UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN AUTÓNOMA DE MÁQUINAS VIRTUALES

Alejandro Chillarón Oliva

Septiembre 2011

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN AUTÓNOMA DE MÁQUINAS VIRTUALES

AUTOR: Alejandro Chillarón Oliva
TUTOR: Jorge E. López de Vergara Méndez

Grupo High Performance Computing and Networking
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2011

Agradecimientos

Quiero comenzar agradeciendo a Jorge E. López de Vergara Méndez, tutor de este Proyecto Fin de Carrera, el tiempo dedicado en las largas reuniones que hemos mantenido y el sin fin de emails intercambiados durante los 19 meses que ha durado este trabajo. Quiero destacar su organización y el grado de perfección que es necesario alcanzar para conseguir su aprobación. Gracias Jorge por haberme dado gran libertad en la toma de decisiones de este Proyecto, así como las facilidades proporcionadas para su desarrollo.

Después de estos duros meses en los que no ha sido posible un desarrollo normal de mi vida, debo agradecer a Eli su apoyo y ánimos, fundamentales para finalizar esta etapa. Sé que eres la persona que más ha sufrido mi falta de tiempo, pero gracias a ti he tomado todas las decisiones que ahora podemos decir que han sido acertadas y que esperemos que hagan que nos compense el tiempo empleado. Sin duda este Proyecto Fin de Carrera se ha terminado gracias a tu forma de ser y por eso espero que te sientas orgullosa de lo que juntos hemos conseguido.

Mis padres, mi hermana, mi hermano y mi familia. Sé que tenéis tantas o más ganas que yo de que esto termine, pues por fin lo hemos logrado. Gracias por hacerme caso en todas las historias que os he contado y no entendíais ni una palabra, incluso hay alguien por ahí que se leyó esta memoria entera.

Mis amigos y compañeros de la carrera: Diego, Jose y Salinas. Siempre he pensado que terminar nuestros estudios era un trabajo de equipo y que yo solo no lo lograría. Hoy puedo decir que esto es cierto. Quiero agradeceros todos los momentos que hemos vivido juntos.

Son muchas las personas que se han interesado por el estado de este Proyecto durante su duración, a todos ellos debo agradecer ese interés y comprensión.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen

La tendencia actual en los servicios de IT es que los usuarios, las compañías y las grandes corporaciones accedan a sus recursos computacionales a través de la red en lo que se ha llamado Computación en la Nube, permitiendo una mejor utilización del hardware y reduciendo costes. En este tipo de infraestructuras, es común desplegar máquinas virtuales para satisfacer los requerimientos de los usuarios. Por este motivo, también es importante proveer la potencia de procesamiento según la demanda, realojando la carga de cada máquina virtual en servidores que la puedan gestionar de forma adecuada. Sin embargo, esta tarea puede convertirse en un reto. Para solucionarlo, se ha diseñado e implementado un sistema autónomo para gestionar estas máquinas virtuales de la siguiente manera: todos los servidores monitorizan su rendimiento y contabilizan cada máquina virtual que se está ejecutando en ellos. Esta información es compartida entre todos los servidores del sistema. Cuando un servidor está sobrecargado, éste busca los mejores servidores en el sistema para realojar sus máquinas virtuales, tratando de resolver el problema de sobrecarga. El sistema sigue el bucle Monitorización-Análisis-Planificación-Ejecución, satisfaciendo las reglas de auto-configuración, auto-cura, auto-optimización y auto-protección. En este Proyecto Final de Carrera se presenta la arquitectura del sistema desarrollado y se proporcionan los resultados obtenidos en las pruebas realizadas.

Abstract

Current trend in IT services is that users, companies and large corporations access computing resources through the network in what is named Cloud Computing, enabling a better utilization of hardware and reducing costs. In this type of infrastructures, it is common to deploy virtual machines to deal with the users' requirements. Then, it is also important to provide the necessary computing power when needed, allocating the load of each virtual machine in a server that can handle it. However, this can be a challenging task. For this, it has been designed and implemented an autonomic system to manage these virtual machines in the following way: all servers monitor their performance as a whole and also accounting each virtual machines running on them. This information is shared among all the servers in the system. When one server is overloaded, it looks for the better servers in the system where to offload those virtual machines, trying to solve this problem. The system follows the Monitor-Analysis-Plan-Execute loop, fulfilling the self-configuration, self-healing, self-optimization, and self-protection rules. In this project it is presented the architecture of the developed system, and provided the results of the tests done.

Palabras clave

Máquina virtual, virtualización, gestión autónoma, KVM

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

La tendencia actual en el campo de los servicios de Tecnologías de la Información es que los usuarios, las empresas y las grandes corporaciones accedan a los recursos computacionales a través de la red (Internet o Intranet), lo que se ha denominado Computación en la Nube o "Cloud Computing" [1]. Esta tecnología elimina la necesidad de instalar máquinas localmente y permite utilizar otras más potentes y con mayor capacidad de almacenamiento. Dicha infraestructura de computación por lo general se sirve de manera virtualizada, mediante el uso de máquinas virtuales, que presentan un conjunto de características muy útiles en este escenario:

La idea central de una **máquina virtual** es permitir ejecutar de forma simultánea varios sistemas operativos en un mismo hardware. Utilizar máquinas virtuales permite, por tanto, reducir los costes de mantenimiento de equipos, mejorar el aprovechamiento de los recursos disponibles en la máquina real y reducir el uso de memoria física, entre otras cosas [2].

Además, las máquinas virtuales tienen como ventaja que proveen gran seguridad, puesto que son totalmente independientes del sistema anfitrión. Cuando se están ejecutando varias máquinas virtuales en una misma máquina física, los distintos usuarios no están en riesgo de exhibir o perder información confidencial, gracias a la independencia con el sistema operativo anfitrión antes mencionada.

Por último, una máquina virtual puede ser trasladada a otro equipo diferente de forma sencilla y al abrirla tener exactamente la misma máquina que se estaba ejecutando en el primer servidor. Esto supone una gran ventaja que facilita la gestión en caso de fallos de las máquinas físicas que contienen las máquinas virtuales.

Estos motivos, hacen pensar que la computación en nube mediante la integración de las máquinas virtuales en empresas con un gran número de equipos será cada vez más amplia, sobre todo en el campo de tecnologías de información donde es necesario dar mantenimiento a un gran número de servidores de la forma más eficiente posible [2].

Por otro lado, en todo sistema se debe implantar un módulo de **gestión** que facilite las operaciones de mantenimiento, actualización y monitorización. Cuando se trata de varios elementos interconectados, la gestión se denomina gestión de red, que consiste en la planificación, organización, supervisión y control de elementos de comunicaciones para garantizar un nivel de servicio y de acuerdo a un coste [3].

En la actualidad, se busca que la gestión de red sea lo más autónoma posible, es decir, que los elementos que forman dicha red se autogestionen sin necesidad de intervención de un administrador de red. Obviamente, la Gestión Autónoma supondrá una reducción de los costes de mantenimiento [4]. Las características de un Sistema Autónomo se enumeran a partir de lo que se conoce como Self-CHOP [4] (*Configuration, Healing, Optimization, Protection*): Auto-configuración, auto-cura, auto-optimización, auto-protección.

En este escenario, se ha diseñado un sistema de gestión autónoma de máquinas virtuales que presenta las ventajas de la gestión autónoma y de las máquinas virtuales mejorando la eficiencia con la que se accede y se trabaja en una máquina virtual remota.

1.2. OBJETIVOS

El objetivo fundamental de este Proyecto Final de Carrera ha sido el diseño e implementación de un sistema de gestión autónoma de máquinas virtuales, capaz de migrar de forma autónoma las diferentes máquinas virtuales que se encuentren ejecutándose en los servidores que forman el sistema. Al ser un sistema autónomo cumple las características Self-CHOP:

- Auto-configuración: Las máquinas virtuales se alojan en el servidor más adecuado.
- Auto-cura: Si un servidor supera el umbral de carga permitido intenta migrar máquinas virtuales a otros servidores hasta que desaparece el problema.
- Auto-optimización: Se distribuye la carga entre los servidores migrando las máquinas virtuales de los más a los menos cargados.
- Auto-protección: Los servidores están en estado de alerta permanente vigilando que no se sobrepasa la carga de servidor permitida.

Para conseguir cumplir las características Self-CHOP, el sistema se basa en un ciclo MAPE (Monitorización, Análisis, Planificación y Ejecución) que realiza las acciones a partir de la información que se almacena en una base de conocimiento (Knowledge-Base). El planteamiento de este sistema pretende mejorar la eficiencia con la que se ejecutan las máquinas virtuales en los diferentes servidores de los que se dispone.
En cada servidor se ha implementado un elemento denominado Gestor, encargado de realizar el ciclo MAPE. El Gestor monitoriza la carga y otros parámetros tanto de las máquinas virtuales como del servidor y hace pública la información relevante cuando le es solicitada en difusión por otro servidor. Así el servidor que solicitó información podrá tener en cuenta el estado del resto de servidores del sistema antes de tomar decisiones.
La gestión de las máquinas virtuales se realiza utilizando una biblioteca de gestión de máquinas virtuales, llamada libvirt [5], capaz de actuar como interfaz con diferentes tecnologías de virtualización.

La *Figura 1*, muestra un esquema general del funcionamiento y de la interconexión de los diferentes componentes del sistema.

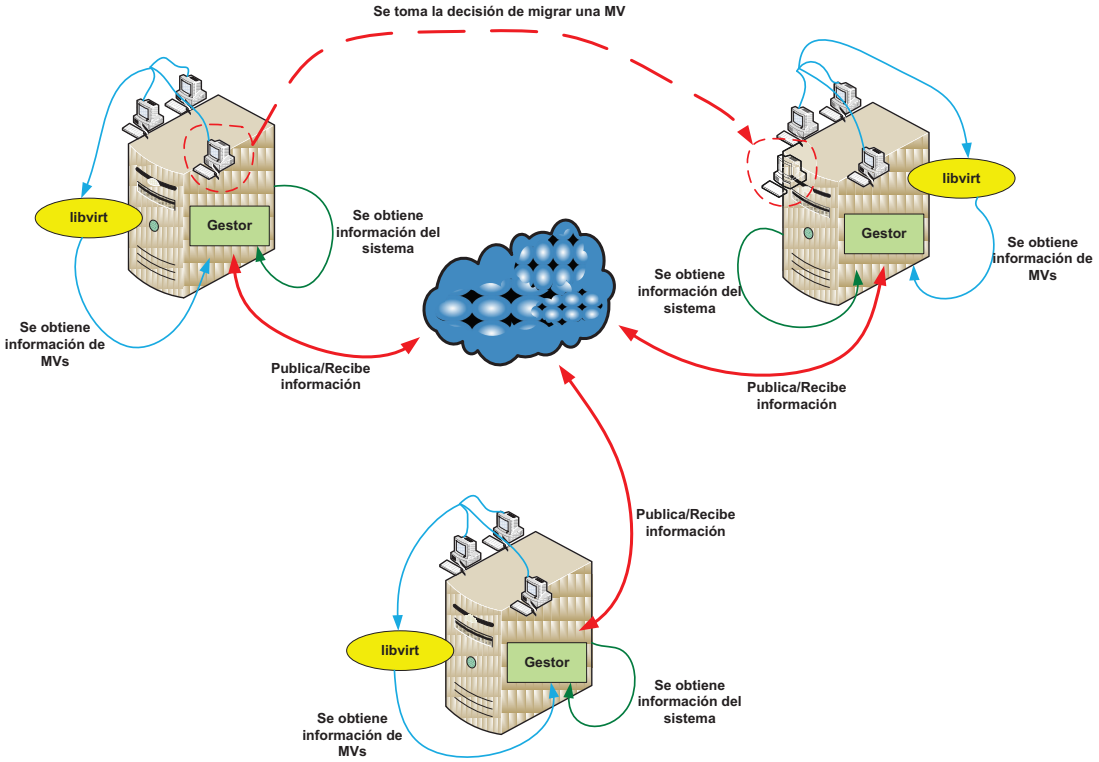


Figura 1: Esquema general del funcionamiento del sistema

1.3. FASES

A continuación se enumeran y detallan las fases que se han seguido durante la elaboración del Proyecto Final de Carrera:

1. Estudiar y analizar el estado del arte de la Gestión Autónoma de Redes. En esta fase se han estudiado las soluciones más comunes a problemas de Gestión Autónoma, analizadas en [4] y se han estudiado las ontologías como un medio para dotar de conocimiento a un sistema autónomo [6].
2. Estudiar y analizar el estado del arte de los sistemas de virtualización [2] empleados comúnmente en entornos de Computación en Nube [1].
3. Análisis de la solución. En esta fase se analiza el uso que se dará a las herramientas utilizadas en la solución.
4. Diseño del sistema autónomo implementado. En esta fase a partir de los conceptos estudiados previamente se han tenido en cuenta diferentes alternativas de funcionamiento.
5. Implementación del sistema autónomo, según las siguientes tareas:
    a. Desarrollar el sistema Gestor. En esta subfase se ha creado la parte "inteligente" del sistema, encargada de decidir qué acción realizar en cada momento.
    b. Desarrollar el sistema de monitorización. En esta subfase se ha creado la parte que monitoriza los parámetros de interés del sistema.
    c. Desarrollar el sistema de comunicación entre servidores. En esta subfase se ha creado un sistema de intercambio de información entre servidores.
6. Pruebas sobre el sistema completo. Las pruebas han consistido en crear de forma artificial situaciones que generen cambios en el sistema para validar el desarrollo realizado. Para la realización de las pruebas ha sido necesario:
    a. Crear y configurar varias máquinas virtuales en dos servidores proporcionados por la EPS. Durante esta fase se han documentado todos los pasos seguidos en la instalación y configuración de las máquinas virtuales.

   b. Generar carga sobre las máquinas virtuales.
   c. Registrar los valores monitorizados para analizarlos posteriormente.
   d. Generar gráficas que posibiliten la visualización gráfica de los resultados obtenidos.
7. Redacción de la memoria del proyecto. En esta fase se han descrito los pasos realizados durante las fases anteriores, adjuntando los resultados de cada una de ellas. Finalmente se han redactado unas conclusiones y posibles alternativas para mejorar el sistema final.

1.4. ESTRUCTURA DE LA MEMORIA

La memoria del Proyecto Final de Carrera está dividida en seis capítulos, tres anexos, pliego de condiciones y presupuesto.

- El Capítulo 1 es la introducción al Proyecto Final de Carrera. Este capítulo describe su motivación y sus objetivos así como las fases en las que se divide el trabajo realizado y la estructura del volumen.

- El Capítulo 2 realiza un análisis del estado actual de la virtualización y la gestión autónoma. Este capítulo realiza una introducción de las herramientas utilizadas en el Proyecto.

- El Capítulo 3 explica cómo se obtienen los datos necesarios para realizar una monitorización de los servidores involucrados en el sistema de gestión autónoma. En este capítulo se detalla qué parámetros son obtenidos con cada herramienta.

- El Capítulo 4 detalla el diseño y la implementación del sistema. En este capítulo se explica la interconexión de todos los elementos utilizados en la monitorización y los algoritmos utilizados en la toma de decisiones. Se proporciona un esquema de interconexión de los módulos implementados

- El Capítulo 5 muestra un estudio sobre el sistema implementado. En este capítulo se detallan las pruebas realizadas sobre la implementación con objeto de evaluar la validez de la solución propuesta.

- El Capítulo 6 presenta las conclusiones obtenidas de los resultados del Capítulo 5 y del proceso de desarrollo. Este capítulo también muestra el trabajo futuro y las posibilidades de mejora del Proyecto Final de Carrera.

- El Anexo A detalla los pasos que hay que realizar en todos los servidores en los que se vaya a ejecuta el sistema de gestión autónoma de máquinas virtuales. Este anexo describe la instalación de software, la creación de usuarios, la configuración de conexiones SSH, etc.

- El Anexo B muestra el procedimiento de ejecución y parada de los servidores y del software desarrollado.

- El Anexo C muestra una pequeña descripción de las clases y métodos implementados en el Proyecto Final de Carrera.

2. ESTADO DEL ARTE

2.1. INTRODUCCIÓN

Este capítulo pretende dar una visión general de la situación actual de la virtualización, la gestión autónoma y el nexo de unión entre ambas, comentando los estudios realizados por otros autores que han tratado de buscar soluciones a problemas similares al tratado en este Proyecto Final de Carrera y así explotar las ventajas de la virtualización vistas en el aparatado anterior.
También se realizará una breve presentación de las herramientas utilizadas en el desarrollo de este proyecto, explicando su funcionalidad y con qué tipo de sistemas se suelen integrar.

2.2. VIRTUALIZACIÓN

La idea central de virtualización consiste en poder ejecutar varios sistemas operativos sobre un mismo servidor que está ejecutando su propio sistema operativo. Para conseguir reutilizar los recursos físicos de un servidor mediante la virtualización, es necesario que exista una capa llamada Hipervisor entre la máquina física y los sistemas operativos virtualizados. Al sistema operativo virtualizado se le denomina Máquina Virtual y posee todas las propiedades de una máquina física, permitiendo realizar las mismas tareas que se podrían realizar en un servidor convencional.
El Hipervisor es el encargado de gestionar los recursos de la máquina física sobre la que se ejecutan las máquinas virtuales y de repartir entre las mismas de forma dinámica dichos recursos.
Como se observa en la *Figura 2* [7], el Hipervisor es una capa que se encuentra justo por encima del Dominio 0 ó sistema operativo anfitrión, permitiendo que las máquinas virtuales tengan completa independencia tanto del Dominio 0 como del resto de las máquinas virtuales.

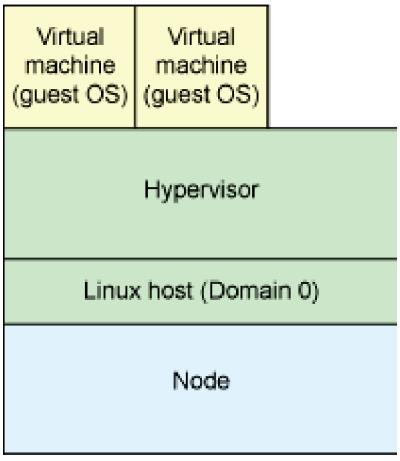


Figura 2: Capas de la virtualización [7]

La principal ventaja de la virtualización se deriva de la oportunidad de reducir los costes asociados al mantenimiento de equipos y al gasto de recursos que estos producen. Además, por regla general un servidor no llega a estar completamente aprovechado y por lo tanto se desperdicia frecuentemente tiempo de procesador que podría ser utilizado en otra tarea. Típicamente las empresas tienden a utilizar un servidor para cada

aplicación y según un estudio de Eaton [8], estas aplicaciones apenas utilizan del 10% al 15% de CPU. En el informe de Eaton, se muestra que con la virtualización de servidores se puede requerir solo un tercio de los equipos que usualmente se utilizan en los centros de datos.
Las predicciones de futuro realizadas por compañías dedicadas a la virtualización como VMware [9], apuntan a que las empresas integrarán cada vez más las tecnologías de virtualización en sus servidores y que aparecerán mejoras notables en la tecnología debido al aumento en su uso. Por otro lado se prevé que el Computación en Nube [1] se vea beneficiada gracias a la virtualización.
También se han realizado estudios sobre el impacto que tiene la virtualización en la red [10], obteniéndose resultados poco favorables, ya que la virtualización causa inestabilidad y retardos anómalos. El estudio sobre la red de Amazon presentado en [10] demuestra que manipular el tiempo de procesador de las máquinas virtuales tiene consecuencias perjudiciales sobre la red. En la modalidad más básica ofrecida por Amazon, las máquinas virtuales solo se ejecutan un 50% del tiempo de procesador, l cual causa retardos para los que protocolos como TCP no están diseñados.

Existen varios hipervisores disponibles sobre los que ejecutar máquinas virtuales. Entre ellos destacan varios por ser los más usados y los que más capacidades ofrecen:

- **VMware** [9] es una empresa dedicada a la virtualización cuyos hipervisores se denominan VMware ESX y VMware ESXi. La principal diferencia de estos hipervisores con el resto es que no se ejecutan sobre un sistema operativo anfitrión, si no que se instalan y ejecutan directamente sobre el hardware del servidor.
- **KVM** [11] (Kernel-based Virtual Machine) es una solución de virtualización completa de código abierto para Linux, a diferencia de VMware, KVM, es un hipervisor que se ejecuta sobre un sistema operativo anfitrión Linux. KVM requiere el uso de QEMU que es un emulador de procesadores.
- **Xen** [12], es un hipervisor alternativo a los anteriores licenciado como software libre que soporta varias arquitecturas de CPU: x86, x86_64, IA64, ARM, etc. El hipervisor Xen, al igual que VMware se ejecuta directamente sobre el hardware del servidor. Un servidor ejecutando Xen contiene tres componentes: el hipervisor Xen, el dominio 0 y las máquinas virtuales. El dominio 0 es lanzado por el hipervisor al iniciar el sistema y puede ejecutar cualquier sistema operativo. Este dominio es el único que tiene privilegios para poder acceder directamente al hardware y es usado para realizar labores de administración.
- **VirtualBox** [13] licenciado en parte bajo GNU General Public License es un hipervisor que se ejecuta sobre un sistema operativo anfitrión al igual que KVM. VirtualBox se puede ejecutar sobre Windows, Linux, Macintosh y OpenSolaris.
- **Hyper-V** [14] es un hipervisor desarrollado por Microsoft, que al igual que VMware y Xen, no necesita un sistema operativo anfitrión (dominio 0). Hyper-V se instala directamente sobre el hardware, por lo tanto reduce la carga sobre el procesador al no requerir un dominio 0.

2.2.1. MIGRACIÓN DE MÁQUINAS VIRTUALES

Todos los hipervisores presentados en el apartado anterior proveen la función de migrar máquinas virtuales entre servidores. En este Proyecto Final de Carrera es necesario que la migración de las máquinas virtuales se realice sin parar la ejecución de la máquina

virtual que se va a migrar, lo que se conoce como *live migration*. Este requisito también lo cumplen todos los hipervisores ya presentados pero solamente se detallará el procedimiento usado por VMware [9] y KVM [11].
VMware ha desarrollado un producto llamado VMotion [15] utilizado para realizar la migración de máquinas virtuales sin interrupción de servicio. VMotion automáticamente optimiza y realoja las máquinas virtuales para una utilización óptima del hardware. El procedimiento que utiliza VMware para realizar la migración en vivo de máquinas virtuales se divide en tres pasos:

1. El estado de la máquina virtual se encapsula es encapsulado por un conjunto de archivos almacenados en un almacenamiento compartido como Fibre Channel, iSCSI o NAS.
2. La memoria activa y el estado que precisa ejecución inmediata de la máquina virtual se transfiere rápidamente sobre una red de alta velocidad de transmisión permitiendo a la máquina virtual que instantáneamente cambie del estado de ejecución en el ESX de origen al estado de ejecución en el ESX de destino. VMotion hace que el tiempo de transmisión sea imperceptible para los usuarios manteniendo una monitorización de las transacciones de memoria en un mapa de bits. Una vez que la memoria entera y el estado de la máquina virtual han sido copiados al ESX de destino, VMotion cambia a suspendido el estado de la máquina virtual en el ESX de origen, copia el mapa de bits al ESX de destino y reanuda la ejecución de la máquina virtual en el ESX de destino. Este proceso tarda menos de dos segundos en una red Gigabit Ethernet.
3. Por último, las redes que están siendo usadas por la máquina virtual también están virtualizadas por el ESX, asegurando que incluso después de la migración, la identidad de la red de la máquina virtual y las conexiones de red serán preservadas. VMotion gestiona la dirección MAC virtual como parte del proceso. Una vez que la máquina en el destino es activada, VMotion realiza un ping al router de la red para asegurar que está al tanto de la nueva localización física de la MAC virtual, lo que permite actualizar las tablas de MACs de los switches.

