Gestión Bajo Demanda de Laboratorios Informáticos Universitarios

Carlos Agra Ramos,
Abel Míguez Rodríguez,
David Rodríguez Pérez

Directores:
Prof. Rubén Santiago Montero
Prof. Rafael Moreno Vozmediano.

Curso 2008-2009
Proyecto de Sistemas Informáticos
Facultad de Informática
Universidad Complutense de Madrid

Declaración de Conformidad

Los alumnos:

"Carlos Agra Ramos", "Abel Míguez Rodríguez", "David Rodríguez Perez" aquí firmantes autorizan a la "Universidad Complutense de Madrid" a difundir y utilizar con **fines académicos, no comerciales y mencionando expresamente a sus autores**, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado".

Madrid, 3 de Julio de 2009

Carlos Agra Ramos	Abel Míguez Rodríguez	David Rodríguez Pérez

Agradecimientos

Quisiéramos agradecer a nuestros profesores directores el apoyo y las ideas aportadas (así como las consultas sobre Open Nebula).Y al padre de Carlos por la ayuda en una impresión de calidad del documento.

Abel:

En este escrito, que representa la culminación del proyecto fin de carrera y un punto de inflexión en mi vida, me gustaría agradecer a todas las personas y vivencias que de cierta forma han influenciado en mi persona... a mi hermano **Ernesto** por animarme siempre a seguir con todo; a mis padres **Julio y Susana** que tanto han tenido que luchar por que yo pueda estar ahora donde estoy, afortunadamente, jamas sabré lo que es tener que salir de tu país a la fuerza y no poder volver.

A todos mis amigos y amigas, que más de una vez en estos años han tenido que poner la oreja y un hombro en el que apoyarme...

Y... “Last but not least”... al par de maravillosos compañeros de SSII, **Carlos Agra** y **David Rodriguez**, que he tenido, sin los cuales no podría haber hecho el proyecto.

Carlos:

Quisiera agradecer a nuestras familias por aguantar los domingos nuestras reuniones en nuestras casas, así como la generosa hospitalidad en cada una de nuestras reuniones (lo que incluyó bizcochos, helados y vichyssoise). Y, claro está, a ese grandioso trabajo en grupo con mis compañeros, ¡somos equipo!.

David:

Quisiera agradecer a mi familia por los días que no he estado a todo por el proyecto, y en especial dedicárselo a mi padre, quien me gustaría que lo hubiera visto terminado. También a mis propios compañeros, ¡buen trabajo chicos!

Índice

Palabras clave para la búsqueda bibliográfica:

•Laboratorios docentes
•Aulas informáticas
•Virtualización
•OpenNebula
•Aulas virtuales
•Ahorro energético
•Interfaz web
•Acceso remoto
•Servlets
•IPTables

Resumen

La infraestructura “Virtual Lab” propuesta se presenta como una mejora en la gestión energética de los laboratorios informáticos, una simplificación del mantenimiento de los puestos sobre los que los alumnos realizan sus prácticas y un aumento de las posibilidades que los laboratorios pueden ofrecer a sus alumnos.

“Virtual Lab”, gracias a la virtualización permite desacoplar el software de los equipos hardware lo que permitiría mejorar el aprovechamiento energético de los laboratorios universitarios, al permitir tener abiertos y en funcionamiento exclusivamente un número determinado de puestos que ofrecerían todas las posibles configuraciones. “Vlab” permite a los alumnos trabajar con estructuras complejas, como clusters de computadoras; acceder remotamente desde sus casas para realizar prácticas sobre los puestos de laboratorios de la universidad; incluso, si las licencias de software lo permiten, se podría poner a disposición de los alumnos las imágenes de las Máquinas Virtuales, con lo que la reproducción de la configuración del laboratorio en sus casas sería automática.

Overview

The “Virtual Lab” infraestructure is presented as an improve in energetic management of computer laboratories, a simplification of the students working systems's manteinance and an increase of the posibilities the laboratories can offer to them.

“Virtual Lab” uses virtualization to decouple the software from the hardware in order to improve the energetic utilization of university laboratories, allowing to keep open and working exclusively a determined number of computers that will offer all possible configurations. “Vlab” allows students work with complex structures, as computer clusters; remote access from their home to laboratory systems to do practices; and also, if software licences allows it, it could be possible to give the students the Virtual Machines images, so they could use the laboratory configuration in their homes.

Virtual Lab

Qué es Virtual Lab:

La infraestructura que proponemos, llamada “Virtual Lab”, mejoraría en diferentes aspectos una herramienta de aprendizaje como son los laboratorios informáticos.

Proponemos la explotación de la virtualización como solución, para muchos problemas que actualmente encontramos los alumnos en el uso de laboratorios, y la vez como tecnología que permitiría ampliar las capacidades de uso académico de los laboratorios.

La configuración de los laboratorios por parte de los profesores consistiría en introducir los parámetros que definen los atributos de las máquinas virtuales, la memoria disponible, la imagen de disco que se utilizará (S.O. y su configuración pertinente) y el número de puestos de prácticas disponibles para los alumnos.

Los alumnos se conectan por medio de protocolos de administración remota a las máquinas virtuales ya configuradas y listas para su utilización, estas máquinas virtuales no se ejecutan en los puestos del laboratorio, reduciendo la inversión hardware en los mismos.
El sistema de virtualización donde se ejecutan los puestos virtuales centraliza tanto la inversión hardware como la administración de los laboratorios, además de permitir su uso desde cualquier equipo, domicilio de los alumnos u ordenadores portátiles; no exclusivamente los laboratorios del centro, siempre y cuando la política de seguridad implementada lo permita.

La infraestructura que da soporte a dicho sistema de virtualización consiste en clusters de computadoras gestionadas por “Open Nebula” (www.opennebula.org) un gestor de infraestructuras virtuales desarrollado por el grupo *“Distributed Systems Architecture Research Group”* de la “Universidad Complutense de Madrid”.
Open Nebula permite centralizar la gestión de la infraestructura virtual a la vez que permite la escalabilidad del sistema, la adición de nuevos equipos para dar soporte a la infraestructura virtual es muy sencilla y en caso de necesidades puntuales OpenNebula permite la utilización de proveedores de servicios externos como Amazon Elastic Compute Cloud (Amazon EC2).

”Virtual Lab”, si las licencias software lo permiten, podría ofrecer a los alumnos las imágenes ya configuradas para que los alumnos las descargaran y así utilizar en casa una instalación ya configurada del software que necesitan para la realización de las prácticas.

Objetivos y beneficios:

- Laboratorios en cualquier sitio.
- Llevar el laboratorio a casa.
- Abaratamiento del coste del laboratorio.
- Simplificación de la administración del laboratorio.
- Posibilidad de consolidar.

• Entornos aislados de entrenamiento, con acceso a internet controlado.

• Adaptar la infraestructura de prácticas a diferentes usos.

• Provisión dinámica de entornos de práctica complejos.

• Despliegue de laboratorios y entornos con un click.

• Control y Monitorización de las MV desplegadas.

• Transparencia de la virtualización para el alumno.

• Acceso mediante terminales gráficos.

• Se dispondrá de dos interfaces de uso. De administrador de sistema y de usuario.

Adoptar “Virtual lab”, motivación:

Los laboratorios de un centro académico son herramientas imprescindibles para el aprendizaje del alumnado pero se observan ciertos problemas para hacer uso de los mismos.

Problemática a la hora de la realización de prácticas fuera del horario académico por la necesidad de unas aplicaciones concretas sólo disponibles en unos laboratorios concretos que pueden no estar disponibles en el momento en que el alumno los necesite. A esto se junta la dificultad de la imitación de las configuraciones en los ordenadores personales de los alumnos, dándose problemas de compatibilidad al tener que utilizar distintas versiones de los programas. También la imposibilidad del uso de los mismos por falta de puestos disponibles con una determinada configuración en momentos de alta afluencia, en muchos casos nos hemos encontrado con que, aunque hay puestos disponibles, no los podemos usar por no estar configurados con las aplicaciones que necesitamos.

También hay casos en los que la infraestructura actual no puede utilizarse para el aprendizaje de ciertas asignaturas de la titulación, como por ejemplo:

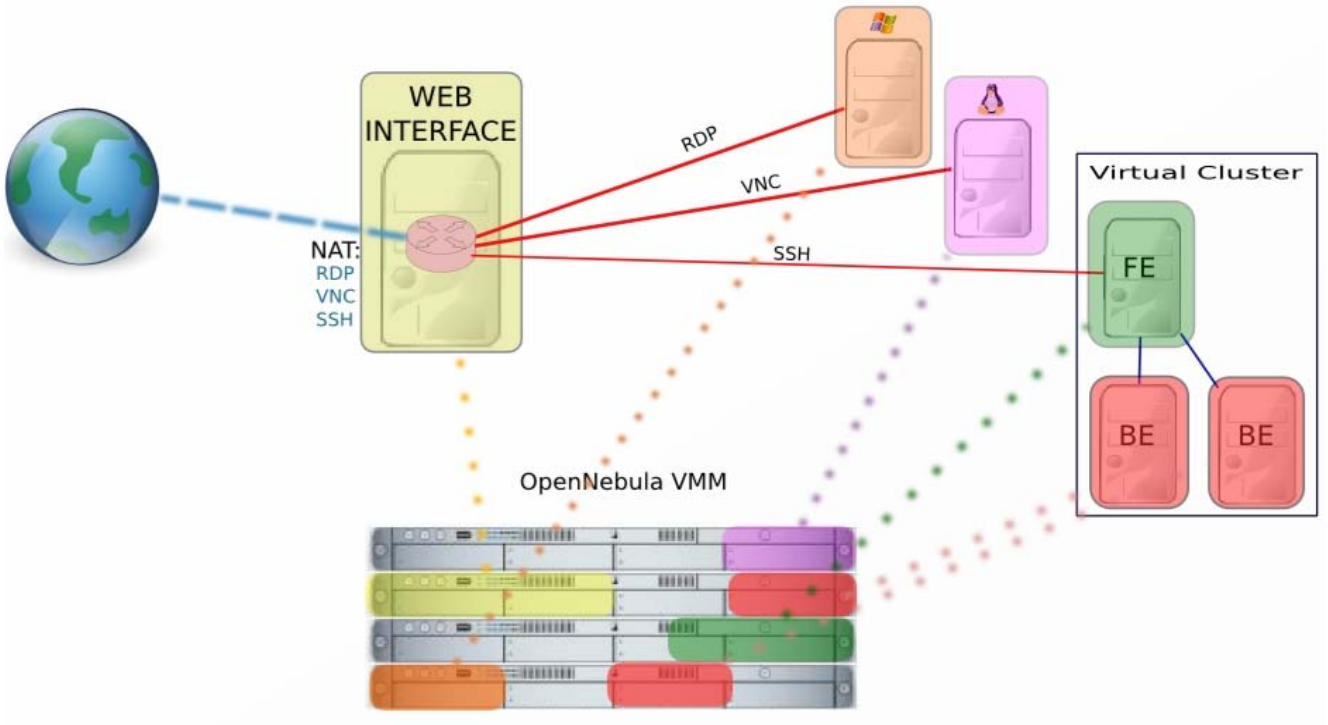
- *“Evaluación del rendimiento de configuraciones”* para profundizar en la asignatura y poder realmente realizar evaluaciones de distintas configuraciones se requiere cambiar parámetros del kernel que solo pueden ser realizados por un usuario con permisos de administración.
- *“Procesamiento Paralelo”* se podrían realizar practicas sobre clusters que asentaran los conceptos introducidos en las clases teóricas.
- *“Laboratorio de Redes”* también se requiere de permisos de super usuario para realizar configuraciones de equipos como routers.
- *“Seguridad de computadores”* aunque actualmente la titulación no dispone de esta asignatura, con la infraestructura virtual se podría dar soporte a prácticas de una asignatura que implique la auditoria de redes de computadores.

Para dar solución a todos estos problemas proponemos “Virtual Lab” como infraestructura de laboratorios académicos.

