o conjunto de líneas a medida que se desarrolla la ejecución, las cuales va mostrando en la pagina Web.

El modo sincrónico e interactivo son similares; la diferencia es que el modo sincrónico envía el pedido de ejecución y espera toda la salida de programa completa, y luego esta salida es mostrada en la pagina Web. Si el programa fallara por algún motivo el usuario no sabría qué es lo que esta pasando o en que momento el programa se detuvo, dejándole la pantalla de salida en blanco. En el caso del modo interactivo, por cada línea o conjunto de líneas que retorne el programa que se este ejecutando en el cluster, se irán mostrando una por una en la pagina Web, de esta forma si el programa se llegara a detener o fallar inesperadamente le dará una idea al usuario donde podría llegar a estar el error ya que la pantalla de salida tendría al menos algunas líneas.

Las ejecuciones sobre una arquitectura de cluster, generalmente, suelen parecerse mas al modo de ejecución asincrónico donde se envía un programa a ejecutar al cluster y se espera una salida en un archivo con una demora importante de tiempo (horas, unos pocos días), los tipos de programas que se corren en clusters suelen tener estos requerimientos.

Se supone que el programa fue anteriormente probado, asegurando la ausencia de errores, y luego ejecutado en el cluster, por lo tanto “no debería fallar”.

El modo sincrónico e interactivo de la aplicación surge por el uso práctico que se le da en algunas cátedras de la carrera, donde los programas no suelen tener demoras importantes, son más bien simples y el uso del cluster es a modo educativo, con fines académicos.

Como se dijo anteriormente, la aplicación tiene dos modos de acceso, el modo exclusivo y el modo compartido, en este último puede negarse la ejecución de un programa si es que la arquitectura de cluster está ocupada, y en este caso, la aplicación reintentará ejecutar hasta que se encuentre libre el cluster aunque también la aplicación le da al usuario la posibilidad de cancelar el pedido de ejecución.

Mientras haya un pedido de ejecución activo no se podrá hacer ningún otro pedido de ejecución al cluster, por lo tanto el usuario deberá esperar a que se produzca la ejecución del pedido pendiente o cancelar dicho pedido para poder realizar otro.

4.5.2 Area de programas en ejecución

El área de procesos en ejecución muestra al usuario que programas enviados se están ejecutando.

Para el caso de usuarios con un perfil de usuario común sólo muestra los programas en ejecución propios, mientras que para el caso de usuarios con perfil de administrador muestra no sólo los programas en ejecución propios sino también los de todos los usuarios del sistema.

Se sabe que un programa que se envía a ejecutar puede “colgarse”; ante esta situación, dentro de esta área de la aplicación Web se permite detener la ejecución.

Un administrador puede detener no solo sus propios programas en ejecución sino también programas de otros usuarios descuidados que no los hayan detenido.

4.5.3 Area de cambio de contraseña

Todo usuario, ya sea administrador o no, tiene la posibilidad desde este área de cambiar la contraseña para ingresar al sistema.

4.5.4 Area de manejos de usuario

El área de manejo de usuarios es un área exclusiva de usuarios con perfil de administrador, permite visualizar los usuarios de sistema, hacer altas, bajas y modificaciones de los mismos como también establecer el perfil de cada usuario.

4.5.5 Area de manejo de clusters

Este área también es de uso exclusivo de usuarios con perfil de administrador, permite visualizar los clusters del sistema y los equipos de cada uno, se pueden hacer altas, bajas y modificaciones tanto de clusters como de maquinas.

El hecho de agregar un cluster o un equipo a un cluster no implica que pueda usarse, todo equipo debe estar correctamente configurado para funcionar en la arquitectura de cluster. Esta es una tarea que no puede hacerse de forma automática y posteriormente se explica la manera de configurar una maquina que quede operativa en el cluster.

En este área también se especifica el modo en que se encuentra el cluster ya sea exclusivo o compartido.

No existe una política que determine cuando el cluster estará en un modo u otro y que ayude a determinar el cambio de modo de forma automática, por lo tanto, el cambio de modo exclusivo a compartido y viceversa lo hace algún administrador en forma manual en este área.

4.5.6 Area de ambientes de ejecución

Los ambientes de ejecución MPI suelen tener motores (“daemons”) de arranque y detención, y los programas no podrán ser ejecutados si estos no están activos.

El área de ambientes de ejecución es otra área de uso exclusivo de usuarios con perfil de administrador y permite arrancar o detener estos motores de ejecución.

4.5.7 Area de asignación de accesos exclusivos

En esta área se permite asignar a un usuario un periodo de tiempo para que utilice la arquitectura de cluster de forma exclusiva.

Esta área también es de uso exclusivo de usuario con perfil de administrador.

4.5.8 Area de registros de ejecución

Este es un área, exclusiva para usuarios con perfil administrador, que permite ver el registro de las ejecuciones que se realizaron, cada registro contiene la siguiente información:

- 1 El usuario que realizó la ejecución.
- 2 El archivo que ejecutó.
- 3 El ambiente que utilizó para la ejecución del archivo (LAM o MPICH2).
- 4 El modo que utilizó para la ejecución (sincrónico, asincrónico o interactivo).
- 5 La fecha y hora en que se realizó la ejecución.

El área permite ver todos los registros de ejecución o bien un periodo particular en el que se hayan hecho ejecuciones sobre la arquitectura de cluster.

4.6 Aspectos de seguridad

En relación a la seguridad se tomaron solo medidas mínimas y es tema de estudio el caso de la seguridad en este tipo de arquitecturas.

El sistema permite ejecutar remotamente una aplicación que no se sabe que tipo de código pueda contener (esto incluye la posibilidad que exista código malicioso que pueda dañar todo el sistema). Por lo tanto, toda la aplicación Web, incluyendo el servidor y la estructura de archivos para cada usuario, trabajan bajo un único usuario de sistema sin privilegios de administrador (no utiliza el usuario root).

De todos modos esto permite que no se puedan modificar estructuras de sistema sin permisos pero no evita, por ejemplo, el hecho de que se pueda leer el contenido del equipo, como evitar que puedan borrarse archivos en los cuales el usuario tenga permisos, en particular por la forma del sistema cada usuario de aplicación tiene el mismo usuario de sistema por lo tanto tiene permisos para modificar la estructura de archivos de otros usuarios de aplicación.

Igualmente constituye una medida mínima para evitar daños mucho mayores.

Como se menciona en un párrafo anterior, toda ejecución de usuario está registrada y la aplicación Web provee un mecanismo transparente sobre la estructura de archivos que le impide obtener información extra a la que ya tiene en su espacio de trabajo, y además cada usuario es bien conocido.

Que un usuario intente obtener información adicional sobre cuestiones de sistema no estaría hablando de algo casual sino de algo malintencionado, y esta intención puede registrarse.

Capítulo 5

Configuración de la arquitectura

5.1 Introducción

La arquitectura para que la aplicación Web funcione correctamente requiere una configuración apropiada tanto del servidor como de cada máquina en uno o varios clusters.

La configuración abarca:

- Sistema operativo.
- Crear un usuario de sistema.
- Espacio de trabajo de los usuarios.
- JAVA.
- Ambientes de ejecución MPI.
- Configuración SSH, RSH (acceso sin contraseña).
- Variables de ambiente.
- NFS.

Lo anterior es para todos los equipos incluyendo al servidor, aunque para este último se debe configurar:

- Servidor Web.
- Motor de base de datos MySQL.
- Espacio de trabajo de los usuarios.

5.2 Configuración del servidor

5.2.1 Sistema operativo

El sistema usado en este caso es Fedora Core 5 pero puede utilizarse cualquier distribución de Linux.

Cualquiera sea el sistema operativo debe tener algunos paquetes instalados, la forma de instalarlos depende de la distribución (solo se va a explicar el caso de Fedora Core 5).

Los paquetes son:

- GCC: compilador C para Linux.

En Fedora la instalación es como usuario root de la siguiente forma:

```
root# yum install gcc
```

- G++: compilador C++ para Linux.

En Fedora la instalación es como usuario root de la siguiente forma:

```
root# yum install gcc-g++-4.1.1-1.FC5.
```

- portmap: es utilizado port NFS para mapear puertos libres.

En Fedora la instalación es como usuario root de la siguiente forma:

```
root# yum install portmap
```

- nfs-utils: utilidades para usar NFS.

En Fedora la instalación es como usuario root de la siguiente forma:

```
root# yum install nfs-utils
```

- xinetd: para levantar servicios por demanda

xinetd es un “daemon” para los servicios de Internet. En lugar de tener muchos servidores distintos corriendo a la vez, sólo se carga xinetd. En el momento en que se hace una petición de servicio, xinetd lanza el servidor correspondiente.

En Fedora la instalación es como usuario root de la siguiente forma:

```
root# yum install xinetd
```

- ssh, rsh: para acceso remoto, usado internamente para la comunicación entre las maquinas del cluster.

En Fedora estos servicios vienen incluidos pero de no tenerlos la instalación se puede hacer como usuario root de la siguiente forma:

```
root# yum install ssh-server ssh rsh-server rsh
```


5.2.2 Espacio de trabajo de los usuarios de la aplicación

Como se mencionó anteriormente, cada usuario tiene un espacio de trabajo, un área donde almacena archivos y directorios.

Los espacios de trabajo de los usuarios deben estar en el sistema de archivos del servidor (aunque podrían residir en otro equipo), y son independientes del sistema de archivos del sistema operativo. Esto significa, por ejemplo, que el directorio raíz del espacio de trabajo de un usuario no necesariamente es el directorio raíz del sistema de archivos del servidor, cada usuario tiene una vista transparente de dónde están ubicados sus archivos, solo ve sus archivos como si tuviera un disco o partición asignada solo para él.

Físicamente los espacios de trabajo deben residir en algún lugar del sistema de archivos del sistema operativo, la política para asignar este lugar fue crear un directorio dentro del sistema de archivos del servidor donde cada usuario tiene su directorio propio de trabajo; el nombre de directorio de usuario coincide con el nombre de usuario a nivel aplicación.

Por ejemplo:

Crear un directorio base en el sistema de archivos local:

```
root# mkdir /home/usuarios
```

Esto crea un directorio /home/usuarios en el sistema de archivos del servidor, luego para cada usuario a nivel aplicación hay que crear un directorio dentro con el nombre del usuario.

Entonces si el nombre a nivel de aplicación del usuario es Usuario1, se debe crear el directorio correspondiente:

```
root# mkdir /home/usuarios/Usuario1
```

Este será el espacio de trabajo de este usuario, en la aplicación Web el usuario tiene una visión de este directorio transparente, para él es el directorio raíz(/).

Es posible que un administrador pueda cambiar el nombre de usuario de algún usuario en particular; en este caso, el nombre del directorio perteneciente a ese usuario también será modificado, para el usuario esto es transparente y no altera la visión que tenga del sistema.

El directorio base no necesariamente tiene que ser /home/usuarios, se puede personalizar y la aplicación Web sabe como lograr la transparencia deseada.

5.2.3 Usuario de sistema

La correspondencia entre usuarios de la aplicación y usuarios de sistema se realiza mediante la creación de un único usuario de sistemas para todo usuario de aplicación, por lo tanto se debe crear este usuario y el correspondiente grupo en el sistema operativo.

La forma de hacerlo es como usuario root de la siguiente forma:

##Crear el grupo

```
root# groupadd lrusuario
```

##Crea el usuario perteneciente al grupo lrusuario y con un home particular ##

```
root# adduser -d /home/usuarios -s /bin/bash -g lrusuario lrusuario
```

lrusuario es entonces el usuario de sistema que utilizaran los usuarios de aplicación, el home directory de este usuario coincide con el directorio base donde estarán los espacios de trabajo de los usuarios (esto no necesariamente tendría que coincidir pero en cierta forma es mas práctico porque solo se comparte un solo directorio mediante NFS con lo demás equipos, sino coincidieran entonces se deben compartir dos directorios mediante NFS, el home del usuario de sistema y el área de trabajo de los usuarios de aplicación).

Ahora, los directorios creados hasta el momento tienen como propietario a root, nadie puede escribir en ellos, el usuario lrusuario debe ser propietario de su home directory y además en los directorios que representan los espacios de trabajo de los usuarios, esto porque cada usuario de aplicación usa este usuario de sistema, por lo tanto se debe cambiar el propietario:

##Cambia el propietario al directorio base

```
root# chown -R lrusuario:lrusuario /home/usuarios
```

##Cambia el propietario a un directorio de usuario

```
root# chown -R lrusuario:lrusuario /home/usuarios/Usuario1
```

5.2.4 Configuración JAVA

La aplicación corre bajo una maquina virtual JAVA por lo que este entorno debe estar configurado, al menos lo que se denomina JRE

JRE (*Java Runtime Environment*) proporciona únicamente un subconjunto del lenguaje de programación JAVA sólo para ejecución. Un usuario necesita el JRE para ejecutar las aplicaciones desarrolladas en lenguaje JAVA, mientras que para desarrollar nuevas aplicaciones en dicho lenguaje es necesario un entorno de desarrollo, denominado JSDK (*Java Standard Development Kit*), que incluye al JRE.

Para este caso se instalo JAVA completo, JSDK.

Se descarga el JSDK que para Linux generalmente viene en un paquete llamado:

```
jdk-<version>-linux-i586.rpm.bin
```

La versión utilizada en este caso fue la 1.5.0_9.

La instalación se debe hacer con usuario root de la siguiente forma:

```
root# chmod a+x jdk-1_5_0_09-linux-i586.rpm.bin
root# ./jdk-1_5_0_09-linux-i586.rpm.bin
```

Esto deja instalado JAVA completo en /usr/java aunque puede cambiarse el directorio moviendo el contenido a otro lugar.

Para que el sistema tome los comandos JAVA se deben modificar las variables de ambiente, editando como root el archivo /etc/profile y agregando las siguientes líneas:

```
export JAVA_HOME=/usr/java/jdk1.5.0_09
export PATH=$JAVA_HOME/bin:$PATH
```

Para probar que esto realmente funciona solo se debería, en principio, reiniciar la maquina, pero se puede ser menos brusco y simplemente ejecutar primero:

```
root# ./etc/profile
```

Esto para que se tomen los cambios de las variables de ambiente y luego ejecutar:

```
root# java -versión
```

Si todo funciona bien la ejecución de esto hace referencia a la versión instalada de JAVA, algo como:

```
java version "1.5.0_09"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_09)
Java HotSpot(TM) Server VM (build 1.5.0_09, mixed mode)
```

5.2.5 Servidor Web

El servidor Web utilizado es Apache Tomcat que además de ser servidor Web es contenedor de aplicaciones JAVA, por lo tanto debe haber una correspondencia entre el servidor y el JRE para que poder correr aplicaciones.

El servidor se obtiene de la pagina Web del proyecto Jakarta Tomcat, la distribución para Linux viene en un archivo .tar.gz.

La versión utilizada en este caso es Jakarta Tomcat 5.5.20.

Como usuario root, se descomprime el contenido de la distribución de Tomcat y se lo deja en algún directorio por ejemplo /tomcat

```
root# gunzip apache-tomcat-5.5.20.tar.gz
root# tar apache-tomcat-5.5.20.tar
root# mv apache-tomcat-5.5.20 /tomcat
```

Luego hay que hacer la correspondencia entre el servidor y JAVA, si la variable de ambiente JAVA_HOME está configurada no suele ser necesario hacer otro tipo de configuración adicional.

De no ser así, o bien porque no se quiere agregar la variable de ambiente o se quiere correr bajo algún usuario en particular en otro contexto, se debe modifica el archivo llamado catalina.sh que se encuentra en:

```
<directorio de instalación>/bin
```

y agregarle la línea:

```
JRE_HOME = directorio donde esta el JDK o JRE.
```

Cuando se ejecuta un programa JAVA desde el servidor Web, este corre bajo el usuario con el que el servidor esta instalado. En este caso, el servidor fue instalado y configurado con el usuario root, lo cual trae riesgos de seguridad importantes porque se podrían ejecutar tareas con permisos de “súper usuario”.

La solución para no dar permisos de “súper usuario” es cambiar el propietario del servidor Web a un usuario sin estos privilegios.

El manejo de usuarios propuesto es que cada usuario de aplicación corre bajo un único usuario de sistema, por lo tanto el usuario creado anteriormente, lrusuario, es el propietario del servidor Web, ya se hizo referencia anteriormente a esto aclarando que es útil para no hacer cambios de usuarios entre el usuario del servidor Web y el usuario de sistema que representa al usuario de aplicación.