La *Figura 3* [15], representa de forma esquemática una migración sobre dos ESX utilizando VMotion.

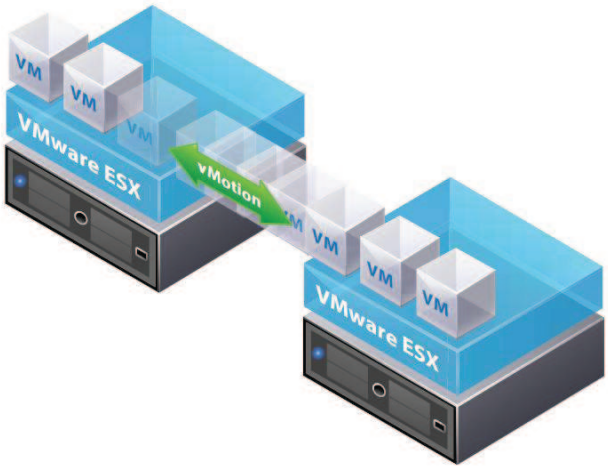


Figura 3: Migración VMware [15]

KVM utiliza un algoritmo bastante similar al de VMware para realizar la migración de máquinas virtuales en vivo.

Primero, se registra la máquina virtual en el destino y se habilita el registro de páginas sucias. Se procede con la transferencia de memoria mientras la máquina virtual continúa ejecutándose en el servidor de origen. Después se transfiere toda la memoria e iterativamente se transfieren todas las páginas sucias (páginas que han sido escritas por la máquina virtual). Se para la máquina virtual en el origen y se sincroniza la imagen de la máquina virtual. Se transfiere el estado tan rápido como sea posible, sin limitación de ancho de banda. Se continúa con el estado de la máquina virtual en el destino y se envía un paquete en difusión para anunciar la nueva localización de la MAC virtual.

2.3. GESTIÓN AUTÓNOMA

Hoy en día, los entornos de computación están adquiriendo un alto nivel de sofisticación que va en aumento [16], envuelven un gran número de soluciones software complejas que cooperan con entornos distribuidos. Este software se desarrolla en plataformas heterogéneas y sus servicios de configuración son generalmente propietarios. Por lo tanto, la gestión de este software (instalación, configuración, modificación, reparación…) es una tarea muy compleja que consume numerosos recursos: humanos y hardware. Una solución a este problema, consiste en implementar la administración como una tarea autónoma.
Además de las ventajas económicas, la gestión autónoma provee a los sistemas que la implementan mayor velocidad en la resolución de problemas, errores o cambios en la configuración que puedan afectar a la integridad del sistema [17], permitiendo que el sistema se pueda recuperar de estos cambios, eligiendo de forma automática el mejor cambio de configuración para asegurar una cierta calidad de servicio.
La gestión autónoma, como norma general, se basa en un simple proceso de ejecución que consiste en [17]: Recopilar, Analizar, Decidir y Actuar, como muestra la *Figura 4* [17].

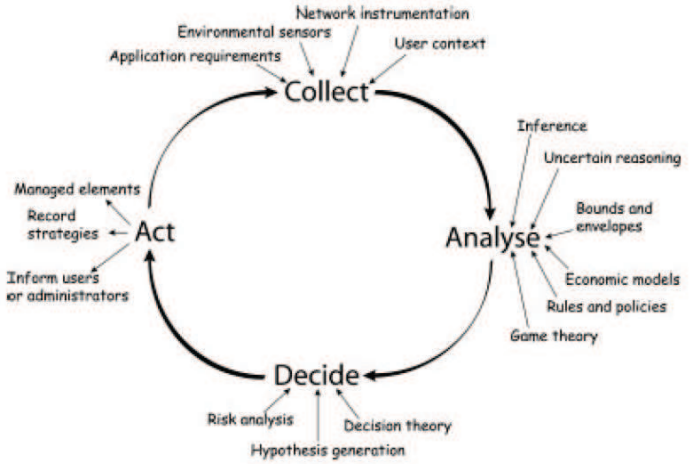


Figura 4: Bucle de control autónomo [17]

Otra de las ventajas de la gestión autónoma consiste en que es posible reconfigurar y adaptar el comportamiento del sistema modificando las reglas aplicadas al sistema autónomo [4]. Además esto es posible hacerlo sin necesidad de parar el sistema que se está gestionando.
Como idea general, se debe tener presente que un sistema autónomo se concibe como aquel que continuamente se monitoriza a sí mismo y ajusta sus operaciones en respuesta a un estímulo interno o externo [4]. Esto es necesario para satisfacer las características Self-CHOP de los sistemas autónomos vistas en la introducción:

- *Self-Configurability* (Auto-Configuración): Consiste en poder detectar cambios en el resto de componentes del sistema o del entorno que tiene influencia en los objetivos de la gestión para así poder lanzar mecanismos de adaptación sin perturbar al resto del sistema.
- *Self-Healing* (Auto-Cura): Consiste en poder restablecer las funcionalidades del sistema si ocurre algún tipo de fallo.
- *Self-Optimization* (Auto-Optimización): Consiste en llevar a cabo tareas de gestión y ejecutar cualquier servicio de la manera más eficiente posible, como por ejemplo proporcionando los recursos disponibles y cambiando los ajustes del entorno de forma que cualquier cambio en la gestión actual no podría resultar en un cambio mejor.
- *Self-Protection* (Auto-Protección): Capacidad de un sistema para tomar las medidas necesarias para proteger sus operaciones de cualquier influencia externa no planeada.

Un sistema autónomo debería ser capaz de tomar decisiones independientes con el fin de modificar sus operaciones para lograr ciertos objetivos.
Como se apuntó en la introducción, todo sistema autónomo tiene un elemento gestor que es el encargado de gestionar uno o más elementos gestionables. El gestor siempre está ocupado con cuatro tareas principales: monitorizar, analizar, planear y ejecutar. Estas tareas se conocen comúnmente como ciclo MAPE [4] como se mencionó en el capítulo de introducción. En entornos dinámicos un sistema autónomo debe mostrar cierto grado de inteligencia a la hora de llevar a cabo las funciones del ciclo MAPE. En este contexto la inteligencia se refiere a la capacidad de aprender, en el sentido de mejorar las operaciones. Una solución es limitar la inteligencia a un gestor autónomo centralizado que controle el resto de componentes del sistema. El otro extremo de la inteligencia distribuida en el sistema autónomo sería usar un sistema capaz de aprender del comportamiento colectivo de pequeños núcleos inteligentes.

2.4. VIRTUALIZACIÓN Y GESTIÓN AUTÓNOMA

Anteriormente se ha comentado que el principal problema en los centros de proceso de datos es que el promedio de utilización de los servidores es muy bajo en comparación con su capacidad de procesamiento. Esto hace que no se aprovechen los recursos de los servidores de forma óptima, lo que implica un aumento de los costes tanto de operación como de mantenimiento. Las principales causas de la baja utilización se deben principalmente a: la necesidad de garantizar un buen funcionamiento en los momentos en los que exista un pico de demanda, al sobre aprovisionamiento de la capacidad de proceso y a utilizar un solo Sistema Operativo por máquina.
Para solucionar este problema se hace patente el uso de las máquinas virtuales junto con el uso de la gestión autónoma para diseñar sistemas capaces de ubicar las máquinas virtuales en los servidores que más convenga según el momento.
En este escenario, resuelto en este Proyecto Final de Carrera, existen otros sistemas que utilizan diferentes algoritmos para resolverlo como MFR (Measure-Forecast-Remap) [18].
MFR consiste en realizar una medida del histórico de datos, pronosticar una demanda futura y recolocar las máquinas virtuales en las máquinas físicas, de forma que la migración de las máquinas virtuales ocurra sin ningún tipo de interrupción de servicio. Este algoritmo se basa en la predicción y por lo tanto requiere un método muy exacto para estimar la demanda futura basándose en la observación de los acontecimientos

pasados. Normalmente en este algoritmo, la calidad de la predicción suele disminuir cuanto mayor sea el tiempo para el cual se quiere realizar dicha predicción.
Por otro lado, este tipo de soluciones presentan como inconveniente que generan una sobrecarga en el sistema cuando se necesita realizar un realojo de máquinas virtuales [19], produciéndose un impacto, en el sistema y en las aplicaciones, relacionado con una sobrecarga en la capacidad.

VMware también ofrece un producto que proporciona gestión autónoma sobre las máquinas virtuales. Este producto se denomina Distributed Resource Scheduler (DRS) y su función consiste en supervisar continuamente la utilización de los recursos y asignarlos de forma inteligente entre las máquinas virtuales, basándose en reglas definidas previamente. Si una máquina virtual experimenta un aumento en la carga, el DRS asigna automáticamente recursos adicionales mediante la redistribución de las máquinas virtuales entre los servidores físicos del grupo de recursos. Los recursos se asignan a la máquina virtual mediante su migración a otro servidor con más recursos disponibles o mediante la creación de más espacio para ella en el mismo servidor migrando otras máquinas virtuales a distintos servidores. La migración en caliente de máquinas virtuales a distintos servidores físicos se realiza de forma completamente transparente para los usuarios finales mediante VMotion. DRS es un software propietario, válido solamente para la gestión de máquinas virtuales que se ejecutan sobre un hipervisor de VMware.

2.4.1. ALGORITMOS DE PLANIFICACIÓN

Recolocar las máquinas virtuales en las máquinas físicas requiere cierta componente de planificación, similar a la utilizada por los procesadores en la planificación de procesos o por los sistemas de memoria que no utilizan paginación. La similitud viene dada desde el momento en que asignar máquinas virtuales a servidores es equivalente a asignar procesos a procesadores. En ambos casos se intentará buscar mejorar el tiempo de respuesta, aumentar la productividad y aumentar la eficiencia del servidor.
El algoritmo MFR comentado anteriormente utiliza una versión del algoritmo de planificación Primero el Proceso más Corto, que se basa en predecir la demanda que tendrá un determinado proceso en función de la demanda que ha tenido el proceso en el pasado. En este Proyecto Final de Carrera también se utilizará este algoritmo para predecir la carga que generará una máquina virtual basándose en la carga que está generando en el momento de la predicción.
La reubicación de las máquinas virtuales puede ser similar a la asignación de procesos en las particiones de memoria, donde cabe destacar los métodos de Best-fit, First-fit y Next-fit.

- Best-fit elegirá siempre el bloque de memoria de tamaño más próximo al solicitado. Este algoritmo en general proporciona los peores resultados puesto que al buscar el hueco más pequeño para el proceso garantiza que el fragmento que deja es lo más pequeño posible y por lo tanto habrá que compactar más frecuentemente.
- First-fit es el más rápido, consiste en asignar el proceso al primer bloque de memoria en el que quepa. Tiene como inconveniente que el extremo inicial de la memoria tiende a estar completo y por lo tanto obliga a recorrer todos estos bloques para encontrar uno libre.
- Next-fit busca el siguiente bloque de memoria en el que quepa el proceso, buscando desde la última asignación que se hizo. Este algoritmo lleva a asignar

los procesos al bloque de memoria más grande, el cual con el tiempo se dividirá en fragmentos pequeños.

Este Proyecto Final de Carreara utilizará una combinación de los algoritmos First-fit y Next-fit para la recolocación de las máquinas virtuales en los servidores. La diferencia principal será que en vez de espacio en memoria, se tendrá en cuenta la carga que genera la máquina virtual sobre el servidor y la carga de los servidores. Así, se deberá comprobar si la carga del servidor cuando se le asigne una nueva máquina virtual permanecerá (predicción) dentro de unos límites, es decir, si "cabe" en el servidor (First-fit y Next-fit).

2.5. HERRAMIENTAS UTILIZADAS

Para la implementación del sistema introducido en los apartados anteriores, se han escogido una serie de herramientas que facilitan la obtención de los parámetros necesarios para conseguir que el sistema funcione de forma adecuada, así como la implantación de la base de la arquitectura del sistema como por ejemplo el hipervisor, el sistema de intercambio de mensajes, la monitorización del servidor y la monitorización y la gestión de las máquinas virtuales.

La justificación de la elección de estas herramientas radica en que todas ellas están licenciadas bajo GPL y proporcionan APIs para Java, lo que facilita la integración de cada herramienta en el sistema general implementado.

A continuación se presentan las herramientas utilizadas en el desarrollo del Proyecto Final de Carrera. El capítulo siguiente muestra como se utilizarán en el desarrollo.

2.5.1. KVM

Como se ha explicado en las secciones anteriores, KVM [11] es un hipervisor que se ejecuta sobre la mayoría de los sistemas operativos Linux. Esto facilita el uso de servidores con un sistema Linux ya instalado. De esta manera, no se requiere la instalación de un hipervisor como Xen o ESX, permitiendo utilizar el servidor para tareas diferentes a la virtualización.

2.5.2. LIBVIRT

libvirt [5], es una API de código abierto, que se utiliza como herramienta para gestionar plataformas de virtualización. Esta API soporta los hipervisores: Xen [12], KVM [11], VirtualBox [13], VMware (ESX, GSX) [9] y VMware Workstation and Player [9].

En este proyecto se ha utilizado libvirt como interfaz de gestión del hipervisor KVM, haciendo uso de la API de Java. En este proyecto se obtiene a través de libvirt información relativa al número de máquinas virtuales ejecutándose o no ejecutándose e información de los recursos que estas máquinas virtuales están consumiendo.

La ventaja de la utilización de libvirt es que la implementación de este Proyecto Final de Carrera es aplicable al resto de hipervisores soportados por libvirt. Solo sería necesario realizar unos pequeños cambios en el código.

2.5.3. SIGAR

Sigar [20] es una API Java que proporciona información del sistema independientemente de la plataforma. En este proyecto se ha utilizado para obtener

información sobre el estado del servidor y sus características como memoria, carga, uso de memoria, número de CPUs, etc.

2.5.4. XMLBLASTER

xmlBlaster [21] es un MOM (Message Oriented Middleware) que se utiliza para el intercambio de mensajes, utilizando el sistema de publicador/subscriptor. Los mensajes pueden contener todo tipo de información como imágenes, objetos Java, texto plano, etc.
En este proyecto se ha utilizado xmlBlaster para el intercambio de información entre los distintos servidores del sistema, como por ejemplo, información de migración de una máquina virtual, información de la entrada o salida de un servidor en el sistema, envío del estado del servidor en un determinado momento, etc.

2.6. CONCLUSIONES

El uso de máquinas virtuales ayuda a mejorar la utilización de los servidores sin que la calidad del servicio disminuya, esto se debe a que típicamente los servidores se encuentran desaprovechados.
La gestión autónoma reduce los costes de operación y mantenimiento de los centros de proceso de datos. Además la resolución de problemas es inmediata y proporciona la posibilidad de adaptar el comportamiento del sistema según los requerimientos del momento.
Tener la habilidad de realojar máquinas virtuales en servidores de forma autónoma y según las necesidades del momento proporciona la capacidad de mejorar la utilización del sistema sin empeorar el rendimiento y sin asignar los recursos de forma estática, de forma que en los picos de demanda siempre se podrán cubrir las necesidades con un simple realojo de máquinas virtuales.
En los siguientes capítulos, ya a partir de la base presentada en éste, se describirá el diseño de este proyecto, explicando el funcionamiento general y la aplicación práctica, para posteriormente obtener unas conclusiones en función de resultados obtenidos a partir de simulaciones.

3. ANÁLISIS DEL SISTEMA DE GESTIÓN AUTÓNOMA

3.1. INTRODUCCIÓN

Para aprovechar las ventajas expuestas en los capítulos anteriores es necesario llevar a cabo un análisis de la solución. El sistema permite reducir el número de servidores necesarios para ofrecer varios servicios, haciendo uso de la virtualización y la migración de las máquinas virtuales de servidor en servidor de forma autónoma. Además, esto es posible hacerlo sin que disminuya la calidad de los servicios ofrecidos.
Este capítulo explica cómo se recopilan los datos necesarios para hacer esto posible haciendo uso de las herramientas introducidas en el apartado anterior. También se detallarán los requisitos necesarios que deben de cumplir los servidores del sistema para un correcto funcionamiento del mismo.

3.2. FUNCIÓN DEL SISTEMA

Si se tiene en cuenta un ejemplo típico de arquitectura en una empresa, en la que son necesarios tres servicios como un servidor web, una base de datos y un servidor de correo electrónico, lo más conservador es asignar un servidor físico a cada servicio, ya que no es posible prever con exactitud la demanda que tendrá cada uno. Además, la demanda es variable en función de las horas del día, siendo muy alta para determinados momentos y muy baja o casi nula para otros. Estos factores hacen que tener tres servidores no sea eficiente ya que lo normal es que estos estén desaprovechados la mayor parte del tiempo.
Una posible solución es tener todos los servicios en un único servidor, pero existen picos de demanda en los que los servicios necesitan hacer uso de todos los recursos disponibles del sistema y aunque estos picos suelen ser de una duración corta, si no es posible satisfacer la demanda del pico, el sistema se ralentiza considerablemente.
Otra solución podría consistir en que cada servicio se ejecute sobre una máquina virtual diferente en el mismo servidor y que cuando exista el pico de demanda mover manualmente alguna de las máquinas virtuales a otro servidor. Este sistema tiene como inconveniente la necesidad de que un administrador esté permanentemente vigilando el sistema, ya que no es posible hacer una predicción exacta de cuándo tendrá lugar el pico.
Una tercera opción consiste en que utilizando dos servidores y máquinas virtuales, se monitorice de forma autónoma la demanda que tiene cada servidor y si es necesario el sistema autónomo movería las máquinas virtuales en función de sus necesidades. Este razonamiento se basa en la baja probabilidad de que los tres servicios experimenten un pico de demanda al mismo tiempo, proporcionando así la posibilidad de que si uno de los tres servicios está experimentando una demanda muy alta, es posible que un servidor ejecute solamente esa máquina virtual, mientras que el otro está ejecutando las otras dos.

La idea de este Proyecto Final de Carrera está basada en la tercera opción. Esto, además de utilizar menos recursos que en la primera opción, permite que el tiempo de reacción sea mucho menor que en la segunda, ya que la tarea de migrar las máquinas virtuales se ha automatizado, junto con la decisión de qué máquinas virtuales migrar. Esta decisión

es óptima, es decir, moverá las máquinas virtuales necesarias para satisfacer un cierto nivel de servicio en todos los servidores. Además, al ser una decisión automatizada solamente tarda unos milisegundos en tomarse, por lo que la calidad del servicio no disminuye y todo sucede de forma transparente a los usuarios.
El ejemplo anterior es una simplificación del fin último de este Proyecto Final de Carrera. El sistema de gestión autónoma de máquinas virtuales deberá gestionar un pool de servidores en el que solamente estén activos los necesarios para ejecutar las máquinas virtuales según el momento y el resto deberán estar en stand-by. Así, cuando se detecten momentos de carga, se activan los que estaban suspendidos, y se moverán las máquinas virtuales necesarias a los servidores activados. De esta forma, en un pico se podrían tener tantos servidores encendidos como fuera necesario, y en el resto de casos, todos durmiendo.

3.3. APLICACIÓN DE LAS HERRAMIENTAS UTILIZADAS

En este apartado se pretende proporcionar una visión de cómo se han integrado en el sistema las herramientas introducidas en el capítulo anterior. Además se explicará que tipo de información se ha obtenido de cada una y como se ha usado.

3.3.1. LIBVIRT Y KVM

Como se explicó en el apartado anterior, KVM es un hipervisor integrable con la mayoría de sistemas Linux. En este proyecto se ha instalado KVM junto con libvirt sobre Ubuntu Server 10.04. Todas las máquinas virtuales empleadas para las pruebas de este proyecto han sido ejecutadas sobre KVM.
Sobre este hipervisor se han creado varias máquinas virtuales de prueba en las diferentes máquinas físicas de las que se ha dispuesto.
Para crear las máquinas virtuales se ha utilizado la herramienta de gestión virt-manager [22]. Virt-manager es una interfaz gráfica que permite crear, modificar y monitorizar las máquinas virtuales sobre KVM. Además esta herramienta posibilita asignar a cada máquina virtual unos recursos diferentes según los requerimientos, como son la memoria, el disco duro y las CPUs.

En el apartado anterior se ha señalado que libvirt es una API que se utiliza para obtener información sobre el hipervisor, en este caso KVM. En el caso de este proyecto, se ha utilizado la API de Java de libvirt para realizar llamadas sobre el hipervisor y así obtener información relevante que posteriormente se utiliza para tomar decisiones o simplemente para compartir con otros servidores.

La información obtenida para el funcionamiento de este proyecto haciendo uso de libvirt es la siguiente:

- Hostname
- Máquinas virtuales activas e inactivas
- Bytes transmitidos por máquina virtual
- Bytes recibidos por máquina virtual
- Uso de procesador de cada máquina virtual

Además de la información anterior, también se usa libvirt para realizar las **migraciones de máquinas virtuales** a diferentes servidores.

3.3.1.1. Otra información necesaria

Además de la información obtenida con libvirt es necesaria otra información para una correcta caracterización del servidor, así como una serie de reglas que se deben seguir a la hora de crear máquinas virtuales para una correcta migración de las mismas entre los servidores que componen el sistema.

Se ha obtenido el uso que se está haciendo de la interfaz de red del servidor para considerarlo un parámetro más de decisión cuando hay que migrar máquinas virtuales. Para ello es necesario que las máquinas virtuales no tengan sus interfaces de red configuradas en modo bridge, pues sería necesario que el servidor de destino tuviera definido el mismo bridge. En caso contrario la migración fallaría.

Como los nombres de las interfaces podrían ser diferentes en cada servidor, se ha simplificado la tarea a obtener el uso que está haciendo la máquina virtual de la interfaz asociada a la ruta por defecto. Para ello es necesario comprobar en el archivo XML de definición de la máquina virtual que se está utilizando la interfaz por defecto y no otra. Este archivo XML se obtiene con libvirt.