Diseño del sistema:

El diseño de “Virtual Lab” se compone de diferentes módulos:

- Interfaz web, desarrollada con tecnologías JAVA para que los usuarios trabajen sobre el mismo.
- Base de datos MySQL, para la gestión de usuarios, laboratorios y máquinas virtuales.
- Gestor de Infraestructura virtuales, OpenNebula.
- Utilidades de sincronización y de configuración dinámica del rutado de paquetes para el acceso a los nuevos puestos de laboratorio a través de un entorno controlado.



Funcionamiento:

El acceso al sistema se realizaría a través de una interfaz web instalada en un equipo que actuaria como router entre las máquinas virtuales y la red externa para ofrecer un entorno controlado.

La interfaz Web ofrece diferentes opciones al profesor como al alumno, el profesor da de alta laboratorios con sus parámetros y el alumno selecciona sobre que laboratorio desea trabajar. Al poco de iniciarse la máquina virtual el alumno recibe unos parámetros de configuración para acceder a la misma de forma remota.

Como las máquinas virtuales se encuentran en una red local cuyo router es el equipo donde reside la interfaz web, este debe ser capaz de configurar de forma dinámica el redireccionado de paquetes cada vez que se lanza una nueva máquina virtual y así lo usuarios puedan acceder a las mismas.

De la provisión de máquinas virtuales y la disposición de las mismas en los diferentes servidores se encarga Open Nebula, con el cual la interfaz se comunica para ofrecer a los usuarios de administración el estado actualizado de los puestos virtuales en funcionamiento y su ocupación.

Conclusiones:

Objetivos logrados:

- Acceso al puesto del laboratorio seleccionado desde cualquier ordenador con conexión a internet.
- Posibilidad de utilizar las imágenes preconfiguradas en los equipos de los alumnos donde tengan configurado un hypervisor: KVM, VirtualBox, VMWare, etc... (es posible que sea necesaria una conversión del formato de imagen de disco).
- El acceso a las máquinas virtuales se realiza a través de un equipo donde se puede configurar iptables con las necesidades de filtrado y control requerido, las máquinas virtuales acceden a internet a través de un entorno controlado.
- Interfaz web de administración que permite la configuración de laboratorios y modificar los parámetros de los mismos , el despliegue de puestos en un “click”, la monitorización y el control de las MV desplegadas.
- Interfaz web de usuario que ofrece al alumno un listado de los laboratorios a los que se puede conectar, una vez seleccionado la interfaz despliega el puesto y le ofrece al usuario los parámetros de acceso a través de terminales gráficos (o de otros servicios que el profesor ha configurado para el laboratorio).
- Posibilidad de ofrecer a los alumnos el acceso a las máquinas virtuales con privilegios de administración.

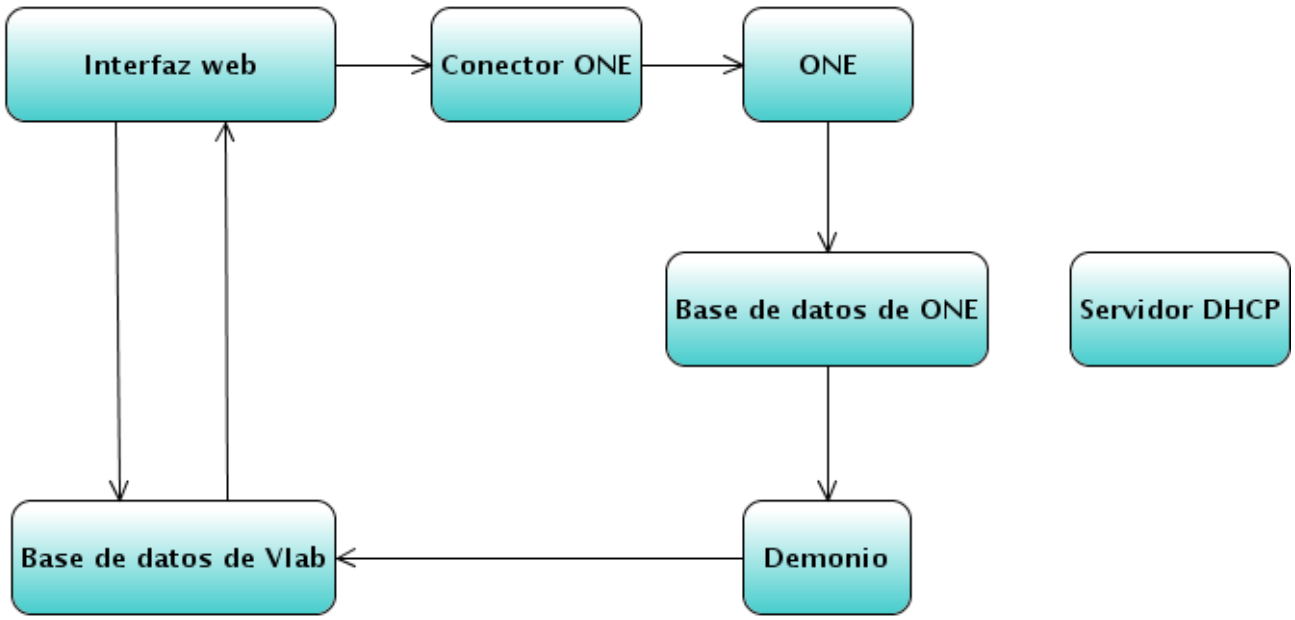
Objetivos no alcanzados:

- Aislamiento de las máquinas virtuales entre sí.
- Provisión dinámica de entornos de práctica complejos, configuración de clusters virtuales por medio de templates que definan un conjunto de máquinas en lugar de una única instancia.

Estructura del Sistema

En este diagrama se muestra la estructura general del Virtual Lab, así como la transferencia de datos que se produce entre los distintos elementos. Todos los conceptos se explicarán con mayor detalle más adelante.

Como se puede ver, el flujo de datos es casi completamente circular. A grandes rasgos, se dispone de una interfaz web, un conector para Open Nebula, el propio Open Nebula (junto con su base de datos), un demonio de sincronización de bases de datos, una base de datos propia de Virtual Lab y un servidor DHCP. El sentido de las flechas indica qué elemento envía la información y cuál lo recibe.



•Interfaz web: envía datos al conector ONE, para comunicarse con OpenNebula, y envía y recibe datos de la Base de datos de Vlab. Cuando se realiza cualquier acción que necesite actuar sobre OpenNebula, como lanzar una máquina virtual, se envía la información al conector. Los valores leídos de la base de datos de Vlab son los que se muestran en la interfaz y son los que pueden ser modificados desde la misma.

•Conector ONE: se trata de un cliente que se comunica por medio de llamadas “XMLRPC” con OpenNebula, se encarga de transportar las peticiones enviadas desde la interfaz a OpenNebula para que las ejecute, por ejemplo, apagar/iniciar una máquina virtual.

•ONE: OpenNebula ofrece a “Virtual Lab” una infraestructura virtual sobre la que desplegar los puestos de laboratorio. OpenNebula centraliza toda la administración de la infraestructura además de permitir la escalabilidad de la misma, para mas información consultar la sección sobre OpenNebula.

•Base de datos de ONE: es un componente interno a OpenNebula donde se registran todas las máquinas virtuales que están y han estado en funcionamiento para su monitorización e historial de uso.

•Demonio de sincronización: como las bases de datos de ONE y Vlab son distintas, se dispone de este demonio para la sincronización entre ambas. Lee el contenido de la base de ONE y actualiza la información necesaria en la base de datos de Vlab.

•Base de datos de Vlab: contiene la información necesaria para ser mostrada a los usuarios y administradores de Vlab, así como los datos necesarios para la identificación de los mismos. Todos los datos se escriben desde la interfaz o desde el demonio de sincronización, exceptuando los datos de los usuarios.

La idea principal es la de la transparencia para el usuario. Para ello, se ha conseguido que todas las características posibles de Vlab se puedan aprovechar desde la interfaz web. Cualquier petición realizada generará una serie de acciones que no serán visibles para el usuario de las cuales solo conocerá el resultado.

Tecnologías Utilizadas

OpenNebula

OpenNebula permite utilizar un cluster físico como una infraestructura virtual flexible que se adapta dinámicamente a las demandas cambiantes. OpenNebula se apoya en plataformas existentes de virtualización para crear una nueva capa de virtualización entre el servicio y la infraestructura física. Esta nueva capa soporta la ejecución de servicios en un cluster físico, extendiendo los beneficios de los VMMs (del inglés, Virtual Machine Monitors) desde un solo recurso físico a un cluster de recursos. OpenNebula desacopla de manera efectiva un servidor (desplegado como una máquina virtual preconfigurada) no sólo de la infraestructura física sino de su localización espacial.

Beneficios de OpenNebula: OpenNebula se apoya en la funcionalidad proporcionada por los hypervisors de máquinas virtuales para proporcionar los siguientes beneficios en un entorno multi-host:

Para el dueño de la infraestructura:

- Control centralizado de la carga de trabajo de las máquinas virtuales y las infraestructuras distribuidas.
- Balance de la carga de trabajo para mejorar la eficiencia y la utilización.
- Consolidación del servidor a un reducido número de sistemas físicos, reduciendo así espacio, esfuerzos de administración, requisitos de energía o de enfriamiento o la posibilidad de apagado de sistemas sin interferir en la carga de trabajo
- Redimensionado dinámico de la infraestructura física añadiendo nuevos hosts
- Particionado dinámico de clusters para ejecutar servicios varios.
- Soporte para cargas de trabajo heterogéneas con requisitos variados de software, permitiendo la ejecución de software con requisitos mínimos
- Escalado dinámico de la infraestructura privada, para satisfacer demandas fluctuantes.

Para el usuario:

- Provisión bajo demanda de máquinas virtuales para satisfacer las necesidades del servicio y de los usuarios.

Para los integrantes del sistema:

- Arquitectura e interfaz abierta y flexible.
- Open source software: licencia de Apache
- Integración con cualquier componente en el “ecosistema” de la virtualización, como por ejemplo hypervisors, managers de imágenes virtuales...

Características:

- Interfaz de usuario: interfaz de consola estilo UNIX para administrar el ciclo de vida de las máquinas virtuales e interfaz XML-RPC API y libvirt.
- Virtualización: conectores de Xen. KVM y libvirt, así como plugin para Amazon EC2 cloud.
- Imágenes: mecanismos generales para transferir y clonar imágenes de máquinas virtuales.
- Red: definición de redes virtuales de para interconectar máquinas virtuales.
- Tolerancia a fallos: base de datos persistente para almacenar información del host y de las máquinas virtuales.
- Escalabilidad: testeado administrando miles de máquinas virtuales
- Instalación: instalación en un frontend de un cluster UNIX sin necesidad de nuevos servicios en los recursos remotos, y distribuido en Ubuntu 9.04 (Jaunty Jackalope)

AJAX

Ajax, a veces escrito AJAX (abreviatura proveniente del inglés: Asynchronous JavaScript And XML), es un grupo de tecnologías web usadas en el lado del cliente para crear aplicaciones web interactivas. Con Ajax, las aplicaciones web pueden recoger información del servidor de manera asíncrona sin interferir con la visualización ni el comportamiento de la página mostrada. Su utilización ha permitido un aumento en la interactividad de las páginas web y mejor calidad de sus servicios gracias a su modo asíncrono, sin necesidad de participación por parte del usuario.

La transferencia de datos entre el servidor y el cliente se realiza con el objeto XMLHttpRequest. Dicho objeto se crea en el lado del cliente y se envía al servidor para que lo procese y envíe una respuesta. A pesar del nombre de la tecnología, el uso de javascript y xml no es estrictamente necesario, así como las peticiones no tienen que ser estrictamente asíncronas.

Tecnologías constituyentes, Ajax no es una tecnología en si, sino un conjunto de ellas. En concreto, se suele hacer uso de:

- HTML y CSS para los estilos y la apariencia de la página.
- DOM (Document Object Model), al que se accede con javascript para interactuar dinámicamente con la información mostrada en la página.
- Un método para intercambiar información asíncronamente entre el navegador y el servidor, evitando así recargar totalmente la página. El objeto XMLHttpRequest es el utilizado habitualmente (también en nuestro caso), aunque también se puede utilizar un objeto iFrame
- Un formato para la información transferida. Los más utilizados son XML, texto plano y JSON. Una característica a señalar es que dicha información se puede generar dinámicamente del lado del servidor.
- JavaScript o VBScript para reunir todas las citadas tecnologías en el lado del cliente.