Para cambiar el propietario al servidor Web se debe hacer con el usuario root de la siguiente forma:

```
root# chown -R lrusuario.lrusuario /tomcat
```

Luego el servidor Web se puede arrancar o detener, ahora con el usuario lrusuario, de la siguiente forma:

```
#arrancar el servidor:
lrusuario# /tomcat/bin/startup.sh
```

```
#detener el servidor:  
lusuario# /tomcat/bin/shutdown.sh
```

Puede hacerse que Tomcat arranque como servicio creando el archivo /etc/tomcat con el siguiente contenido:

```
#!/bin/sh  
# tomcat web server/container  
  
case "$1" in  
'start')  
    /tomcat/bin/startup.sh  
    ;;  
'stop')  
    /tomcat/bin/shutdown.sh  
    ;;  
*)  
    echo "Usage: $0 { start | stop }"  
    ;;  
esac  
exit 0
```

Luego se hace que inicie en el arranque en los distintos runlevels:

```
root#chkconfig --level 345 tomcat on
```

Las aplicaciones Web, en Tomcat, se deben dejar en el directorio /tomcat/webapps en un archivo con extensión .war que es generado por la mayoría de los ambientes que trabajan sobre esta plataforma, es simplemente un archivo zip con una extensión diferente que puede ser interpretada por tomcat. El servidor automáticamente descomprimirá el archivo en una carpeta con el mismo nombre.

5.2.6 Ambientes de ejecución MPI

La aplicación Web usa tanto Lam MPI como Mpich2 MPI, ambos utilizan el compilador C configurado previamente (gcc) aunque también se pueden configurar para otros lenguajes, pero C se suele utilizar más comúnmente.

Se pueden configurar bajo el usuario root pero ninguna de las dos implementaciones deja correr programas bajo este usuario, ni siquiera dejan correr el servicio.

La configuración de estos ambientes para Fedora solo requiere configurar los paquetes ya instalados y configurados anteriormente, otras distribuciones de Linux tal vez necesiten instalar algún otro conjunto de librerías o servicios que no se incluyen en la instalación. Un ejemplo es Ubuntu o Debian donde se necesita instalar mediante apt el paquete build-essential que incluye librerías y compiladores necesarios.

5.2.6.1 LAM - MPI

La versión utilizada de Lam es la 7.1.2, se descarga gratuitamente de la página Web generalmente en un archivo .tar.gz y como usuario root se procese de la siguiente forma:

```
root# gunzip lam-7.1.2.tar.gz
root# tar -xf lam-7.1.2.tar
```

Esto deja un directorio llamado lam-7.1.2, se ingresa al mismo y se hace la configuración que suele tardar bastante:

```
root# ./configure --prefix=/usr/local/lam-7.1.2 --without-fc
```

Se le está diciendo que instale Lam MPI en el directorio /usr/local/lam-7.1.2, la opción --without-fc le indica que no configure el compilador de Fortran, al menos no es necesario en este caso pero de llegar a necesitarlo habría que configurar los paquetes de Fortran para la distribución de Linux que se utilice.

Luego se ejecuta algo que también suele tardar bastante:

```
root# make
root# make install
```

Después de esto quedara instalado Lam MPI.

Para que pueda funcionar se deben crear las variables de ambiente modificando como usuario root el archivo /etc/profile y agregando las líneas

```
export LAM_HOME=/usr/local/lam-7.1.2
export PATH=$LAM_HOME/bin:$PATH
```

Para que se tomen los cambios hay que reiniciar la maquina o bien ejecutar:

```
root# . /etc/profile
```

pero como vamos a ejecutar con otro usuario es conveniente cargarlo en una consola para ese usuario, así que ejecutamos:

```
root# su lusuario
lusuario# . /etc/profile
```

Más adelante se muestra como configurar las variables de ambiente solo para un usuario particular.

Luego para probar que todo funciona bien se inicia Lam de la siguiente forma:

```
lusuario# lamboot
```

Si todo esta bien muestra en la salida:

```
LAM 7.1.2/MPI 2 C++/ROMIO - Indiana University
```

Luego de esto el motor de Lam estará corriendo.

también es posible pasarle un parámetro haciendo referencia a un archivo que tiene las direcciones de las maquinas del cluster y de esta forma se ejecuta el comando lamboot en cada maquina a la que haga referencia el archivo. La forma de ejecutar lamboot en todo el cluster seria:

```
lusuario# lamboot maquinas
```

Donde maquinas es un archivo con los nombres o direcciones IP de las máquinas del cluster que se quieran utilizar.

Esto sólo funciona si está configurado SHH/RSH adecuadamente como se explica más adelante.

Luego para compilar archivos se utiliza un compilador propio llamado hcc con la siguiente sintaxis:

```
hcc -o <archivo de salida> <archivo fuente .c>
```

Para ejecutar un archivo compilado se hace mediante:

```
mpirun -c <cantidad total de procesos> <archivo ejecutable> <parámetros>
```

Para detener el motor de Lam MPI se hace a través de:

```
lusuario# lamhalt
```

Para ejecutar un programa sobre el cluster utilizando Lam se debe correr el siguiente comando:

```
lusuario#mpirun -c <cantidad de procesos> miPrograma <parámetros>
```

Esto corre el programa miPrograma en el cluster, el parámetro -c indica la cantidad de procesos que hará correr el programa.

Se debe asegurar que lamboot esta corriendo antes de ejecutar este comando, de lo contrario retornara un error.

Existe una forma de ejecutar sin hacer lamboot previamente y es utilizando el comando mpiexec de la siguiente forma:

```
lusuario# mpiexec -machinefile maquinas -c <cantidad de procesos> miPrograma <parámetros>
```

Esto corre lamboot en las maquinas indicadas en el archivo maquinas y luego ejecuta el programa de la misma forma que la explicada anteriormente.

5.2.6.2 MPICH2 - MPI

La versión utilizada es Mpich2 mpi versión 1.0.4p1, se descarga de la página en un formato .tar.gz., se descomprime:

```
root# .gunzip mpich2-1.0.4.tar.gz
root# tar-xf mpich2-1.0.4.tar
```

Esto deja un directorio mpich2-1.0.4, se ingresa a este y se configura casi de la misma forma que Lam, en un proceso que tiene la misma demora:

```
root# ./configure --prefix=/usr/local/mpich2-1.0.4
root# make
root# make install
```

Con esto anterior queda instalado Mpich2 MPI en el directorio /usr/local/mpich2-1.0.4. Luego se modifican, como usuario root, las variables de ambiente en el archivo /etc/profile agregando las líneas:

```
export MPICH2_HOME=/usr/local/mpich2-1.0.4
export PATH=$MPICH2_HOME/bin:$PATH
```

Hay una diferencia con Lam en que para poder ejecutar el motor primero se debe crear, en el home directory del usuario que quiera usarlo, un archivo de configuración. En este caso se utilizó el usuario lrusuario que es el que interesa. Se ingresa al home directory del usuario y se crea el archivo .mpd.conf

```
lrusuario# touch .mpd.conf
```

Se le da permisos solo a ese usuario:

```
lrusuario# chmod 600 .mpd.conf
```

Se edita el archivo agregando la siguiente línea:

```
MPD_SECRETWORD=<alguna contraseña>
```

Esto es para poder correr el motor pidiendo una contraseña al usuario, pero en este caso no es necesario porque el motor se ejecuta desde la aplicación Web y es transparente. Dejando la contraseña en blanco no pide ninguna autenticación. Para correr el motor se puede hacer de dos formas:

La primera:

```
lrusuario# mpd &
```

La segunda:

```
lrusuario# mpdboot
```

La segunda funcionó mejor con la aplicación Web.

De la misma forma que con Lam MPI, es posible pasarle un parámetro haciendo referencia a un archivo que tiene las direcciones de las maquinas del cluster y así ejecutar la aplicación sobre estas maquinas. La forma de hacerlo es la siguiente:

```
lusuario# mpdboot -file maquinas
```

Donde maquinas es un archivo que contiene el nombre o IP de las máquinas a utilizar en el cluster.

Para compilar se hace de la siguiente manera:

```
mpicc -o <archivo de salida> <archivo fuente>
```

Para ejecutar el archivo compilado anteriormente se hace:

```
mpiexec -n <cantidad de procesos> <programa> <parámetros de programa>
```

Y para detener el motor Mpich2 MPI se hace:

```
lusuario# mpdallexit
```

Para la ejecución de programas se utiliza el comando mpiexec de manera similar a Lam, se puede hacer con el parámetro machinefile o sin el, en este último caso se debería correr mpdboot. La ejecución se hace de la siguiente forma:

```
lusuario# mpiexec -n <cantidad de procesos> MiPrograma <parámetros>
```

El comando es semánticamente igual al de Lam con la diferencia sintáctica en el nombre del comando y el parámetro -n que es igual a -c.

Para poder correr sobre el cluster se debe tener configurado el acceso RSH/SSH a cada maquina sin que se pida contraseña para el usuario que vaya a ejecutar.

Esto se explica a continuación.

5.2.7 Configuración SSH, RSH, acceso sin autenticación

Los ambientes MPI necesitan que cada máquina del cluster se comunique con las demás mediante los protocolos RSH/SSH, estos protocolos suelen pedir una contraseña al usuario cuando lo utiliza, pero los ambientes de MPI no necesitan ninguna contraseña, por lo tanto hay que configurar estos protocolos para que no requieran autenticación.

En el caso de Mpich2 requiere esta configuración aun cuando se utilice una sola maquina (cuando se hace un llamado a la misma maquina) es por eso que esta configuración debe estar hecha también sobre el servidor.

La idea de configurar SSH/RSH para no requerir autenticación es la siguiente:

Suponer que el usuario UA de la maquina MA quiera acceder por SSH/RSH a la maquina MB como usuario UB sin que esta le pida una contraseña, se supone que UA existe en la maquina MA y UB existe como usuario en la maquina MB.

Suponer que se está en la maquina MA logueado como usuario UA, entonces se deben generar las claves publicas y privadas para este usuario (se suele pedir una contraseña que hay que dejar en blanco).

Luego la clave pública hay que llevarla a la maquina MB y agregarla en el archivo de claves autorizadas del usuario UB en esa maquina.

Para el caso del servidor, y porque los ambientes MPI lo requieren, es necesario hacer esta configuración para que se pueda hacer SSH/RSH a la misma maquina y con el mismo usuario, donde el usuario a utilizar en lrusuario.

Se procede de la siguiente forma:

Primero se generan las claves pública y privada:

```
lrusuario# ssh-keygen -t rsa
```

Cuando pida la contraseña dejarla en blanco.

Luego quedan dos archivos en el directorio .ssh del home directory del usuario, esto es en /home/usuarios/.ssh quedan los siguientes archivos:

```
id_rsa.pub  
id_rsa
```

Luego el contenido del archivo id_rsa_pub debe copiarse en el directorio .ssh del home directory del usuario lrusuario:

```
lrusuario# cat id_rsa.pub >> /lrhome/lruser/.ssh/authorized_keys
```

Y por ultimo se deben modificar los permisos:

```
lrusuario# chmod 400 /lrhome/lruser/.ssh/authorized_keys
```

Para probar esto se puede hacer ejecutando SSH a la maquina misma y no tendría que pedir ninguna contraseña:

```
lusuario# ssh localhost -l lusuario
```

Mas adelante se explica cómo se configura esto para las demás máquinas del cluster pero viendo las ventajas de utilizar el protocolo NFS.

Un punto más es que se debe crear un archivo en el home directory del usuario llamado `.rhosts` con el contenido de las máquinas que pueden acceder. Este archivo es usado por los ambientes MPI dentro del contexto de RSH para que puedan comunicarse las maquinas, por lo tanto el contenido de ese archivo debe tener las direcciones de las maquinas del cluster.

Una aclaración: en el caso de Fedora, el protocolo RSH que trae no es el standard, por lo tanto al instalarlo se debe hacer referencia al comando que corresponda.

Es posible que el servicio RSH/SSH esté deshabilitado, por lo tanto habría que ver en los archivos en:

```
/etc/xinet.d/rsh
/etc/xinet.d/rlogin
/etc/xinet.d/rexec
```

Y comentar la línea “disable yes” o bien escribiendo “disable no”.

En el caso particular de LAM se puede forzar a que utilice ssh en vez de rsh agregando la variable de ambiente `LAMRSH=ssh`.

5.2.8 Variables de ambiente locales a un usuario

Algunas variables de ambiente suelen ser más prácticas si en vez de definir las en `/etc/profile` se definen dentro del home directory de un usuario en particular en el archivo `.bashrc` (esto porque no siempre se tiene permisos al archivo `/etc/profile` pero también porque puede ahorrar tiempo de configuración si se complementa utilizando el protocolo NFS que se vera a continuación).

Se pueden agregar los path's de los ambientes MPI, el `JAVA_HOME` y las referencias que se deseen.

En este caso se agregan al archivo `.bashrc` del home directory del usuario `lusuario`, esto seria en:

```
/home/usuarios/.bashrc
```

Una aclaración: En Fedora el rsh por defecto no es el estándar y lo toma desde otra ruta, por lo tanto, para que tome el rsh standard hay que agregar:

```
export PATH=/usr/bin:$PATH
```

5.2.9 NFS (Network File System)

NFS es un protocolo, generalmente de sistemas Unix/Linux, que permite compartir directorios de una forma altamente transparente, el usuario no tiene conciencia de si el directorio compartido por NFS es local o remoto.

Este protocolo es muy útil, existen una buena cantidad de configuraciones que deben hacerse en cada maquina del cluster y que pueden evitarse sólo compartiendo un directorio desde una máquina y que las demás máquinas lo vean.

El caso mas importante es el del home directory del usuario lusuario, en cada máquina del cluster este usuario debe existir además de tener la misma configuración de RSH/SSH, variables de ambiente y los espacios de trabajo de los usuarios de aplicación, esto ultimo teniendo en cuenta que los ambientes MPI necesitan para ejecutar un programa, que este exista en toda maquina del cluster en la misma ruta.

Para ahorrar tiempo se comparte el home directory del usuario lusuario con las demás maquinas del cluster.

El servidor debe ser configurado entonces como un servidor NFS.

Los directorios a compartir generalmente se encuentran en:

`/etc/exports`

Para agregar directorios compartidos la línea que se debe ingresar es la siguiente:

(directorio a compartir) (host o red/mascara de red) (permisos)

ejemplo:

```
/directorio1 192.168.0.0/255.255.255.0(rw)
/directorio2 192.168.0.10(ro)
/directorio3 192.168.0.0/255.255.255.0(ro)
/direcotrioN *(rw)
```

* Cualquier host

(rw) lectura escritura

(ro) solo lectura

En el caso del home directory del usuario lusuario la línea es:

```
/home/usuarios 192.168.0.0/255.255.255.0
```

Para indicar al sistema que relea el archivo y active los cambios se puede hacer reiniciando el demonio nfsd:

```
root# /etc/init.d/nfs restart
```

o bien mediante el comando:

```
root#exportfs -ra
```

Luego se explicará como hacer que las demás maquinas vean este directorio.

5.2.10 MySQL

Por último, la aplicación utiliza un motor MySQL para almacenar información, parte de esta información, esta relacionada con distintas rutas de acceso a los programas y lugares específicos de archivos en el sistema operativo, como por ejemplo:

- Rutas de los ambientes mpi.
- Rutas de comandos de sistemas para distintas operaciones sobre la aplicación.
- Ruta de los espacios de trabajo de los usuarios.

Además de lo anterior se tienen registros de ejecuciones, usuarios y clusters con sus respectivas maquinas.

No sólo hay que instalar y configurar el motor de base de datos sino también descargar las librerías para JAVA, que son utilizadas por el servidor Web.

Estas librerías vienen en un archivo con formato .jar que debe ser copiado en el directorio de instalación de Tomcat en el subdirectorío /shared/lib.

La instalación del motor de base de datos es muy sencilla y depende de la distribución de MySQL, para este caso solo se descargó el archivo correspondiente al motor de base de datos y se instaló siguiendo las instrucciones del archivo de documentación que viene adjunto al archivo de instalación.

5.3 Configuración de una máquina perteneciente al cluster

5.3.1 Sistema operativo

Para la mayoría de las máquinas del cluster se utilizó Fedora Core 5, y se configuraron algunas otras máquinas con Ubuntu 6.10.