Una vez que se ha obtenido esta información es necesario averiguar el nombre real de la interfaz para posteriormente obtener su velocidad. Con esta información junto con la obtenida con Sigar referente a los bytes trasmitidos y recibidos, es posible calcular el uso que se está haciendo de la interfaz por defecto del servidor.

Adicionalmente será necesario que todos los servidores implicados en el sistema estén dados de alta en un DNS, no siendo necesario para las máquinas virtuales. Esto es así porque se ha decidido que todo el intercambio de mensajes se realice utilizando el FQDN (Fully Qualified Domain Name) del servidor y no la IP.

3.3.2. SIGAR

Como se ha mencionado, Sigar es una API que proporciona información relativa al sistema. En este proyecto, al igual que con libvirt, se ha utilizado la API de Java para obtener dicha información. Para utilizar esta API es necesario disponer de la librería correspondiente al sistema operativo y al procesador del servidor del que se quiere obtener información.

La información obtenida con Sigar en este proyecto complementa a la obtenida con libvirt para obtener en todo momento un informe detallado del servidor que se está monitorizando, si bien los datos obtenidos con libvirt se centran más en las máquinas virtuales del servidor, con Sigar se obtienen datos del rendimiento del propio servidor.

La información obtenida para el funcionamiento de este proyecto haciendo uso de Sigar es la siguiente:

- Carga del servidor
- Memoria total
- Memoria utilizada
- Número de CPUs

- Bytes recibidos y transmitidos por la interfaz de red

Estos datos proporcionan la información más importante y necesaria para poder caracterizar a un servidor en un determinado momento, de forma que este pueda ser comparado con otros y así predecir en cuál de ellos los procesos se ejecutarán más rápido. Además esto permite asignar un peso a cada característica según se quiera dar más importancia a unas o a otras para realizar dicha predicción.

3.3.3. XMLBLASTER

Como se comentó en el capítulo anterior, xmlBlaster es un Message Oriented Middleware. Este software se ejecuta en un servidor al cual se conectan los clientes para leer o publicar mensajes. Al igual que libvirt y Sigar, xmlBlaster tiene una API con Java que permite publicar y leer mensajes mediante código Java.

En este proyecto se ha utilizado xmlBlaster para publicar y leer mensajes, todos ellos enviados en difusión. La naturaleza de los mensajes utilizados es de dos tipos: mensajes que contienen una cadena de caracteres y mensajes que contienen un objeto Java. Dependiendo de la información que se necesite intercambiar se envía un tipo de mensaje u otro.

La información intercambiada permite a cada servidor conocer el estado del resto de servidores y compartir su propio estado con el resto. Aunque los mensajes son enviados en difusión no todos los servidores tienen por qué recibirlos, ya que podría ser que no estén subscritos a todos los mensajes en todo momento.

Además del estado de cada servidor, el intercambio de mensajes también sirve para que todos los servidores sepan que otros servidores están en el sistema y así poder intercambiarse mensajes con ellos.

Un vez que toda esta información es compartida, los servidores pueden evaluar a cuál migrar máquinas virtuales en función de los parámetros intercambiados y comportarse de formas diferentes en función de si hay más servidores en el sistema o solo hay uno.

A modo de resumen la información intercambiada con mensajes mediante xmlBlaster es:

- Nombre del servidor
- Asentimiento de recepción de nombre de servidor enviando el propio hostname
- Petición de información de estado al resto de servidores
- Envío de la información solicitada
- Mensaje de notificación de que un servidor abandona el sistema
- Mensaje de información indicando que se ha terminado de migrar una máquina virtual
- Mensaje de asentimiento a la notificación de fin de la migración de la máquina virtual.

3.4. CONCLUSIONES

Este Proyecto Final de Carrera intenta proporcionar una solución a la ejecución de procesos utilizando el menor número de recursos posible, con una la calidad de servicio equivalente a la que habría con más recursos, gracias a la inclusión de márgenes de seguridad que habilitan más recursos físicos en caso de ser necesarios.
Con la información obtenida por medio de las herramientas libvirt y Sigar es posible caracterizar un servidor y a sus máquinas virtuales, haciendo posible la comparación entre servidores. Esta comparación se realiza enviando dicha información con xmlBlaster en forma de mensajes que contienen cadenas de caracteres u objetos Java.
En los siguientes capítulos se detallarán los algoritmos utilizados para realizar el intercambio de mensajes, la decisión de las máquinas virtuales que hay que migrar, la auto-monitorización y el funcionamiento general del sistema. Después se expondrán las clases Java que se han desarrollado y su relación entre ellas.

4. DISEÑO E IMPLEMENTACIÓN

4.1. INTRODUCCIÓN

Una vez presentado el problema al que se pretende dar solución y las diferentes tecnologías existentes, tanto referentes a la virtualización como las usadas en este proyecto para obtener información referente a los servidores, en este capítulo se va a detallar como se han interconectado todos los elementos anteriormente descritos.
Se explicará cómo se auto-monitoriza un servidor, cómo sabe qué servidores están actualmente en el sistema, qué máquinas virtuales son originariamente suyas, cómo decide que tiene un problema, el procedimiento para solucionar dicho problema: solicitando información a los demás servidores y decidiendo qué máquinas virtuales migrar a qué servidores. Por último, se explicarán las políticas a emplear en caso de que el servidor no pueda resolver el problema y se mostrará un esquema de interconexión de los módulos implementados en código Java.

4.2. DECISIONES DE DISEÑO

Como se verá en el capítulo 5, en el que se mostrarán las pruebas realizadas sobre el diseño, el parámetro fundamental de los servidores y que por lo tanto es el que se pretende minimizar es la **carga del servidor**, es decir, el tiempo de uso de CPU. Por ello, todos los algoritmos implementados en este proyecto están orientados a tal fin, procurando que la carga de todos los servidores que forman parte del sistema se **mantenga por debajo de un umbral** fijado. Para lograrlo, se migrarán máquinas virtuales desde los servidores que sobrepasen dicho umbral hacia otros servidores que formen parte del sistema, tratando de mantener en el servidor de destino la carga por debajo del umbral estipulado.
Como se dijo en los capítulos anteriores la decisión de qué máquinas mover a qué servidores no solo depende de la carga de los servidores, sino que se tienen en cuenta otros parámetros. Estos parámetros, junto con la carga del servidor, tendrán unos pesos de decisión, los cuales son libremente asignables por el administrador del sistema, pero que para un correcto funcionamiento es recomendable que sean iguales en todos los servidores que componen el sistema.
Por último, los umbrales que intervienen en las decisiones del sistema también son libremente asignables por el administrador y aunque es este caso no es estrictamente necesario que sean iguales en todos los servidores, sí es aconsejable.

Para que el sistema diseñado funcione y el algoritmo de comparación entre servidores sea fiable, existe la restricción de que **todos los servidores involucrados en el sistema deben ser idénticos**, es decir, deben tener el mismo número de procesadores, procesadores iguales y misma cantidad de memoria. Esto debe ser así debido a la imposibilidad en caso contrario de predecir el estado en el que quedará el servidor de destino después de migrar una máquina virtual. Eliminar esta limitación quedará como trabajo futuro, cambiando el método de predicción de carga a partir de benchmarks.

Adicionalmente a esta restricción es requisito indispensable que **los nombres de todos los servidores involucrados en el sistema se encuentren registrados en el DNS**. Esto es así debido a que los servidores almacenan el FQDN del resto de servidores y este es usado para el intercambio de mensajes.

Los parámetros del servidor que se han decidido monitorizar están en la *Tabla 1*, que muestra el nombre del parámetro, cada cuanto tiempo se realiza una medida del mismo y una breve explicación de cómo ha sido calculado.

PARÁMETROS DE MONITORIZACIÓN DE LOS SERVIDORES FÍSICOS		
Parámetro	**Intervalo de medida**	**Resumen de cálculo**
Máquinas virtuales activas	10 segundos	Parámetro obtenido directamente con libvirt
Máquinas virtuales inactivas	10 segundos	Parámetro obtenido haciendo uso de libvirt
Carga del Servidor	10 segundos	Parámetro obtenido directamente con Sigar
Número de CPUs	10 segundos	Parámetro obtenido directamente con Sigar
Memoria usada	10 segundos	Parámetro obtenido directamente con Sigar
Memoria total	10 segundos	Parámetro obtenido directamente con Sigar
Porcentaje de uso de memoria	10 segundos	Parámetro obtenido directamente con Sigar
Uso de interfaz de red	60 segundos	Se hace distinción entre el uso del Downlink y del Uplink. Se obtienen los bytes transmitidos y recibidos con Sigar. Se obtiene la velocidad de la interfaz de red realizando una consulta al Sistema Operativo. Mediante un cálculo se obtiene la utilización de la interfaz

Tabla 1: Parámetros de monitorización de los servidores físicos

El intervalo de medida de uso de la interfaz de red es 60 segundos en vez de 10 para evitar que aparezca demasiado ruido, ya que es un parámetro que previsiblemente variará muy rápido.

Al igual que con los servidores que hospedan las máquinas virtuales, se ha decidido monitorizar una serie de parámetros referentes a las propias máquinas virtuales. Estos parámetros, *Tabla 2*, influyen en la decisión de que máquinas virtuales migrar a que servidores.

PARÁMETROS DE MONITORIZACIÓN DE LAS MÁQUINAS VIRTUALES		
Parámetro	**Intervalo de medida**	**Resumen de cálculo**
Carga que genera la Máquina Virtual	10 segundos	Parámetro obtenido utilizando libvirt y Sigar, junto con cálculos adicionales que se detallarán en los siguientes apartados.
Uso de interfaz de red	60 segundos	Se calcula el uso que tiene la interfaz de red del servidor a la que está conectada la Máquina Virtual. Se hace distinción entre el uso del Downlink y del Uplink. Se obtienen los bytes transmitidos y recibidos con Sigar. Se obtiene la velocidad de la interfaz de red realizando una consulta al Sistema Operativo. Mediante un cálculo se obtiene la utilización de la interfaz

Tabla 2: Parámetros de monitorización de las máquinas virtuales

Por último, se han definido los umbrales que caracterizan el comportamiento del sistema en función de su valor. La *Tabla 3* muestra los nombres y la descripción de dichos umbrales.

UMBRALES DEL SISTEMA	
Nombre	**Descripción**
umbralCarga	Define el umbral de carga a partir del cual se considera que el servidor tiene una carga demasiado alta. Este umbral puede tomar valores en coma flotante de 0 a 1.
umbralAlarmaCarga	Define el número medidas de carga consecutivas que deben superar umbralCarga para que el servidor considere que debe migrar máquinas virtuales.
umbralPenalización	Si un servidor no logra migrar todas las máquinas virtuales para disminuir su carga por debajo de umbralCarga, entonces no podrá volver a intentar migrar máquinas hasta que se realicen el número de medidas de carga indicadas por umbralPenalización. Pero si en una de esas medidas la carga obtenida está por debajo de umbralCarga, entonces no es necesario cumplir el número de medidas fijado por umbralPenalizacion.

Tabla 3: Umbrales del sistema

4.3. VISIÓN GENERAL

En primer lugar se va a pretender mostrar una fotografía de todo el sistema completo similar a la *Figura 1*.

Según se ha introducido en los capítulos previos, cada servidor ejecutará un elemento gestor encargado de:

- monitorizar el estado del servidor
- pedir información al resto de servidores
- analizar la información recibida
- decidir la distribución de las máquinas virtuales para disminuir la carga sin aumentarla demasiado en los servidores de destino

- decidir la penalización que se debe imponer a sí mismo en caso de que no logre una adecuada distribución de las máquinas virtuales
- saber en todo momento que servidores forman parte del sistema
- notificar al resto de servidores que se ha terminado de migrar una máquina virtual

Esta información será obtenida haciendo uso de libvirt y Sigar.
Los gestores de cada servidor estarán en constante comunicación, esto lo harán por medio de xmlBlaster. Para ello los servidores estarán conectados al servidor xmlBlaster a través del cual intercambiarán los mensajes.

La *Figura 5* muestra de una forma simplificada como están los elementos del sistema conectados entre sí.

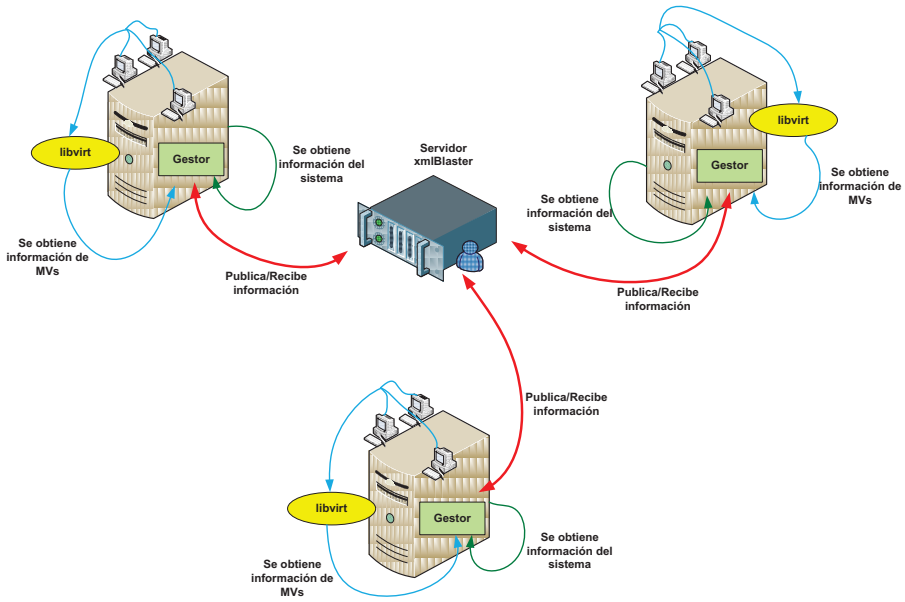


Figura 5: Interconexión de los elementos del sistema

4.4. FUNCIONAMIENTO GENERAL

A continuación se va a describir de forma detallada cómo se han obtenido cada uno de los elementos introducidos en la *Tabla 1: Parámetros de monitorización de los servidores físicos* y *Tabla 2: Parámetros de monitorización de las máquinas virtuales*. Además, se describirá el algoritmo diseñado para migrar las máquinas virtuales cuando se detecta que un servidor sobrepasa el umbral de carga permitido.

4.4.1. CONOCIMIENTO DE LOS SERVIDORES QUE FORMAN PARTE DEL SISTEMA

Lo primero que se debe tener en cuenta es que un servidor solo podrá migrar/recibir máquinas virtuales a/de otros servidores que estén ejecutando el sistema de gestión autónoma de máquinas virtuales. Al conjunto de servidores que estén ejecutando dicho sistema de gestión autónoma se le llamará **grupo de servidores** de ahora en adelante.

En este apartado se explicará cómo un servidor que está ejecutando el sistema de gestión autónoma de máquinas virtuales tiene conocimiento respecto a qué servidores puede migrar máquinas virtuales, es decir cuáles forman parte del grupo de servidores.
El registro que hay que llevar de los servidores es algo dinámico, ya que estos pueden ejecutar o parar el sistema de monitorización en cualquier momento y por lo tanto entrar o salir del grupo de servidores.

La información que registrará cada servidor para elaborar el grupo de servidores serán los FQDN de cada servidor que añadirá o eliminará según se amplíe o se reduzca el grupo de servidores. Los FQDN de cada servidor se enviarán como una cadena de caracteres utilizando xmlBlaster.

Para que cada servidor pueda llevar a cabo el registro de servidores que forman parte del grupo de servidores es necesario diferenciar dos casos: cuando un servidor entra en el grupo de servidores y cuando un servidor sale del grupo de servidores.

4.4.1.1. Entrada de un servidor en el grupo de servidores

Cuando en un servidor se inicia el sistema de gestión autónoma de máquinas virtuales, este debe entrar a formar parte del grupo de servidores que estén conectados al mismo servidor xmlBlaster. Para entrar a formar parte del grupo, la primera acción que realiza el servidor es enviar un mensaje en difusión cuyo contenido es una cadena de caracteres con su FQDN. Este mensaje se recibe por todos los servidores que forman parte del grupo de servidores en ese momento, que contestarán enviando un mensaje similar en difusión con su propio hostname. Estos mensajes de respuesta serán recibidos por todo el grupo de servidores, pero solo almacenará los hostnames recibidos el servidor que acaba de iniciar el sistema de gestión autónoma de máquinas virtuales, ya que los demás comprobarán si tienen el hostname recibido en su registro de hostnames antes de almacenarlo y si ya lo tienen simplemente ignoran el mensaje.

Un caso particular es el primer servidor que entra a formar parte del grupo de servidores. En este caso cuando este servidor envía su propio hostname, no recibe ninguna respuesta dado que no hay más servidores en el grupo.

A continuación se explica un ejemplo de entrada de servidores en el grupo de servidores. En concreto, este ejemplo muestra tres servidores que se van incorporando al grupo de servidores de forma secuencial. Para explicarlo, se han creado 3 figuras: *Figura 6*, *Figura 7* y *Figura 8*, en las que es posible observar el intercambio de mensajes que tiene lugar con la incorporación de cada servidor.

La primera figura muestra el inicio del sistema de gestión autónoma de máquinas virtuales. El servidor en el que se ha iniciado el sistema envía un mensaje con su propio hostname pero como no hay más servidores en el grupo ningún servidor responde con su propio hostname a dicho mensaje. Cuando un servidor entra en el sistema y publica su hostname, éste está disponible para ser leído por otros servidores durante un segundo. Pasado este tiempo el servidor que lo publicó lo borra para que lo servidores nuevos que entren en el sistema después de él, no lo puedan leer.

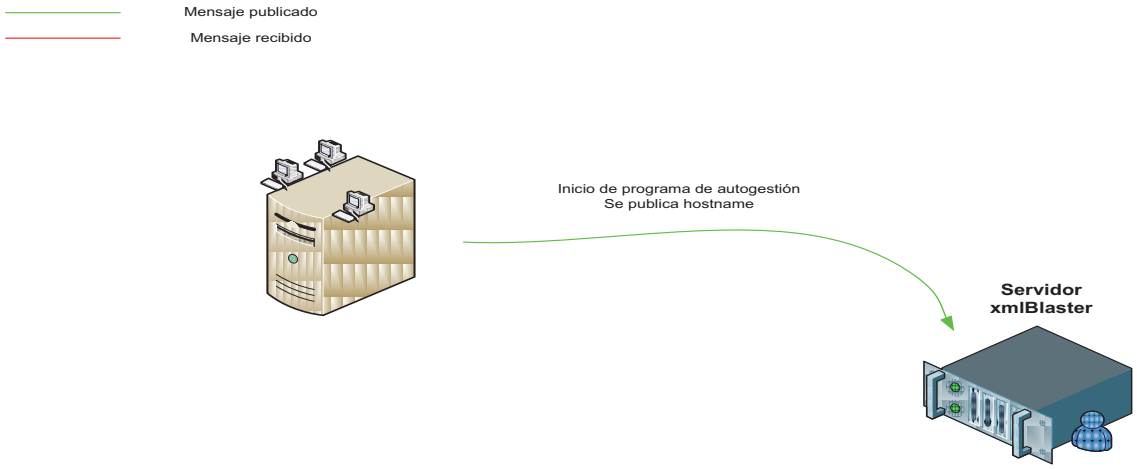


Figura 6: Servidor entrando en grupo de servidores vacío

En la *Figura 7* un nuevo servidor inicia el sistema de gestión autónoma de máquinas virtuales. Este servidor envía un mensaje con su hostname, que se recibe en el servidor que ya formaba parte del grupo de servidores. Al recibir el mensaje el servidor antiguo almacena el hostname que acaba de recibir y responde enviando su propio hostname, que se recibe en el servidor nuevo y lo almacena.

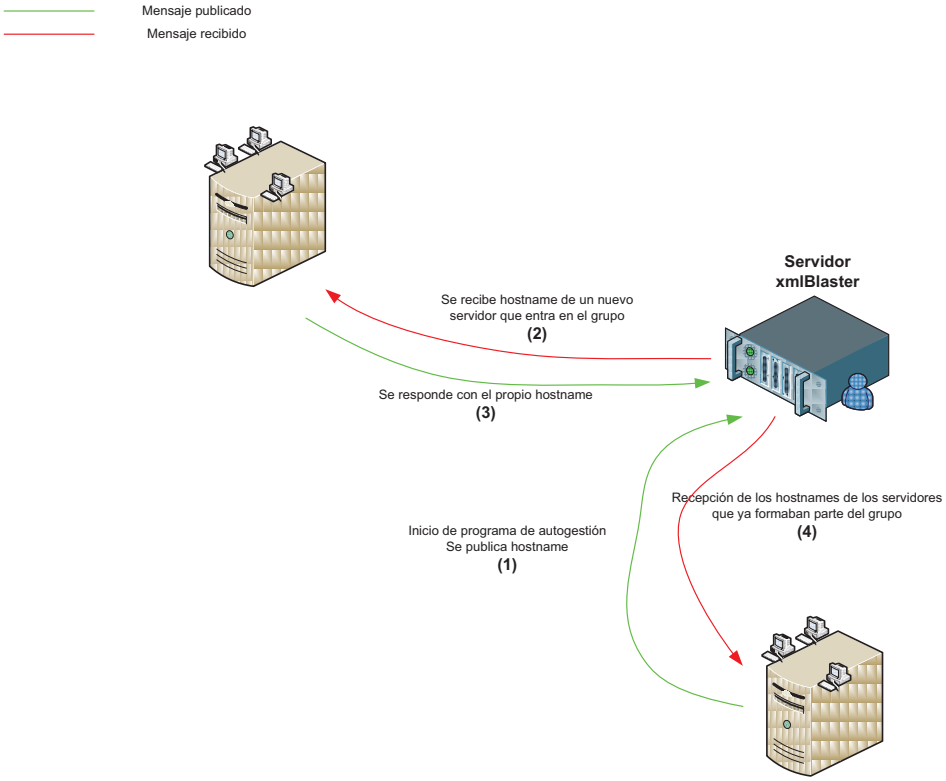


Figura 7: Servidor entrando en grupo de servidores con un solo servidor

En la *Figura 8* un tercer servidor inicia el sistema de gestión autónoma de máquinas virtuales. Al igual que hicieron los otros dos servidores que ya forman parte del grupo, este envía su hostname que es recibido y almacenado por los otros 2. Hecho esto, cada uno de ellos envía su propio hostname, los cuales son recibidos y almacenados por el nuevo servidor.