Ventajas de la utilización de Ajax:

- En muchos casos, las páginas de un sitio web contienen mucho información que es común entre ellas. Utilizando métodos tradicionales, dicho contenido tendría que ser recargado en cada petición. Sin embargo, utilizando Ajax, una aplicación web puede realizar una petición únicamente del contenido que necesita ser actualizado, reduciendo drásticamente el uso de ancho de banda y el tiempo de carga.

- El uso de peticiones asíncronas permite a la interfaz de usuario (web) ser más interactiva y responder rápidamente a las acciones del usuario.

- El uso de Ajax puede reducir el número de conexiones al servidor, puesto que los scripts y los datos del estilo sólo se tienen que pedir una vez.

Desventajas de la utilización de Ajax:

- Las páginas creadas dinámicamente utilizando Ajax no se registran automáticamente en el historial del navegador. Por lo tanto, al intentar volver a una página anterior con el botón "Atrás" de los navegadores podría no llevar al usuario a un estado anterior a la última actualización asíncrona sino llevarle a la anterior página total.

- Cualquier navegador que no soporte Javascript o XMLHttpRequest o que no lo tenga activado, no podrá hacer uso de su funcionalidad.

Uso de Ajax en Vlab:

En nuestro proyecto, nos dimos cuenta de la problemática que existía al mostrar la información sobre los laboratorios a los administradores y a los usuarios. Se dispone de bastantes datos que son interesante que se muestren actualizados en todo momento para tener una visión concreta de los sistemas.

En el caso de los administradores, se ha utilizado Ajax en dos páginas:

- En labstatus.jsp, se actualizan dinámicamente los valores de máquinas virtuales permitidas y lanzadas para un determinado tipo de laboratorio, así como su puerto. Esto es útil, por ejemplo, si el administrador quiere monitorizar cuántas máquinas están en uso y cuántas están libres en cada momento sin necesidad de recargar la página de administración
- En adminlab.jsp, se actualizan dinámicamente los valores de estado, memoria utilizada, memoria máxima, puerto, ONEid y dirección MAC de cada una de las máquinas virtuales de un tipo de laboratorio. Esto es útil, por ejemplo, si el administrador quiere apagar una determinada máquina al comprobar que su estado es incorrecto o que su uso de memoria es elevado, sin tener que comprobar dichos valores a través de ONE.

En el caso de los usuarios, se ha utilizado ajax en una página:

- En connectinfo.jsp, se actualizan dinámicamente los valores de estado y de puerto a utilizar. Esto es útil cuando el usuario ha pedido un puesto de un laboratorio y tiene que esperar a que su máquina virtual se inicialice. Así puede comprobar sus datos de conexión sin tener que recargar la página.

Tecnologías utilizadas:

- En el lado del cliente, se optó por utilizar Javascript por su mayor extensión y por su mayor compatibilidad con el resto de tecnologías de Ajax. En lugar de ser totalmente asíncrono, el refresco de información se realiza cada diez segundos automáticamente. Para la petición de información se ha utilizado XMLHttpRequest por su facilidad de uso y por su compatibilidad con casi todos los navegadores actuales.

- En el lado del servidor, la información requerida se formatea en XML y se devuelve al script del cliente para su tratamiento y visualización.

MySQL

MySQL es un sistema gestor de bases de datos relacionales con licencia GNU GPL creado por la empresa MySQL AB, que fue adquirida por Sun Microsystems en 2008. Sus principales características son:

Interioridades y portabilidad:

•Escrito en C y en C++

•Funciona en diferentes plataformas.

•APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.

•Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiples CPUs si están disponibles.

•Proporciona sistemas de almacenamiento transaccionales y no transaccionales.

•Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.

•Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.

•Un sistema de reserva de memoria muy rápido basado en threads.

•Joins muy rápidos usando un multi-join de un paso optimizado.

•Tablas hash en memoria, que son usadas como tablas temporales.

•Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.

•El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Tipos soportados:

•Diversos tipos: enteros con/sin signo de 1, 2, 3, 4, y 8 bytes de longitud, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, y tipos espaciales OpenGIS.

•Registros de longitud fija y longitud variable.

Sentencias y funciones:

•Soporte completo para operadores y funciones en las cláusulas de consultas SELECT y WHERE.

•Soporte completo para las cláusulas SQL GROUP BY y ORDER BY. Soporte de funciones de

agrupación (COUNT(), COUNT(DISTINCT ...), AVG(), STD(), SUM(), MAX(), MIN(), y GROUP_CONCAT()).

•Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN cumpliendo estándares de sintaxis SQL y ODBC.

•Soporte para alias en tablas y columnas como lo requiere el estándar SQL.

•DELETE, INSERT, REPLACE, y UPDATE devuelven el número de filas que han cambiado (han sido afectadas). Es posible devolver el número de filas que serían afectadas usando un flag al conectar con el servidor.

•Puede mezclar tablas de distintas bases de datos en la misma consulta.

Seguridad:

•Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor.

Escalabilidad y límites:

•Soporte a grandes bases de datos.

•Se permiten hasta 64 índices por tabla. Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes. Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

Conectividad:

•Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma.

Clientes y herramientas:

•MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente mysqlcheck.

Uso en Vlab:

Para llevar una buena gestión de usuarios nos era necesaria la existencia de una base de datos. Elegimos MySQL por estar bajo licencia GNU GPL y seguir el lenguaje SQL de bases de datos, además de por la existencia de múltiples APIs que nos hicieron falta. Además, necesitábamos más bases de datos para la gestión de múltiples laboratorios y conservar la información de los mismos, como el template que ejecutaba una máquina para ése laboratorio concreto, y otra base de datos para las máquinas virtuales, ya que también manejamos información y estados no contemplados en la propia base de datos que maneja OpenNebula. Por último, tenemos una base de datos para la gestión de las direcciones mac (e IPs) a las máquinas virtuales.

Las tablas de la base de datos son las siguientes:

- **macs(mac,free)**

  Mantiene información relativa al "pool" de direcciones MAC para asignar a las Maquinas Virtuales al definir un laboratorio nuevo. La dirección MAC es la que determina la dirección IP que tendrá la máquina virtual en ejecución, configuración realizada por medio de "leases" en el servidor DCHP.

  • mac: la dirección mac

  • free: indica si la mac está asignada

- **users(id,password,fname,lname,admin,idVM)**

  Guarda los administradores del sistema y usuarios que tienen acceso al servicio con la maquina virtual que están usando.

  • id: identificador de usuario

  • password: contraseña del usuario

  • fname: nombre

  • lname: apellido

  • admin: bit que indica el rol del usuario (0==usuario,1==administrador)

  • idVM: id de la máquina virtual en uso por el usuario (si hay alguna)

- **vms(id,lab,status,usedMem,maxMem,port,ONEid)**

  Mantiene información de las máquinas virtuales, tanto información relativa a la ejecución (estado, memoria en uso) como características particulares de cada máquina virtual (dirección MAC, puerto del router redireccionado para acceder a la maquina).

  • id: id de la máquina virtual

  • lab: nombre del laboratorio al que pertenece

  • status: estado (running, boot, shutdown, available)

  • usedMem: memoria usada actualmente

  • maxMem: memoria máxima permitida para la máquina virtual

  • port: puerto utilizado para llegar a la máquina virtual

  • ONEid: identificador utilizado por OpenNebula para referirse a la máquina virtual

  • MAC: la dirección mac utilizada por la máquina virtual

- **labs(name,template,launched,allowed)**

  Mantiene información relativa a los laboratorios dados de alta en la aplicación y que están disponibles para los alumnos, tanto el numero de puestos disponibles como las características generales de las instancias, definidas por medio del template de Open Nebula.

  • name: nombre del laboratorio

  • template: template de OpenNebula para las máquinas de ése laboratorio

  • running: número de máquinas virtuales en ejecución.

  • allowed: número máximo de máquinas virtuales permitidas

  • port: puerto local del servicio de control remoto a utilizar (VNC, ssh...)

Virtualización

Virtualización es un termino que abarca diferentes significados en el área de la computación, en general consiste en la abstracción de recursos de computadoras.

Dentro del término "Virtualización" se incluyen las siguientes tecnologías:

- **"Virtualización de Plataforma"**, consistente en la ocultación al S.O de la Maquina Virtual, las características físicas de los recursos de procesamiento de información y en su lugar, la presentación de una plataforma emulada para su ejecución.

- **"Virtualización de Recursos"**, consistente en la abstracción de un un determinado recurso del sistema, como puede ser Volúmenes de almacenamiento, Recursos de Red, Memoria RAM, etc...

- **"Virtualización de Aplicaciones"**, consistente en el empaquetado de la aplicación y de todas sus dependencias, tanto de librerías como de configuración (registro de windows), de tal forma que incluso se pueda ejecutar en un sistema aplicaciones que tengan dependencias incompatibles entre sí, ya que no es necesario instalar dichas dependencias en el sistema.

En nuestro proyecto hacemos uso de la "Virtualización de Plataforma", concepto que explicaremos con mayor detalle.

Platform Virtualization

Se ejecuta el sistema operativo de la maquina virtual, de ahora en adelante "guest", sobre la virtualización de los recursos hardware que el equipo anfitrión ofrece ("host").

Esta virtualización del hardware del equipo "host" se lleva a cabo por medio de una capa de software llamada "Hypervisor" o VMM (Virtual Machine Manager) que administra y gestiona la CPU, Memoria y dispositivos de Entrada / Salida.

El Hypervisor distribuye los recursos dinámicamente entre las diferentes máquinas virtuales en ejecución, de esta forma la máquina virtual se ejecuta como si estuviese instalada en una plataforma de hardware autónoma.

En general, la máquina virtual no tiene la necesidad de saber si esta siendo ejecutada sobre un hardware emulado, lo que se traduce en no tener que adaptar el software para su ejecución en una máquina virtual. Esto último tiene sus excepciones, lo que nos lleva a la existencia de distintos tipos de "Platform Virtualization"

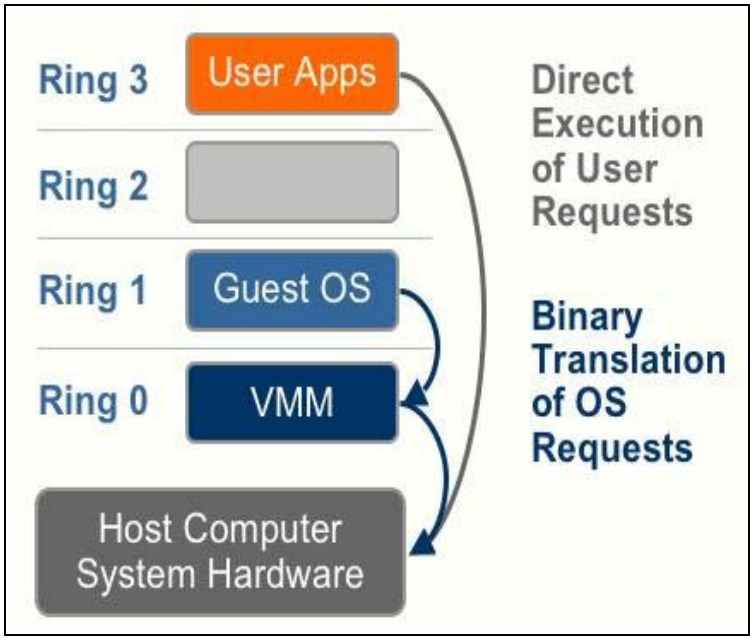
Full Virtualization

Este tipo de virtualización abstrae el hardware subyacente ofreciendo "la simulación completa de hardware" creando un completo sistema virtual, donde un sistema operativo "guest" puede ejecutarse. No se requiere ninguna modificación del sistema operativo "guest" o de sus aplicaciones; el S.O. "guest" y sus aplicaciones no son conscientes de estar siendo ejecutadas dentro de un entorno virtualizado, se ejecuta como si estuviese sobre un sistema físico.