La configuración para las máquinas con Fedora Core 5 es la misma que la hecha en el servidor instalando los mismos paquetes. En el caso de las máquinas con Ubuntu 6.10 se deben instalar los siguientes paquetes utilizando apt-get:

- librerías de desarrollo, incluyen GCC y G++:

```
root# apt-get install build-essential
```

- Portmap: es utilizado port NFS para mapear puertos libres.

```
root# apt-get install portmap
```

- ssh: para acceso remoto, usado internamente para la comunicación entre las máquinas del cluster.

```
root# apt-get install ssh
```

No es necesario para el caso de Ubuntu 6.10 instalar el paquete xinetd.

5.3.2 Usuario de sistema

El usuario de sistema lusuario debe ser creado en cada máquina del cluster. El nombre, uid(identificador de usuario), grupo y gid(identificador de grupo) deben ser los mismos que en el servidor.

Además debe ser creado el home directory del usuario lusuario para lograr la transparencia deseada cuando este haga referencia al home directory ubicado en el servidor mediante NFS.

La creación de grupo y usuario se hace de la misma forma que la realizada en el servidor, pero seguramente al crearlos el gid y el uid son diferentes de los creados en el servidor y además diferentes entre los creados en las otras máquinas del cluster.

Para que todas las máquinas tengan el mismo usuario con las mismas características una opción es como se mencionó en algún momento, tener una autenticación centralizada mediante LDAP, NIS o NIS+, o sólo modificar los archivos donde está la configuración de los usuarios y grupos de sistema, estos archivos son /etc/passwd para el caso de usuarios y /etc/group para el caso de grupos, allí se pueden modificar el uid y gid respectivamente en los archivos para ese usuario. Otra forma es directamente agregar el gid y uid desde el comando al crear el grupo y usuario de la siguiente forma:

```
##crear el grupo
```

```
root# groupadd -g gid lusuario
```

```
##crea el usuario perteneciente al grupo lusuario y con un home particular ##
```

```
root# adduser -d /home/usuarios -s /bin/bash -g lusuario -u uid lusuario
```

5.3.3 Configuración JAVA

La configuración de JAVA es exactamente la misma que la realizada en el servidor.

5.3.4 Ambientes de ejecución MPI

La configuración no cambia con respecto a la del servidor, se debe hacer de la misma forma.

5.3.5 Configuración SSH, RSH, acceso sin autenticación

Como el home directory del usuario lrusuario es compartido mediante NFS, gran parte de esta configuración no se hace, sólo hay que asegurarse que los comandos rsh y ssh estén en la ruta correspondiente y que sean los Standard; además hay que habilitar el servicio como se explico anteriormente para el caso del servidor en los archivos:

```
/etc/xinet.d/rsh  
/etc/xinet.d/rlogin  
/etc/xinet.d/rexec
```

Luego que esto esté configurado y después de haber realizado la configuración de NFS en cada maquina del cluster, se puede probar acceder de una a otra maquina mediante RSH/SSH para testear que no se pida contraseña.

5.3.6 Variables de ambiente locales a un usuario

Las variables de ambiente locales al usuario lrusuario en el archivo .bashrc quedarán compartidas a partir de configurar NFS. Por lo tanto no se debe realizar ninguna configuración.

5.3.7 NFS (Network File System)

Cada máquina del cluster debe hacer referencia desde un directorio local, en este caso /home/usuarios local, al directorio compartido en el servidor, en este caso /home/usuarios del servidor.

Se puede hacer mediante un comando con la siguiente sintaxis:

```
root#mount <servidor>:<directorio remoto compartido> <directorio local>
```

Pero al reiniciar ya no estará más compartido; para este caso es necesario que cada vez que se reinicie haga referencia a ese directorio compartido.

Si se quiere que sea montado al arrancar el sistema se debe modificar el archivo /etc/fstab agregando una línea con la siguiente sintaxis:

```
<servidor>:<directorio remoto compartido> nfs <permisos>,hard,intr 0 0
```

Para montar entonces el home directory del usuario lusuario se agrega la siguiente línea:

```
servidor:/home/usuarios /home/usuarios nfs rw,hard,intr 0 0
```

Luego el home directory de lusuario en la máquina del cluster hace referencia al home directory de este usuario en la máquina servidor permitiéndole ver su contenido.

Al tener el mismo nombre, la transparencia es muy buena, dándole al usuario la sensación de que ese directorio es local.

Capítulo 6

Conclusiones y líneas de investigación y desarrollo abiertas

6.1 Conclusiones

La investigación y el desarrollo dan como resultado una aplicación Web instalada y configurada para el uso de alumnos, docentes e investigadores.

Anterior a la aplicación Web, el acceso al cluster, era en la mayoría de los casos de forma local, y el acceso remoto solo se daba en casos particulares mediante ssh por lo tanto requería de ciertos controles a nivel de firewall. Esto además tenía la desventaja que el usuario podía acceder al sistema a recursos innecesarios como por ejemplo poder ver archivos o directorios ajenos o de sistema.

La aplicación Web permite mayor disponibilidad del cluster, esto es una ventaja que favorece a los distintos tipos de usuarios, en particular a los alumnos que no contaban con un acceso remoto por cuestiones de seguridad, por lo tanto la única posibilidad que tenían de probar los programas sobre el cluster era ir donde la arquitectura está físicamente ubicada.

Otra ventaja de la aplicación Web es la de brindar un espacio de trabajo a cada usuario para manejar sus archivos y directorios, independiente de la estructura de directorios del sistema operativo; esto le da un grado importante de transparencia, y de esta forma el usuario sólo accede a su espacio de trabajo y se evita el acceso a recursos a los cuales no debiera acceder.

Los distintos modos de ejecución permiten varios niveles de interacción de los usuarios con el cluster y se adaptan a los distintos requerimientos de acuerdo a los tipos de programas que se corren sobre la arquitectura.

La aplicación Web permite controlar la forma de acceso al cluster, y de esto surgen dos modos, compartido y exclusivo, De esta manera el cluster puede verse como un único recurso en el que sólo puede estar un solo usuario ejecutando un programa, el modo compartido permite aprovechar ciertos tiempos ociosos de la arquitectura, mientras que el modo exclusivo da a un usuario la posibilidad de disponer del cluster por un periodo de tiempo determinado en los casos en que los programas necesitan de un tiempo de ejecución importante.

No sólo existe la posibilidad de usar el cluster remotamente, a través de la aplicación Web, sino también de administrar distintos aspectos como los ambientes para las librerías de MPI, los distintos usuarios, tipos de acceso, clusters y sus equipos.

No se requieren controles a nivel de firewall dado que no se utiliza ningún tipo de protocolo a excepción del que se necesite para el servidor Web; de todos modos los aspectos de seguridad deben tenerse en cuenta, la arquitectura fue configurada para dar un mínimo nivel de seguridad y la aplicación Web provee un registro de las ejecuciones realizadas para tener un control sobre cuales son las actividades que se desarrollaron y quien las realizo.

Montar un cluster y configurarlo no es una tarea sencilla, adaptarlo para ser usado remotamente vía Web agrega algo mas de complejidad, y mucho depende de la interacción y compatibilidad entre las distintas partes que componen la arquitectura global, sistema operativo, librerías MPI, protocolos ssh/rsh internos, servidor web, JAVA y base de datos. En muchos casos no es fácil hacer que interactúen de forma adecuada.

El desarrollo de la aplicación Web continúa solucionando los problemas que pudieran surgir durante su uso, agregando otras formas de ejecución sobre el cluster, extendiendo el cluster y adaptándolo a una estructura mas amplia como lo es una infraestructura de grid.

6.2 Líneas de investigación y desarrollo abiertas

La realización de una aplicación Web para el uso y administración de una arquitectura multicluster, hace que el uso de este recurso no sea solo exclusivo de un grupo de usuarios pertenecientes a una institución como lo son los docentes, investigadores y alumno de nuestra Facultad, sino que abre un camino a que pueda ser usado por usuarios externos, incrementando la cantidad de personas que utilicen la arquitectura lo que implica además tener un mayor control en cuanto a la ejecución de programas y a la seguridad.

Podría considerarse el caso en que la arquitectura multicluster no sea vista como un solo recurso exclusivo sino como varios, separando al cluster en dos o más “subclusters”, permitiendo satisfacer las demandas de distintos tipos de usuarios simultáneamente.

También se puede considerar el hecho de expandir el cluster uniéndolo a otros clusters, tal vez fuera de los límites de una LAN lo que lleva además a la noción de Grid.

Es por esto que muchos aspectos deben ser contemplados y el desarrollo e investigación sobre una aplicación Web para este tipo de arquitectura continúa en puntos tales como:

- Seguridad.
- Schedulers y gestión de programas a procesarse en el cluster.
- Fragmentación y escalabilidad de una arquitectura multicluster.
- Computación en Grid.

Hacer el control y administración de este tipo de características, sumadas a las características ya implementadas por la aplicación Web, es una tarea compleja si debe hacerse por separado como se venía haciendo sin la aplicación Web; por eso la importancia de esta a modo de único punto de acceso para el control y administración además de servir como un acceso transparente a los usuarios no administradores.

Por lo tanto, expandir la funcionalidad y características del cluster implica expandir la funcionalidad y características de la aplicación Web, aunque hay aspectos relacionados con la configuración que todavía no pueden hacerse vía Web pero que con la investigación y desarrollos futuros pueden lograrse.

6.2.1 Aspectos de seguridad en una arquitectura multicluster

La seguridad en este tipo de arquitectura es de importancia, más aun si se quiere que sea utilizada por un número de personas mucho mayor que lo acceden remotamente. Los usuarios envían sus programas a ejecutar pero no se sabe bien que tipo de código va a ser ejecutado sobre el cluster. Este código puede tener instrucciones que atenten contra la seguridad del sistema.

Es difícil tomar medidas en estos casos porque no se puede evaluar cada código que se vaya a correr en el cluster buscando instrucciones que atenten contra el sistema.

Como primer paso, en la aplicación Web, se pensó en impedir que los programas que ejecuten los usuarios lo hagan bajo un usuario con privilegios de administrador de sistema operativo. Esto es útil porque evita que el usuario pueda escribir áreas del sistema en las que no tiene permisos, pero no evita que pueda leer algunas áreas de sistema y así obtener información, y tampoco evita que un usuario pueda escribir en un área compartida con otro dentro del sistema de archivos en el sistema operativo, teniendo en cuenta que a nivel de sistema los usuarios son los mismos.

De todos modos, como se dijo anteriormente, toda actividad de usuario esta controlada y en esta etapa solo acceden usuarios bien conocidos, como la aplicación Web permite un grado importante de transparencia cualquier intento contra el sistema no es algo casual sino algo malintencionado que pudo haberse registrado.

Una opción es aplicar lo que se conoce como ACLs (*Access Control List*) que permite especificar permisos con mayor granularidad que los clásicos permisos (User, Group, Others) de los sistemas Unix/Linux. Las ACL son difíciles de configurar y demandan cierto tiempo.

Utilizando ACL por ejemplo se puede lograr que el usuario sólo pueda ver el directorio donde residen sus archivos y nada más que eso, pero no evita que un usuario pueda ingresar o modificar áreas compartidas con otros usuarios ya que el usuario del sistema es el mismo.

La idea seria agregar un usuario de sistema por cada usuario de aplicación; ya se mencionaron las complicaciones que trae esto al agregar un nuevo usuario, donde se requieren una gran cantidad de aspectos de configuración por cada uno, pero se piensa en una automatización del proceso lo que resulta una tarea por demás compleja.

Luego de esto, algo mucho más complejo, es la combinación de ACLs con protocolos como LDAP que puede dar un nivel de seguridad donde cada usuario solo pueda acceder nada más que a sus archivos y directorios, sin tener permisos de ningún tipo sobre la estructura de archivos de sistema o de otros usuarios.

6.2.2 Schedulers y manejador de recursos

Una arquitectura multicluster se ve como un recurso exclusivo de manera que sólo un programa puede estar corriendo y no se permite ninguna ejecución hasta que no se libere.

Teniendo en cuenta los tipos de usuarios en la aplicación Web se consideran dos modos de acceso, el exclusivo propiamente dicho donde un usuario tiene asignado un determinado tiempo de uso sobre el cluster y nadie más podrá acceder mientras permanezca este usuario, y el modo compartido donde se aprovechan de mejor forma los tiempos ociosos del cluster.

Se podría tener en cuenta un modo más, suponiendo que un usuario quisiera ejecutar un programa y el cluster estuviera ocupado, podría encolar el pedido para que sea ejecutado luego, cuando el cluster se libere.

Esto es algo que se ha venido haciendo utilizando herramientas denominadas schedulers y manejador de recursos.

Los schedulers deciden las políticas con las que se ejecutan determinados trabajos, determinan prioridades etc. Un ejemplo de scheduler open source es Maui.

Los manejadores de recursos permiten administrar los recursos de una arquitectura multicluster, proveen control sobre trabajos encolados y permiten una mejor distribución de los mismos sobre los recursos del cluster. Un ejemplo de manejador de recursos open source es Torque.

Otros schedulers o manejadores de recurso son Condor, Mosix y OpenMosix, que realizan tareas semejantes a las anteriores, y agregan tolerancia a fallos y migración de procesos.

No son fáciles de configurar y requieren de mucho tiempo no solo para configurarlo sino para aprender a hacerlo funcionar, inclusive algunas agregan algo de “ruido” a la red haciendo broadcast periódicos lo que resulta en una baja en el rendimiento en algunos casos.

Incluir estas herramientas también requiere de estudiar algunos usos de algún tipo de usuario particular, no tendría sentido para un alumno que solo quiere correr un programa que no va a demorar mas de cinco minutos, encolarlo y esperar un día hasta que el cluster se libere.

Seguramente tiene sentido para usuarios que corran programas con una demora importante y que es lo mismo ejecutarlo hoy o mañana.

Esto plantea el hecho de separar el cluster en dos o más “subclusters” o mantener políticas para aprovechar partes del cluster que no se utilizan (esto último lo hacen la mayoría de los manejadores de recursos).

En principio, y de acuerdo a los requerimientos de los usuarios que hasta ahora se tienen , en la aplicación Web sólo alcanza con tener configurado el cluster sin los schedulers y manejadores de recursos, manteniendo las dos políticas de acceso, exclusiva y compartida.

Queda como futura investigación y desarrollo la inclusión de estas herramientas.

6.2.3 Fragmentación y escalabilidad con clusters en redes remotas

En el párrafo anterior se planteo el hecho de separar o fragmentar el cluster en “subclusters” o de aprovechar áreas del cluster que no se estén usando.

Se podría tener un área del cluster sólo para usuarios que no requieran mucha demora en la ejecución, como es el ejemplo de un alumno que su programa demore 5 minutos por lo que no tiene sentido que espere un día para ejecutar.

Fragmentar el cluster se puede lograr en forma estática, estableciendo alguna política de fragmentación que determine que usuario puede usar toda o una parte de una arquitectura multicluster como también que parte no puede utilizar, pero también puede hacerse de forma dinámica aprovechando áreas de la arquitectura que no se estén usando, tarea que muchos manejadores de recursos realizan muy bien.

Existe un problema en la fragmentación y está relacionado con los ambientes de las librerías mpi, tanto Lam como Mpich2 necesitan que la máquina desde donde se “dispara” la ejecución de un programa sea parte del procesamiento de ese programa, por lo tanto, si se desean correr dos programas “disparándolos” desde una misma máquina, donde cada programa se ejecuta sobre partes separadas del cluster, cualquiera de los ambientes mpi dará un error o bien dejara de ejecutar el programa que estaba ejecutando para ejecutar el otro.

Otro planteo es la escalabilidad del cluster; las arquitecturas de cluster siempre estuvieron pensadas para redes LAN y no para redes mas amplias, utilizadas por usuarios locales y no remotos, de hecho los motores de MPI y PVM no están preparados para ejecutar en redes más grandes y necesitan extensiones, a esto se le agrega el hecho de la heterogeneidad tanto de software como de hardware lo que introduce mayor complejidad.

Escalar el cluster a redes remotas no es imposible aunque sea muy difícil y demande mucho tiempo, pero introduce a la idea de computación en Grid.

6.2.4 Generalizaciones a Grid

Un *Grid* es un tipo de sistema distribuido que permite compartir e integrar recursos distribuidos geográficamente, incluyendo procesadores, instrumentos, bases de datos e incluso recursos humanos. Esta clase de arquitectura permite definir múltiples configuraciones colaborativas entre comunidades de usuarios, donde cada usuario “ve” una arquitectura única.