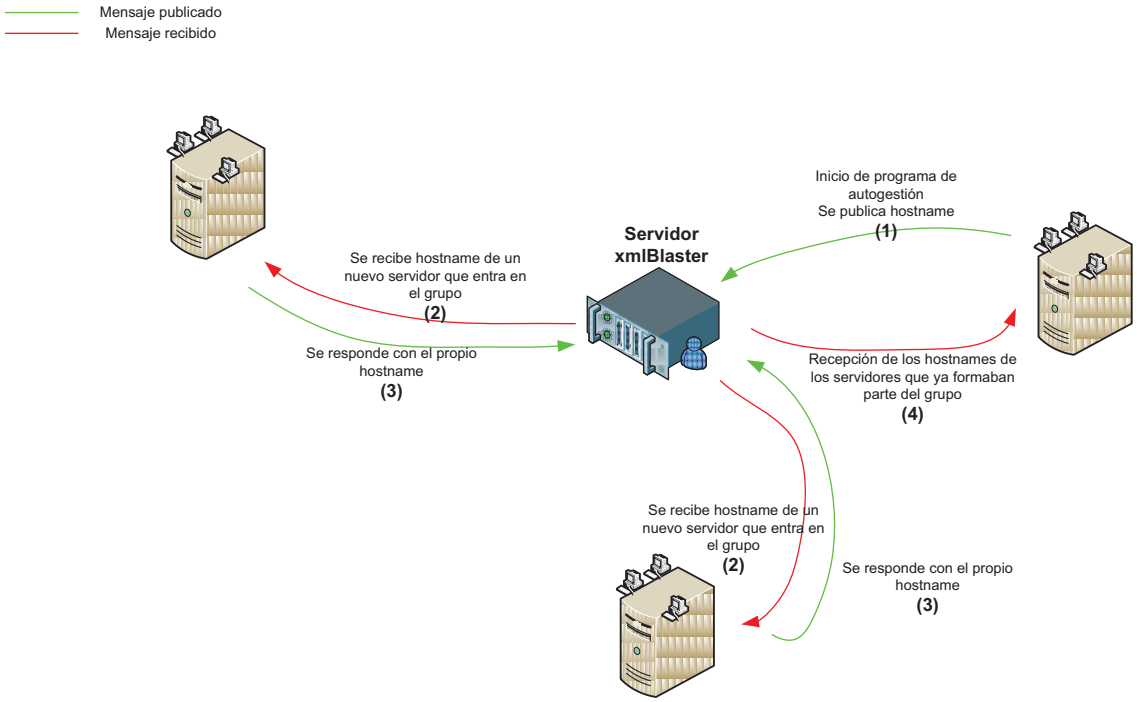


Figura 8: Servidor entrando en grupo de servidores con dos servidores

4.4.1.2. Salida de un servidor del grupo de servidores

Cuando un servidor termina la ejecución del sistema de gestión autónoma de máquinas virtuales debe enviar una notificación al resto de servidores que se encuentran en el grupo de servidores. El propósito de esta notificación es que los servidores borren de su registro el nombre del servidor que abandona el grupo de servidores. El mensaje que envía el servidor que abandona el grupo está formado con el hostname del servidor.

Cuando un servidor termina de ejecutar el sistema de gestión autónoma pregunta al administrador si debe finalizar migrando todas las máquinas virtuales o simplemente finalizar sin migrar ninguna máquina virtual. Por este motivo se deben diferenciar dos casos: salir quedando más servidores en el grupo y salir siendo el último servidor del grupo.

Salir quedando más servidores en el grupo

En este caso existe la posibilidad de migrar todas las máquinas virtuales que se hallen en el servidor al resto de servidores que permanezcan en el grupo. Si se opta por esta opción habrá que esperar a que finalicen todas las posibles migraciones que estén pendientes de terminar y posteriormente se empezarán a migrar las máquinas virtuales del servidor que quiere salir del grupo. Si no es posible migrar todas las máquinas virtuales debido a que empeoran demasiado la carga de los servidores que permanecerán en el grupo, se aplican las políticas de penalización mencionadas en los apartados anteriores y que se detallarán posteriormente. El servidor no terminará de ejecutar el sistema de gestión autónoma hasta que no migre todas las máquinas virtuales, ya que mientras no pueda migrarlas quiere decir que es un servidor necesario para mantener a todos los servidores del grupo por debajo de los umbrales fijados, pero mientras esté en este estado no recibirá máquinas virtuales nuevas provenientes de otros servidores del grupo, ya que estos ya habrán borrado su nombre de su lista y no lo considerarán parte

del grupo. Por otro lado, el servidor que quiere salir del grupo sí es capaz de recibir y almacenar hostnames de nuevos servidores que entren en el grupo. De esta forma tendrá en cuenta a estos nuevos servidores para la migración de máquinas virtuales.

Cuando el servidor termine de migrar las máquinas virtuales el sistema de gestión autónoma terminará.

La *Figura 9*, muestra de forma esquematizada el intercambio de mensajes que se realiza cuando un servidor finaliza el sistema de gestión autónoma y va a abandonar el grupo.

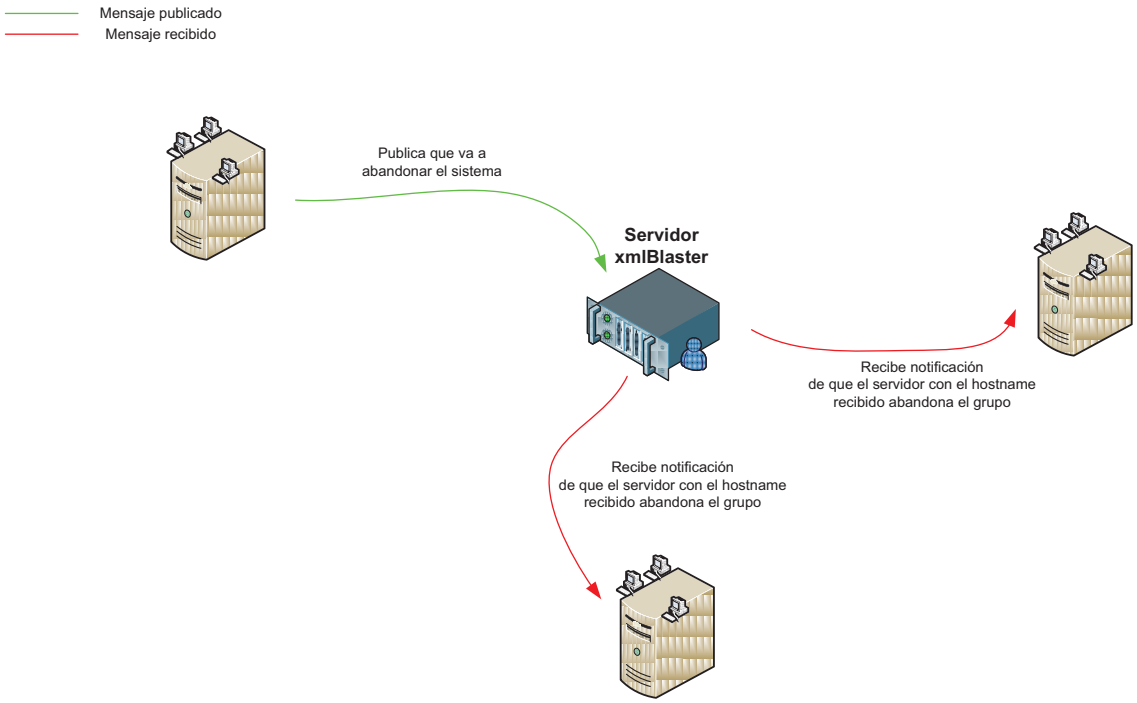


Figura 9: Salida del grupo

Salir siendo el último servidor del grupo

En este caso no será posible salir migrando todas las máquinas virtuales puesto que el sistema no tendrá donde migrarlas y este avisará de ello si se elige esta opción.

En el caso de la salida del último servidor del grupo se deben borrar todas las claves de los mensajes del servidor xmlBlaster que se han utilizado durante la sesión. Esto se debe a que es necesario limpiar estas claves para futuras sesiones sin necesidad de reiniciar el servicio xmlBlaster en el servidor que lo esté ejecutando.

4.4.2. CONOCIMIENTO DE LAS MÁQUINAS VIRTUALES PERTENECIENTES A CADA SERVIDOR

Una de las ventajas de la virtualización que se han comentado en los capítulos previos y el fundamento de este Proyecto Final de Carrera, es la posibilidad de cambiar la máquina virtual de servidor sin necesidad de parada de la misma. Esto tiene como inconveniente que no es posible saber directamente de donde es originaria una máquina virtual, es decir, en que servidor se creó y por lo tanto está definida.

Tener esta información resulta de vital importancia para el sistema de gestión autónoma de máquinas virtuales desarrollado, ya que uno de los parámetros de decisión para la migración mostrado en la *Tabla 1* es llevar un registro de las máquinas virtuales inactivas que se encuentran en el servidor.

El problema reside en que KVM considera que cuando una máquina virtual se migra desde el servidor donde está definida a otro, una vez terminada la migración, en el servidor de origen la máquina virtual quedará como inactiva, cuando en realidad se está ejecutando en otro servidor, por este motivo no debe de ser contabilizada como tal. Por el contrario si la máquina virtual se migra desde un servidor desde el que no fue definida, es decir llegó a él como resultado de una migración previa, al terminar la migración la máquina no quedará registrada en el origen como inactiva.

Por este motivo se ha creado un método que al iniciar el sistema de gestión busca las máquinas virtuales que se han definido en el servidor y las guarda en una lista. A esta lista se la ha llamado *pertenecientes.* Como complemento a *pertenecientes*, se ha creado otra lista que almacena las máquinas virtuales que han sido migradas. A esta lista se la ha llamado *migradas.* Con la información almacenada en estas dos listas es posible determinar si se debe contabilizar una máquina virtual como inactiva de forma correcta.
El funcionamiento del algoritmo es el siguiente:

Al iniciar el sistema de gestión autónoma de máquinas virtuales se crea la lista *pertenecientes* obteniendo las máquinas virtuales activas e inactivas por medio de libvirt y comparando con el archivo de definición de máquinas virtuales del servidor que se encuentra en la ruta */etc/libvirt/qemu*. Este archivo contiene un listado de todas las máquinas que se han definido en el servidor, es decir aquellas que pertenecen al servidor. Gracias a la comparación que se realiza con dicho archivo, se evitará que si al iniciar el sistema se encontraba ejecutándose una máquina virtual que no había sido definida en el servidor, esta no será considerada perteneciente al servidor ya que no aparecerá en el archivo */etc/libvirt/qemu*. Este método tiene como inconveniente que desde el servidor en el que se ha iniciado el sistema de gestión autónoma no se debe haber migrado ninguna máquina virtual de forma manual previamente, ya que si esto ha sucedido, la máquina virtual migrada será contabilizada como una máquina inactiva cuando en realidad se está ejecutando en otro servidor.
Cuando se migre una máquina virtual se comprobará si esta aparece en la lista *pertenecientes* y si esto ocurre la máquina virtual se añadirá a la lista *migradas.* En caso contrario la máquina virtual migrada no se añadirá a ninguna lista.
Por último, para poder contabilizar una máquina virtual como inactiva se realizará previamente una comprobación de que no aparezca en la lista *migradas*. Si la máquina virtual que libvirt da como inactiva aparece en la lista migradas esta no será contabilizada ya que se estará ejecutando en otro servidor.
Será también necesario mantener actualizada la lista *migradas* de forma que si una máquina virtual que fue migrada vuelve al servidor al que pertenece, deje de estar en dicha lista.

A continuación se muestra el pseudocódigo de este proceso que se acaba de describir:

```
Obtener máquinas inactivas con libvirt

Obtener máquinas activas con libvirt

Obtener máquinas definidas en /etc/libvirt/qemu

Si(máquinas inactivas están en /etc/libvirt/qemu)
        Guardar en pertenecientes

Si(máquinas activas están en /etc/libvirt/qemu)
        Guardar en pertenecientes

While(true){
        Si se migra máquina virtual
                Si está en pertenecientes
                        Añadir a migradas

        Si se recibe máquina virtual
                Si está en pertenecientes
                        Quitar de migradas
}
```

4.4.3. AUTOMONITORIZACIÓN

Para conocer el estado del servidor en todo momento es necesario que este monitorice los parámetros vistos en la *Tabla 1* y la *Tabla 2*. Así el servidor podrá conocer su estado y será capaz de tomar decisiones.

Algunos de los parámetros vistos en las tablas varían constantemente con el tiempo y por ello es necesario realizar medidas constantes para actualizar su valor. Otros son fijos y solo es necesario medir una vez y almacenar su valor, como por ejemplo el número de procesadores del servidor.

A continuación se explicará cómo se han obtenido los parámetros más complejos de calcular: la carga del servidor, la carga que genera la máquina virtual y el uso de la interfaz de red.

4.4.3.1. Carga del servidor

La carga del servidor es el uso que se está haciendo del procesador en un momento determinado. En otras palabras, es la cantidad de tiempo que el procesador está trabajando respecto del tiempo total. Un procesador cargado al 100% estaría el 100% del tiempo realizando tareas.

Este parámetro se obtiene directamente con Sigar y se realiza una medición del mismo cada 10 segundos. Una vez obtenido es guardado para poder ser consultado durante los 10 próximos segundos.

4.4.3.2. Carga que genera la máquina virtual

En este punto es necesario diferenciar entre la carga de la máquina virtual y la carga que genera la máquina virtual en el servidor en el que se está ejecutando. Como la carga de la máquina virtual no depende de donde esté hospedada la máquina, para este sistema solo tiene sentido analizar la carga que genera la máquina virtual sobre el servidor. Esto se debe a que, como se ha comentado en los capítulos previos, el sistema de gestión

autónoma de máquinas virtuales siempre pretenderá minimizar la carga de todos los servidores del grupo y no de las máquinas virtuales.

A continuación se detallará el procedimiento seguido para calcular la carga que genera una máquina virtual sobre el servidor en el que se está ejecutando.

Como es lógico, la carga que genera una máquina virtual sobre el servidor depende del número de procesadores y la cantidad de memoria que esta tenga asignada, además de los procesos que se estén ejecutando en la máquina virtual. Una máquina virtual con más procesadores podrá generar más carga en el servidor que otra con menos.

El primer factor que se debe conocer es la ocupación que hace de los procesadores que tiene asignados la máquina virtual. A este factor se le llamará *uso de procesadores*.
El factor *uso de procesadores* se calcula haciendo uso de libvirt. Con libvirt es posible obtener el tiempo de uso de procesador, por lo tanto si se restan dos medidas consecutivas del tiempo de uso del procesador y se dividen por el tiempo entre medidas se obtendrá el tanto por ciento de uso de procesador. La siguiente fórmula muestra el cálculo, donde CPUtime y CPUtime_old son tiempos de procesador obtenidos con libvirt separados por tiempo intervalo de medida que tiene un valor de 10 segundos:

$$uso\ de\ procesadores = \frac{CPUtime - CPUtime_{old}}{intervalo\ de\ medida} \cdot 100$$

Este cálculo dará como resultado valores entre 0 y 100 x (número de procesadores de la máquina virtual) lo que significa que para un valor de 0 la máquina virtual no está generando ninguna carga en el servidor y para una máquina que por ejemplo tuviera dos procesadores si se obtuviera un valor de *uso de procesadores = 150* significaría que la máquina virtual está utilizando 1.5 procesadores de los 2 que tiene asignados.

Una vez obtenido *uso de procesadores*, será necesario conocer los recursos físicos del servidor que consume una máquina virtual, es decir, la cantidad de procesadores del servidor que está utilizando la máquina virtual. Para ello es necesario conocer el número de procesadores del servidor, parámetro obtenido con Sigar. Una vez obtenido este parámetro solo será necesario dividir el *uso de procesadores* entre el número de procesadores del servidor, como se muestra en la siguiente fórmula:

$$recursos\ físicos\ utilizados = \frac{uso\ de\ procesadores}{número\ de\ procesadores\ del\ servidor}$$

Con estos datos, *uso de procesadores* y *recursos físicos utilizados*, junto con la carga del servidor obtenida con Sigar es posible calcular la carga que está generando la máquina virtual sobre el servidor. Este cálculo se realiza multiplicando la carga del servidor por los *recursos físicos utilizados*:

$$\boldsymbol{carga\ generada\ por\ la\ máquina\ virtual\ en\ servidor = carga\ servidor \cdot recursos\ físicos\ utilizados}$$

4.4.3.3. Uso de la interfaz de red

Según se comentó en el capítulo anterior, el sistema implementado tiene como restricción que no se utilice la configuración de bridges para las interfaces de red. Esta

restricción se debe principalmente a que para la migración de máquinas virtuales entre servidores es necesario que si existen bridges configurados estos sean los mismos en los dos servidores que intervienen en la migración para que no se produzca un error.

Dado que no existirán bridges, se ha decidido medir solamente la utilización que hace el servidor de la interfaz por defecto. Se realizan dos mediciones, la utilización que se hace del downlink y la utilización que se hace del uplink. Es necesario mencionar que se realizará una monitorización de la interfaz de red por máquina virtual que se esté ejecutando y no por servidor, por lo tanto no se monitorizará la interfaz de red de un servidor que no tenga máquinas virtuales. El procedimiento seguido para realizar la medición se detalla a continuación.

Para obtener el uso que se está haciendo de la interfaz de red es necesario conocer la velocidad de dicha interfaz. Para ello es necesario conocer el nombre de la interfaz por defecto, es decir, la que queremos monitorizar. Este nombre no tiene por qué ser el mismo en todos los servidores y por ello debe ser comprobado previamente.
Para obtener el nombre de la interfaz primero habrá que comprobar si la máquina virtual está configurada para utilizar un bridge o no. Si está utilizando un bridge, no se monitorizará la interfaz que esté usando dicha máquina virtual.

Para saber si una máquina virtual está utilizando la interfaz por defecto o un bridge es necesario buscar en el archivo XML de definición de la máquina virtual. Esta información se ha obtenido con libvirt. A continuación se muestra un XML de una máquina virtual donde se observa que la interfaz de red utilizada es por defecto y otro XML de una máquina virtual configurada con un bridge, ambos ejemplos están resaltados en rojo.

```
<domain type='kvm'>
  <name>lange</name>
  <uuid>178a5bcc-4a8c-dcd0-3ac1-c6e7677781ce</uuid>
  <memory>1048576</memory>
  <currentMemory>1048576</currentMemory>
  <vcpu>2</vcpu>
  <os>
    <type arch='i686' machine='pc-0.12'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/bin/kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source file='/var/lib/libvirt/images/lange.img'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <disk type='block' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
    </disk>
    <interface type='network'>
      <mac address='52:54:00:4d:f4:31'/>
      <source network='por defecto'/>
      <model type='virtio'/>
    </interface>
    <console type='pty'>
      <target port='0'/>
    </console>
    <console type='pty'>
      <target port='0'/>
    </console>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes'/>
    <sound model='es1370'/>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
    </video>
  </devices>
</domain>
```

```
<domain type='kvm'>
  <name>umu-kvm</name>
  <uuid>ae9eb621-1606-9af8-640d-6b4d92351363</uuid>
  <memory>1048576</memory>
  <currentMemory>1048576</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-0.12'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/bin/qemu</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source file='/var/lib/kvm/umu-kvm/pasito_um3.qcow'/>
      <target dev='hda' bus='ide'/>
    </disk>
    <interface type='bridge'>
      <mac address='00:16:3e:00:b8:cb'/>
      <source bridge='br0'/>
    </interface>
    <input type='mouse' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' listen='127.0.0.1'/>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
    </video>
  </devices>
</domain>
```

Una vez que se sabe si la máquina está utilizando la interfaz por defecto, se procede a averiguar el nombre de dicha interfaz. Este nombre se obtiene haciendo uso de Sigar que provee un método con el que es posible obtener la tabla de rutas del servidor. Una vez obtenida la tabla de rutas es posible identificar la interfaz de la ruta por defecto porque tiene la entrada 0.0.0.0 y de ahí se obtendrá el nombre de la interfaz.

Conociendo el nombre de la interfaz de red y haciendo uso de Sigar se obtienen de forma sencilla el número de bytes trasmitidos y recibidos por dicha interfaz.

El último parámetro necesario para calcular el uso de la interfaz es la velocidad a la que es capaz de transmitir el servidor por dicha interfaz de red. Este parámetro se obtiene ejecutando un comando del sistema llamado ethtool. La forma de ejecución es la siguiente: *ethtool nombre_interfaz* y la salida que muestra está a continuación dónde se ha marcado en rojo la línea que muestra la velocidad.

```
Settings for eth0:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Link partner advertised link modes:  Not reported
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: No
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 1
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: on
        Supports Wake-on: pumbag
        Wake-on: g
        Current message level: 0x00000001 (1)
        Link detected: yes
```

Se ha decidido que el tiempo entre medidas de bytes transmitidos o recibidos sea de 60 segundos. Una vez que se tienen dos medidas consecutivas se calcula la utilización media de la interfaz en los 60 segundos de la siguiente manera: se restan los bytes transmitidos en la medida actual menos los bytes transmitidos en la medida anterior y se divide el resultado entre la velocidad de la interfaz por el tiempo transcurrido entre medidas. El procedimiento para los bytes recibidos es análogo.

4.4.4. AUTODETECCIÓN DE PROBLEMAS

Por tratarse de un sistema autónomo cada servidor se monitoriza a sí mismo y en función de los datos que recopila decide si está en una situación que merece ser atendida para mejorar el rendimiento. Como se comentó en los capítulos previos, el sistema autónomo sólo considerará como parámetro crítico la carga del servidor. Esto quiere decir que de todos los parámetros monitorizados solo se considerará que existe un problema cuando la carga del servidor sea excesivamente alta durante un tiempo prolongado, quedando a elección del administrador del sistema decidir el umbral de carga que no debe rebasar el servidor y el tiempo que debe rebasar dicho umbral para considerar migrar máquinas virtuales. El algoritmo de detección de problema funciona de la forma que se describe a continuación.

Cada 10 segundos se mide la carga del servidor. Si supera el umbral de carga fijado como máximo, se incrementa un contador que cuando llegue al valor máximo permitido provocará que se ponga en marcha el algoritmo de decisión de migración de máquinas virtuales. Cada vez que se mida la carga del servidor, si esta no supera el umbral máximo de carga, el contador se reseteará a cero, por lo tanto solo se iniciará el algoritmo de migración cuando haya un número de medidas de carga superiores al umbral consecutivas, igual al valor máximo permitido de estas medidas. El siguiente pseudocódigo ilustra este algoritmo.

```
if(carga >= umbralCarga){
```

```
        contador ++
        if(contador == umbralContador && existen más servidores en grupo)
                iniciarAlgoritmoDeMigración
                contador = 0
        }
else {
        contador = 0
}
```

4.4.5. SOLICITUD DE INFORMACIÓN A LOS SERVIDORES DEL GRUPO

Esta es la primera fase del algoritmo de migración que se inicia cuando el servidor detecta que su carga ha sido demasiado alta durante un tiempo demasiado largo. Esta fase consiste en recopilar información sobre el estado de todos los servidores del grupo para posteriormente poder realizar una comparación entre ellos y el servidor que la ha solicitado y, si es posible, redistribuir las máquinas virtuales del servidor que tiene la carga demasiado alta para que esté en los márgenes permitidos.