El resultado es un sistema virtualizado el cual consta de las mismas capacidades de ejecución de software que un sistema funcionando directamente sobre Hardware, en particular, puede ejecutar cualquier sistema operativo (que soporte la arquitectura virtualizada). Esta aproximación a la virtualización es ventajosa pues capacita al software de independencia con respecto al hardware, permitiendo la migración de aplicaciones y cargas de trabajo entre distintos sistemas físicos.

Sin embargo, el reto principal en "full virtualization" es la interceptación y simulación de operaciones privilegiadas, como las instrucciones de E/S. El efecto de todas las operaciones realizadas en una máquina virtual deben mantenerse dentro del espacio dedicado a dicha máquina virtual, las operaciones virtuales no tienen permitido modificar el estado de cualquier otra máquina virtual, el hypervisor o el hardware.

Algunas instrucciones pueden ser ejecutadas directamente sobre el hardware, puesto que sus efectos están completamente controlados por el programa de control, como posiciones de memoria y registros aritméticos; otras instrucciones con la capacidad de alterar el estado e información fuera de la maquina virtual, no pueden ser ejecutadas directamente y deben ser en su lugar capturadas y simuladas por el hypervisor.



La necesidad de tener que capturar ciertas instrucciones por parte del hypervisor, tener que ofrecer a la maquina virtual una imagen de un sistema completo, una BIOS virtual, espacio de memoria virtual y dispositivos virtuales; incurre en una cierta penalización (“overhead”) en el rendimiento de la máquina virtual.

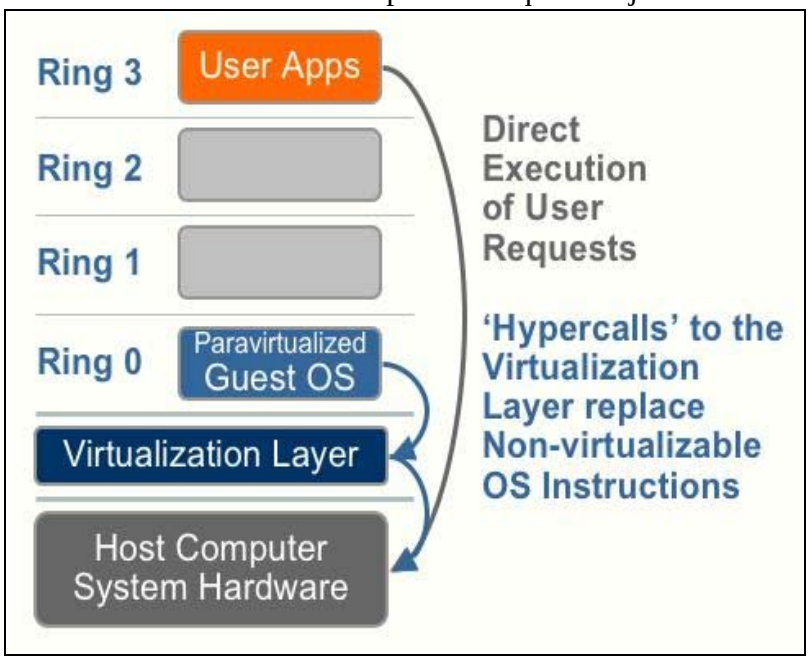
Un hypervisor que haga uso de la Virtualización completa debe disponer del soporte para virtualización por parte del procesador en la forma de un conjunto de instrucciones especiales, conocidas como "Intel VT" y AMD Pacífica, en procesadores Intel y AMD respectivamente.

KVM, es una solución para implementar virtualización completa con Linux sobre hardware x86. Con KVM es posible la ejecución de distintas máquinas virtuales no modificadas con sistemas operativos Windows o Linux. Está formada por un módulo del núcleo, (con el nombre kvm.ko) incluido en Linux desde la versión 2.6.20, y módulos específicos dependiendo del procesador, kvm-intel.ko y kvm-amd.ko. KVM utiliza una versión modificada de QEMU que ofrece a cada máquina virtual su propio hardware virtualizado: tarjetas de red, discos, adaptadores gráficos, etc...

Para-Virtualization

La Para-virtualization ofrece a cada maquina virtual una abstracción del hardware que es similar pero no idéntico al hardware subyacente. Un sistema de Para-virtualización intenta proveer la mayor cantidad de los servicios posibles desde el hardware en lugar de desde una abstracción del mismo, los servicios que no se pueden ejecutar directamente en el hardware son ejecutados por el hypervisor.

La Paravirtualizacion requiere de la modificación de los sistemas operativos que se ejecutan en las maquinas virtuales, cambiando las llamadas al hardware con capacidad de alterar el estado fuera de la máquina virtual por llamadas al hypervisor encargado de realizar las operaciones de una forma controlada, de esta forma el hypervisor no tiene que interceptar las llamadas sino que las máquinas virtuales son las encargadas de ponerse en contacto con él, este sistema reduce el "overhead" producido por la necesidad de evitar accesos no deseados, permitiendo un rendimiento cercano al obtenible en un sistema no virtualizado.



Xen desde un inicio es un hypervisor para-virtualizado, rasgo que le dotaba de un un alta rendimiento en la ejecución de máquinas virtuales debido al poco "overhead" producido por la "para-virtualizacion", al precio de no poder ejecutar instancias del S.O. de la familia Windows. Desde la version 3.0, Xen haciendo uso de las tecnologías de virtualización implementadas en los procesadores Intel y AMD (Intel VT y AMD Pacífica) puede lanzar instancias de los S.O. windows, la tecnología que hace uso de ambas implementaciones, se conoce con el nombre de HVM (Hardware Virtual Machine)

Ventajas de la Virtualizacion:

Aislamiento: las máquinas virtuales son totalmente independientes, entre sí y con el hypervisor. Por tanto un fallo en una aplicación o en una máquina virtual afectará únicamente a esa máquina virtual. El resto de máquinas virtuales y el hypervisor seguirán funcionando normalmente.

Seguridad: cada máquina tiene un acceso privilegiado (root o administrador) independiente. Por tanto, un ataque de seguridad en una máquina virtual sólo afectará a esa máquina.

Flexibilidad: podemos crear las máquinas virtuales con las características de CPU, memoria, disco y red que necesitemos, sin necesidad de "comprar" un ordenador con esas características. También podemos tener máquinas virtuales con distintos sistemas operativos, ejecutándose dentro de una misma máquina física.

Agilidad: la creación de una máquina virtual es un proceso muy rápido, básicamente la ejecución de un comando. Por tanto, si necesitamos un nuevo servidor lo podremos tener casi al instante, sin pasar por el proceso de compra, configuración, etc.

Portabilidad: toda la configuración de una máquina virtual reside en uno o varios ficheros. Esto hace que sea muy fácil clonar o transportar la máquina virtual a otro servidor físico, simplemente copiando y moviendo dichos ficheros que encapsulan la máquina virtual.

Recuperación: si se dispone de una copia de los ficheros de configuración de la máquina virtual, en caso de desastre la recuperación será muy rápida, simplemente arrancar la máquina virtual con los ficheros de configuración guardados. No es necesario reinstalar, recuperar backups y otros procedimientos largos que se aplican en las máquinas físicas.

Consolidacion del hardware: un mal extendido en los CPDs actuales es el gran número de servidores, muchos de ellos infrautilizados. Si se virtualiza un número de esos sistemas infrautilizados en un solo servidor físico, se ahorrará energía, espacio, capacidad de refrigeración y administración, debido a que se ha reducido el número de servidores físicos.

La Virtualizacion aporta a nuestro proyecto las siguientes ventajas:

- Permitir al alumno el total control de la instancia para la realización de prácticas; para analizar en profundidad el sistema operativo e incluso en ciertas practicas, se requiere disponer de permiso total en el sistema.
- Permitir al alumno disponer de un sistema complejo completo, por ejemplo un cluster para la realización de practicas de procesamiento paralelo.
- Permitir al alumno probar software en sistemas con diferentes configuraciones, versiones de librerías, sistemas operativos, etc...
- Configurar la estructura hardware de los puestos de trabajo por medio de la edición de un fichero de texto.
- Permitir al alumno reproducir la configuración del sistema en la universidad en su propia casa, poniendo a disposición del alumnado las imágenes virtuales para poder ejecutar instancias en su propia casa, siempre y cuando las licencias software lo permitan.
- La complejidad de creación de nuevos puestos de laboratorios para los alumnos se reduce sencillamente a copiar la imagen original y lanzar la instancia.

- Desacoplando el software instalado de los equipos hardware, se puede solicitar cualquier configuración disponible desde cualquier ordenador de la red, incluso desde el domicilio de los alumnos.
- El ahorro energético al consolidar no solo los servidores, sino los laboratorios. En una infraestructura normal de laboratorios, sin hacer uso de “Virtual Lab”, es necesario mantener laboratorios con distintas configuraciones abiertos y en funcionamiento para que estén disponibles para los alumnos. Considerando que a lo largo del día aproximadamente el 80% de los equipos que no están en uso permanecen encendidos el ahorro energético de tener esos laboratorios cerrados, con los ordenadores apagados, es considerable.

Java Server Pages

Java Server Pages (JSP) es una tecnología Java destinada a generar contenido web dinámicamente. Tiene una estrecha relación con los servlets, de hecho, es una forma alternativa de construirlos. Todo lo que puede hacer un servlet lo puede hacer un JSP y viceversa. Los JSP se ejecutarán en contenedores de servlets (en nuestro caso, Tomcat) y al ejecutarse lo que harán es crear un programa Java que será el que vaya generando el contenido web que finalmente será mostrado.

Principales ventajas:

•Alta portabilidad, ya que usa máquinas virtuales de Java.

•El uso de Java facilita la programación, ya que es un lenguaje muy extendido y su uso de clases facilita la comprensión y organización del código y del manejo de datos, así como la utilización de diversos APIs ya existentes para éste lenguaje.

•Usa también una combinación de código Java y HTML.

•Cada JSP se maneja en una hebra independiente, en su propio contexto.

•La hebra creada no se crea cada vez que se la llama, persiste de una llamada a otra.

Uso en VLAB:

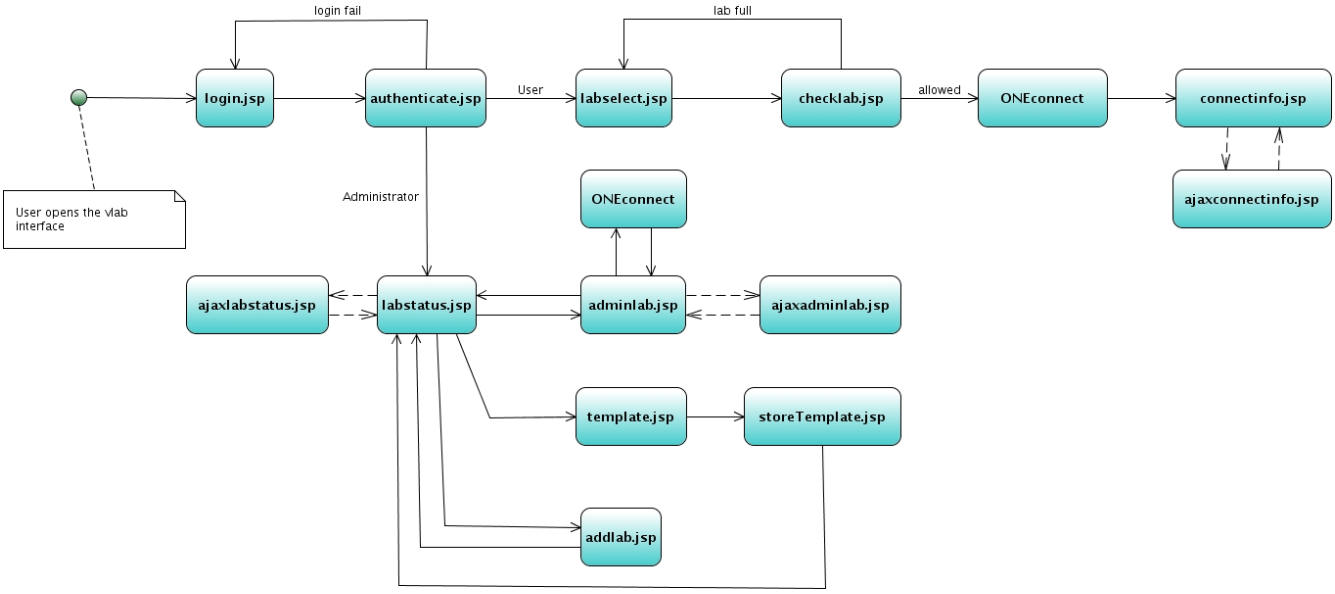
Visto que necesitábamos manejar bases de datos y ejecutar código en el propio servidor, optamos por JSP para la creación de toda la interfaz web, permitiéndonos conectar a la base de datos y llamar al servlet conector de Open Nebula para poder manejar el sistema de máquinas virtuales desde la interfaz.