Existen distintos proyectos sobre temas de grid, las aplicaciones que suelen utilizarse como banco de pruebas son entre otras:

- Algoritmos de tratamiento de imágenes (en particular de reconstrucción 3D).
- Algoritmos de reconocimiento de secuencias (en particular de ADN o genómicas).
- Algoritmos de simulación paramétrica (en particular de fenómenos naturales como incendios o inundaciones).

Se pueden considerar varios equipos como nodos de grid distribuidos geográficamente, estos nodos pueden formar entre ellos un cluster de manera que permitan procesar algún tipo de programa de forma paralela. Un nodo puede tener además algún cluster tradicional funcionando incrementando la capacidad de ese nodo.

Una arquitectura de este tipo tiene varias diferencias con respecto a un cluster tradicional. Entre otras cosas se pueden mencionar: la distancia que separa los equipos, un mayor control de seguridad, el overhead en las comunicaciones y el agregado de capas de software de control. La mayoría de los middlewares de grid funcionan bajo el concepto de Web Service por lo que generalmente se tiene en cada nodo uno o varios servicios Web escuchando por alguna petición.

Adaptar la aplicación Web a una arquitectura de este tipo implica incluir cada nodo geográficamente distribuido y utilizar algún middleware de grid como puede ser Globus Toolkit, se deben considerar además otros tipos de lenguajes y librerías como pueden ser JAVA o Python, aunque existe una discusión basada en la baja performance de estos lenguajes con respecto a MPI utilizando el lenguaje C, se sabe que MPI tiene problemas cuando se quiere utilizar en un entorno mas amplio que una red LAN; como se mencionó anteriormente existen extensiones para trabajar con esta librería fuera de una red LAN. Para el tema de grid, se esta trabajando en una implementación de la librería MPI, llamada MPICH-G2 que es una extensión de MPICH adaptada para trabajar sobre grid.

La idea de incluir lenguajes como JAVA o Python tiene sus fundamentos en la heterogeneidad: un programa escrito en C debe ser compilado para cada plataforma software/hardware diferente, en cambio JAVA y Python son multiplataforma.

La idea de grid aún está en desarrollo. y por lo tanto la aplicación Web se irá adaptando de acuerdo a la evolución de las herramientas para el uso en grid.

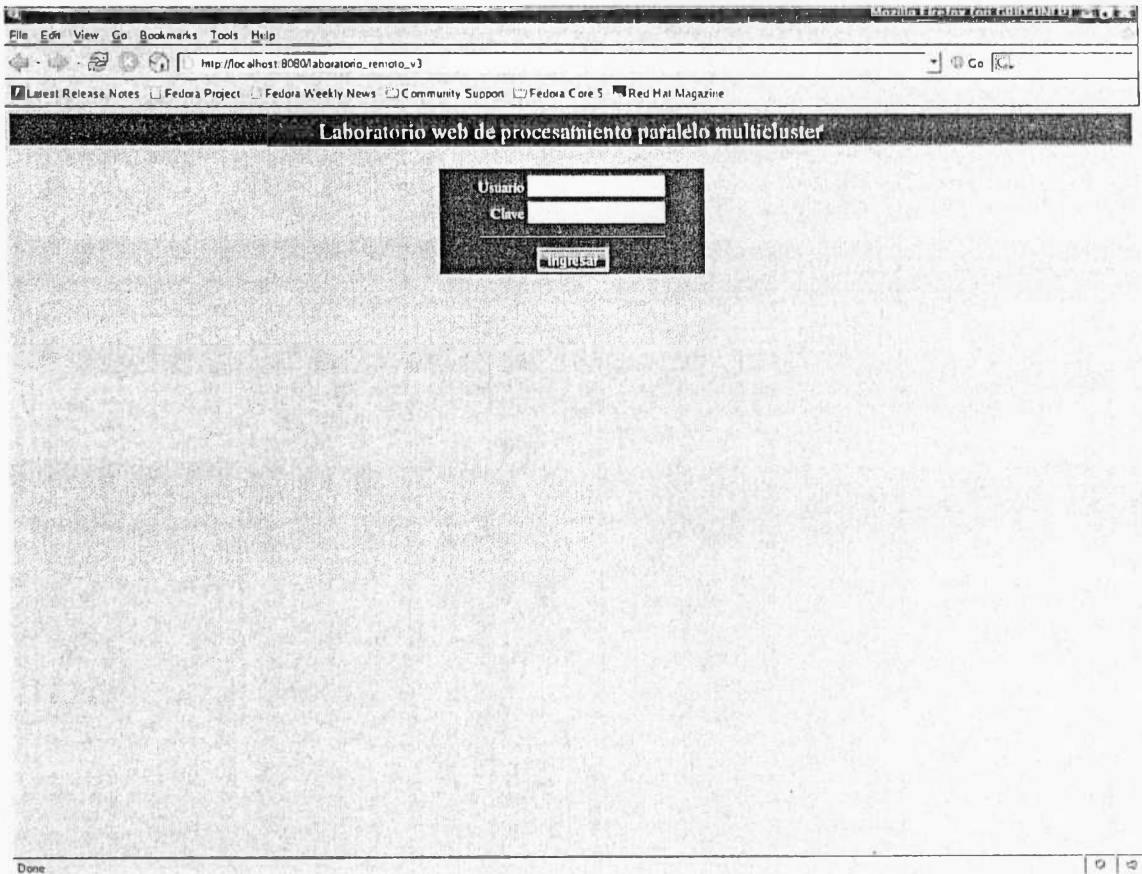
Anexos

Anexo A

Manual de usuario

A1. Pantalla inicial – autenticación de usuario

La pantalla inicial es una pantalla de autenticación clásica donde se pide ingresar un usuario y contraseña.



El cluster tiene dos modos, uno exclusivo y otro compartido.

Si el cluster está en modo exclusivo sólo un usuario podrá ingresar de acuerdo a la reserva del periodo de tiempo; mientras este usuario tenga el acceso exclusivo no podrá ingresar ningún otro usuario al sistema.

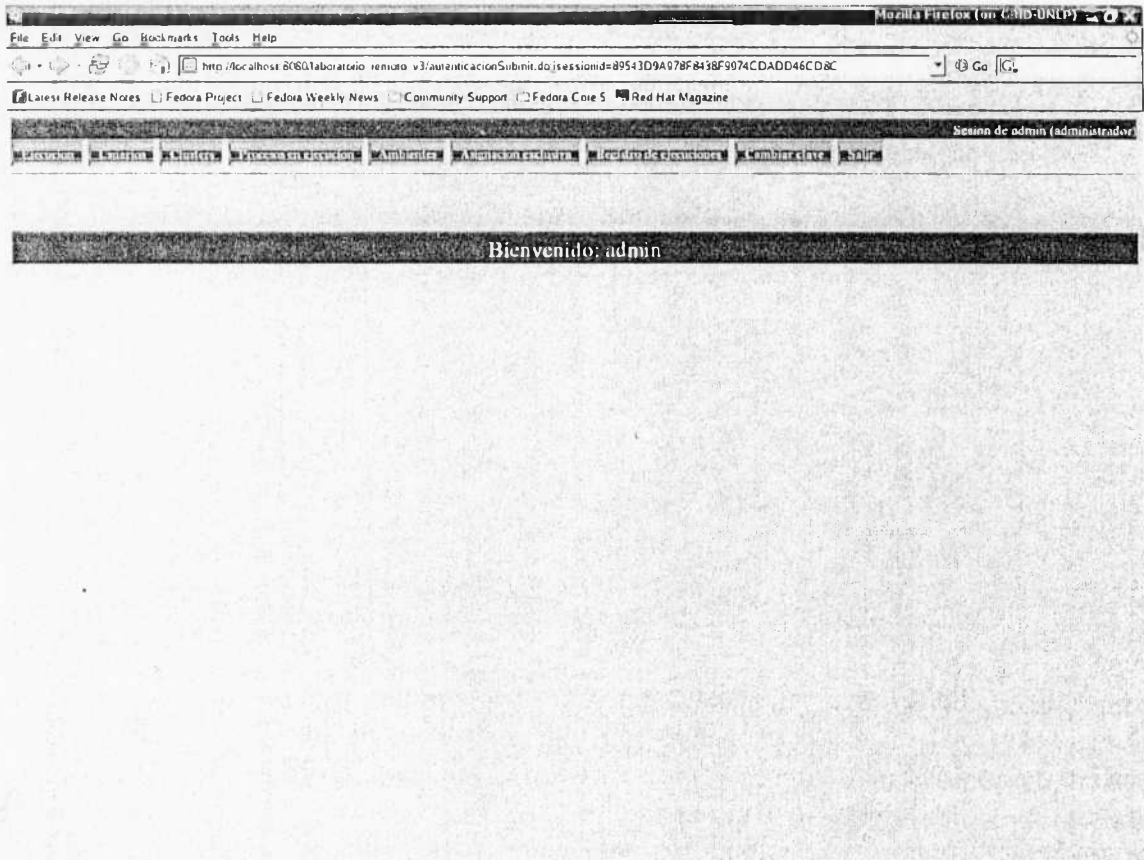
Si el cluster está en modo compartido cualquier usuario podrá ingresar pero no tendrá permitido ejecutar ningún programa si el cluster esta ocupado procesando alguna ejecución.

A2. Pantalla principal

Luego que el usuario se autentique en el sistema, la pantalla principal dependerá del perfil del usuario, por lo tanto existen dos modos:

- Modo Administrador
- Modo Usuario

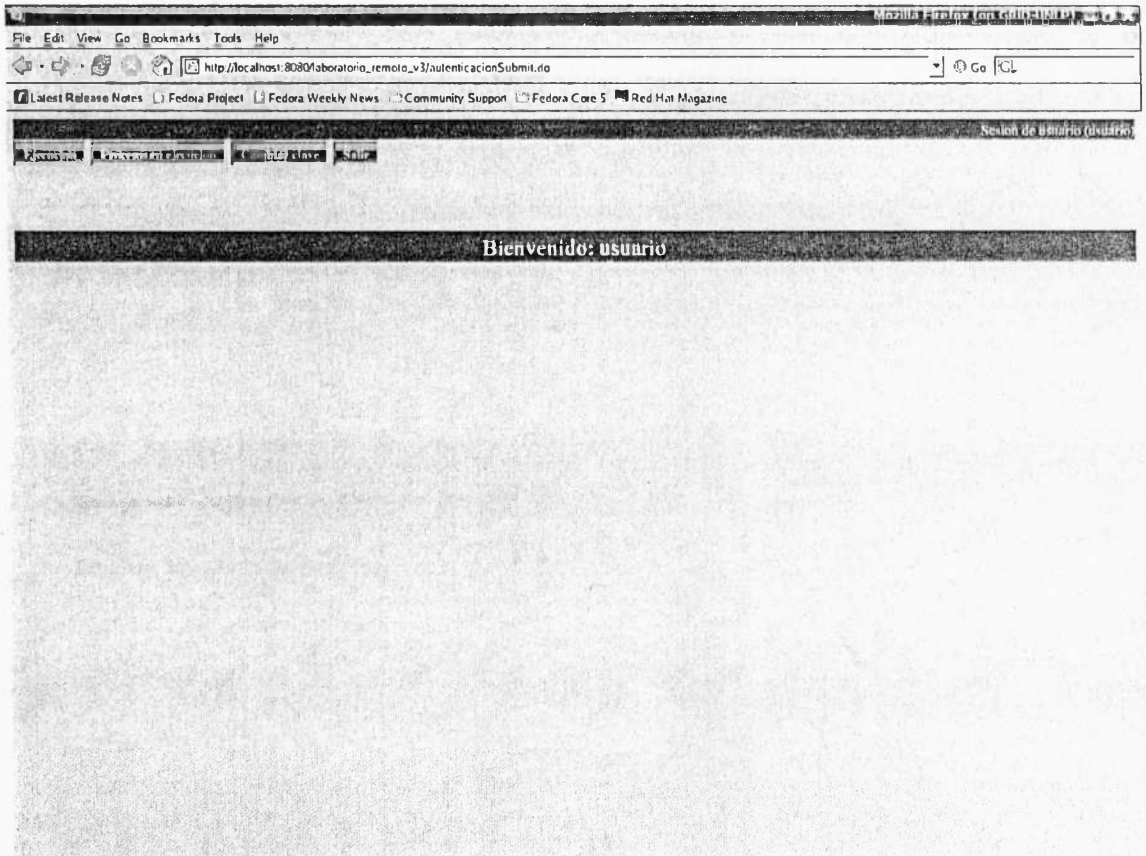
A2.1 Modo Administrador



La pantalla principal para un usuario administrador tiene un menú en la parte superior que le permite acceder a las distintas áreas a saber:

- Área de ejecución
- Área de manejo de usuarios
- Área de manejo de clusters
- Área de procesos en ejecución
- Área de ambientes
- Área de asignaciones exclusivas
- Área de registros de ejecuciones
- Área de cambio de contraseña
- Salir

A2.2 Modo Usuario

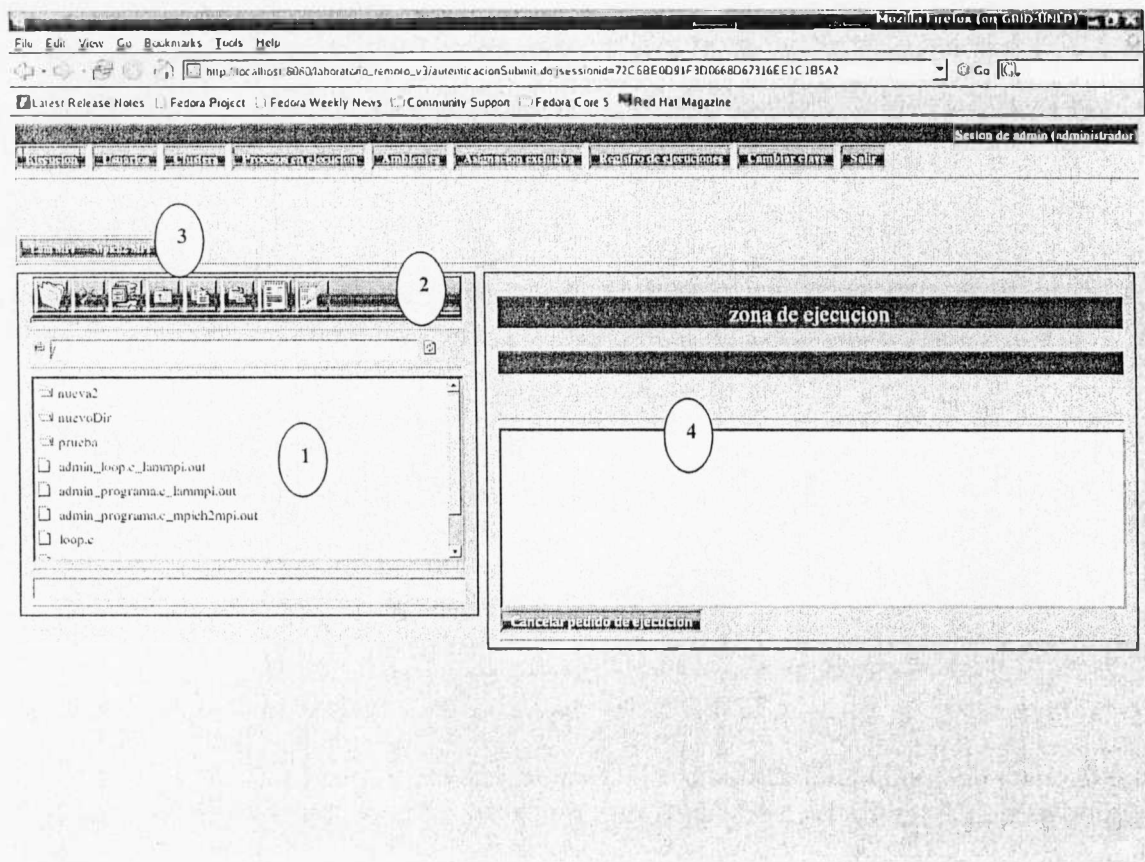


A diferencia del modo administrador, el menú superior en la pantalla principal tiene una versión reducida, las áreas en este caso solo son:

- Área de ejecución
- Área de procesos en ejecución
- Área de cambio de contraseña
- Salir