La recopilación de información se inicia enviando tantos mensajes como servidores haya en el grupo excepto el que lo envía. El mensaje está formado por una clave que indica que el mensaje es una solicitud de información y en el contenido el hostname de uno de los servidores del grupo. Todos los servidores recibirán todos los mensajes pero solo actuarán cuando reciban el mensaje que contiene su hostname. El motivo por el que no se envía un único mensaje al que respondan todos los servidores del grupo, es evitar una posible pérdida de las respuestas y así dar tiempo al sistema para que las almacene de forma correcta.

La primera acción que realizarán los servidores cuando reciban la petición de información es dejar de auto-monitorizarse, es decir, dejarán de recopilar datos sobre su estado. Esta acción se realiza para evitar que los servidores que van a recibir máquinas virtuales intenten migrar las suyas y para que cuando termine de migrar máquinas el servidor que solicitó la migración, el resto de servidores del grupo redescubran sus máquinas virtuales. Posteriormente, los servidores, enviarán un mensaje con la información de su estado en ese momento. Dicho mensaje contendrá los parámetros de la *Tabla 1* junto el nombre del servidor que envía el mensaje. La información recibida se almacena para usarla posteriormente en el algoritmo de decisión de migración de máquinas virtuales a servidores. Cuando el servidor recibe la información de todos los servidores del grupo informa al módulo gestor del sistema de gestión autónoma del servidor de que ya tiene las propiedades de todos los servidores preparadas y éste inicia el algoritmo de reubicación de las máquinas virtuales.

4.4.6. ALGORITMO DE REUBICACIÓN DE MÁQUINAS VIRTUALES

La función de este algoritmo consiste en disminuir la carga del servidor que lo ha iniciado por debajo del umbral de carga permitido, migrando máquinas virtuales desde este servidor a los servidores del resto del grupo, comprobando antes de realizar la migración que en el servidor de destino la carga no será mayor que el umbral fijado. Según esto, este algoritmo tendrá dos componentes de predicción:

- Predecir la carga que habrá en el destino *después* de la migración
- Predecir la carga que habrá en el origen *después* de la migración

Es necesario predecir la carga que existirá en el destino justo después de la migración para mantener el servidor que recibe la máquina virtual por debajo del umbral de carga

fijado, ya que no se realizarán migraciones de máquinas virtuales que incrementen tanto la carga en el destino que hagan que no se cumpla la restricción impuesta por el umbral de carga. Así mismo, será necesario predecir la carga que quedará en el servidor justo después de migrar una máquina virtual para saber cuántas máquinas virtuales será necesario migrar para disminuir la carga del servidor por debajo del umbral fijado.
A continuación se describen los pasos que realiza el algoritmo de decisión.

4.4.6.1. Ordenar servidores según puntuación

Una vez recibidas y almacenadas todas las propiedades de los servidores del resto del grupo, se realiza una ordenación de los servidores según sus propiedades. Este orden se realiza asignando una nota a cada servidor que dependerá de las propiedades que ha enviado. El administrador del sistema tiene la capacidad de decidir qué peso asigna a cada propiedad, dividiendo las propiedades en positivas y negativas para calcular la nota media final del servidor.
La *Tabla 4: Propiedades positivas del servidor* indica las propiedades que son deseables en un servidor, es decir, que conviene que sean lo más elevadas posible.

PROPIEDADES POSITIVAS
Memoria total del servidor
Número de CPUs del servidor

Tabla 4: Propiedades positivas del servidor

Por el contrario las propiedades negativas conviene que sean lo más pequeñas posible ya que empeorarán la nota media del servidor. Estas propiedades se muestran en la *Tabla 5: Propiedades negativas del servidor.*

PROPIEDADES NEGATIVAS
Carga del servidor
Máquinas virtuales activas
Máquinas virtuales inactivas
Porcentaje de uso de memoria

Tabla 5: Propiedades negativas del servidor

El cálculo de la nota media del servidor se realiza calculando la nota media positiva del servidor por el peso de las propiedades positivas menos la nota media negativa del servidor por el peso de las propiedades negativas. El siguiente cuadro ilustra los cálculos.

$$nota\ positiva = número\ de\ CPUS \cdot peso\ de\ CPUs + memoria\ total \cdot peso\ de\ memoria$$

$$nota\ negativa = carga\ de\ CPU \cdot peso\ de\ carga + MV_{activas} \cdot peso\ activas + MV_{inactivas} \cdot peso\ inactivas + porcentaje\ de\ uso\ de\ memoria \cdot peso\ de\ porcentaje\ de\ uso\ de\ memoria$$

$$\boldsymbol{puntuación\ del\ servidor = nota\ positiva \cdot 0.333 - nota\ negativa \cdot 0.666}$$

Los factores 0.333 y 0.666 son el peso que tienen la nota positiva y la nota negativa respectivamente. Estos factores se calculan teniendo en cuenta que las propiedades positivas son 2 respecto a 6 propiedades totales, luego representan el 33.3% del total, mientras que las propiedades negativas son 4 respecto a 6 luego representan un 66.6% respecto del total.
Calculada la puntuación media de todos los servidores se procede a ordenar los servidores del grupo de mayor a menor puntuación como muestra el ejemplo de la *Figura 10*, para el que se han utilizado los pesos: peso de CPUs=0.4, peso de memoria=0.6, peso de carga=0.4, peso activas=0.2, peso inactivas=0.05 y peso de porcentaje de uso de memoria=0.35.

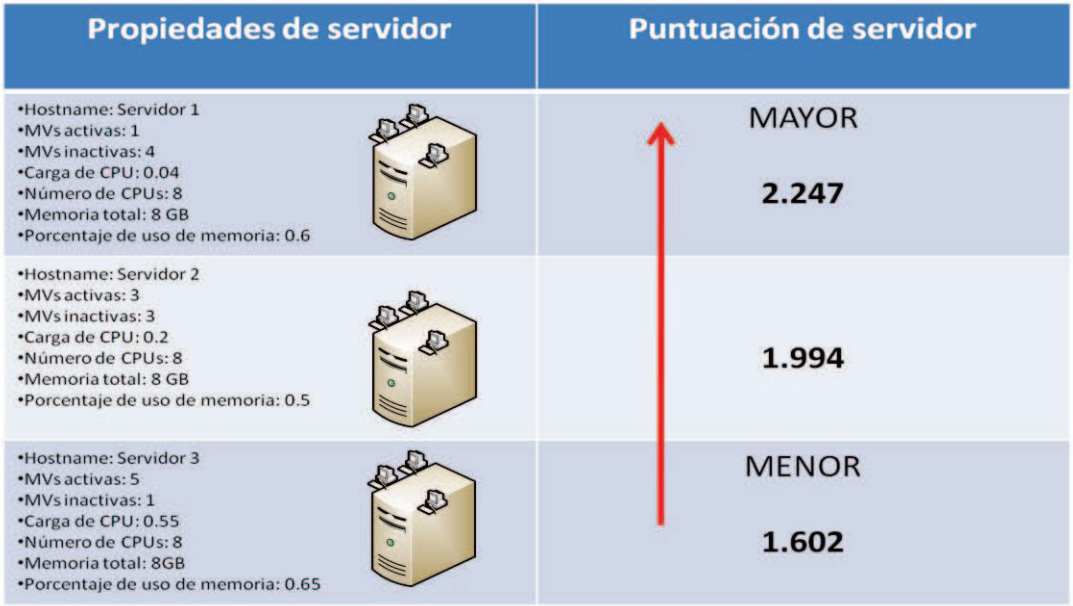


Figura 10: Orden de servidores según puntuación

4.4.6.2. Ordenar Máquinas Virtuales según puntuación

De forma análoga a la ordenación que se ha hecho con los servidores, se asigna una puntuación a las máquinas virtuales del servidor que tiene el problema para ordenarlas también en una lista de mayor a menor puntuación. Sin embargo, ahora tener una puntuación alta significa que la máquina virtual está causando más problemas que otra con una puntuación más baja. Esto se debe a que la puntuación de la máquina virtual se calcula con los parámetros *uso que hace la máquina virtual de la interfaz de red por defecto* y *carga que genera la máquina virtual sobre el servidor*.
El administrador del sistema tiene nuevamente la capacidad de decidir qué peso asigna a cada propiedad de máquina virtual, siendo estas propiedades las mostradas en la *Tabla 6.*

PROPIEDADES DE LA MÁQUINA VIRTUAL
Utilización de Downlink de la Interfaz de Red
Utilización de Uplink de la Interfaz de Red
Carga que genera la MV sobre el servidor

Tabla 6: Propiedades de máquina virtual

Finalmente el siguiente cuadro muestra cómo se calcula la puntuación de una máquina virtual.

$$puntuación\ MV = utlizacion\ downlink \cdot\ peso\ downlink + utilizacion\ uplink \cdot\ peso\ uplink + carga\ generada\ por\ la\ máquina\ virtual\ en\ servidor \cdot\ pesocargaMV$$

Aunque los tres factores de puntuación de la máquina virtual son importantes, la carga que genera la máquina virtual sobre el servidor debe tener un peso significativamente mayor que los otros dos factores. Esto se debe a que, como se comentó anteriormente, las decisiones de migración de máquinas virtuales se realizan en función de la carga generada en el servidor. Sería interesante añadir en un futuro más parámetros de puntuación de máquinas virtuales, como por ejemplo la memoria consumida por una máquina virtual.

La *Figura 11* ilustra un ejemplo de la ordenación de las máquinas virtuales según su puntuación.

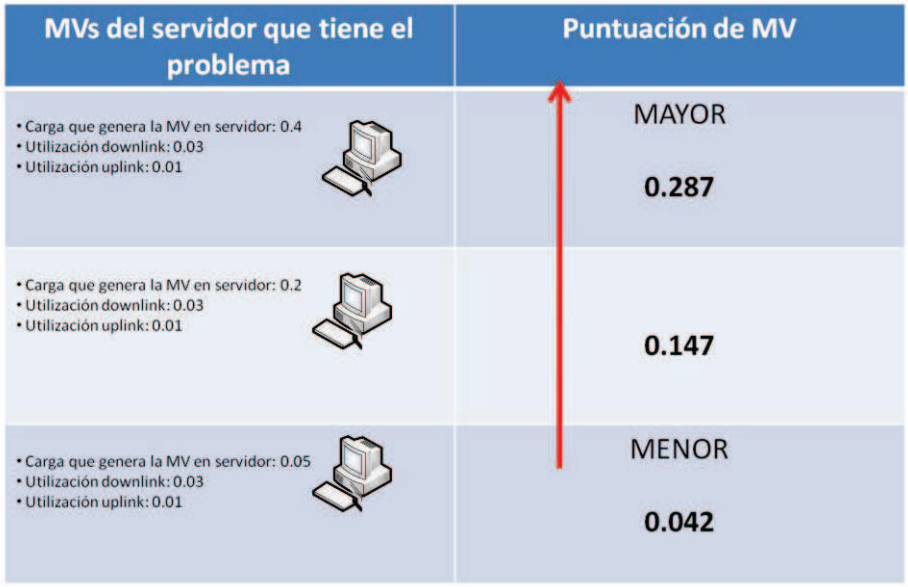


Figura 11: Ordenación de máquinas virtuales

4.4.6.3. Asignación de máquinas virtuales a servidores

Teniendo ordenadas por puntuación tanto las máquinas virtuales como los servidores llega el momento de asignar máquinas virtuales a servidores hasta que la carga en el servidor de origen sea menor que el umbral fijado y siempre respetando que en el destino la carga no sobrepase dicho umbral.

La *Figura 12* ilustra la primera parte del algoritmo, que consiste en asignar la máquina virtual con mayor puntuación al servidor con mayor puntuación, pero antes de que se produzca la asignación se realiza una predicción de la carga que tendrá el servidor de origen después de la migración. Si esta carga es menor que el umbral fijado se asigna dicha máquina virtual a dicho servidor, en caso contrario se intentará asignar la máquina virtual al siguiente servidor de la lista hasta que no queden más servidores en dicha lista cuando se continuará con la siguiente máquina virtual.

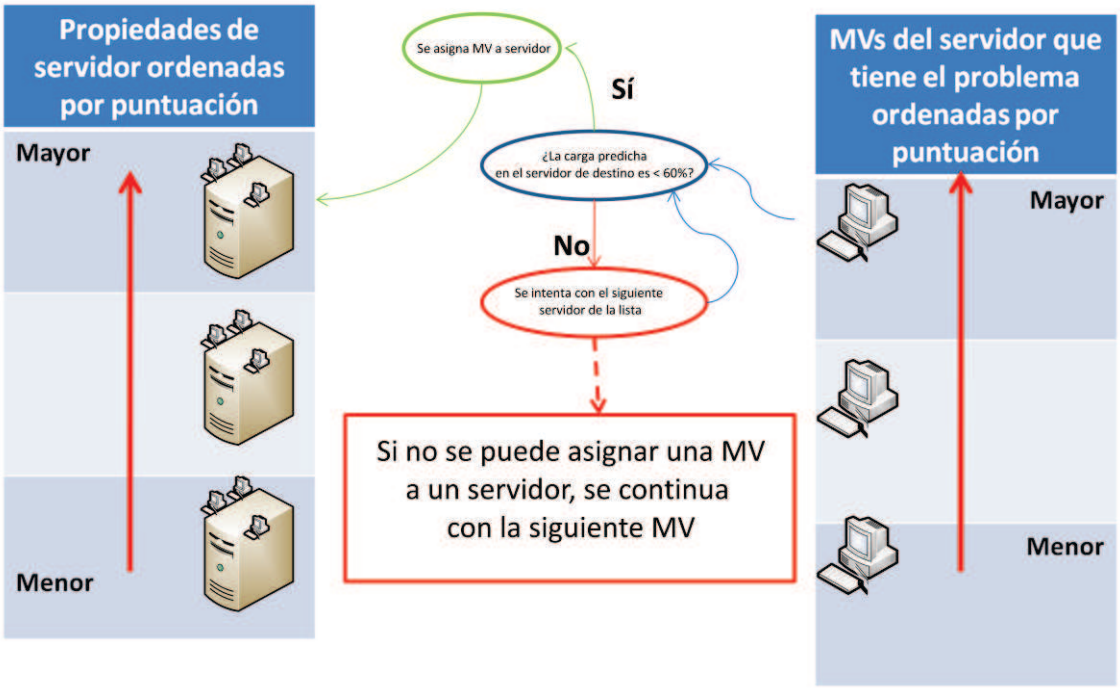


Figura 12: Asignación de MV a servidores

Una vez que se ha conseguido asignar una máquina virtual a un servidor se procede a realizar una reordenación de la lista de servidores ya que el servidor al que se le ha asignado la máquina virtual habrá aumentado su carga y por lo tanto habrá variado su puntuación. Ahora quedará una máquina menos que asignar tal y como se muestra en la *Figura 13*.

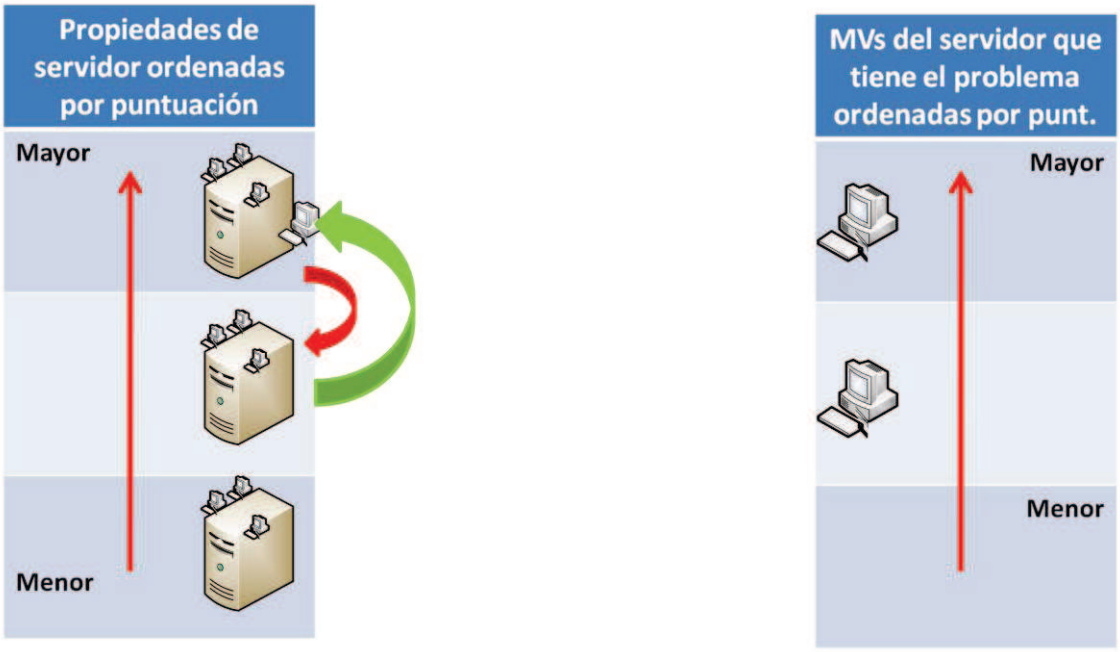


Figura 13: Reordenación de servidores

Terminada la reordenación de los servidores de mayor a menor puntuación, se predice la carga que tendrá el servidor de origen que cuando termine la migración de la máquina virtual que se acaba de asignar. Si la carga predicha está por debajo del umbral de carga fijado, terminará el algoritmo, en caso contrario se repetirá todo el proceso con la siguiente máquina virtual de la lista. Si no se logra disminuir la carga del servidor de origen por debajo del umbral, al menos se habrá conseguido disminuir la carga con las máquinas que se hayan conseguido asignar.
Cuando termina el algoritmo, ya sea de forma satisfactoria o no, se procede a realizar las migraciones de las máquinas virtuales que hayan podido ser asignadas a sus servidores de destino. Una vez terminada la migración de las máquinas virtuales, se informa al resto de los servidores del grupo para que busquen si tienen máquinas nuevas y reanuden su auto-monitorización.

4.4.7. TIEMPO DE ESPERA POR NO RESOLVER EL PROBLEMA

Este tiempo de espera ocurre cuando no se ha podido disminuir la carga del servidor por debajo del umbral de carga por medio de las migraciones. El motivo por el que no se ha podido disminuir la carga se debe a que si se migraban máquinas virtuales al resto de servidores del grupo, estos aumentarían su carga demasiado y pasarían a estar por encima del umbral de carga, situación que no es admisible.
Cuando esto ocurre, el servidor de origen seguirá teniendo una carga por encima del umbral, aunque probablemente se hayan podido migrar algunas máquinas virtuales y por lo tanto habrá sido posible disminuir la carga que tenía el servidor cuando solicitó información al resto de servidores del grupo. Siendo esta la situación, no es admisible que el servidor con una carga demasiado alta solicite de forma instantánea migrar máquinas virtuales de nuevo porque lo más probable es que siga sin ser posible reubicar sus máquinas virtuales. Por este motivo, existe un parámetro que indica un tiempo de espera que entra en acción cuando no se ha podido resolver el problema.

Este tiempo de penalización que es asignable por el administrador del sistema, impide que el servidor solicite de nuevo información al resto de servidores del grupo hasta que este no se cumpla. La motivación de este parámetro es que en este tiempo bien terminen procesos del servidor de origen y que por lo tanto disminuya su carga o bien que el resto de servidores del grupo terminen sus procesos y que sea posible migrar máquinas virtuales.

Aunque inicialmente se asigna un tiempo de penalización, este no es fijo si no que es dinámico. Su funcionamiento es el siguiente:

- Si no se ha solucionado el problema se debe cumplir el tiempo de penalización asignado.
- En el siguiente intento si no se consigue solucionar el problema, el tiempo de penalización se incrementa 10 segundos.
- Cada vez que no se solucione el problema el tiempo de penalización se incrementa 10 segundos.
- Si la carga del servidor disminuye por debajo del umbral durante el tiempo de penalización, se habrá resuelto el problema y el servidor dejará de estar en penalización.

4.5. IMPLEMENTACIÓN

En este apartado se muestra la interconexión de los diferentes módulos desarrollados en código Java para realizar la implementación del diseño.

La *Figura 14*, representa de forma esquemática el código desarrollado. En esta figura se muestran las clases implementadas y las APIs que se han utilizado para obtener información adicional: libvirt, xmlBlaster y Sigar. Las flechas representan la interconexión y la relación entre cada una de estas clases, indicando las flechas verdes que se obtiene información para la clase desde la que se inicia la flecha desde la clase en la que termina la flecha. Las flechas rojas representan una transferencia de información desde la clase en la que se inicia la flecha hacia la clase en la que termina.

Para una mayor comprensión de la función de cada clase, consultar el Anexo C, dónde se proporciona una breve descripción de cada clase y se listan todos sus métodos junto con una breve explicación de la función realizada por cada uno de ellos.

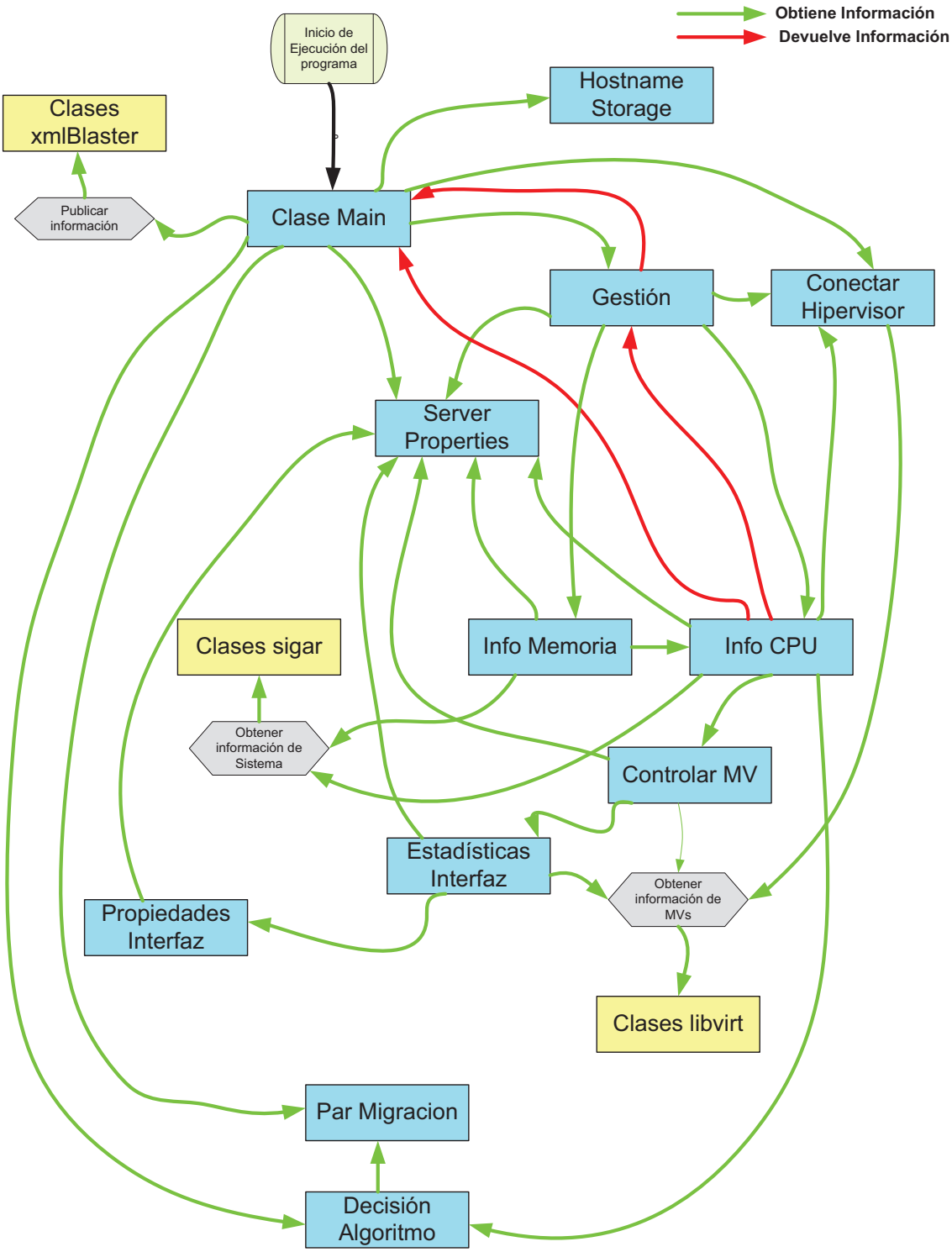


Figura 14: Implementación, interconexión de elementos

4.6. CONCLUSIONES

Este capítulo detalla el diseño del proyecto, así como las decisiones que se han tomado para conseguir dar solución al problema planteado.