Módulos del Sistema

Interfaz web

Debido a las características del sistema, desde un principio se vio claro que era necesaria una interfaz web. Como la problemática que intentamos resolver es la poca flexibilidad y malos horarios de los laboratorios físicos, una interfaz web es la solución más adecuada, permitiendo al alumno acceder a ella desde su propia casa.

La estructura general de la interfaz es la siguiente:



Como se puede observar, la estructura de la interfaz es medianamente compleja, con lo que explicaremos concretamente la función de cada componente:

●Login.jsp: es la pantalla inicial de la interfaz web. Aquí es donde accederán tanto alumnos como administradores para identificarse en el sistema. Una vez que se introducen los datos, estos se envían a “authenticate.jsp” para la comprobación de su veracidad.

●Authenticate.jsp: no es visible para el usuario final. Se encarga de comprobar en la base de datos de virtual lab si los datos introducidos son correctos. De no ser correctos, se envía al usuario de nuevo a “login.jsp” indicando el error. Si los datos son correctos, se genera un JavaBean que contendrá los datos del usuario para que éste sólo pueda acceder a las páginas que le están designadas (por ejemplo, un alumno no puede acceder a la página de administración de laboratorios), se comprueba el rol del usuario y se redirecciona a “labselect.jsp” si el usuario es un alumno o a “labstatus” si el usuario es un administrador del sistema.

●Labselect.jsp: en esta página el alumno elige el laboratorio al que se quiere conectar. Se extraen de la base de datos los laboratorios disponibles y se muestran en un desplegable para su elección. Una vez ha elegido un laboratorio y pulsa aceptar, la información se envía a “checklab.jsp”

●Checklab.jsp: no es visible para el usuario final. Se encarga de comprobar en la base de datos si el laboratorio seleccionado dispone de puestos libres. Si no es posible realizar la conexión (porque se ha excedido del número de máquinas permitidas o porque no quedan direcciones MAC libres), se redirige de nuevo a “labselect.jsp” informando al alumno del error. Si se puede realizar la conexión, los datos se envían al servlet “ONEconnect” para su proceso. Una vez que ONEconnect ha recibido la información, se redirige al alumno a “connectinfo.jsp”

●Connectinfo.jsp: esta pantalla mantiene informado al alumno sobre la conexión que acaba de pedir. Se le muestra el estado de su máquina, el puerto al que debe conectarse remotamente y el nombre del laboratorio. Aquí, el alumno debe esperar hasta que el estado sea “Running” para poder acceder a su puesto. La información mostrada se refresca automática y asíncronamente con “ajaxconnectinfo.jsp”

●Ajaxconnectinfo.jsp: no es visible para el usuario final. Se encarga de procesar las peticiones asíncronas de “connectinfo.jsp” y de enviarle la respuesta (ver el apartado tecnologías: Ajax para más detalles).

●Labstatus.jsp: es la pantalla inicial de administración. Aquí se muestran al administrador los tipos de laboratorios disponibles, un breve resumen de los mismos y la posibilidad de administrarlos y/o añadir nuevos tipos. La información mostrada se refresca automática y asíncronamente con “ajaxlabstatus.jsp”. Si se desea añadir un nuevo laboratorio, se llama a “addlab.jsp”. Si se desea modificar el template de uno de los tipos de laboratorio, se llama a “template.jsp”. Si se desea administrar uno de los tipos de laboratorio para ver información más concreta, se llama a “adminlab.jsp”

●Ajaxlabstatus.jsp: no es visible para el usuario final. Se encarga de procesar las peticiones asíncronas de “labstatus.jsp” y de enviarle la respuesta (ver el apartado tecnologías: Ajax para más

detalles).

●Addlab.jsp: en esta ventana se definen los parámetros del nuevo tipo de laboratorio a incluir y se almacena la información en la base de datos del sistema.

●Template.jsp: en esta ventana se puede ver el template asociado actualmente al tipo de laboratorio, así como su modificación o actualización como el administrador considere oportuno. Una vez que se pulsa el botón de aceptar, se envía la información a “storetemplate.jsp”

●Storetemplate.jsp: no es visible para el usuario final. Almacena en la base de datos la información enviada por “template.jsp”.

●Adminlab.jsp: en esta ventana se muestran las máquinas virtuales de cada tipo de laboratorio, así como su estado, su memoria actual, memoria máxima, la dirección física (MAC) y el puerto externo que se utiliza para su conexión. Aquí el administrador puede realizar las acciones de ONE sobre cada una de las máquinas desplegadas. La información mostrada se actualiza automática y asíncronamente con “ajaxadminlab.jsp”. Cuando el administrador realiza cualquiera de las acciones permitidas, éstas se envían al servlet ONEconnect para su proceso (ver el apartado ONEconnect para más información).

●Ajaxadminlab.jsp: no es visible para el usuario final. Se encarga de procesar las peticiones asíncronas de “adminlab.jsp” y de enviarle la respuesta (ver el apartado tecnologías: Ajax para más detalles).

●Logout.jsp: no está reflejado en el diagrama, pero es accesible desde cualquier página de la interfaz (exceptuando la de login). Se encarga de terminar la sesión del usuario y desconectarle de manera segura (Nota: el hecho de que un alumno cierre su sesión no implica que su máquina se apague y/o suspenda).

El conector OpenNebula

El sistema hace uso de OpenNebula Virtual Infrastructure Engine, un gestor de máquinas virtuales. Para poder usar sus características nos apoyamos en el API desarrollado en java, ya que la interfaz web está hecha en JSP y así desarrollamos un objeto Servlet como conector entre ONE y la interfaz.

En checklab.jsp, si se ha comprobado la disponibilidad de máquinas virtuales, se llamará al Servlet ONEconnect para que cree y ejecute una máquina virtual para ponerla a disposición del usuario.

En adminlab.jsp, el administrador dispone de varias opciones sobre las máquinas virtuales, que serán ejecutadas a través del mismo ONEconnect. Éstas opciones son encender, apagar, suspender, y activar (una máquina previamente suspendida).

Funcionamiento:

Al implementarse como un servlet, la clase extenderá la clase HttpServlet. Por ello tendrá un método doGet y un doPost, que se encargarán de recoger el request y realizar la acción correspondiente, así como un init que en éste caso inicializará la conexión a la base de datos que se usará (MySQL).

Los otros métodos que controlarán el funcionamiento de OpenNebula son:

- allocateVM(HttpServletRequest request, HttpServletResponse response): realiza un allocate de la máquina virtual, es decir, la crea en un host (controlado por OpenNebula) con los parámetros obtenidos de la base de datos (template incluido) y actualiza el puerto asignado en la base de datos.

- actionVM(OpenNebulaConstants.VMAction action, HttpServletRequest request, HttpServletResponse response): realiza la acción indicada en el parámetro action, en éste caso suspend, shutdown y resume están soportadas.

- infoVM(HttpServletRequest request, HttpServletResponse response):realiza el equivalente a una llamada info de OpenNebula, mostrando el resultado por pantalla.

Servidor DHCP

Para configurar la red en las máquinas virtuales, hacemos uso de un servidor DHCP, en nuestro caso decidimos usar "dhcp3-server". La asignación de IP's a las máquinas virtuales se realiza por medio de pares "[MAC - IP]" de tal forma que toda máquina virtual con una determinada MAC (que se asigna al crearla en la aplicación) tenga siempre la misma dirección IP. Para ésta asociación hemos realizado una equivalencia entre MAC e IP asignada, de tal modo que los últimos 4 bytes de la MAC se traducen en la dirección IP que se le asignará a la máquina. Aunque usamos una aproximación similar a la utilizada en OpenNebula en la asociación de IP y MAC , en el proyecto no se hace uso de las Redes Virtuales de ONE.

Demonio de sincronización

Dado que OpenNebula y nuestro sistema tienen bases de datos separadas, había que encontrar un mecanismo de sincronización entre ambas bases de datos. Para ello hemos creado un pequeño programa demonio (se ejecuta cada cierto tiempo automáticamente en el sistema) que se encarga de ésta actualización.

Está programado en C++ y realiza consultas a la base de datos sqlite3 de OpenNebula (guardada en el archivo one.db) y actualiza la información de estado y memoria en uso de la máquina virtual en la tabla vms de nuestra base de datos MySQL con los valores obtenidos de la tabla vm_pool. Utiliza para ello una clase llamada dbupdate.

Estructura de la clase dbupdate:

La clase dbupdate se utiliza para la conexión con ambas bases de datos y ejecutar la consulta y actualización de las mismas. Consta de las siguientes funciones:

•dbupdate(): constructor, inicializa las conexiones con las bases de datos,

•start(): método de la clase que realiza la consulta de la base de datos SQLite3 y la actualización de la base de datos MySQL. Al ejecutarse la operación sqlite3_exec(db, "SELECT * FROM vm_pool",callback,this,&zErrMsg) con la operación de consulta, realizará la función callback (de la que hablaremos más adelante) a cada fila del resultado de la consulta.

•update(char* stat, char* vmid, char* mem): realiza la actualización sobre la base de datos MySQL con los datos de cada linea de la consulta a la base de datos sqlite3, cogiendo los datos de estado y memoria en uso. En caso de que el estado sea "available" reiniciará los datos de la base de datos MySQL.

•~dbupdate(): destructor de la clase.

Fuera de la clase tenemos dos funciones que son utilizadas:

•stateConvert(char* state, char* lcm_state): devuelve un char* con el estado en forma de cadena de caracteres, pues la base de datos de OpenNebula guarda el estado en forma numérica (aunque la consulta los devuelva como cadenas de caracteres, siguen siendo números) mediante los valores de state y lcm_state.

•callback(void *NotUsed, int argc, char **argv, char **azColName): se ejecutará a cada línea de la consulta SQLite 3, y en éste caso hemos sacado de cada línea los valores de oid(identificador de máquina virtual), memory(memoria en uso), state (estado) y lcm_state(estado dentro de su ciclo de vida) para llamar al método update de la clase dbupdate con los parámetros necesarios para la actualización.

Instalación y configuración de “Virtual Lab”

En esta sección pasamos a describir los pasos para instalar y configurar la infraestructura necesaria para la ejecución de un laboratorio virtual.

Ha sido instalado y probado en servidores Ubuntu 8.10 “Intrepid Ibex” y 9.04 “Jaunty Jacklope”.

Requisitos:

- OpenNebula Virtual Infrastructure Engine >= 1.2
- OpenNebula Java Client >= 1.0
- Java Development Kit 6
- Apache-Tomcat >= 6.0
- Mysql server >= 5.0
- DHCP Server (dhcp3)
- libmysql++-dev >= 3.0
- libsqlite3-dev >= 3.5
- bridge-utils
- iptables

Instalación de los componentes:

Instalación de OpenNebula.

- Ubuntu 9.04, instalación por medio de paquetes Ubuntu:

  https://help.ubuntu.com/community/OpenNebula

- Ubuntu 8.10, instalación según la documentación de "Open Nebula" :

  http://www.opennebula.org/doku.php?id=documentation

Instalacion de Apache - Tomcat.

En general se puede proceder a instalarlo por medio de los paquetes de la distribución correspondiente, pero en ubuntu 8.10 tuvimos problemas, así que realizamos una instalación manual. Para mas información consultar la documentación correspondiente de la página del proyecto:

http://tomcat.apache.org/

Instalación del Interfaz Web.