A3. Área de ejecución

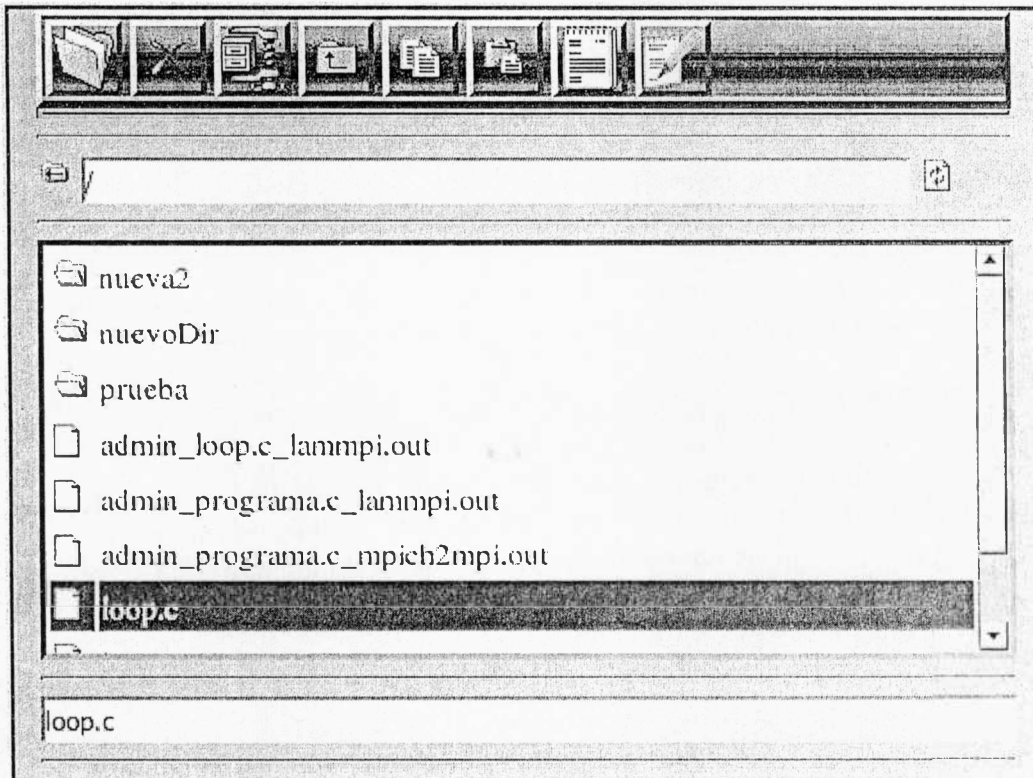
Esta área tiene un espacio de trabajo que permite al usuario administrar archivos y directorios, además compilar y ejecutar programas en los ambientes Lam y Mpich2 MPI.



- 1- Espacio de trabajo: es un área de pantalla que muestra un sistema de archivos para el usuario logueado.
- 2- Panel del área de trabajo: permite realizar distintas operaciones sobre el área de trabajo, como por ejemplo crear, borrar o modificar carpetas, subir archivos o editarlos, descomprimir archivos zip etc.
- 3- Menú de ambientes de ejecución: permite compilar o ejecutar un archivo seleccionado en el área de trabajo con alguno de los ambientes de ejecución Lam o Mpich2.
- 4- Área de ejecución: es un área de pantalla que permite ver las salidas de las ejecuciones realizadas ya sea en compilación o ejecución de un programa.

A3.1 Espacio de trabajo

El espacio de trabajo es un área de pantalla que permite a un usuario administrar directorios y archivos.



En la parte superior tiene un panel que permite realizar distintas operaciones, entre el panel y el visor se muestra la ruta actual que en el caso del gráfico se indica con / que actualmente está en la raíz, en la parte inferior se muestra el archivo seleccionado actualmente que en este caso es “loop.c”.

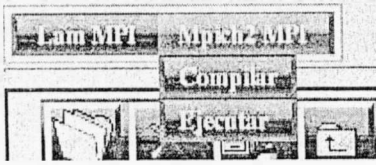
Para acceder a un directorio se hace doble-click sobre el mismo y para volver se hace click en la flecha a la izquierda del campo que muestra la ruta actual, a la derecha de este campo se encuentra un icono que permite recargar el visor.

Según el gráfico, de izquierda a derecha las operaciones que se muestran en el panel son:

- Crear directorio
- Eliminar un archivo o directorio
- Descomprimir archivos .zip que puedan haber subido previamente.
- Subir un archivo: permite subir solo un archivo, pero puede ser un archivo .zip que contenga varias carpetas o archivos y ser descomprimido con la operación anterior.
- Copiar un archivo
- Pegar un archivo, utilizada en conjunto con la operación anterior como el “copiar-pegar clásico”
- Visualizar un archivo, cuyo contenido se verá en el área de ejecución.
- Editar archivo.

A3.2 Menú de ambientes de ejecución

El menú de ambientes de ejecución permite seleccionar la distribución de MPI que se desee utilizar; las dos distribuciones que se pueden utilizar son Lam y Mpich2.



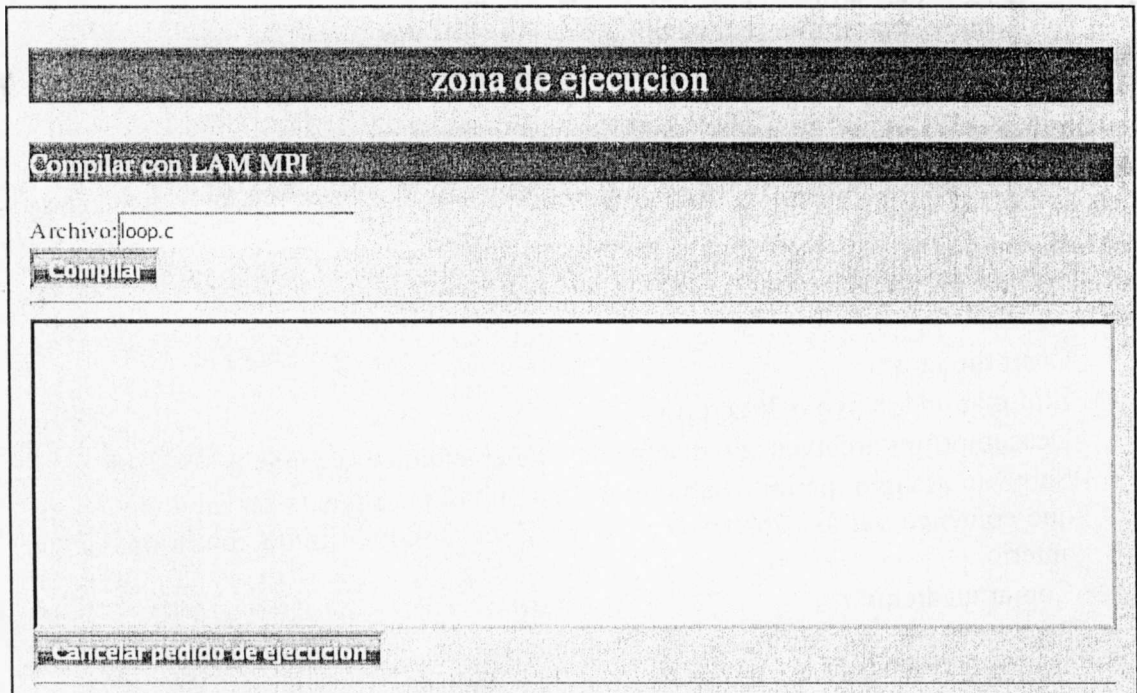
Se puede elegir la operación a realizar sobre el archivo seleccionado en el visor de archivos, y las operaciones pueden ser compilar o ejecutar dicho archivo.

A3.3 Zona de ejecución

La zona de ejecución tiene un visor que muestra los resultados obtenidos de acuerdo a las distintas operaciones realizadas (ya sea una compilación de un archivo fuente, una ejecución de un programa o una visualización de un archivo). De acuerdo a lo que se realice la vista de la zona de ejecución cambiará de estilo.

A3.3.1 Compilación

Al seleccionar en el menú de ambientes de ejecución la opción de compilar ya sea en Lam o en Mpich2, se mostrará lo siguiente en la zona de ejecución:

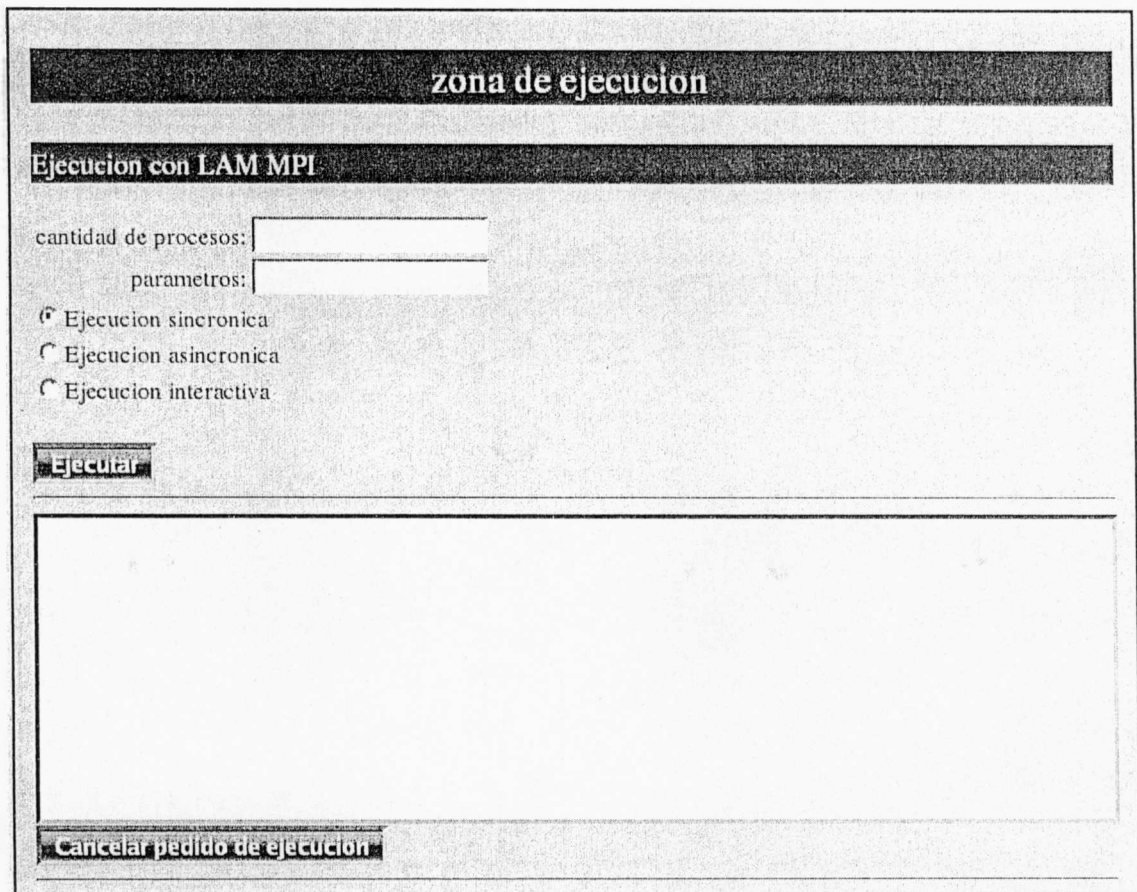


El campo que muestra el archivo indica el archivo seleccionado en el visor de archivos actualmente; será modificado si se selecciona otro archivo en el mismo.

El resultado de la compilación saldrá en el visor sea con éxito o con error.

A3.3.2 Ejecución

Al seleccionar en el menú de ambientes de ejecución la opción de ejecutar ya sea en Lam o en Mpich2, se mostrará lo siguiente en la zona de ejecución:



The screenshot shows a window titled "zona de ejecución". Inside, there is a section "Ejecucion con LAM MPI". Below this, there are two input fields: "cantidad de procesos:" and "parametros:". Underneath these fields are three radio button options: "Ejecucion sincronica" (which is selected), "Ejecucion asincronica", and "Ejecucion interactiva". At the bottom left of the form area is a button labeled "Ejecutar". At the bottom right is a button labeled "Cancelar pedido de ejecución".

Los campos a llenar hacen referencia a la cantidad total de procesos (obligatorio) y los parámetros del programa a ejecutar (puede quedar en blanco).

Luego se permite elegir entre los tres modos de ejecución disponibles:

- Sincrónico: que deja la salida en el visor luego de procesar todo el programa.
- Asíncronico: que deja la salida en un archivo, que se vera al recargar el visor de archivos del espacio de trabajo.
- Interactivo: que deja la salida en el visor pero a diferencia del modo sincrónico no espera a que todo el programa sea procesado sino que va dejando de a líneas a medida que el proceso se ejecute y vaya dejando líneas en la salida.

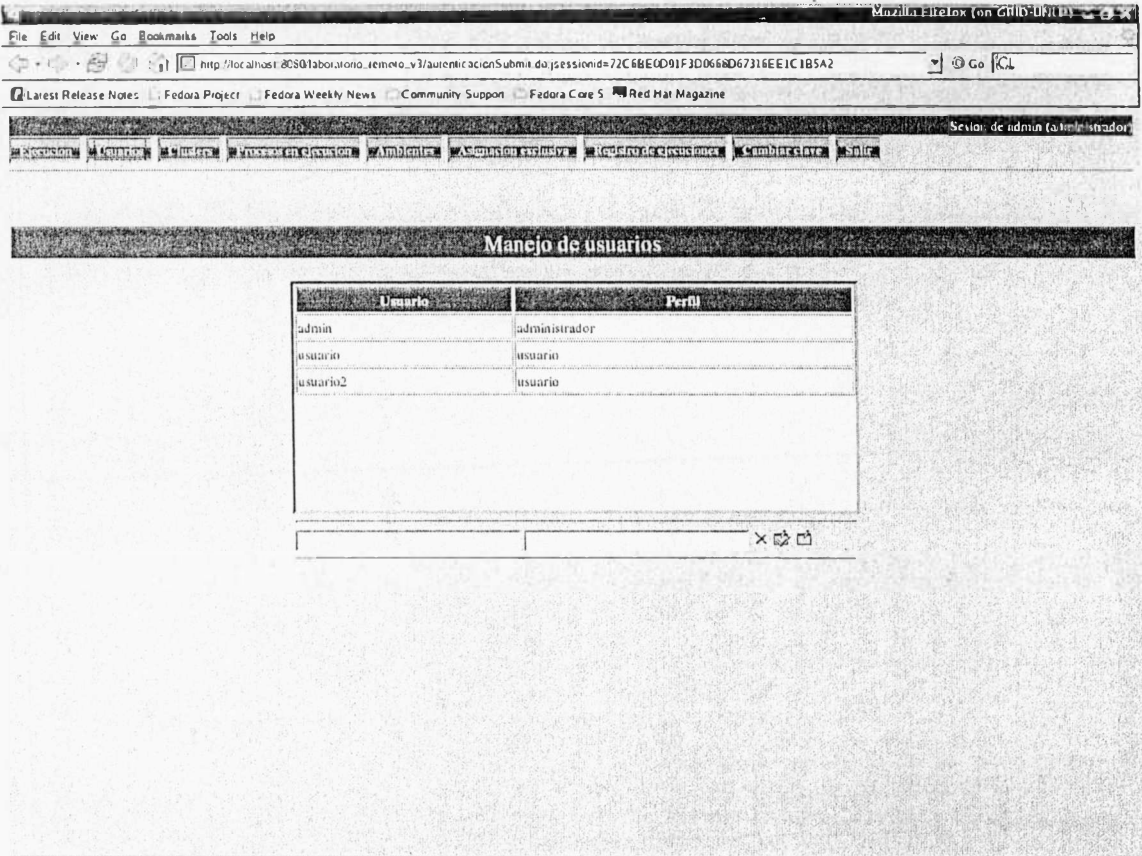
En el caso del cluster compartido, toda ejecución será realizada si el cluster esta desocupado, en caso contrario se reintentara la ejecución; este pedido de ejecución puede ser cancelado.

A3.3.3 Visualización

Cualquier archivo que se quiera visualizar, inclusive salidas de ejecución dejadas en archivos debido a una ejecución asíncronica, pueden verse en el visor del área de ejecución.

A4. Área de manejo de usuarios

Esta área permite administrar los usuarios del sistema: visualizar, agregar, modificar y eliminar usuarios.