Todas las acciones que se realizan en este sistema están encaminadas a mantener la carga de los servidores del grupo por debajo del umbral de carga fijado. Esto se consigue con el movimiento de máquinas virtuales entre los servidores que forman el grupo. Los servidores auto-monitorizan ciertos parámetros que intercambiarán con el

resto. Esta información les servirá para poder comparar el estado de los servidores y decidir dónde mover las máquinas virtuales.
Las pruebas que se han realizado sobre el diseño y la implementación, mostrando la precisión de las predicciones de carga futura y la cantidad de movimientos de máquinas virtuales que se deben realizar para conseguir una carga menor que el umbral en todos los servidores del grupo.

5. VALIDACIÓN DEL DISEÑO

5.1. INTRODUCCIÓN

El diseño y la posterior implementación de la solución al problema propuesto, dan como resultado un sistema real que debe ser probado para ver en qué medida los resultados prácticos se asemejan a los resultados teóricos esperados. Las pruebas presentadas en este capítulo se han realizado montando el sistema implementado sobre un entorno formado por dos servidores con sistema operativo Ubuntu Server 10.04.2 LTS 64 bits.
No serán objeto de este capítulo las pruebas realizadas durante el desarrollo, destinadas a comprobar el correcto funcionamiento, si no que serán pruebas diseñadas con objeto de evaluar la validez de la solución propuesta.
Las pruebas realizadas también servirán para realizar un estudio del comportamiento de los servidores cuando se está produciendo una migración de una máquina virtual. Se estudiarán los efectos que tiene una migración en la memoria del servidor y en el uso de procesador.
Para explicar la validación del diseño, en primer lugar se explicará la metodología empleada en la realización de 9 casos de prueba. En cada uno se detalla la descripción del caso, el resultado esperado y el resultado obtenido.
Seguidamente, se realiza un estudio de los errores cometidos en el algoritmo de predicción de carga.
Finalmente, se adjunta un estudio de los efectos que produce la migración de máquinas virtuales en caliente sobre la red y sobre el procesador. Este estudio facilita el conocimiento de las aplicaciones prácticas del diseño una vez realizadas las validaciones.

5.2. METODOLOGÍA

Para la realización de las pruebas sobre el sistema implementado se han diseñado varios casos de prueba, destinados a testear diferentes situaciones para las que se ha previsto una solución en el diseño.
Cada caso de prueba constará de una descripción que expondrá la situación de los servidores que componen el grupo de servidores y como se ha llegado a dicha situación, de forma que el caso de prueba sea reproducible. En la descripción también constará el valor de los parámetros relevantes escogidos por el administrador para realizar el experimento.
Dada la descripción se explicará cuál es el resultado esperado según el diseño y cuál debe ser la situación final del sistema.
Posteriormente se mostrarán de forma gráfica los resultados obtenidos, analizando si se han cumplido las expectativas previstas y los errores cometidos respecto del resultado teórico. Para este cometido se mostrarán gráficas de la carga de los servidores, con las predicciones hechas durante la ejecución del programa, gráficas de la memoria de los servidores y gráficas con los errores cometidos en las predicciones.

5.3. CASO DE PRUEBA 1

5.3.1. DESCRIPCIÓN

Este caso representa la situación más simple posible y servirá para ilustrar el formato de las gráficas expuestas así como una breve explicación de cómo han sido obtenidas.

La prueba se ha realizado sobre dos servidores idénticos que ejecutan el sistema de gestión autónoma de máquinas virtuales diseñado.
El procedimiento seguido ha consistido en aumentar la carga de uno de los dos servidores por encima del umbral de carga máximo permitido, fijado en este caso al 60%, pero dejando el nivel de carga en un valor próximo a este umbral. Este nivel de carga se ha conseguido haciendo que las máquinas virtuales de dicho servidor ejecuten tareas que requieren uso de procesador.
Conseguido el nivel de carga adecuado, se inicia en el servidor el sistema de gestión autónoma de máquinas virtuales. Transcurridos unos segundos desde el inicio del sistema de gestión en el servidor con la carga elevada, se inicia el sistema en el otro servidor, cuyas máquinas virtuales no están ejecutando ningún proceso y por lo tanto su carga es casi nula.
Se fija el número de medidas de carga mayores que el umbral para considerar que existe un problema a dos.

5.3.2. RESULTADO ESPERADO

Mientras que el servidor con una carga elevada, por encima del 60%, sea el único que forma el grupo de servidores, no hará intentos de migración de máquinas virtuales porque no tendrá otros servidores a los que migrar.
Una vez que se inicie el sistema de gestión autónoma en el otro servidor y transcurran dos medidas consecutivas de carga mayores que el 60 % en el servidor con carga elevada se migrarán máquinas virtuales al nuevo servidor que ha entrado en el grupo, quedando los dos servidores que forman el grupo de servidores con una carga menor que el umbral de carga fijado (60 %).

5.3.3. RESULTADO OBTENIDO

La *Figura 15* muestra la carga en función del tiempo de los dos servidores que forman parte del grupo de servidores. Todas las figuras de las pruebas realizadas seguirán el mismo formato, representado:

- la línea **azul** el servidor 1
- la línea **roja** el servidor 2
- la línea **negra** la predicción de carga en el **servidor de origen** una vez que termine la migración
- la línea **verde** la predicción de carga que tendrá el **servidor de destino** una vez que termine la migración.

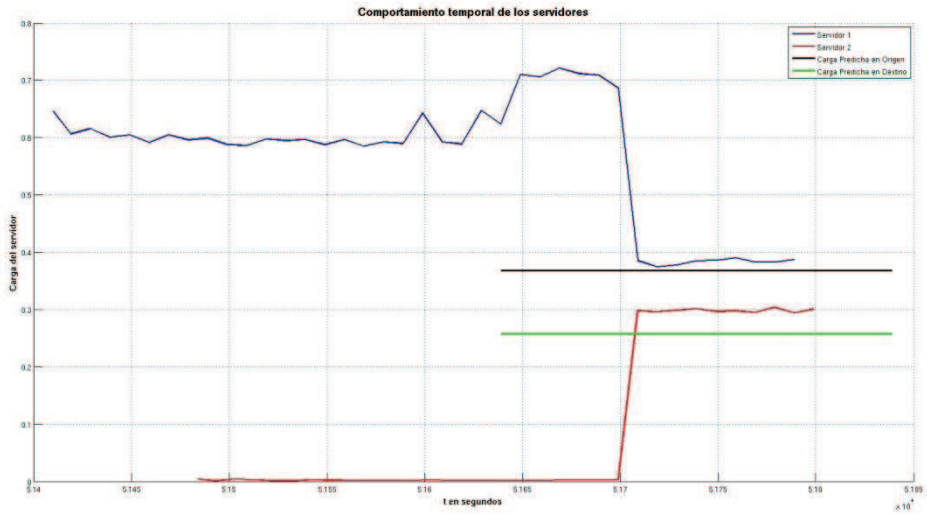


Figura 15: Carga, Prueba1

En la figura, la línea azul muestra que el servidor 1 tiene una carga que oscila en torno a valores del 60%. Inicialmente no hay más servidores en el grupo y por lo tanto aunque se toman más de dos medidas de carga consecutivas mayores que el 60%, el servidor no hace intentos de migración de máquinas virtuales. Esto se sabe porque no se pinta ninguna predicción en este intervalo de tiempo y según el diseño si no se realiza predicción no hay intento de migración.
Cuando entra el servidor 2 a formar parte del grupo de servidores, el servidor 1 puede solicitar migrar máquinas virtuales pero según se muestra en la figura esto no ocurre hasta que no se producen de nuevo dos medidas consecutivas de la carga mayores que el 60%. En este momento aparecen las líneas negra y verde que representan las predicciones que se realizan sobre la carga, una vez terminada la migración de máquinas virtuales, del servidor 1 (origen) y el servidor 2 (destino) respectivamente.
Justo después de la predicción de carga comienza la migración de máquinas virtuales, en este caso una, y transcurridos unos segundos termina la migración descendiendo la carga del servidor 1 y aumentando la carga del servidor 2, quedando ambos servidores por debajo del 60 %.
Como se observa en la figura la predicción de carga en el origen es algo más precisa que en el destino pero ambas representan un valor válido para poder ser tenido en cuenta.
Desde el momento de la predicción hasta que finaliza la migración trascurren aproximadamente 70 segundos por lo tanto ese es el tiempo que se ha tardado en migrar la máquina virtual.

Para este caso de prueba se va a detallar como se ha realizado el estudio sobre los errores cometidos en las predicciones, tanto en la carga estimada en el origen como en el destino. Al final del capítulo se mostrarán los errores de todos los casos de prueba pero solamente se explicarán los cálculos en este caso.

Para analizar el error cometido en la predicción, se deben distinguir dos casos ya que se realizan dos predicciones: predicción de carga en el origen después de la migración y predicción de carga en el destino después de migración. El cálculo en ambos casos es completamente análogo y se realiza de la siguiente manera:
El primer paso consiste en calcular los mínimos cuadrados de la carga del servidor en cuestión después de la migración, cogiendo unos valores de carga coherentes, es decir,

que no sean demasiado lejanos en el tiempo al fin de la migración y que el servidor no haya sido alterado por otras perturbaciones.
Realizado este cálculo se obtiene el valor absoluto de la resta de la recta obtenida por mínimos cuadrados y del valor predicho. Este cálculo proporciona el error absoluto.
Finalmente conociendo el error absoluto es posible conocer el error porcentual respecto a la carga predicha.

La *Figura 16* muestra gráficamente los resultados de estos cálculos para el caso de prueba 1 para la predicción de carga realizada para el servidor 1 (origen).
Esta figura representa con puntos rojos los valores de carga medidos después de la migración de la máquina virtual y con una línea azul la aproximación lineal por mínimos cuadrados a estos valores medidos. La línea negra es la carga del servidor obtenida en la predicción, por lo tanto el error en valor absoluto, representado en rojo, será el valor absoluto de la resta de la representación lineal por mínimos cuadrados y la predicción (línea azul – línea negra).

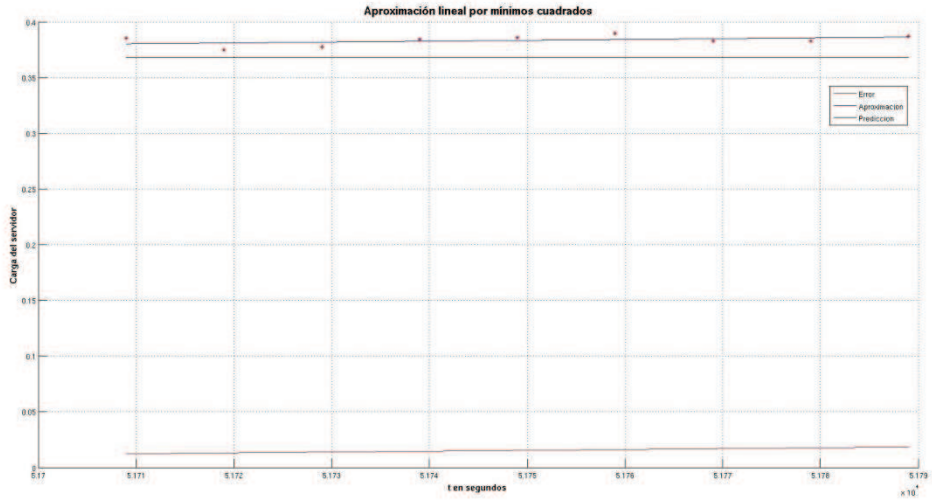


Figura 16: Mínimos cuadrados origen, Prueba 1

Conociendo el valor absoluto del error cometido en la predicción, es posible conocer qué porcentaje representa este error respecto del valor predicho. Este cálculo se muestra en el siguiente cuadro.

$$porcentaje\ de\ error = \frac{error\ en\ valor\ absoluto \cdot\ 100}{carga\ predicha}$$

A continuación se muestran de forma gráfica los errores cometidos en la predicción en origen para el caso de prueba 1 en las figuras: *Figura 17* y *Figura 18*. Estos errores son más realistas cuanto menor es el tiempo que ha pasado desde que se hizo la predicción, es decir, la predicción está hecha para la carga que tendrá el servidor justo después de la finalizar la migración de las máquinas virtuales.

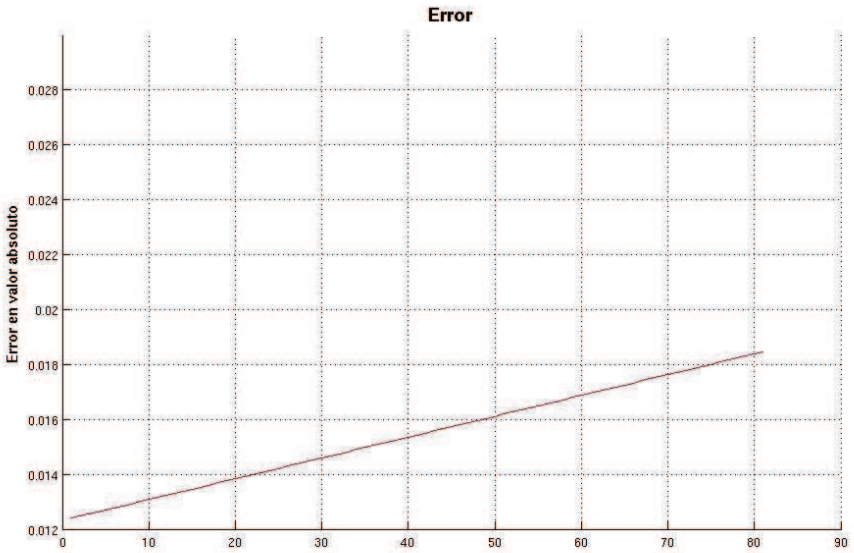


Figura 17: Error absoluto origen, Prueba 1

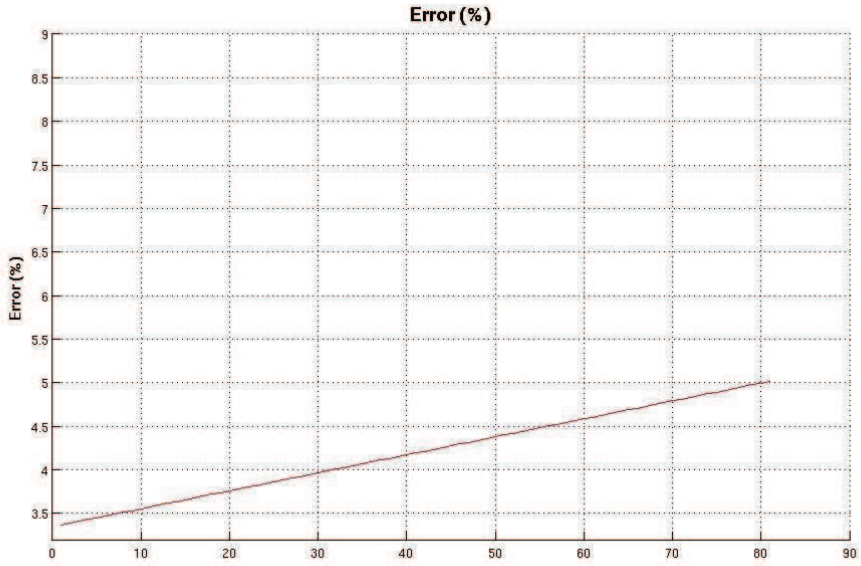


Figura 18: Error porcentual origen, Prueba 1

Las gráficas de error porcentual representan valores muy grandes que pueden llevar a sacar conclusiones equivocadas ya que están calculadas respecto del error calculado en valor absoluto que es el realmente importante. Estos valores elevados se deben a que un error absoluto pequeño representa un valor porcentual muy alto respecto de los valores de carga que son también pequeños.

Estas gráficas se obtendrán para cada prueba realizada pero solamente se mostrará el proceso de obtención para este caso. Terminados todos los casos de prueba se mostrarán de forma conjunta todos los errores obtenidos en los distintos casos.

A continuación, se muestran las gráficas del error cometido en el destino (servidor 2) calculadas de forma análoga a como se ha hecho con el origen (servidor 1).

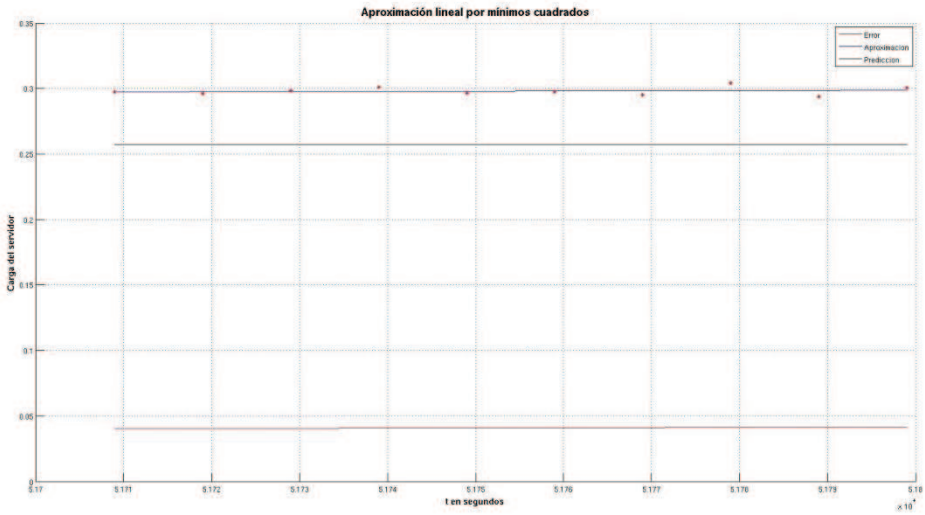


Figura 19: Mínimos cuadrados destino, Prueba 1

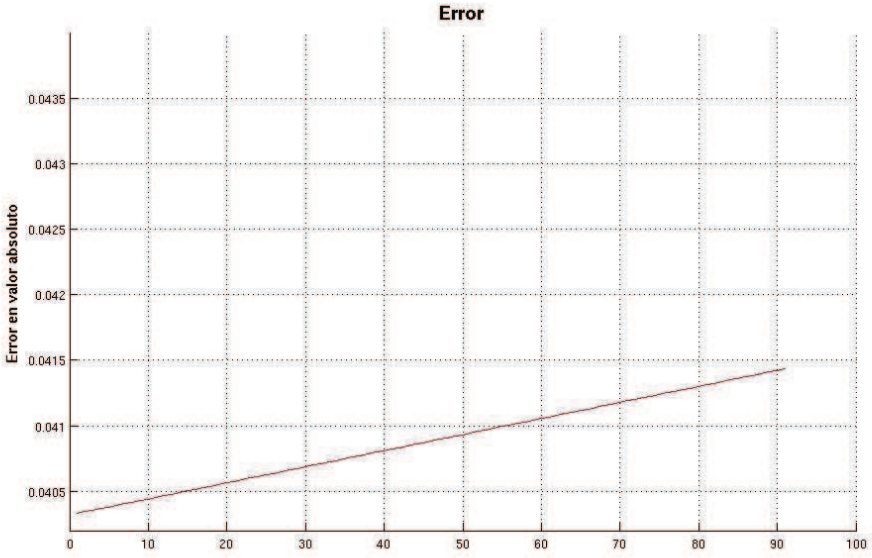


Figura 20: Error absoluto destino, Prueba 1

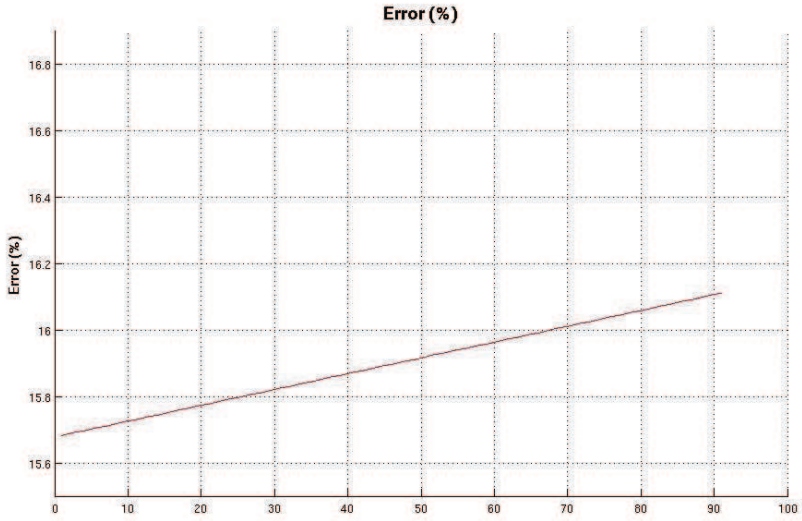


Figura 21: Error porcentual destino, Prueba 1

Existe la posibilidad de analizar el comportamiento del grupo de servidores desde el punto de vista de la memoria del servidor. Para ello se han obtenido las gráficas que muestran el porcentaje de uso de memoria de los servidores del grupo. Estas gráficas hacen posible visualizar que cuando una máquina virtual se migra de servidor, el servidor de origen libera toda la memoria asignada a la máquina virtual que pasa a ocuparse en el servidor de destino cuando termina la migración. Así mismo es especialmente relevante el hecho de que una vez terminada la migración de la máquina virtual, se tiende a ocupar algo de memoria en el destino que no corresponde a la máquina virtual. Esta memoria es liberada unos segundos después y hace sospechar que es memoria que ocupa el proceso de migración. Este comportamiento no se observa demasiado bien en este caso de prueba pero será especialmente relevante en los siguientes.
La *Figura 22* ilustra este comportamiento para el caso de prueba 1.