Despliegue del código en la carpeta “webapps” del directorio de instalación de Tomcat

```
$TOMCAT_HOME/webapps/interface
```

Instalación de las librerias y utilidades.

Dependencias del demonio de sincronización:

```
$ sudo aptitude install "libmysql++-dev" "libsqlite3-dev"
```

Utilidades de red, para la configuración de las MV:

```
$ sudo aptitude install dhcp3-server bridge-utils
```

Configuración:

Configuración de la Base de datos MySQL.

El proyecto requiere de unas ciertas características en la base de datos MySQL, la existencia del usuario “vlab” con password “one” y una base de datos “vlab” con las tablas “users”, “labs”, “macs” y “vms”. Para más información sobre la estructura de cada una de las tablas, los campos que lo componen y su utilización en la aplicación, remitirse a la sección “Tecnologías utilizadas – MySQL “ (Poner número de sección).

Nos conectamos con un usuario que tenga privilegios.

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.0.67-0ubuntu6 (Ubuntu)
mysql>
```

Creación de la base de datos “vlab”

```
mysql> CREATE DATABASE vlab;
mysql> USE vlab;
Database changed
```

Creación de la tabla “users”

```
mysql> CREATE TABLE `vlab`.`users` (`id` INT NOT NULL,`password` VARCHAR(50)
       NOT NULL,`fname` VARCHAR(50) NOT NULL,`lname` VARCHAR(50) NOT NULL,
       `admin` BIT NOT NULL DEFAULT 0,`idVM` INT NOT NULL,PRIMARY KEY
       (`id`))ENGINE = MyISAM;
```

Creación de la tabla “labs”

```
mysql> CREATE TABLE `vlab`.`labs` (`name` VARCHAR(50) NOT NULL,`template`
       LONGTEXT NOT NULL,`running` INT NOT NULL,`allowed` INT NOT NULL,`port`
       INT NOT NULL,PRIMARY KEY (`name`))ENGINE = MyISAM;
```

Creación de la tabla “macs”

```
mysql> CREATE TABLE `vlab`.`macs` (`mac` varchar(50) NOT NULL,`free` BIT NOT
       NULL,PRIMARY KEY (`mac`))ENGINE = MyISAM;
```

Creación de la tabla “vms”

```
mysql> CREATE TABLE `vlab`.`vms` (`id` INT NOT NULL,`lab` VARCHAR(50) NOT
       NULL,`status` VARCHAR(15) NOT NULL,`usedMem` INT NOT NULL,`maxMem` INT
       NOT NULL,`port` INT NOT NULL,`ONEid` INT NOT NULL,`MAC`
       VARCHAR(50),PRIMARY KEY (`id`))ENGINE = MyISAM;
```

Creación del usuario que Virtual Lab utiliza para administrar la base de datos, lanzamos la sentencia GRANT, indicando los permisos que otorgamos, la base de datos y los objetos de la misma sobre los que estamos asignando privilegios, el nombre del usuario y el password.

```
mysql> GRANT SELECT, INSERT ON vlab.* TO 'vlab'@'localhost' IDENTIFIED BY
'one';
```

Para terminar se debe configurar la cuenta de administrador de la aplicación, añadiéndola a la tabla “users” con el “flag” de administrador a “1”.

```
mysql> INSERT INTO users VALUES($id,$password,$first_name,$last_name,1,0)
```

El resto de usuarios se añaden a través de la interfaz del administrador.

Configuración de Open Nebula:

Se requiere la configuración de Open Nebula en función de la infraestructura hardware que va a dar soporte a el Laboratorio Virtual, toda la información relativa a como configurar Open Nebula para ofrecer un "Cloud" privado se encuentra en la documentación del proyecto:

http://www.opennebula.org/doku.php?id=documentation

Configuración del Sistema:

Además de la configuración de Open Nebula, se deben configurar el sistema. Estas instrucciones son relativas a un sistema Ubuntu o similar (que haga uso del comando sudo para ejecutar aplicaciones con permisos de super-usuario).

El script encargado de realizar dinámicamente la configuración NAT y de redirección de puertos para acceder remotamente a las máquinas virtuales hace uso de "iptables", esta aplicación debe ser ejecutada con permisos de superusuario. Para que este script se ejecute correctamente, debemos configurar "sudo" para que no requiera la contraseña del usuario al ejecutar iptables:

Modificar el fichero de configuración de la utilidad “sudo”:

```
$ sudo visudo
```

Introducir el siguiente Cmnd_Alias:

```
"Cmnd_Alias IPTABLES=/sbin/iptables".
```

Conceder permiso al usuario para poder ejecutar "iptables" sin requerir contraseña.

```
$NOMBRE_DE_USUARIO ALL=NOPASSWD:IPTABLES
```

Ejemplo de fichero de configuración:

```
 1 # /etc/sudoers
 2 #
 3 # This file MUST be edited with the 'visudo' command as root.
 4 #
 5 # See the man page for details on how to write a sudoers file.
 6 #
 7 Defaults  env_reset
 8
 9 # Host alias specification
10 # User alias specification
11 # Cmnd alias specification
12 Cmnd_Alias IPTABLES=/sbin/iptables
13
14 # User privilege specification
15 root  ALL=(ALL) ALL
16
17 # Uncomment to allow members of group sudo to not need a password
18 # (Note that later entries override this, so you might need to move
19 # it further down)
20 # %sudo ALL=NOPASSWD: ALL
21
22 # Members of the admin group may gain root privileges
23 %admin ALL=(ALL) ALL
24
25 # Gran access to Cmnd_Alias to user abel
26 abel ALL=NOPASSWD:IPTABLES
```

Configuración de la Red:

Para configurar la estructura de red del proyecto, hacemos uso de las aplicaciones incluidas en el paquete "bridge-utils" que nos permiten crear un interfaz bridge al que conectar tanto interfaces físicas del servidor como interfaces virtuales pertenecientes a las máquinas virtuales. “Iptables” se encarga de realizar las traducciones NAT y la redirección de puertos que permite a los alumnos conectarse a las máquinas virtuales desde su domicilio.

Crear el interfaz bridge:

```
$ sudo brctl addbr br5
```

Iniciarlo y configurarlo:

```
$ sudo ip link set br5 up
$ sudo ifconfig br5 192.168.0.1
```

Nota: por defecto la aplicación conecta las máquinas virtuales al bridge con nombre "br5"

Configuración iptables:

Se considera que la interfaz con acceso a internet es “eth0”, modificar las reglas con la configuración de red correspondiente al servidor donde se ha desplegado “virtual Lab”.

Activamos forwarding y masquerade:

```
$ sudo iptables -t nat -A POSTROUTING -o br0 -j MASQUERADE
```

Aceptar Forwarding de paquetes entre las interfaces:

```
$ sudo iptables -A FORWARD -P -j ACCEPT
```

Aceptar Paquetes ICMP:

```
$ sudo iptables -A INPUT -i eth0 -p ICMP -j ACCEPT
```

Nota: esto es recomendable a la hora de depurar, se pueden rechazar en un sistema en producción.

Acepto paquetes destinados al interfaz web:

```
$ sudo iptables -A INPUT -p TCP -i eth0 –dport 8080 -j ACCEPT
```

Acepto paquetes de conexiones ya establecidas :

```
$ sudo iptables -A INPUT -p TCP -m state --state RELATED -j ACCEPT
```

Configurar acciones por defecto:

```
$ sudo iptables -P INPUT DROP
$ sudo iptables -P OUTPUT DROP
```

Activar forwarding a nivel kernel:

```
$ sudo sysctl -e net.ipv4.conf.all.forwarding=1
```

Servidor DHCP:

Para configurar la red en las máquinas virtuales, hacemos uso de un servidor DHCP, en nuestro caso decidimos usar "dhcp3-server". La siguiente configuración corresponde a dicho servidor, pero se puede utilizar otro servidor cuya configuración será análoga, remitirse a la documentación correspondiente.

La asignación de IP's a las máquinas virtuales se realiza por medio de pares "[MAC - IP]" de tal forma que toda máquina virtual con una determinada MAC (que se asigna al crearla en la aplicación) tenga siempre la misma dirección IP.

La configuración del servidor se realiza modificando el fichero "/etc/dhcp3/dhcpd.conf" con los valores deseados, por ejemplo el siguiente fichero de configuración asigna 5 direcciones IP a MACs en el intervalo 02:D2:C0:A8:00:0A - 02:D2:C0:A8:00:AE en la red 192.168.0.0.

```
 1 ddns-update-style none;
 2
 3 log-facility local7;
 4
 5 subnet 192.168.0.0 netmask 255.255.255.0 {
 6 range 192.168.0.20 192.168.0.200;
 7 option domain-name-servers 80.58.61.250;
 8 option routers 192.168.0.1;
 9 option broadcast-address 192.168.0.255;
10 default-lease-time 600;
11 max-lease-time 14400;
12 host vm0{
13     hardware ethernet 02:D2:C0:A8:00:0A;
14     fixed-address 192.168.0.10;
15 }
16 host vm1{
17     hardware ethernet 02:D2:C0:A8:00:0B;
18     fixed-address 192.168.0.11;
19 }
20 host vm2{
21     hardware ethernet 02:D2:C0:A8:00:0C;
22     fixed-address 192.168.0.12;
23 }
24 host vm3{
25     hardware ethernet 02:D2:C0:A8:00:0D;
26     fixed-address 192.168.0.13;
27 }
28 host vm4{
29     hardware ethernet 02:D2:C0:A8:00:0E;
30     fixed-address 192.168.0.14;
31 }
32 }
```

A continuación tenemos que configurar la interfaz donde se servirán las peticiones DHCP, editamos el fichero /etc/default/dhcp3-server y modificamos la siguiente linea con el interfaz correspondiente, por defecto el bridge donde están conectadas las máquinas virtuales, br5:

```
 8
 9 # On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
10 # Separate multiple interfaces with spaces, e.g. "eth0 eth1".
11 INTERFACES="br5"
12
```

Ejecución de "Virtual Lab"

Iniciar Servidor DHCP:

```
$ sudo /etc/init.d/dhcp3-server start
```

Iniciar el servidor contenedor servlet (tomcat):

```
$ sudo /etc/init.d/tomcat start
```

Iniciar "Open Nebula":

```
$ one start
```

Iniciar demonio de sincronización entre la aplicación y Open Nebula:

```
$TOMCAT_HOME/webapps/interface/WEB-INF/misc/ddbbupd &
```

Nota: $TOMCAT_HOME/ es el directorio de instalación del contenedor TOMCAT.

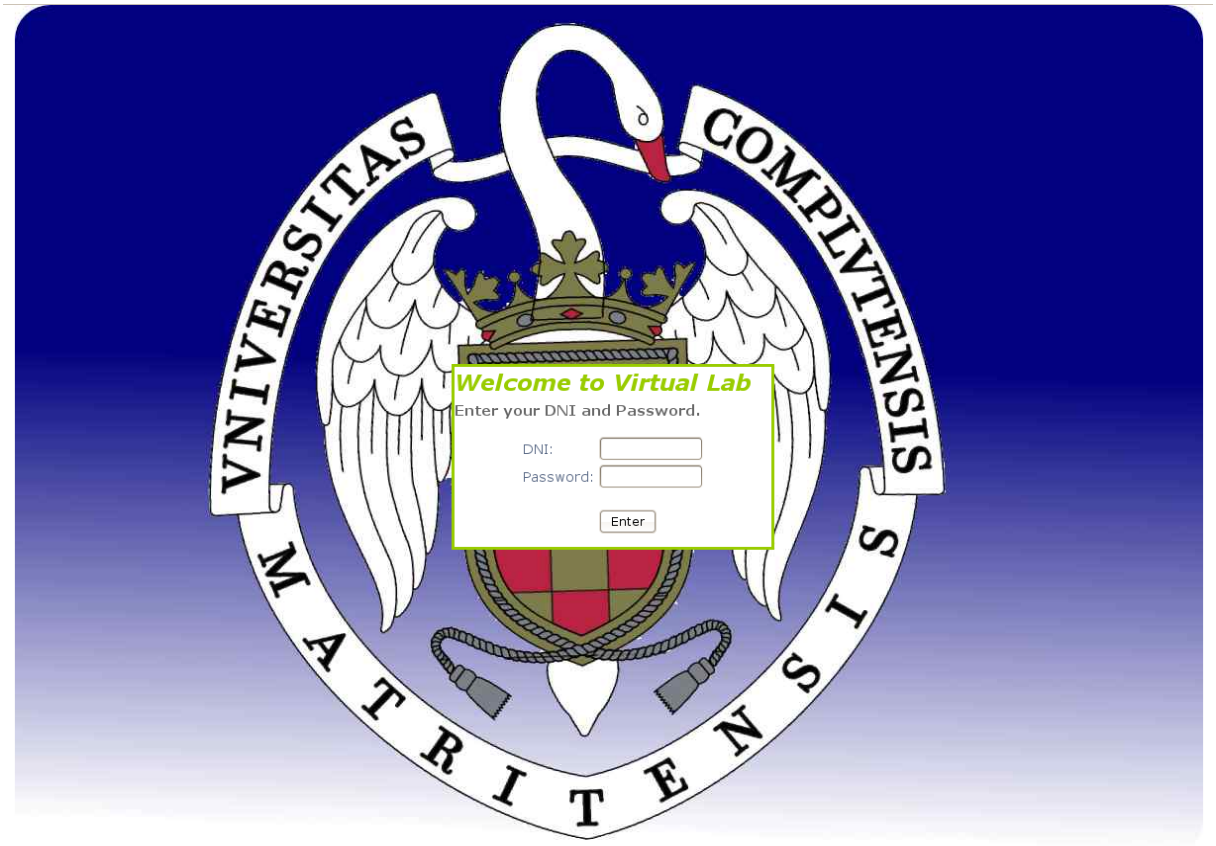
La aplicación esta lista para usarse, se puede probar conectándose con un navegador WEB en la máquina donde está instalado "Virtual Lab" a:

```
http://localhost:8080/interface
```

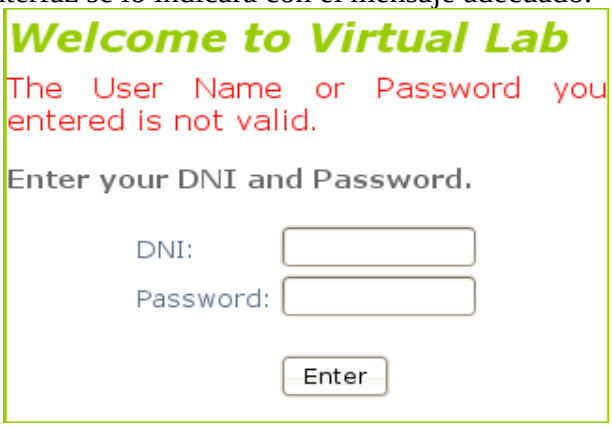
Una vez dados de alta a los alumnos y administradores, se puede dar la dirección web que determina el servidor “Virtual Lab” para su uso.

Manual de Vlab

Al abrir la interfaz de Virtual Lab, se accede a la pantalla de login:

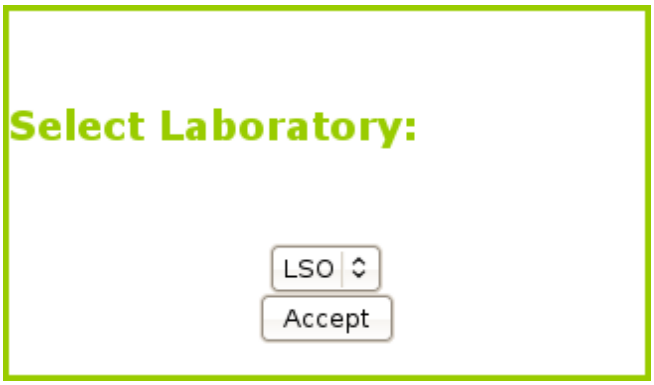


Aquí, debe introducir su DNI y la contraseña suministrada por los técnicos. Si introduce información incorrecta, la interfaz se lo indicará con el mensaje adecuado.

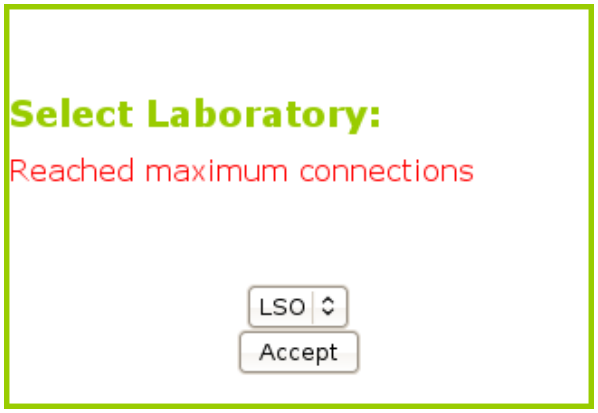


Manual de usuario

Una vez que acceda a la interfaz de usuario, se le mostrará la siguiente página:

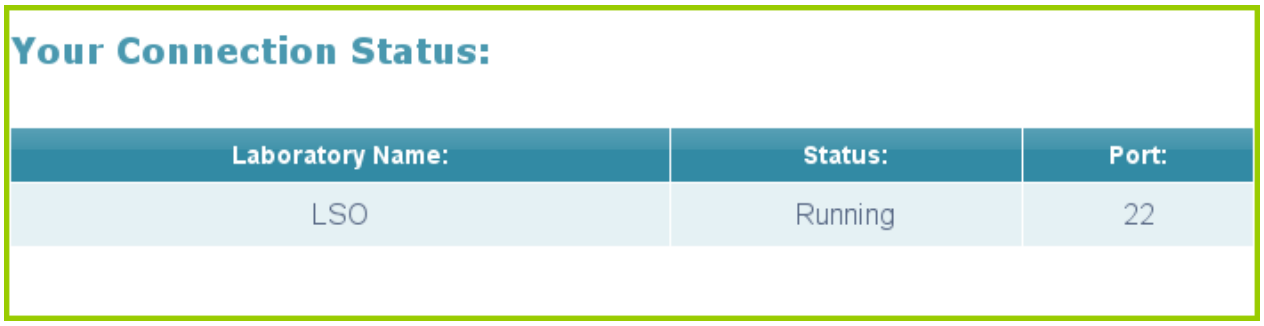


Seleccione en el desplegable uno de los laboratorios disponibles y pulse el botón de aceptar. Si ha seleccionado un laboratorio que está lleno, la interfaz le mostrará este mensaje:



Si recibe el mensaje “Reached maximum connections” significa que el número de máquinas disponibles para el laboratorio ha llegado al máximo con lo que tendrá que esperar a que se libere algún puesto, también puede solicitar al profesor que aumente el número de puestos disponibles. Si se le muestra el mensaje “Unable to get a free lab”, póngase en contacto con el administrador del sistema.

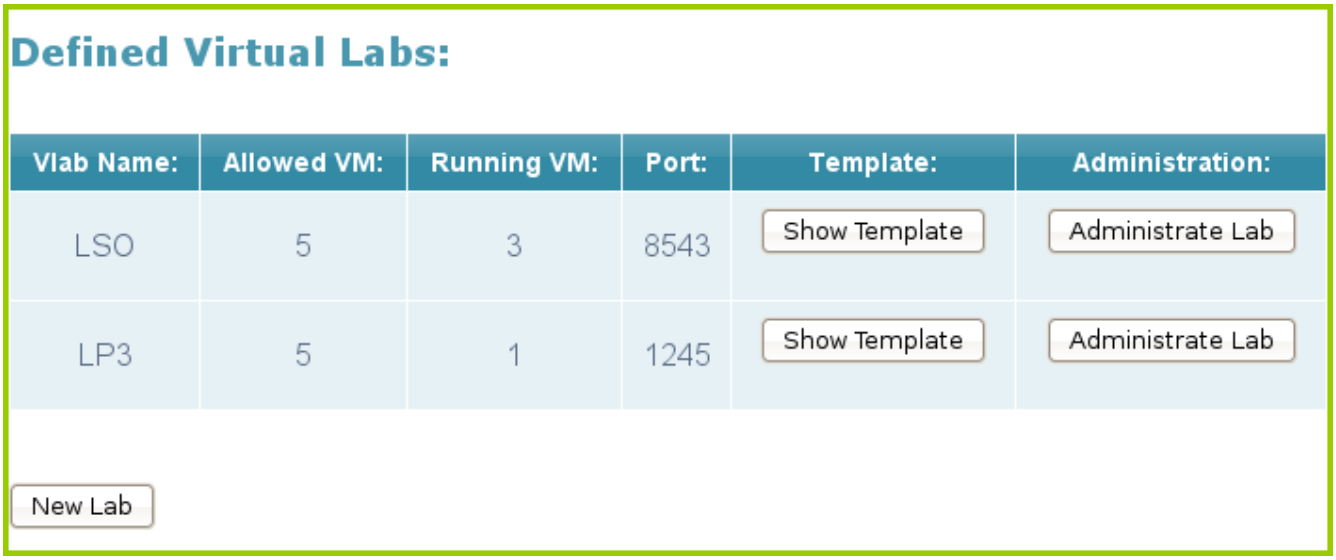
Si todo ha ido correctamente, se le mostrará la siguiente pantalla:



Aquí debe esperar a que el estado de su máquina virtual sea 'Running' (la página se actualizará automáticamente). Si el estado de la máquina nunca pasa a 'Running' después de aproximadamente dos minutos, póngase en contacto con el administrador del sistema. Una vez que la máquina esté ejecutada, podrá conectarse a ella mediante escritorio remoto (o el servicio configurado por el profesor) a través del puerto indicado.

Manual de administrador

Una vez que acceda a la interfaz de administrador, se le mostrará la siguiente pantalla:



Aquí se le mostrarán los laboratorios definidos en el sistema. La información mostrada de cada laboratorio es:

●Vlab Name: nombre del laboratorio.

●Allowed VM: número máximo de máquinas virtuales permitidas para ese tipo de laboratorio.

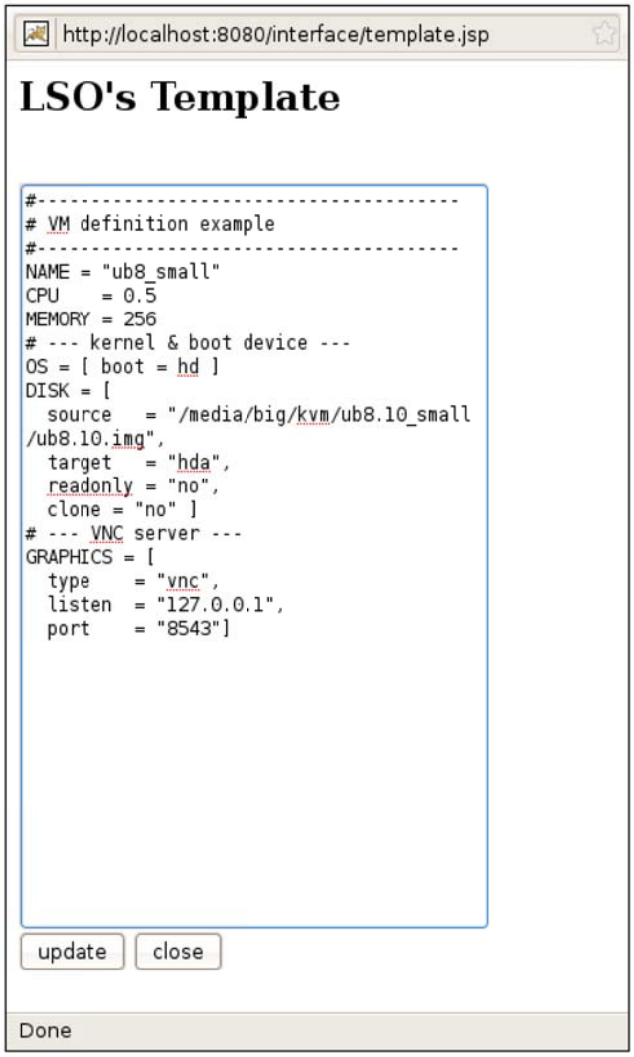
●Running VM: número actual de máquinas virtuales en ejecución. Si el número es igual o superior a las permitidas, los alumnos no podrán ejecutar otro laboratorio hasta que uno quede libre o el administrador aumente el número máximo de permitidas.

●Port: el puerto del servicio que las máquinas virtuales ofrecen para conectarse a las mismas, VNC, Remote Desktop de Windows, ssh, http, etc...

●Show Template: el template de ONE que define los rasgos de las máquinas virtuales disponibles en dicho laboratorio

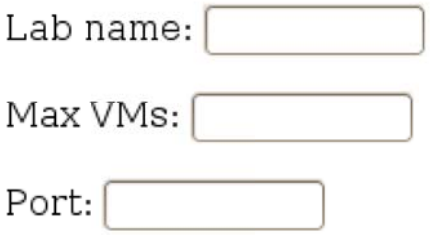
●Administrate Lab: lleva a la pantalla de administración de cada instancia de máquina virtuales.

Si pulsa 'Show Template' se le mostrará la siguiente pantalla:



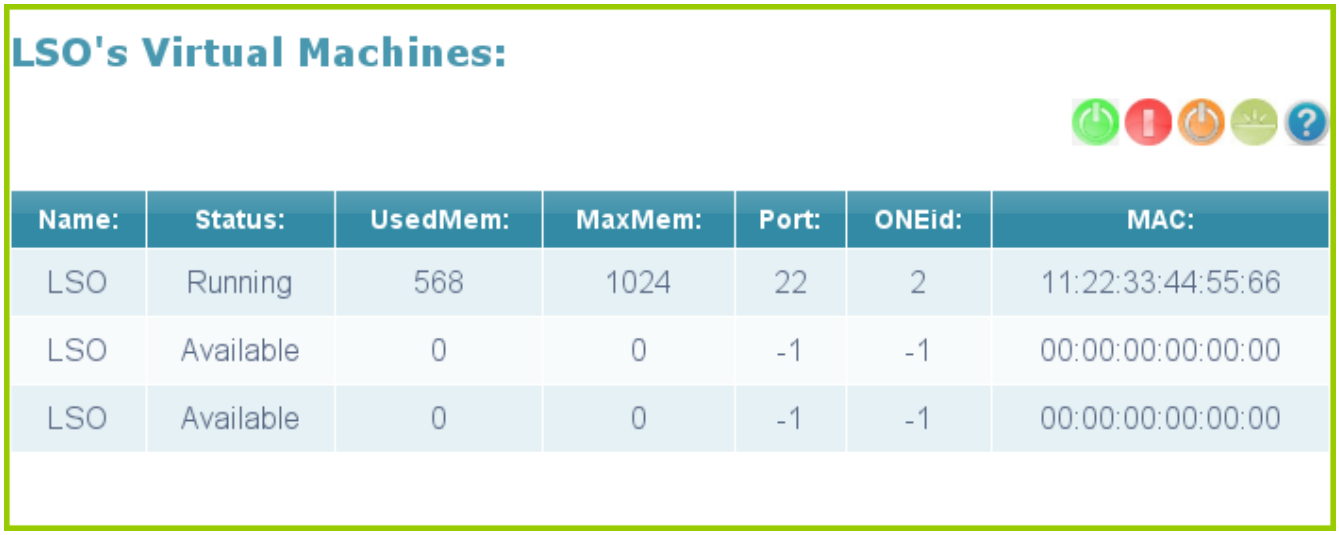
Aquí podrá editar el template asociado a cada tipo de laboratorio por si necesita ser modificado una vez creado.
Tenga en cuenta que las modificaciones introducidas serán efectivas para las máquinas virtuales que sean lanzadas desde el momento de la actualización. Las que se encuentren ya en ejecución mantendrán la anterior definición. Se recomienda apagar todas las máquinas antes de realizar una actualización de template. Cuando haya terminado de editarlo, pulse 'update' para guardarlo en la base de datos, o pulse 'close' para salir sin guardar los cambios

Estando en la Interfaz del Administrador, si pulsa en 'New Lab', se le abrirá una nueva ventana en la que podrá introducir los datos correspondientes del nuevo laboratorio:



Inserte aquí el nombre del nuevo laboratorio, el número máximo de máquinas permitidas que tendrá y el puerto del servicio que quiera ofrezcan las mismas. Introduzca también el template de ONE en el cuadro de texto (no aparece en la ilustración) y pulse 'Ok' cuando haya terminado.

Si desde la ventana principal de administración pulsa el botón 'Administrate Lab' en cualquiera de los tipos de laboratorio, llegará a esta pantalla:



Aquí se le mostrarán cada una de las máquinas virtuales que se hayan definido y/o que estén en ejecución. Para cada máquina se muestran los valores:

- Name: nombre del laboratorio.
- Status: estado de la máquina
- UsedMem: memoria en uso de una máquina en ejecución.
- MaxMem: memoria máxima permitida para la máquina virtual
- Port: puerto externo al que hay que acceder para conectarse a la máquina
- ONEid: el id de OpenNebula asignado a la máquina virtual
- MAC: dirección física de la máquina virtual.

Nota: Es importante distinguir el puerto, del servicio que ofrece la MV, interno a la red virtual (el definido en el Laboratorio) y el puerto externo que en el servidor "Virtual Lab" ha sido redireccionado para conectarse a una instancia particular del laboratorio (dicho puerto es el que aparece en esta pantalla).

Para realizar las tareas de administración, debe seleccionar una de las filas de la tabla (la que corresponda al laboratorio a administrar), cambiando su aspecto a este:

Name:	Status:	UsedMem:	MaxMem:	Port:	ONEid:	MAC:
LSO	Running	568	1024	22	2	11:22:33:44:55:66

Una vez seleccionado la máquina virtual, puede hacer uso de los botones de administración:

- Enciende la máquina virtual seleccionada.

- Apaga la máquina seleccionada.

- Suspende la máquina seleccionada

- Activa la máquina previamente suspendida

- Obtiene la información de la máquina virtual

Posibles mejoras al sistema actual

•Ejecutar máquinas virtuales en máquinas locales concretas a elección (por motivos de rendimiento), usando los equipos de laboratorio como hosts de ONE donde lanzar las máquinas virtuales, la interfaz se encargaría de identificar el host desde donde se hace la petición e indicar a ONE que despliegue allí la máquina virtual. La máquina virtual se ejecutaría en el puesto del laboratorio y la conexión sería local.

•Tener particiones de disco asociadas a cada alumno donde poder almacenar información académica, prácticas, documentación, etc... que se montaría en las máquinas virtuales que solicita el alumno automáticamente.

•Asignación de múltiples máquinas virtuales a un usuario, estructuras complejas de entrenamiento, templates que definan un conjunto de máquinas en lugar de una única instancia.

•Permitir la configuración de ciertos rasgos de la aplicación por medio de ficheros de configuración, como por ejemplo: el rango de puertos externos para conectarse a las máquinas virtuales o IPs a asignar en las máquinas virtuales, el usuario de la base de datos y contraseña que la aplicación usara, etc...

•Mostrar al profesor un listado de máquinas ya configuradas con detalles de las mismas en lugar de tener que introducir la información del template a mano.

•Mostrar al alumno, solo los laboratorios donde está matriculado.

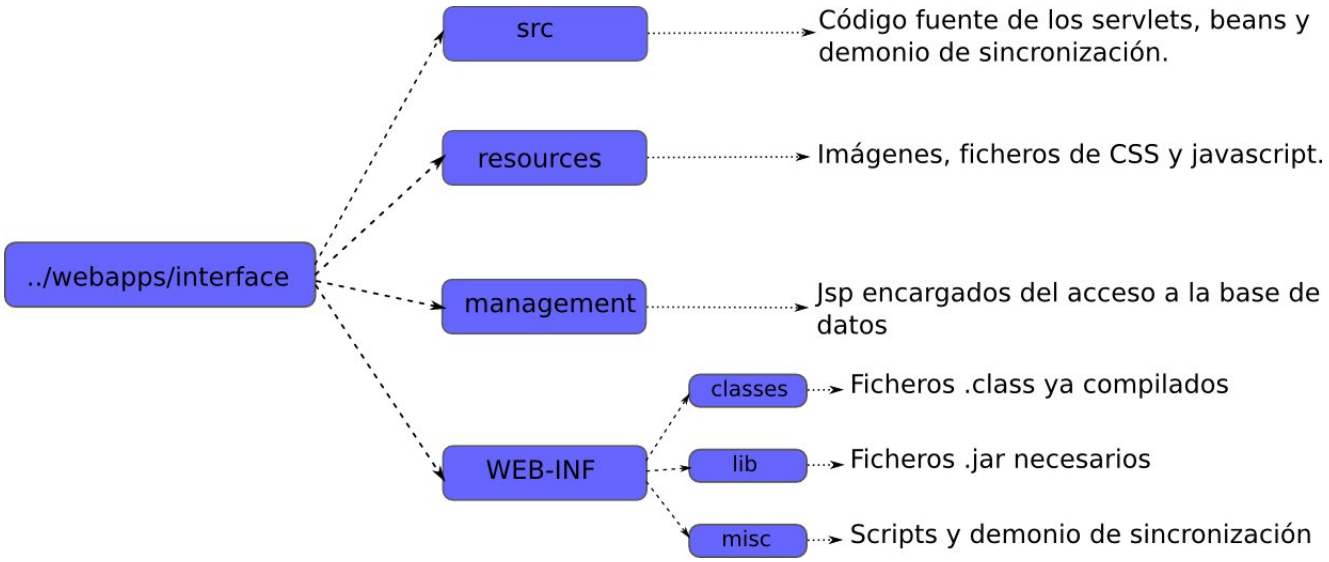
•Acceso seguro a la interfaz web haciendo uso de protocolos encriptados como https.

Bibliografía

“Open Nebula” website: http://www.opennebula.org/
"Xen User guide" http://www.xen.org
"Kvm User guide" http://www.linux-kvm.org
"Full Virtualization vs Para-Virtualization" http://www.techweek.es
Wikipedia http://www.wikipedia.org
MySQL Official Page: http://www.mysql.com
Online web tutorials: http://www.w3schools.com/
“Java Server Pages, Third Edition” Hans Bergsten, editorial O'Reilly.
“Beginning J2EE 1.4: From Novice to Professional”, editorial Apress.

Anexo A

Diagráma de despliegue del proyecto:



Configuración del proyecto web:

Fichero WEB-INF/web.xml donde se configuran diferentes parámetros del servicio Web en el servidor TOMCAT, entre las opciones configurables se encuentran:

- Página de inicio, *linea 18.*
- Conector con la base de datos (para utilizar otra base de datos), *linea 10.*
- Nombre de la base de datos, *linea 10.*
- Parametros de acceso a la Base de datos, usuario y password, *linea 10.*
- Url asociada al servlet oneconnect, *linea 27.*

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
    version="2.5">
 3   <display-name>home</display-name>
 4   <!-- Used by the JSTL database actions -->^M
 5   <context-param>
 6     <param-name>
 7       javax.servlet.jsp.jstl.sql.dataSource
 8     </param-name>
 9     <param-value>
10       jdbc:mysql:///vlab,org.gjt.mm.mysql.Driver,vlab,one
11     </param-value>
12   </context-param>
13   <context-param>
```

```
14      <param-name> jdbcURL  </param-name>
15      <param-value>  jdbc:mysql:///vlab </param-value>
16    </context-param>
17    <welcome-file-list>
18      <welcome-file>login.jsp</welcome-file>
19    </welcome-file-list>
20    <servlet>
21      <description></description>
22      <servlet-name>oneconnect</servlet-name>
23      <servlet-class>ONEconnect</servlet-class>
24    </servlet>
25    <servlet-mapping>
26      <servlet-name>oneconnect</servlet-name>
27      <url-pattern>/ONEconnect/*</url-pattern>
28    </servlet-mapping>
29 </web-app>
```