Se tiene un visor en el cual se muestran los usuarios y el perfil de cada uno; en la parte inferior existe un panel el cual permite dar alta a un usuario, modificar un usuario seleccionado o bien eliminarlo.



A5. Área de manejo de clusters

Esta área permite administrar todo lo relacionado con los clusters.

Manejo de clusters

Modos de acceso a arquitectura

Exclusivo 1

Compartido

Clusters

| Cluster | Descripción |
|-----------|--------------------------------|
| ClusterP3 | Cluster formado por pentium 3. |
| Cluster2 | Cluster de prueba numero 2. |

2

Máquinas

Todas las máquinas Solo el cluster seleccionado

| IP | Nombre | Estado |
|------------|-------------|----------------------------------|
| 10.10.0.72 | maquina 72. | <input type="radio"/> |
| 10.10.0.73 | maquina 73. | <input type="radio"/> |
| 10.10.0.74 | maquina 74. | <input type="radio"/> |
| 10.10.0.75 | maquina 75. | <input type="radio"/> |
| 10.10.0.76 | maquina 76. | <input type="radio"/> |
| 10.10.0.77 | maquina 77. | <input type="radio"/> |
| 10.10.0.78 | maquina 78. | <input checked="" type="radio"/> |

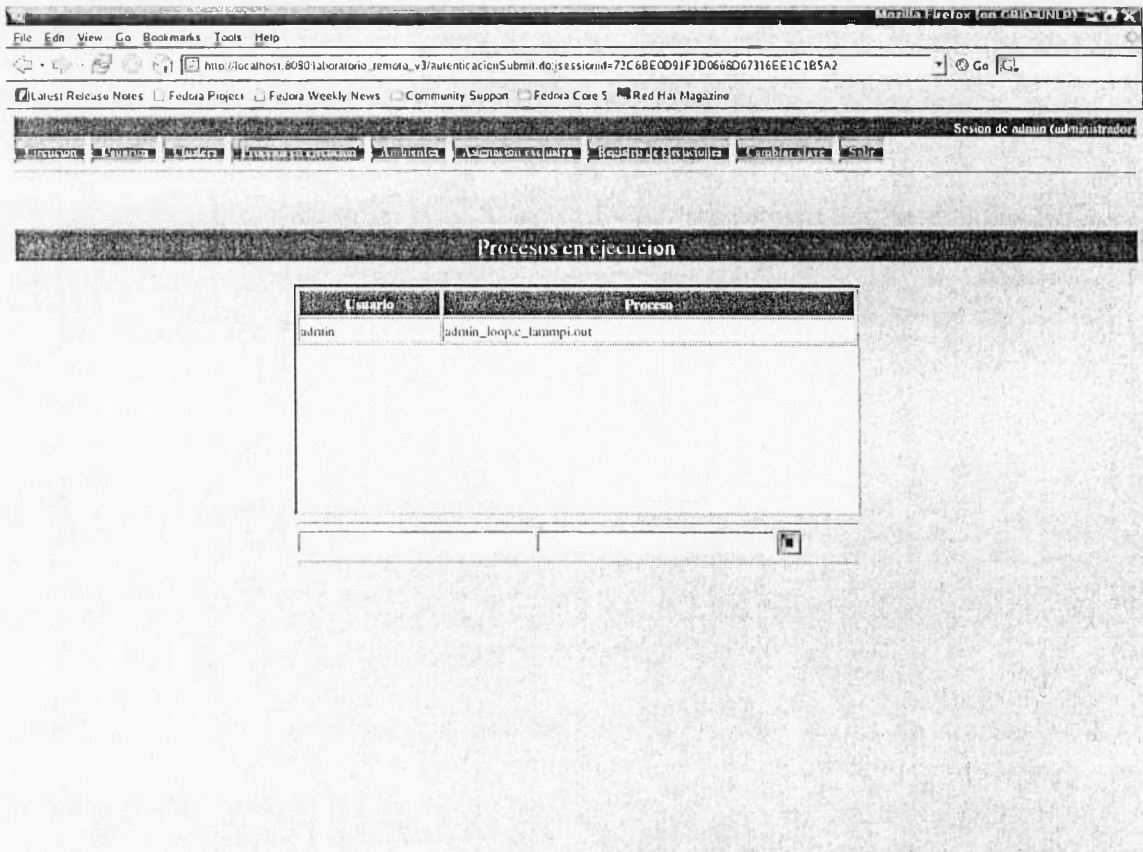
3

- 1- Cambios de modo de acceso a la arquitectura: permite a un administrador cambiar el modo en que se encuentra el cluster, puede ser exclusivo y compartido.
- 2- Vista de clusters: permite visualizar los clusters en el sistema, además de agregar, modificar o eliminar clusters.
- 3- Vista de máquinas: muestra las máquinas que forman parte de los clusters; incluye además el estado en que se encuentra, si es en verde está encendida, si es en rojo está apagada. Pueden visualizarse todas las máquinas o solo las máquinas del cluster seleccionado en la vista de clusters.
- 4- En caso de mostrar las máquinas pertenecientes a un cluster en particular, se incluye la funcionalidad de agregar, modificar o eliminar máquinas.

A6. Área de procesos en ejecución

Este área muestra los procesos que están en ejecución; para el caso de los usuarios comunes sólo pueden visualizar los procesos propios, en cambio, en el caso de administradores puede ver no sólo los propios sino también los procesos en ejecución de cualquier otro usuario.

Existe la posibilidad de detener cualquier proceso en ejecución seleccionando el mismo y pulsando el botón stop.



Existe un visor que muestra los procesos y en la parte inferior los dos campos sirven para mostrar el proceso seleccionado, a la derecha de estos campos está el botón stop que permite detener el proceso seleccionado.

A7. Área de ambientes de ejecución MPI

Este área permite ver el estado en que se encuentran los daemons para los ambientes MPI, tanto Lam como Mpich2; además de esto permite arrancar o detener estos ambientes.

| Ambiente | Comenzar | Detener | Estado |
|------------|----------------------------------|----------------------------------|--------|
| Lam MPI | <input type="button" value="▶"/> | <input type="button" value="■"/> | ● |
| Mpich2 MPI | <input type="button" value="▶"/> | <input type="button" value="■"/> | ● |

A la izquierda se muestran los ambientes en forma de tabla, el campo estado muestra el estado actual de los mismos; si es verde estarán ejecutándose y si están en rojo estarán detenidos.

Para arrancar algún daemon se debe presionar el botón del campo comenzar y para detenerlo el botón del campo detener.

El resultado de comenzar o detener el daemon se visualizará en el visor que se encuentra a la derecha de la pantalla.

A8. Área de asignaciones exclusivas

Como se mencionó anteriormente, el cluster puede estar en dos modos, uno compartido y otro exclusivo; en este ultimo caso el usuario que quiera acceder tendrá que tener asignado un periodo de tiempo.

En este área se asignan los periodos de tiempo a los usuarios para que accedan al cluster exclusivamente.

Manejo de asignaciones exclusivas

| Usuario | Fecha y hora de inicio | Fecha y hora de fin |
|---------|------------------------|-----------------------|
| usuario | 2007-02-01 10:25:00.0 | 2007-02-28 10:25:00.0 |

Fecha y hora de inicio:

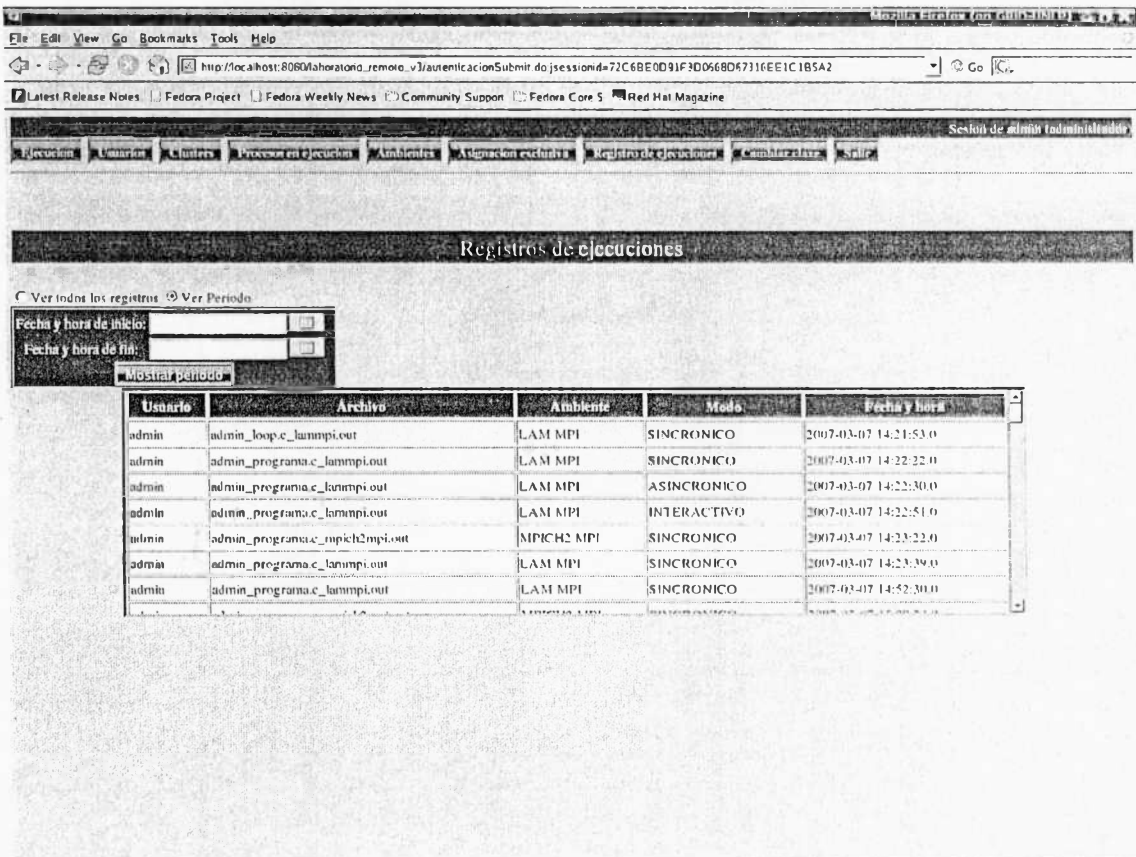
Fecha y hora de fin:

A la izquierda se visualizan los usuarios del sistema; al seleccionar un usuario se visualiza a la derecha los periodos en que el usuario seleccionado tiene asignado el cluster de forma exclusiva.

En el panel de la derecha además se permite agregar, modificar o eliminar asignaciones al usuario seleccionado.

A9. Área de registro de ejecuciones

Este área permite ver el registro de todas las ejecuciones realizadas para llevar un control.



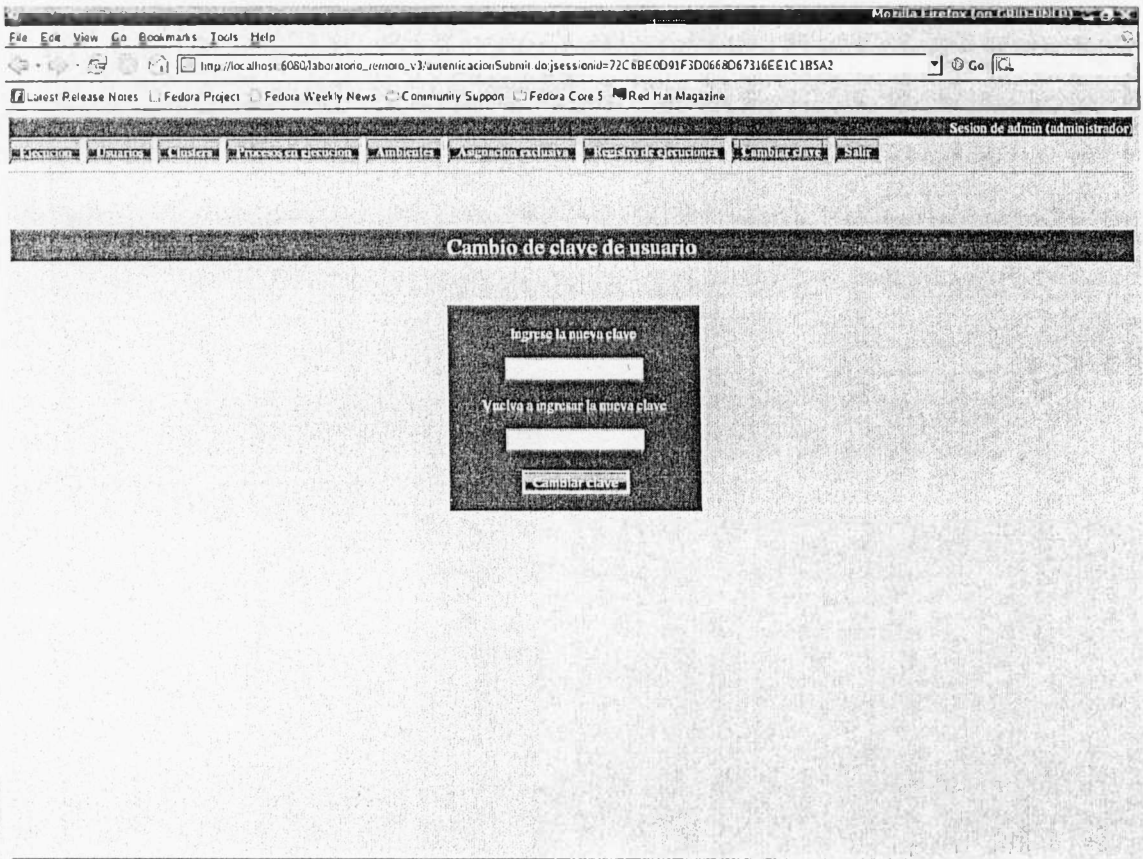
The screenshot shows a web browser window with the URL `http://ocahost:8080/aheratonc_remoto_v3/autenticacionSubmit.do?sessionId=72C6BE0D91F3D0668D67316EE1C185A2`. The page title is "Registros de ejecuciones". Below the title, there are search filters for "Fecha y hora de inicio" and "Fecha y hora de fin", and a "Mostrar periodo" button. The main content is a table with the following data:

| Usuario | Archivo | Ambiente | Modo | Fecha y hora |
|---------|--------------------------------|------------|-------------|-----------------------|
| admin | admin_loop_e_lammpi.out | LAM MPI | SINCRONICO | 2007-03-07 14:21:53.0 |
| admin | admin_programa_e_lammpi.out | LAM MPI | SINCRONICO | 2007-03-07 14:22:22.0 |
| admin | admin_programa_e_lammpi.out | LAM MPI | ASINCRONICO | 2007-03-07 14:22:30.0 |
| admin | admin_programa_e_lammpi.out | LAM MPI | INTERACTIVO | 2007-03-07 14:22:51.0 |
| admin | admin_programa_e_mpich2mpi.out | MPICH2 MPI | SINCRONICO | 2007-03-07 14:23:22.0 |
| admin | admin_programa_e_lammpi.out | LAM MPI | SINCRONICO | 2007-03-07 14:23:39.0 |
| admin | admin_programa_e_lammpi.out | LAM MPI | SINCRONICO | 2007-03-07 14:52:30.0 |

Se permiten ver todos los registros o bien solo un periodo de tiempo determinado; cada registro muestra el usuario que realizó la ejecución, el programa que ejecutó, el ambiente en que lo hizo (Lam o Mpich2), el modo de ejecución que utilizó (sincrónico, asincrónico o interactivo) y la fecha y la hora en que fue realizada dicha ejecución.