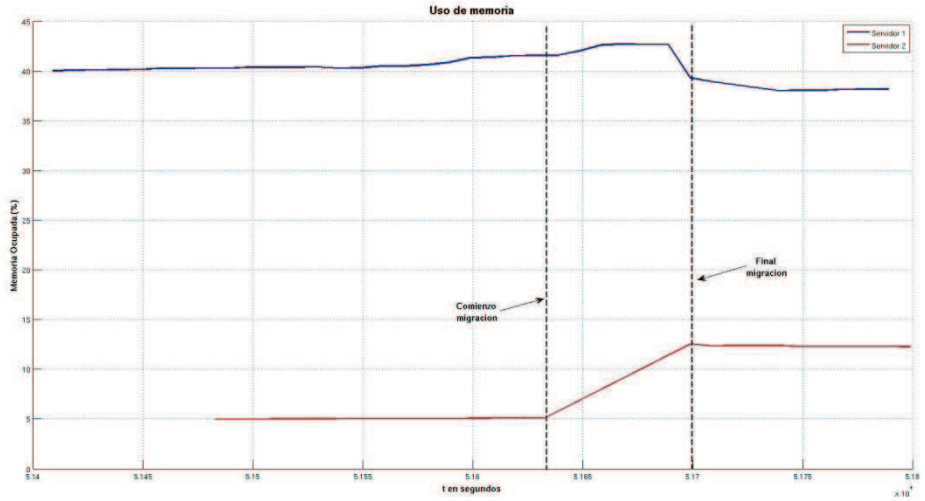


Figura 22: Memoria, Prueba 1

5.4. CASO DE PRUEBA 2

5.4.1. DESCRIPCIÓN

Se dispone de dos servidores sobre los que ejecutar el sistema de gestión autónoma de máquinas virtuales. Al igual que en el caso anterior se fija el umbral de carga máxima permitida al 60% y el número de mediciones de carga mayores que dicho umbral debe ser dos para considerar que existe un problema.
Se inicia el sistema de gestión autónoma en uno de los dos servidores y se ejecutan procesos sobre las máquinas virtuales de dicho servidor con el fin de aumentar la carga del servidor a valores lo más altos posible (mayor del 70%); a diferencia del caso anterior, en el que la carga se elevó a valores ligeramente por encima del 60%.
Transcurrido un tiempo se inicia el sistema de gestión autónoma en el segundo servidor, el cual tiene sus máquinas virtuales a un nivel de utilización muy bajo y por lo tanto el servidor tiene una carga muy baja.

5.4.2. RESULTADO ESPERADO

Al tener el servidor valores muy altos de carga, se espera que para conseguir que su carga esté por debajo del valor máximo permitido (60%) este tenga que migrar más de una máquina virtual sin que el servidor de destino supere el umbral de carga.
Como inicialmente solo habrá un servidor en el grupo, este no realizará intentos de migración de máquinas virtuales y por lo tanto no habrá predicciones de carga hasta que no haya dos servidores en el grupo.

5.4.3. RESULTADO OBTENIDO

La *Figura 23* muestra la carga de los servidores en función del tiempo. Como se puede observar, el servidor 1 tiene valores muy elevados de carga que están por encima del 60%. Mientras que este servidor está solo en el grupo de servidores no se realiza ningún intento de migración de máquinas virtuales pese a que hay constantes medidas consecutivas por encima del 60% de carga y solo son necesarias dos para considerar que el servidor debe migrar máquinas virtuales.

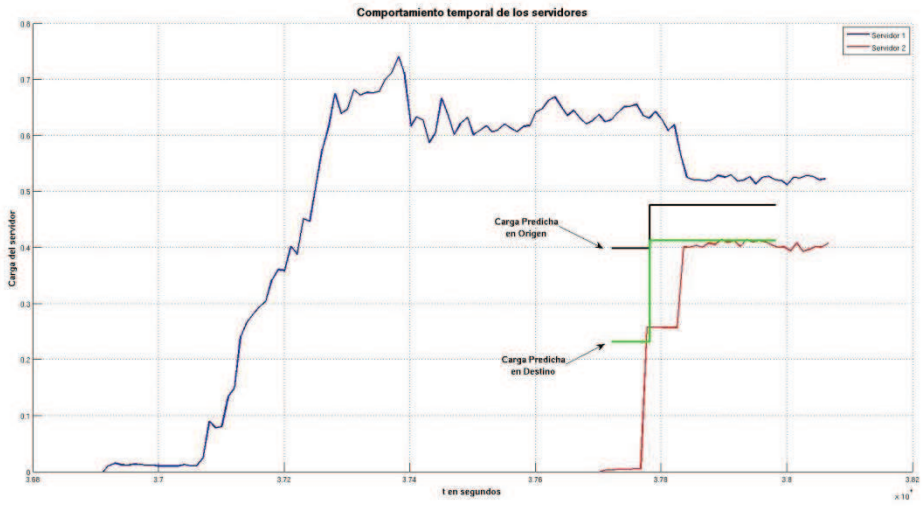


Figura 23: Carga, Prueba 2

Cuando se inicia el sistema de gestión autónoma de máquinas virtuales en el servidor 2 y se toman dos medidas consecutivas de carga por encima del 60% en el servidor 1, se inicia el algoritmo de migración de máquinas virtuales que realiza una primera predicción de carga tanto en el servidor de origen como de destino. Esta primera predicción para el servidor 1 (origen) es bastante desacertada, representada por el primer tramo de la línea negra, como se ve en la carga que sigue teniendo el servidor 1 cuando termina la primera migración. Por este motivo, terminada la migración dado que el servidor 1 sigue con una carga mayor que el 60% se inicia de nuevo el algoritmo de decisión, migrando una segunda máquina virtual ahora la predicción en el origen está más acertada, segundo tramo de la línea negra, y se consigue disminuir la carga por debajo del umbral, sin aumentarla demasiado en el destino (servidor 2).

Estos errores en la predicción en el origen tan grandes se deben a que el servidor 1 estaba saturado, es decir, no podía satisfacer las necesidades de todas las máquinas virtuales que se estaban ejecutando sobre él. Cuando un servidor está en este estado, al

migrar una máquina virtual su carga apenas disminuye porque hay otras máquinas virtuales que están demandando trabajo que debe ser realizado y pasan a ser planificadas por el hipervisor de forma más continuada cuando se ha terminado la migración.

Este caso de prueba también puede ser visto desde el punto de vista de la memoria. La *Figura 24* ilustra las variaciones de la memoria de ambos servidores durante la prueba. En esta figura se observan las dos migraciones que tienen lugar en el experimento, apreciándose en cada migración como disminuye la cantidad de memoria utilizada del servidor 1 y aumenta la del servidor 2. Observando la memoria del servidor 2 al final de la segunda migración es fácilmente apreciable la característica comentada en el caso de prueba 1 en el que al finalizar la migración la cantidad de memoria ocupada es ligeramente superior a lo que debería ser. Esta característica se debe a que para conseguir migrar la máquina virtual es necesario utilizar algo más de memoria que la que tiene la propia máquina virtual en sí misma. Transcurridos unos segundos, esta memoria adicional se libera, quedando ocupada la memoria que corresponde a la máquina virtual migrada.

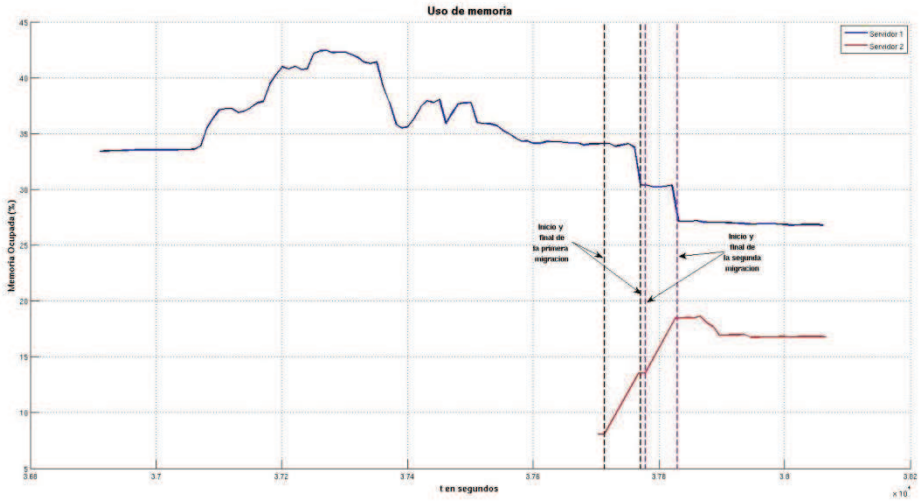


Figura 24: Memoria, Prueba 2

5.5. CASO DE PRUEBA 3

5.5.1. DESCRIPCIÓN

El grupo de servidores estará formado por dos servidores, siendo la carga máxima permitida el 60% y el número de medidas consecutivas de carga superiores a este umbral para considerar que existe un problema debe ser dos. El tiempo de penalización inicial elegido es de 190 segundos, incrementándose en 10 segundos cada vez que se intenten migrar máquinas virtuales de forma insatisfactoria, como se describió en el diseño.
Se inicia el programa de gestión autónoma en ambos servidores. Uno de los servidores tendrá una carga muy elevada, por encima del 70% debido a que sus máquinas virtuales están ejecutando muchos procesos. El otro servidor del grupo tendrá una carga casi nula.
Cuando se estabilice el sistema se aumentará la carga del servidor que originariamente tenía la carga más baja de forma manual. Posteriormente se migrara máquinas de forma manual desde el servidor con la carga originaria más baja.

5.5.2. RESULTADO ESPERADO

Inicialmente, se deben transferir máquinas virtuales desde el servidor con la carga más alta al que la tienen casi nula. El sistema se debe estabilizar con ambos servidores con cargas por debajo del 60% pero con un valor bastante próximo a este umbral.
Cuando se aumente la carga del servidor 2 de forma manual por encima del 60%, este intentará migrar máquinas al servidor 1 pero no podrá porque aumentaría la carga de éste por encima del 60%.
Cuando se migren de forma manual máquinas desde el servidor 2 al servidor 1, la carga de este último aumentará por encima del umbral fijado al 60% pero al tratarse de una migración manual, el servidor admitirá las máquinas. Por supuesto, el servidor 1 intentará migrar máquinas virtuales si es posible hacerlo y en caso contrario tendrá que respetar los tiempos de penalización diseñados antes de poder solicitar una nueva migración de máquinas virtuales.

5.5.3. RESULTADO OBTENIDO

La *Figura 25* muestra gráficamente las variaciones de la carga en función del tiempo de los dos servidores que forman el grupo. Como se puede apreciar, en este caso aparecen dos líneas negras y dos líneas verdes. Esto es debido a que cada una de las líneas del mismo color representa la predicción de carga para el servidor 1 o el 2; siendo el color negro la predicción para el origen y el verde la predicción para el destino. Tanto el servidor 1 como el 2 pueden ser origen o destino, dependiendo del caso.

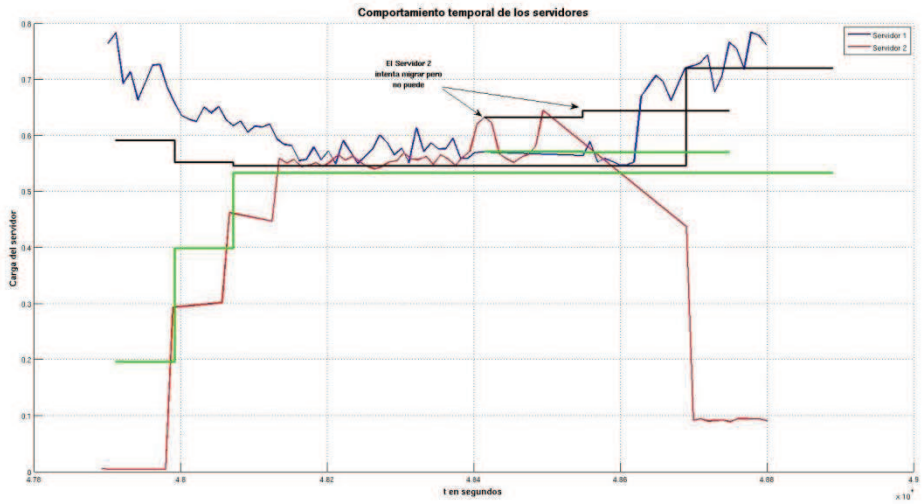


Figura 25: Carga, Prueba 3

La primera parte de la figura, donde la carga del servidor 1 es de más del 70% y la del servidor 2 es prácticamente nula, representa la migración de máquinas virtuales desde el servidor 1 al servidor 2. Nuevamente al igual que en el caso anterior, la predicción de carga realizada para el servidor 1 (origen) es bastante desacertada por los mismos motivos que se explicaron en el caso anterior, el servidor está saturado. Aunque es apreciable que cuanto más disminuye la carga más acertada es la predicción en el servidor de origen.

Cuando el servidor 1 consigue estar por debajo del 60% de carga, deja de migrar máquinas virtuales y el sistema se estabiliza, quedando los dos servidores con una carga similar por debajo del 60% pero con un valor próximo a este umbral.
Pasado un tiempo, se aumenta de forma manual la carga del servidor 2 para situarla por encima del 60%. Cuando esto ocurre el servidor inicia el algoritmo de decisión, pero concluye que no puede migrar máquinas virtuales porque aumentaría la carga del servidor 1 por encima del 60%. Por este motivo las predicciones pintadas en negro y verde son la carga para el servidor 2 que es ahora el origen y el servidor 1 que es ahora el destino. Como no se va a llevar a cabo la migración de ninguna máquina virtual, las predicciones dicen que todo va a seguir igual. Ahora entra el juego el algoritmo de penalización y el servidor 2 debería esperar 190 segundos antes de solicitar una nueva migración, pero este tiempo de penalización no se respeta como se observa en la figura ya que se hace un nuevo intento transcurridos solamente 134 segundos desde el anterior intento. Esto se debe a que desde el primer intento de migración hasta el segundo la carga del servidor 2 ha disminuido por debajo del 60% y por lo tanto la penalización queda anulada. Como el segundo intento tampoco es satisfactorio se inicia de nuevo la penalización con 190 segundos.
Posteriormente se inicia la migración manual de máquinas virtuales desde el servidor 2 al servidor 1, bajando la carga del servidor 2 por debajo del 60% y quedando anulada la petición de migración y el tiempo de penalización. Con esta operación el servidor 1 aumenta su carga por encima del 60% provocando un intento de migración que no puede ser llevado a cabo porque aumentaría la carga del servidor 2 por encima del umbral permitido. El servidor 2 entra en el tiempo de penalización que no llega a cumplirse porque se apaga el sistema de gestión autónoma en ambos servidores 108 segundos después de este intento fallido de migración.

La gráfica del uso de memoria que tienen los servidores del grupo se puede apreciar en la *Figura 26*. Esta figura revela el movimiento de máquinas virtuales que ha tenido lugar durante el experimento. Como se observa, inicialmente se migran de forma autónoma tres máquinas virtuales desde el servidor 1 al servidor 2. Transcurrido un tiempo en el que el sistema se encuentra estable, se migran máquinas cuatro máquinas desde el servidor 2 al servidor 1.

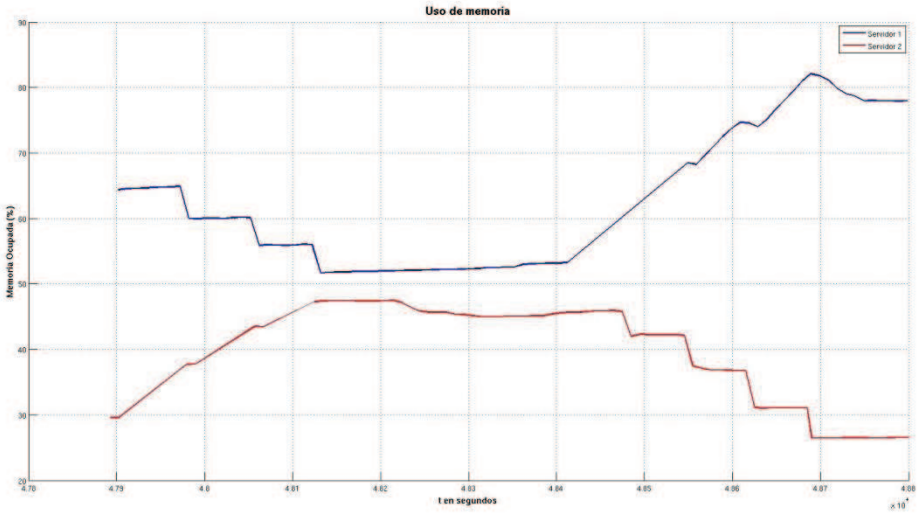


Figura 26: Memoria, Prueba 3

5.6. CASO DE PRUEBA 4

5.6.1. DESCRIPCIÓN

Este caso de prueba pretende demostrar la migración de máquinas virtuales de forma autónoma desde los dos servidores que forman el grupo.
El procedimiento que se sigue consiste en fijar la carga máxima permitida en el servidor al 60% y el número de medidas consecutivas necesarias superiores a este valor para considerar que existe un problema es dos.
Se aumenta la carga de uno de los dos servidores por encima del 60% mientras que el otro servidor se deja con una carga casi nula. Cuando se estabilice el sistema se elevará la carga del otro servidor por encima del 60%.

5.6.2. RESULTADO ESPERADO

Al elevar la carga del servidor 1, este deberá migrar máquinas virtuales al servidor 2, quedando el sistema estabilizado al finalizar las migraciones necesarias. Pasado un tiempo con el sistema estable, se eleva de forma manual la carga del servidor 2 para que se produzcan migraciones de máquinas virtuales desde el servidor 2 al servidor 1, quedando finalmente la carga de ambos servidores por debajo del 60%.

5.6.3. RESULTADO OBTENIDO

La *Figura 27* muestra la carga frente al tiempo del grupo de servidores para este caso de prueba.

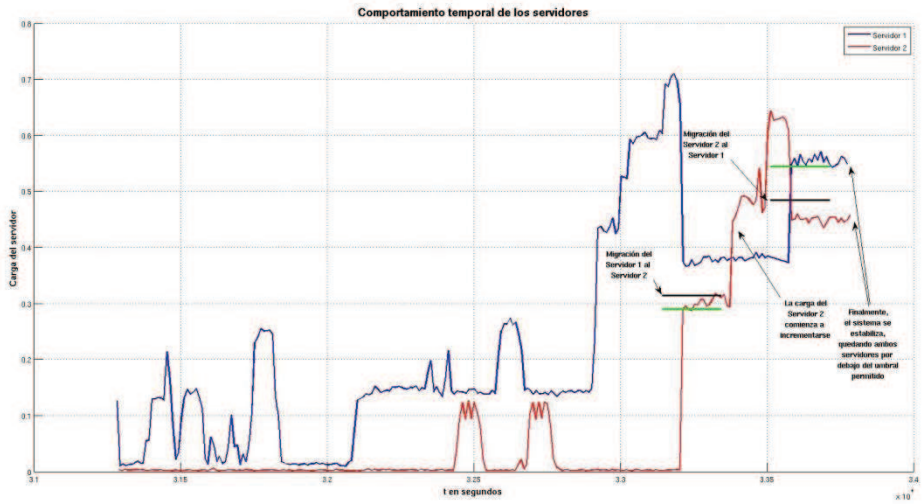


Figura 27: Carga, Prueba 4

En esta figura se observa cómo se van encendiendo las máquinas virtuales en ambos servidores, esto está representado por los picos de carga que se observan al principio de la gráfica.
Cuando se eleva la carga del servidor 1 por encima del 60%, se produce una migración de máquinas virtuales al servidor 2, disminuyendo la carga del servidor 1 al 37% y aumentando la carga del servidor 2 hasta el 30% aproximadamente. Unos segundos después aumenta la carga del servidor 2 hasta valores por encima del 60%, momento en el que se inicia el algoritmo de migración de máquinas virtuales desde el servidor 2

hasta el servidor 1 quedando finalmente ambos servidores con cargas menores al 60%, en concreto el servidor 1 con una carga del 56% y el servidor 2 con 45%.
Como se observa en la figura, se han realizado dos predicciones una para la migración desde el servidor 1 al servidor 2 y otra para la migración desde el servidor 2 al servidor 1. En este punto es necesario recordar que las líneas negras siempre representan la predicción de carga para el servidor de origen de la migración y que las líneas verdes siempre representan la predicción de carga para el servidor de destino de la migración.

La *Figura 28*, muestra la memoria involucrada en este caso de prueba. Esta figura muestra que el número de máquinas virtuales migradas desde el servidor 1 al servidor 2 es una, al igual que en la migración que se realiza desde el servidor 2 al servidor1 donde también se migra una máquina virtual.
En esta figura es fácilmente apreciable la cantidad de memoria adicional que se usa en el servidor de destino cuando tiene lugar una migración. Pasados uno segundos desde el fin de la migración esta memoria se libera.

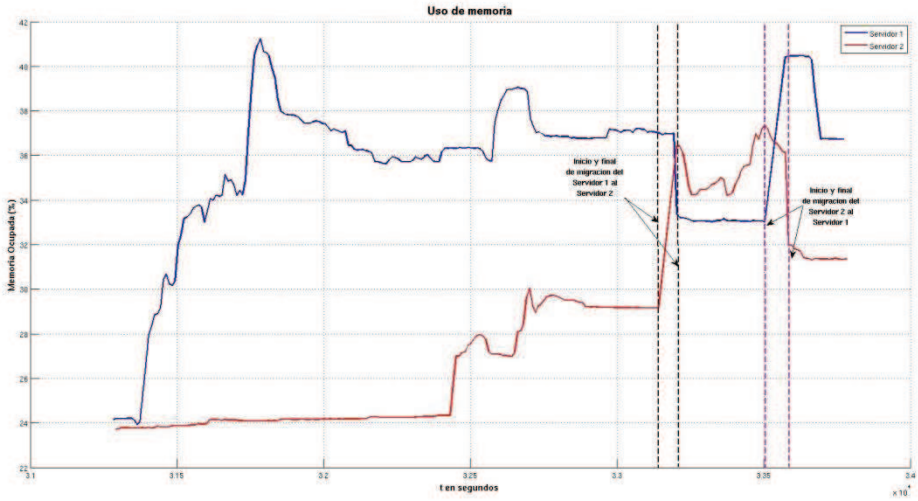


Figura 28: Memoria, Prueba 4

5.7. CASO DE PRUEBA 5

5.7.1. DESCRIPCIÓN

El número de servidores que forman el grupo es de dos. Se fija el umbral de carga al 60% y el número de mediciones consecutivas de carga mayores que el umbral para considerar que existe un problema es dos.
Inicialmente la carga de uno de los dos servidores es muy elevada, por encima del 70 % y la del otro baja, en torno al 40%. Se espera a que ambos servidores tengan su carga por debajo del 60% y se aumenta la carga de forma manual de uno de ellos.