A10. Área de cambio de contraseña

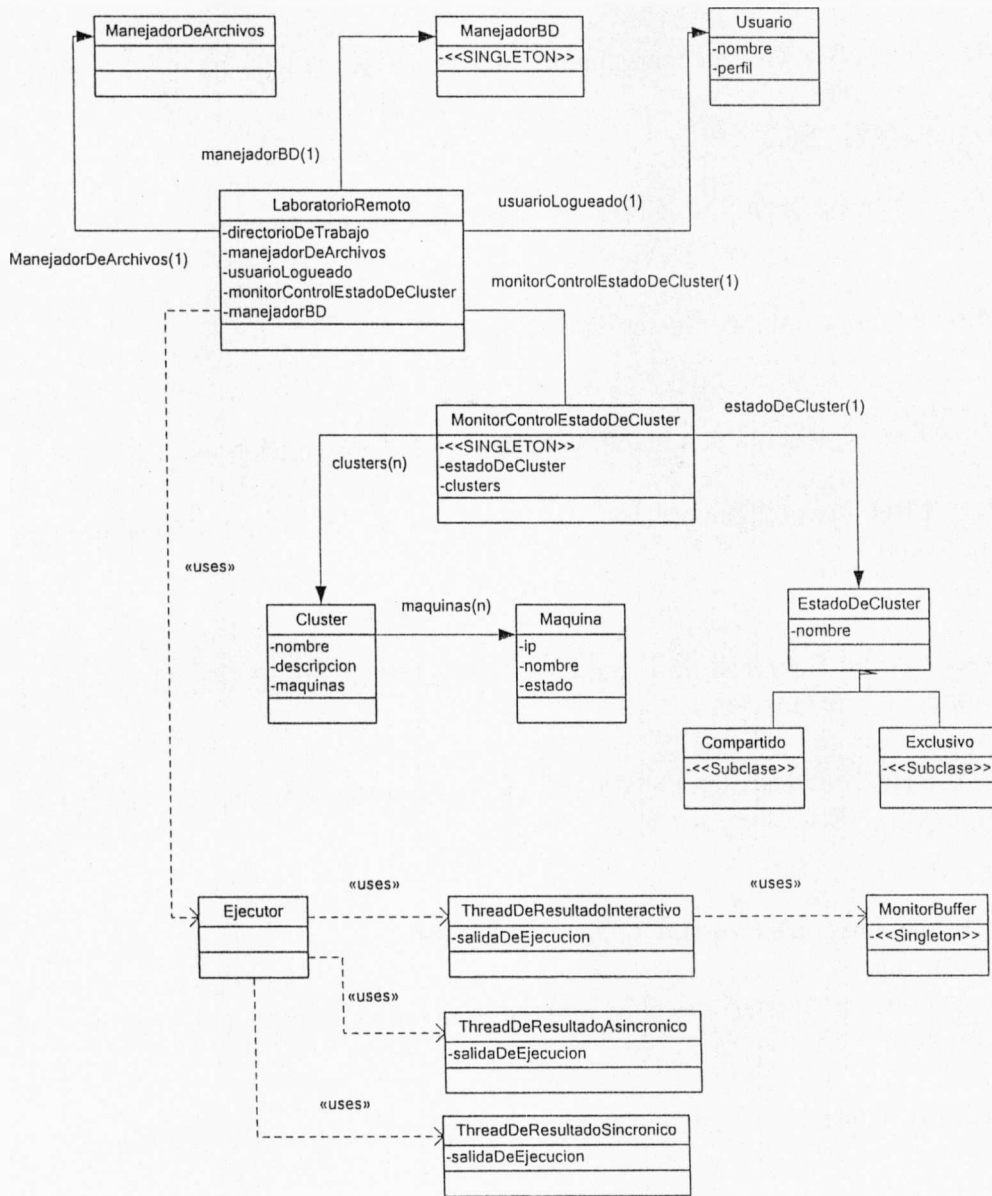
Este área permite a un usuario cambiar su contraseña.



Anexo B

Documentación y Diagramas

B1. Diagrama de clases



B2. Descripción de los objetos

- **LaboratorioRemoto**: clase principal que representa el modelo de la aplicación.
- **Usuario**: representa un usuario de la aplicación.
- **ManejadorBD**: Clase encargada de interactuar con la base de datos, es única en todo el sistema.
- **ManejadorDeArchivos**: Clase encargada de lo relacionado con el manejo de archivos.
- **Cluster**: representa un cluster en el sistema.
- **Maquina**: representa una máquina perteneciente a un cluster en el sistema.
- **EstadoDeCluster**: Superclase que engloba a los estados posibles de la arquitectura.
- **Compartido**: Subclase de EstadoDeCluster, representa un estado posible de la arquitectura de cluster.
- **Exclusivo**: Subclase de EstadoDeCluster, representa un estado posible de la arquitectura de cluster.
- **MonitorControlEstadoDeCluster**: Clase única en el sistema, conocida por cada instancia de la clase LaboratorioRemoto, encargada de controlar la arquitectura de cluster y excluir mutuamente dicha arquitectura, funciona como un monitor.
- **Ejecutor**: Clase encargada de ejecutar programas o comandos de sistema.
- **ThreadDeResultadoSincronico**: Utilizada por el Ejecutor para la ejecución sincrónica.
- **ThreadDeResultadoAsincronico**: Utilizada por el Ejecutor para la ejecución asincrónica.
- **ThreadDeResultadoInteractivo**: Utilizada por el Ejecutor para la ejecución interactiva.
- **MonitorBuffer**: Utilizada por las clases ThreadDeResultadoInteractivo.

B3. Descripción de casos de uso

Actores

Usuarios
Administradores

B3.1 Descripción básica de casos de usos

Casos de usos de usuarios

Ingresar al sistema
Utilizar área de ejecución
 Utilizar espacio de trabajo
 Navegar estructura de archivos
 Crear directorio
 Eliminar archivo o directorio
 Subir un archivo
 Descomprimir un archivo .zip
 Visualizar archivo
 Editar archivo
 Copiar archivo
 Pegar archivo
 Realizar compilaciones
 Compilar en Lam MPI
 Compilar en Mpich2 MPI
 Realizar ejecuciones
 Ejecutar en Lam MPI
 Ejecutar sincrónicamente
 Ejecutar asincrónicamente
 Ejecutar Interactivamente
 Ejecutar en Mpich2 MPI
 Ejecutar sincrónicamente
 Ejecutar asincrónicamente
 Ejecutar Interactivamente
 Cancelar reintentos de ejecuciones
Detener programas en ejecución
Cambiar contraseña
Salir

Casos de usos de administradores

Ingresar al sistema

Utilizar área de ejecución (ídem usuario)

Utilizar área de clusters

Establecer acceso compartido o exclusivo

Realizar alta de cluster

Realizar baja de cluster

Realizar modificación de cluster

Realizar alta de maquina en un cluster

Realizar baja de maquina en un cluster

Realizar modificación de maquina en un cluster

Utilizar área de usuarios

Realizar alta de usuario

Realizar baja de usuario

Realizar modificación de usuario

Utilizar área de ambientes

Arrancar Lam mpi

Detener Lam mpi

Arrancar Mpich2 mpi

Detener Mpich2 mpi

Utilizar área de registro de ejecuciones

Utilizar área de asignaciones de accesos exclusivos

Realizar alta de asignación exclusiva a usuario

Realizar baja de asignación exclusiva a usuario

Realizar modificación de asignación exclusiva a usuario

Detener programas propios o ajenos en ejecución

Cambiar contraseña (ídem usuario)

Salir (ídem usuario)

B3.2 Descripción detallada de casos de usos particulares

Caso de uso: Ingresar al sistema

Actores: Usuario

Propósito: Un usuario intenta ingresar al sistema.

Visión general: Un usuario ingresa su nombre y contraseña e intenta ingresar al sistema. El sistema le permite el acceso solo si es un usuario válido y además el cluster está en estado compartido o en caso de estar exclusivo el usuario tiene una asignación exclusiva.

Curso de eventos:

1. El usuario ingresa su nombre de usuario.
2. El usuario ingresa su contraseña.
3. Da curso a la solicitud de ingreso.

Cursos alternativos:

Línea 3: El usuario ingresado o la contraseña es inválida, no se permite el ingreso, se indica el error y se vuelve a pedir usuario y contraseña.

Línea 3: El cluster está en estado exclusivo pero el usuario no tiene una asignación exclusiva al cluster, no se permite el ingreso y se indica el error.

Caso de uso: Ingresar al sistema

Actores: Administrador

Propósito: Un administrador intenta ingresar al sistema.

Visión general: Un administrador ingresa su nombre y contraseña e intenta ingresar al sistema. El sistema sólo permite el acceso si es un usuario válido. De serlo siempre permite su acceso independientemente del estado del cluster por tener un perfil de administrador.

Curso de eventos:

1. El usuario ingresa su nombre de usuario.
2. El usuario ingresa su contraseña.
3. Da curso a la solicitud de ingreso.

Cursos alternativos:

Línea 3: El usuario ingresado o la contraseña es inválida, no se permite el ingreso, se indica el error y se vuelve a pedir usuario y contraseña.

Caso de uso: Compilar en Lam MPI

Actores: Usuario, Administrador

Propósito: Compilar un archivo con la librería Lam MPI.

Visión general: Un usuario o administrador selecciona la opción de compilar en Lam MPI, luego en el visor de archivos selecciona un archivo fuente escrito en lenguaje C, y envía a compilar; solo se permitirá compilar si el archivo tiene extensión .c. El sistema retorna el resultado de la compilación.

Curso de eventos:

1. Seleccionar la opción de compilar en Lam MPI.
2. Seleccionar un archivo fuente .c en el visor de archivos.
3. Enviar a compilar.
4. El sistema retorna el resultado de la compilación informando el código de error 0.
5. El sistema deja el archivo compilado con extensión .out.

Cursos alternativos:

Línea 3: No existe un archivo seleccionado como indica la línea 2, no se envía la petición y se indica el error.

Línea 3: El archivo seleccionado no tiene extensión .c, no se envía la petición y se indica el error.

Línea 4: La compilación da un error, se informa la salida y la línea 5 no se cumple.

Caso de uso: Ejecutar sincrónicamente en Lam MPI

Actores: Usuario, Administrador

Propósito: Ejecutar un archivo con la librería Lam MPI de forma sincrónica.

Visión general: Un usuario o administrador selecciona la opción de ejecutar en Lam MPI, luego en el visor de archivos selecciona un archivo con extensión out previamente compilado con Lam MPI, ingresa la cantidad de procesos y los parámetros de programa, selecciona el modo sincrónico y envía a ejecutar, sólo se permitirá ejecutar si el archivo tiene extensión .out y si el cluster no está siendo usado. El sistema retorna el resultado completo de la ejecución en pantalla.

Curso de eventos:

1. Seleccionar la opción de ejecutar en Lam MPI.
2. Seleccionar un archivo con extensión .out en el visor de archivos previamente compilado con Lam MPI.
3. Seleccionar la opción sincrónica.
4. Enviar a ejecutar.
5. El sistema retorna el resultado completo de la ejecución en pantalla sea exitoso (exitcode 0) o erróneo.

Cursos alternativos:

Línea 4: No existe un archivo seleccionado como indica la línea 2, no se envía la petición y se indica el error.

Línea 4: El archivo seleccionado no tiene extensión .out, no se envía la petición y se indica el error.

Línea 4: La ejecución da un error debido a que no está corriendo el daemon de Lam MPI, se debe avisar a un administrador para que lo arranque.

Línea 5: Se informa que el cluster está ocupado, el sistema automáticamente reintenta la ejecución, el usuario puede detener este reintento cancelando el pedido de ejecución.

Caso de uso: Ejecutar interactivamente en Lam MPI

Actores: Usuario, Administrador

Propósito: Ejecutar un archivo con la librería Lam MPI de forma interactiva.

Visión general: Un usuario o administrador selecciona la opción de ejecutar en Lam MPI, luego en el visor de archivos selecciona un archivo con extensión out previamente compilado con Lam MPI, ingresa la cantidad de procesos y los parámetros de programa, selecciona el modo interactivo y envía a ejecutar; sólo se permitirá ejecutar si el archivo tiene extensión .out y si el cluster no está siendo usado. El sistema va retornando en pantalla el resultado de la ejecución línea por línea a medida que se va realizando.

Curso de eventos:

1. Seleccionar la opción de ejecutar en Lam MPI.
2. Seleccionar un archivo con extensión .out en el visor de archivos previamente compilado con Lam MPI.
3. Seleccionar la opción sincrónica.
4. Enviar a ejecutar.

5. El sistema retorna el resultado de la ejecución en pantalla sea exitoso (exitcode 0) o erróneo a medida que la ejecución va llevándose a cabo.

6. El sistema indica cuando el programa termino su ejecución.

Cursos alternativos:

Línea 4: No existe un archivo seleccionado como indica la línea 2, no se envía la petición y se indica el error.

Línea 4: El archivo seleccionado no tiene extensión .out, no se envía la petición y se indica el error.

Línea 4: La ejecución da un error debido a que no esta corriendo el daemon de Lam MPI, se debe avisar a un administrador para que lo arranque.

Línea 5: Se informa que el cluster está ocupado, el sistema automáticamente reintenta la ejecución, el usuario puede detener este reintento cancelando el pedido de ejecución.

Línea 6: Puede no aparecer la indicación de finalización de programa y ya no se escriben más líneas de la salida de programa en pantalla, puede ser un alerta que el programa se “colgó”, la forma interactiva da al usuario la posibilidad de entender una situación como esta, pudiendo detener el programa.

Caso de uso: Ejecutar asincrónicamente en Lam MPI

Actores: Usuario, Administrador

Propósito: Ejecutar un archivo con la librería Lam MPI de forma asincrónica.

Visión general: Un usuario o administrador selecciona la opción de ejecutar en Lam MPI, luego en el visor de archivos selecciona un archivo con extensión out previamente compilado con Lam MPI, ingresa la cantidad de procesos y los parámetros de programa, selecciona el modo asincrónico y envía a ejecutar; solo se permitirá ejecutar si el archivo tiene extensión .out y si el cluster no está siendo usado. El sistema retorna el resultado completo de la ejecución en un archivo con extensión .txt.

Curso de eventos:

1. Seleccionar la opción de ejecutar en Lam MPI.

2. Seleccionar un archivo con extensión .out en el visor de archivos previamente compilado con Lam MPI.

3. Seleccionar la opción sincrónica.

4. Enviar a ejecutar.

5. El sistema retorna el resultado completo de la ejecución en un archivo sea exitoso (exitcode 0) o erróneo.

6. Pasado un tiempo razonable de acuerdo a la ejecución del programa se puede actualizar el visor para ver el archivo resultado.

Cursos alternativos:

Línea 4: No existe un archivo seleccionado como indica la línea 2, no se envía la petición y se indica el error.

Línea 4: El archivo seleccionado no tiene extensión .out, no se envía la petición y se indica el error.

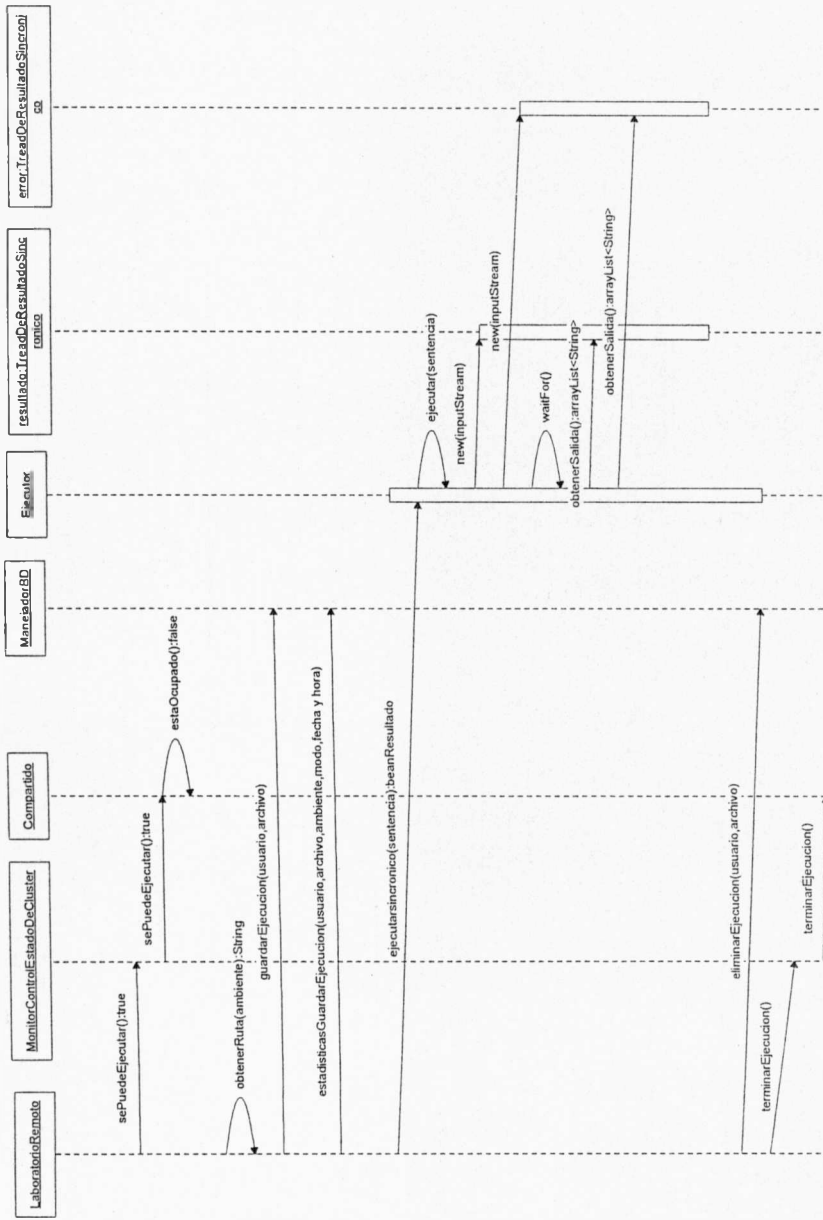
Línea 4: La ejecución da un error debido a que no esta corriendo el daemon de Lam MPI, se debe avisar a un administrador para que lo arranque.

Línea 5: Se informa que el cluster esta ocupado, el sistema automáticamente reintenta la ejecución, el usuario puede detener este reintento cancelando el pedido de ejecución.

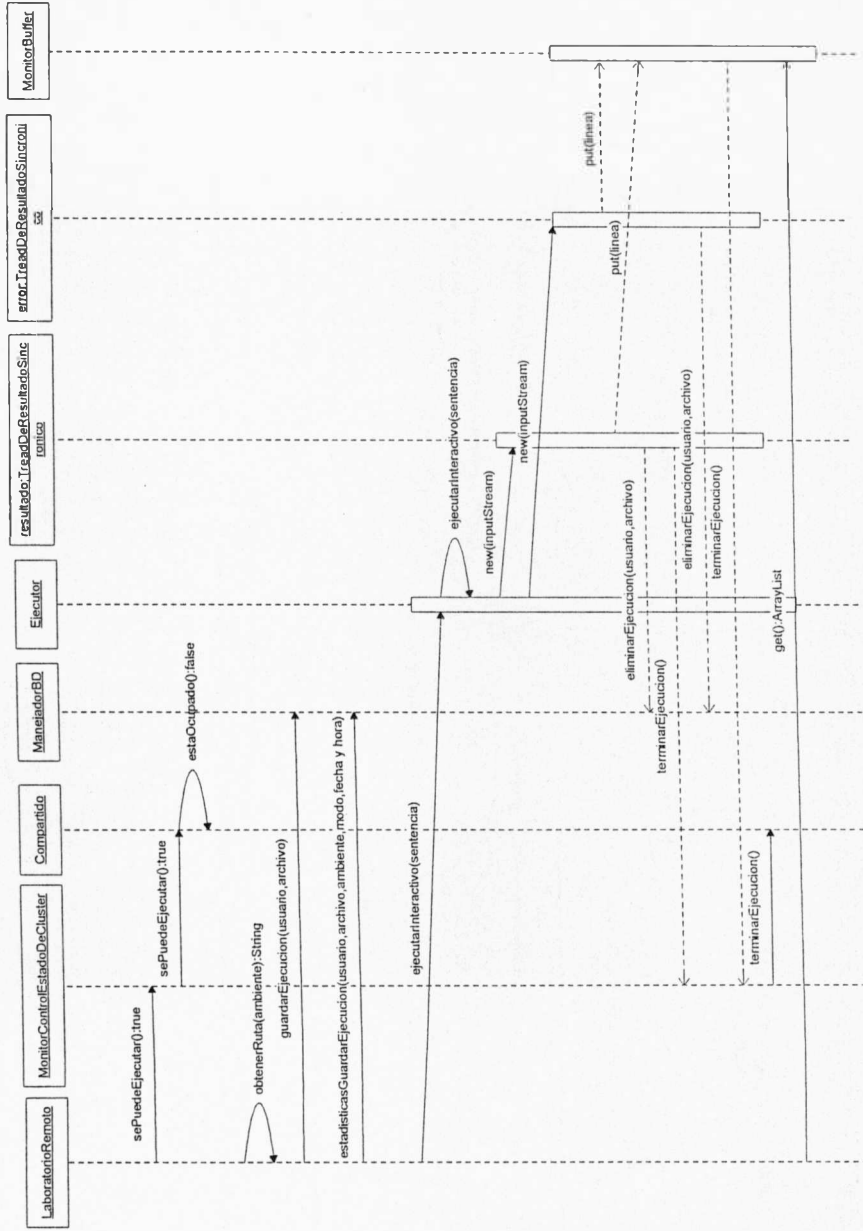
Línea 6: Si pasado un tiempo razonable el visor no muestra el archivo puede que todavía el programa no haya terminado o esté “colgado”, es posible detenerlo.

B4. Diagramas de interacción

Ejecucion sincronica, estado del cluster compartido, no ocupado



Ejecucion interactiva, estado del cluster compartido, no ocupado



Autenticación en estado exclusivo (pattern state)



Autenticación en estado compartido (pattern state)



B6. Tablas resultantes

- **usuarios:** Almacena los usuarios del sistema

usuarios(usuario, clave, idPerfil)

- **perfiles:** Almacena los perfiles de los usuarios del sistema

perfiles(id, nombrePerfil)

- **clusters:** Almacena los clusters del sistema

clusters(id, nombre, descripción)

- **maquinas:** Almacena los equipos pertenecientes a los clusters del sistema

maquinas(ip, nombre, idCluster)

- **ejecución:** Almacena registros de que archivos está corriendo un determinado usuario, una vez terminada la ejecución del archivo se elimina el registro

ejecución(usuario, archivo)

- **estadisticaEjecuciones:** Se utiliza para mantener una estadística de las ejecuciones, almacena registros que indican que archivos corrió un determinado usuario en una fecha y hora determinada, sobre que ambiente de ejecución y en que modo lo hizo.

estadisticaEjecuciones(usuario, archivo, ambiente, modo, fechaHora)

- **rutas:** Se utiliza para almacenar las rutas de los distintos comandos o procesos que utiliza la aplicación para interactuar con el sistema operativo dentro del sistema de archivos.

rutas(descripción, ruta)

- **estadoCluster:** Se utiliza para almacenar el estado actual de la arquitectura de cluster, pudiendo ser este compartido o exclusivo.

estadoCluster(id, estado)

- **accesosExclusivos:** Almacena los periodos de tiempo en fecha y hora en que los usuarios tienen un acceso exclusivo a la arquitectura de cluster.

accesosExclusivos(usuario, fechaHoraInicio, fechaHoraFin)

Anexo C

Herramientas utilizadas durante el desarrollo

C1. Introducción

Fue importante el uso de herramientas de distribución libre y gratuita, muchas de estas son de código abierto y además, existe una fuente importante de información en Internet a manera de soporte técnico.

Estas herramientas van desde el sistema operativo, herramientas de desarrollo y herramientas para ejecución en clusters.

C2. Sistema operativo

El sistema operativo utilizado tanto para el servidor como para la mayoría de los equipos del cluster es una distribución Linux llamada **Fedora core 5**.

El proyecto Fedora es un proyecto de código abierto patrocinado por Red Hat y soportado por la comunidad. Su objetivo es trabajar con la comunidad Linux para construir un sistema operativo completo de propósito general, exclusivamente a partir de software libre.

Puede obtenerse más información en los siguientes sitios:

- en Español <http://www.redhat.es/fedora/>
- en Ingles <http://www.redhat.com/fedora/>

Para otras máquinas del cluster fue configurada la distribución de Linux **Ubuntu 6.10**. Los desarrolladores de Ubuntu se basan en gran medida en el trabajo de las comunidades de Debian y GNOME.

Es patrocinado por Canonical Ltd, una empresa privada fundada y financiada por el empresario sudafricano Mark Shuttleworth.

Puede obtenerse más información en los siguientes sitios:

- en Español <http://www.ubuntu-es.org>
- en Ingles <http://www.ubuntu.com>

C3. Lenguaje de programación

El lenguaje de programación elegido para el desarrollo del sistema fue **JAVA**.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los 90. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es interpretado por una maquina virtual JAVA lo que le permite ser independiente de la plataforma. La versión utilizada es la J2SE 1.5.

JAVA presenta además lo que se conoce como J2EE que es una plataforma de programación (parte de la plataforma JAVA) para desarrollar y ejecutar software de aplicaciones en lenguajes de programación con arquitectura de n niveles distribuida.

Una herramienta muy conocida dentro de la plataforma J2EE es el uso de **SERVLETS**, “programa” que se ejecuta en un servidor; este servidor puede ser un servidor Web con la particularidad de tener un contenedor de aplicaciones (Ej.: Tomcat) o bien puede ser un servidor de aplicaciones (Ej.: OC4J Oracle, Sun Java Sistema Application Server).

Otra herramienta muy usada en esta plataforma es la tecnología **JSP** (Java Server Pages).

JSP es una tecnología Java que permite a los programadores generar contenido dinámico para Web, en forma de documentos HTML, XML, o de otro tipo. Las JSP's permite al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento Web.

En las **JSP** se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente.

Puede obtenerse más información en el siguiente sitio:

<http://java.sun.com/javase/downloads/index.jsp>

C4. Servidor Web y contenedor de aplicaciones

El servidor Web elegido fue **Apache Tomcat**.

Tomcat (también llamado **Jakarta Tomcat** o **Apache Tomcat**) funciona como un servidor Web con contenedor de servlets desarrollado bajo el proyecto jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystem.

Tomcat puede funcionar como servidor Web por sí mismo, es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en JAVA, funciona en cualquier sistema operativo que disponga de la maquina virtual JAVA.

Puede obtenerse más información en el siguiente sitio:

<http://tomcat.apache.org/>

C5. Ambiente de desarrollo

El ambiente de desarrollo JAVA puede dividirse en las herramientas para compilar/ejecutar aplicaciones JAVA y el ambiente grafico que permite trabajar de un modo mas amigable e interactivo.

Las herramientas para compilar/ejecutar aplicaciones JAVA son provistas por Sun Microsystem y se le denomina **JDK** (Java Development Kit), la versión utilizada es la JDK 1.5 que puede ser descargada del sitio del lenguaje de programación JAVA mencionado anteriormente.

Existen muchos ambientes gráficos para desarrollo en JAVA conocidos en ingles como IDE (Entorno integrado de desarrollo); se pueden mencionar algunos como Jdeveloper, JBuilder, eclipse etc., muchos gratuitos y otros no tanto.

El ambiente elegido fue **Eclipse** que es una plataforma de software de código abierto independiente de la plataforma, típicamente ha sido usada para desarrollar un entorno integrado de desarrollo, como el IDE de JAVA llamado Java Development Toolkit (JDT) y el compilador (ECJ) que es parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones.

Eclipse en si mismo no permite trabajar con servlets, para poder hacerlo se debe incluir plugins para desarrollo con servlets, los cuales hay varios.

En particular se usó una distribución de Eclipse llamada **LOMBOZ**, que es un plugin gratuito y abierto de Eclipse para el entorno de desarrollo J2EE; del sitio de Lomboz se pueden descargar gratuitamente tanto el plugin como una versión de eclipse configurada con este plugin incluido para trabajar con todo tipo de aplicaciones distribuidas. Puede obtenerse más información en el siguiente sitio:

<http://www.objectlearn.com/index.jsp>

Fue importante también trabajar con el patrón MVC (Model View Controller), para lograr esto en la tecnología antes mencionada se debe trabajar utilizando **STRUTS**.

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de “software libre” y su compatibilidad con todas las plataformas en que Java Enterprise esté disponible, lo convierte en una herramienta altamente disponible.

Se debe incluir un plugin más a Lomboz para poder trabajar con Struts; existen varios pero el usado en este caso fue el de Amateras que puede obtenerse en el siguiente sitio:

http://amateras.sourceforge.jp/cgi-bin/fswiki_en/wiki.cgi?page=StrutsIDE

C6. Ambientes de ejecución en MPI

La librería de pasaje de mensajes (MPI, siglas en inglés de *Message Passing Interface*) es un protocolo de comunicación entre computadoras; existen varias distribuciones, dependiendo de estas se podrán usar en lenguajes como C, C++, ADA, FORTRAN y/o JAVA.

Dentro de las utilizadas en el sistema están:

- **LAM**

Es una implementación de esta librería, es de código abierto, sólo trabaja en ambientes Unix/Linux y puede obtenerse más información en el siguiente sitio:

www.lam-mpi.org/

- **MPICH2**

Es otra implementación de esta librería, también es de código abierto, además de trabajar bajo sistemas Unix/Linux puede hacerlo en Windows y MacOS, hay más información en el siguiente sitio:

<http://www-unix.mcs.anl.gov/mpi/mpich2/>

C7. Motor de base de datos

El motor de base de datos elegido para trabajar con el sistema fue **MYSQL 5**. MySQL es un sistema de gestión de base de datos multithread. MySql AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. MySQL está poseído y patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Más información se puede obtener del sitio:

<http://www.mysql.com/>

Bibliografía

- [1] Grama A, Gupta A, Karypis G, Kumar V. "Introduction to parallel computing". Second Edition. Pearson Addison Wesley, 2003.
- [2] Jordan H, Alaghband G. "Fundamentals of parallel computing". Prentice Hall, 2002.
- [3] Dongarra J, Foster I, Fox G, Gropp W, Kennedy K, Torczon L, White A. "The Sourcebook of Parallel Computing". Morgan Kaufmann Publishers. Elsevier Science, 2003.
- [4] Z. Juhasz (Editor), P. Kacsuk (Editor), D. Kranzlmuller (Editor). "Distributed and Parallel Systems: Cluster and Grid Computing" (The International Series in Engineering and Computer Science). Springer; 1 edition (September 21, 2004).
- [5] Foster I., Kesselman C., Kaufmann M. "The Grid 2: Blueprint for a New Computing Infrastructure". The Morgan Kaufmann Series in Computer Architecture and Design. 2 edition (November 18, 2003).
- [6] Berman F., Fox G., Hey A. "Grid Computing: Making The Global Infrastructure a Reality". John Wiley & Sons (April 8, 2003).
- [7] The TOP500 project <http://www.top500.org/>.
- [8] Jiang, Yeung. "Scalable Inter-Cluster Communication System for Clustered Multiprocessors". 1997.
- [9] AJAX ORG <http://www.ajax.org/>.
- [10] AJAX en español <http://www.ajax.com.es/index.php?title=Portada>.
- [11] LAM/MPI Parallel Computing <http://www.lam-mpi.org>.
- [12] MPICH2/MPI Parallel Computing <http://www-unix.mcs.anl.gov/mpi/mpich2/>.
- [13] Torque Resource Manager
<http://www.clusterresources.com/pages/products/torque-resource-manager.php>
- [14] Maui Cluster Scheduler <http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php>.
- [15] Ejecución de comandos vía Web:
<http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-traps.html?page=4>
<http://www.rgagnon.com/javadetails/java-0014.html>
<http://2fwww.programacion.com/java/articulo/tt011204/>
- [16] RIA:
<http://es.wikipedia.org/wiki/RIA>
http://es.wikipedia.org/wiki/Rich_Internet_Application

- [17] Java Sun Microsystems <http://java.sun.com/>.
- [18] Proyecto Jakarta Tomcat <http://tomcat.apache.org/>.
- [19] Proyecto Apache Struts <http://struts.apache.org/>.
- [20] Fedora en español <http://www.redhat.es/fedora/>.
- [21] Fedora project <http://fedoraproject.org/wiki/>.
- [22] Proyecto Eclipse www.eclipse.org.
- [23] Proyecto Lomboz <http://lomboz.objectweb.org/>.
- [24] Proyecto Amateras struts IDE <http://amateras.sourceforge.jp>.
- [25] Capacitación en UML y diseño orientado a objetos, Java, Bases de datos relacionales (SqlServer), JDBC, XML, Servlets, JSP, Tomcat, Struts – LIFIA – Facultad de Informática.



BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

| | | | |
|-----------------------------------|----------------------------|--------|--|
| TES 07/11 DIF-03096 SALA | DONACION.....FACULTAD..... | TES | |
| | \$..... | 07/11 | |
| | Fecha.....12-3-08 | | |
| | Inv. E.....Inv. B..... | 003096 | |

UNIVERSIDAD NACIONAL DE LA PLATA
 FACULTAD DE INFORMÁTICA
 Biblioteca
 50 Y 120 La Plata
 catalogo:info@unlp.edu.ar
 biblioteca@info.unlp.edu.ar

DIF-03096