5.7.2. RESULTADO ESPERADO

El servidor con la carga más alta debe migrar máquinas virtuales al que la tiene más baja para conseguir que ambos tengan una carga inferior al 60%. Cuando se incremente de forma manual la carga del primer servidor por encima del 60%, este intentará migrar máquinas virtuales de nuevo al primero pero no debe poder conseguir disminuir la carga por debajo del 60%.

5.7.3. RESULTADO OBTENIDO

Inicialmente el servidor 1 tiene una carga del 75% y el servidor 2 del 40%. En esta situación el servidor 1 necesita migrar máquinas virtuales al servidor 2, consiguiendo realizar dicha migración y quedar con una carga del 52% mientras que el servidor 2 queda con 58%. Esta situación puede considerarse estable.
De forma manual se aumenta la carga del servidor 1 por encima del 60%, nuevamente este intentará migrar máquinas virtuales pero teniendo el servidor 2 una carga tan cercana al 60%, el servidor 1 solo podrá mover maquinas que estén generando muy poca carga. El servidor 1 mueve una máquina virtual que está generando muy poca carga y por lo tanto no soluciona el problema pero tampoco lo crea en el servidor 2. Con esto se consigue prevenir al servidor 1 de que la máquina migrada comience a generar mucha carga y empeore la situación. Este comportamiento se observa en *Figura 29*.

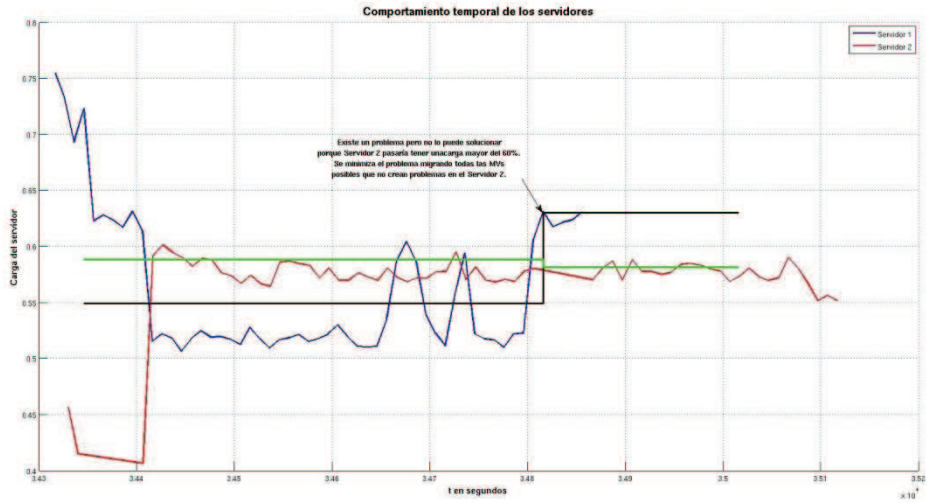


Figura 29: Carga, Prueba 5

Es interesante observar el efecto que tiene este caso de prueba sobre la memoria de los servidores del grupo. La *Figura 30* ilustra el comportamiento.

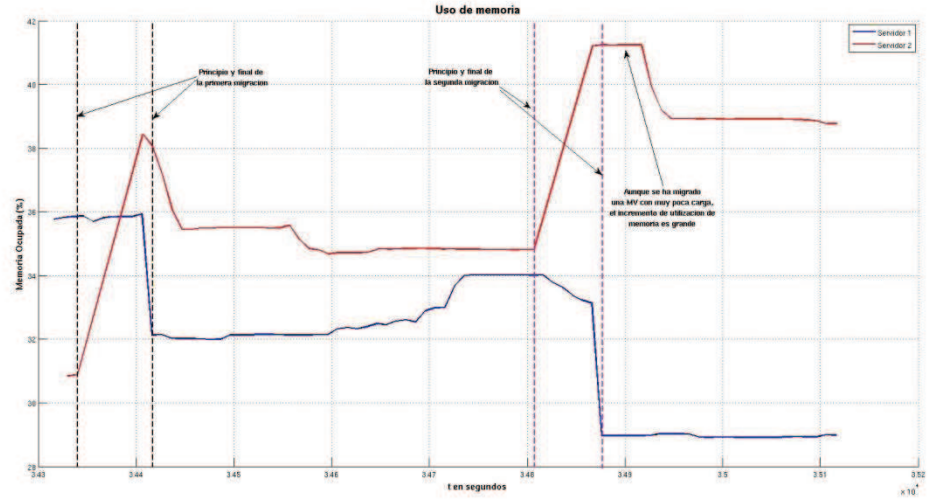


Figura 30: Memoria, Prueba 5

Como se observa y se ha comentado previamente se producen dos migraciones desde el servidor 1 al servidor 2. La segunda migración, aunque se trata de una máquina virtual que está generando muy poca carga sobre el servidor, el incremento de memoria en el destino es muy grande. Esto se debe a que en una migración se transfiere toda la memoria que tenga asignada la máquina virtual, independientemente de que esté siendo usada o no.

5.8. CASO DE PRUEBA 6

5.8.1. DESCRIPCIÓN

Este caso muestra el correcto funcionamiento de la opción implementada para terminar de ejecutar el sistema de gestión autónoma de máquinas virtuales migrando todas las máquinas virtuales que se estén ejecutando en el servidor.
El procedimiento seguido consiste en tener dos servidores formando el grupo de servidores y fijar el umbral de carga al 60%. Uno de los dos servidores tiene varias máquinas virtuales ejecutando y generando una carga alta, ~50%. El otro servidor también tiene varias máquinas virtuales activas pero apenas generan carga.
Con el sistema en esta situación, se finaliza el sistema de gestión en el primer servidor con la opción de migrar todas las máquinas virtuales activas.

5.8.2. RESULTADO ESPERADO

El sistema debe migrar todas las máquinas virtuales al servidor que permanecerá en el grupo, siempre y cuando la carga generada en este servidor no sea mayor del 60%. En caso de que esto ocurra, el sistema de gestión no finalizará hasta que sea posible migrar todas las máquinas virtuales.

5.8.3. RESULTADO OBTENIDO

La *Figura 31*, muestra el comportamiento de los dos servidores del grupo cuando se inicia la finalización del sistema de gestión en el servidor 1 migrando todas las máquinas virtuales que tiene activas.
En este caso de prueba se migran 6 máquinas virtuales desde el servidor 1 al servidor 2, realizando comprobaciones de la carga en el destino para cumplir la restricción de esta se mantenga por debajo del 60% en el servidor 2.
Una vez que el servidor 1 ha podido migrar todas las máquinas, termina el programa de gestión.

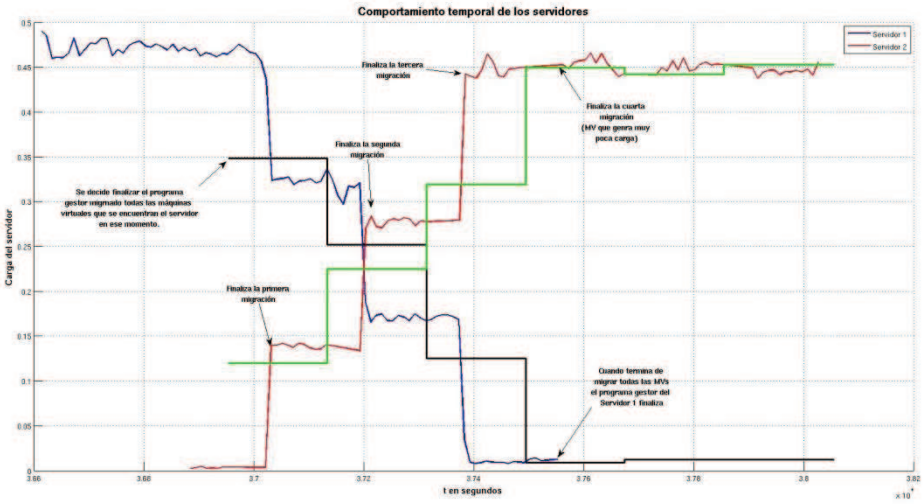


Figura 31: Carga, Prueba 6

En la *Figura 32*, se observa la gráfica de la memoria de los servidores del grupo involucrada en esta prueba. Es apreciable que el incremento de memoria en el servidor 2 es casi igual al decremento de memoria del servidor 1como era de esperar. Nuevamente es apreciable la memoria adicional que se utiliza en la migración de las máquinas virtuales en el destino.

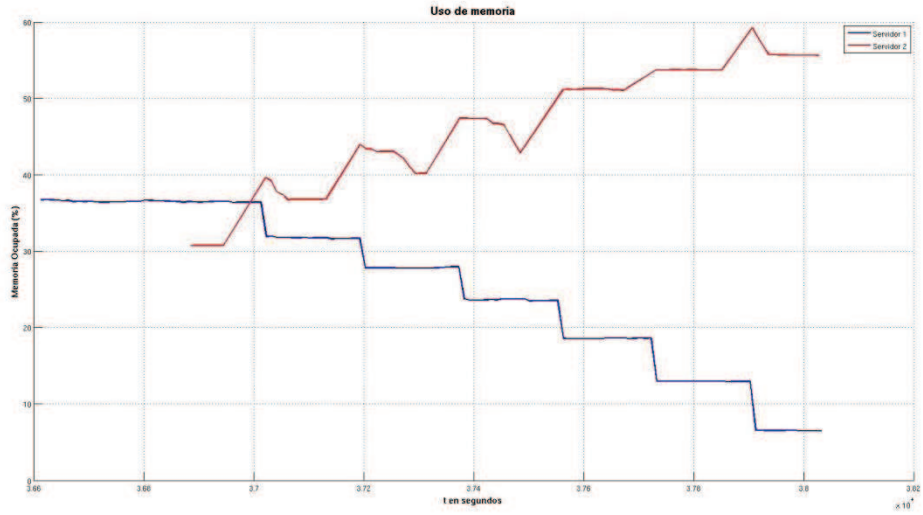


Figura 32: Memoria, Prueba 6

5.9. CASO DE PRUEBA 7

5.9.1. DESCRIPCIÓN

En este caso se pretende probar el algoritmo implementado para el tiempo de penalización impuesto a un servidor cuando este no puede migrar máquinas virtuales.
Para la realización de esta prueba se fija el umbral de carga máxima permitida en los servidores al 20% y el número de mediciones de carga consecutivas superiores a este umbral necesarias para considerar que existe un problema se fija a dos. Se ha escogido este umbral de 20% por la dificultad que entraña conseguir cargas más altas para la

realización de esta prueba. El tiempo de penalización inicial se ha fijado en 190 segundos.
El procedimiento consiste en conseguir que los dos servidores que forman el grupo tengan una carga superior al 20% y una vez conseguido iniciar el sistema de gestión autónoma de máquinas virtuales. Pasado un tiempo se disminuye la carga de uno de los servidores para que el otro pueda migrar máquinas virtuales y así estabilizar el sistema.

5.9.2. RESULTADO ESPERADO

Cuando los dos servidores del grupo se encuentren con una carga superior al 20% ambos intentarán migrar máquinas virtuales pero no podrán y entrarán en tiempo de penalización que debe incrementarse 10 segundos por cada intento fallido de migración.
Cuando la carga de uno de los dos servidores disminuya, el sistema de gestión del servidor con la carga más alta debe migrar máquinas virtuales al otro servidor del grupo para estabilizar la carga.

5.9.3. RESULTADO OBTENIDO

Según se puede observar en la *Figura 33*, los dos servidores que forman el grupo sobrepasan el umbral de carga máxima permitida fijado al 20%. En esta situación ambos servidores realizan intentos de migración de máquinas virtuales pero ninguno consigue llevarlos a cabo. El primer intento tiene una penalización de 190 segundos. Transcurridos, se intenta de nuevo pero no se consigue porque la situación no ha cambiado. El segundo intento tiene una penalización de 200 segundos, tiempo durante el cual uno de los dos servidores disminuye su carga. Transcurrido este tiempo el servidor 1 puede realizar una migración de una máquina virtual, quedando ambos servidores por debajo del umbral de carga fijado (20%).

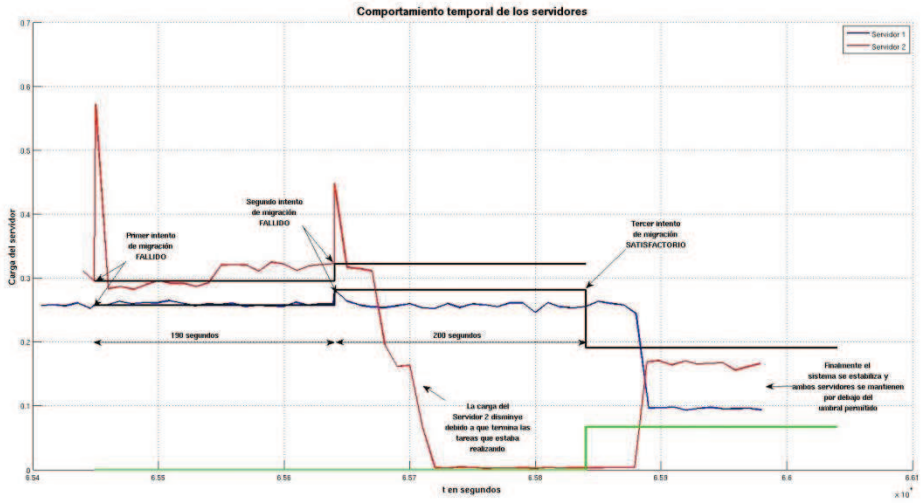


Figura 33: Carga, Prueba 7

La *Figura 34*, muestra el comportamiento que tiene la memoria de los servidores en esta prueba. En esta figura es posible apreciar que solo se ha migrado una máquina virtual.

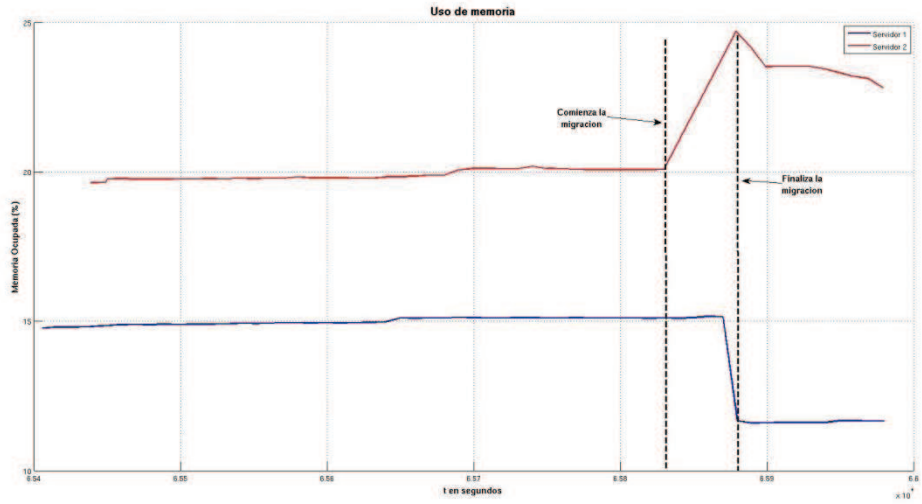


Figura 34: Memoria. Prueba 7

5.10. ERRORES COMETIDOS EN LAS PREDICCIONES

Como se ha podido observar en los casos de prueba presentados en los apartados anteriores, las predicciones de carga realizadas tanto para la carga futura del servidor de origen de la migración como para el servidor de destino de la migración, no son totalmente exactas.
Con objeto de intentar obtener algún patrón de error en estas predicciones, se han obtenido y analizado los errores cometidos en las predicciones respecto del valor de carga medido una vez terminada la migración de la máquina virtual.

El procedimiento seguido para medir el error cometido en las predicciones de carga es el introducido en el caso de prueba 1. Como se dijo, este procedimiento consiste en:

- Obtener una recta de carga por mínimos cuadrados con los valores medidos después de la migración de la máquina virtual.
- Calcular el valor absoluto de la resta de la recta anterior con la recta de carga que resulta de la predicción.
- Calcular el porcentaje que representa el error absoluto calculado respecto del valor predicho.

Estos pasos dan como resultado un error absoluto y un error porcentual, los cuales tienen mayor sentido cuanto más cerca del final de la migración son calculados, ya que, una vez terminada la migración de la máquina virtual, el servidor puede aumentar o disminuir su carga según los procesos que se estén ejecutando en él.

A continuación, en la *Figura 35* se muestran los errores obtenidos para varias predicciones en el servidor de origen de la migración de la máquina virtual.

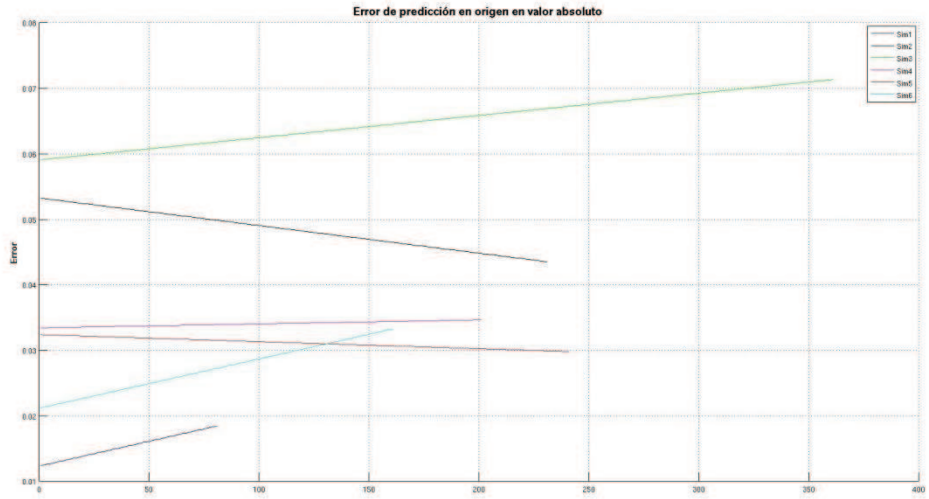


Figura 35: Error absoluto en la predicción de carga del servidor de origen de la migración

Como se observa en la figura, estos errores están entre un 1% y un 7%, teniendo presente que son errores en valor absoluto, es decir que la predicción de carga en el servidor de origen puede fallar entre ±1 y ±7 para los casos de prueba.
Estos valores demuestran que el error cometido en la predicción es aleatorio y por este motivo no es posible introducir un valor de corrección en la predicción.

La *Figura 36* muestra los porcentajes de error respecto del valor de carga calculado en la predicción. Igual que en el caso anterior, los valores obtenidos no muestran indicios de patrones de error con los que poder aplicar un factor de corrección a la predicción.

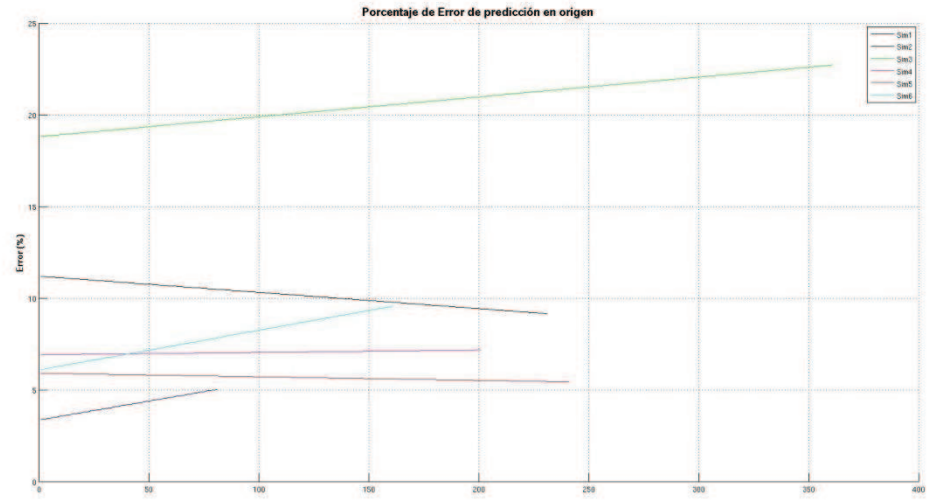


Figura 36: Porcentaje de error en la predicción de carga en el servidor de origen de la migración

Siguiendo un procedimiento análogo al anterior, se han obtenido los errores obtenidos para las predicciones de carga en el servidor de destino de la migración de las máquinas virtuales.
La *Figura 37* muestra los errores en valor absoluto cometidos en las predicciones de carga para los servidores de destino de la migración. Estos errores van desde 0.5% hasta

4% lo que quiere decir que la predicción de carga debería ser ±0.5-4 dado que se trata de valores absolutos.

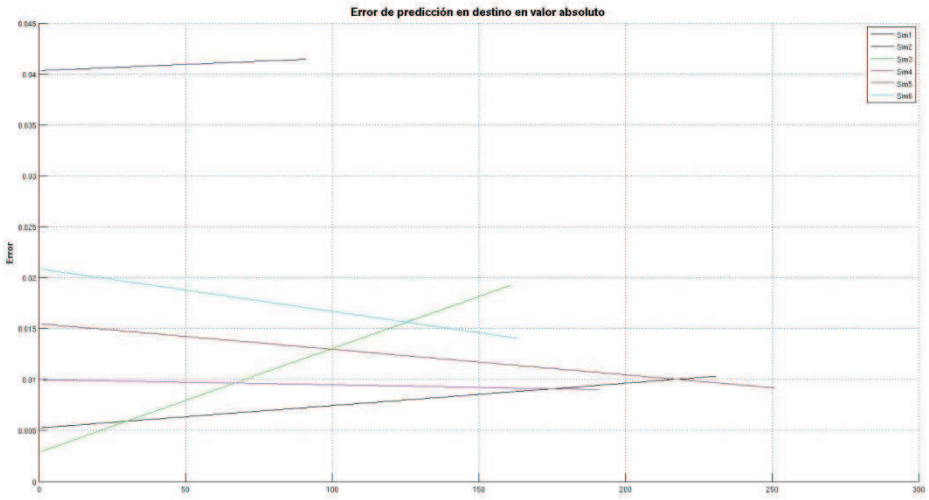


Figura 37: Error absoluto en la predicción de carga del servidor de destino de la migración

La *Figura 38* muestra los porcentajes de error respecto del valor de carga calculado en la predicción. Igual que en el caso anterior, los valores obtenidos no muestran indicios de patrones de error con los que poder aplicar un factor de corrección a la predicción.

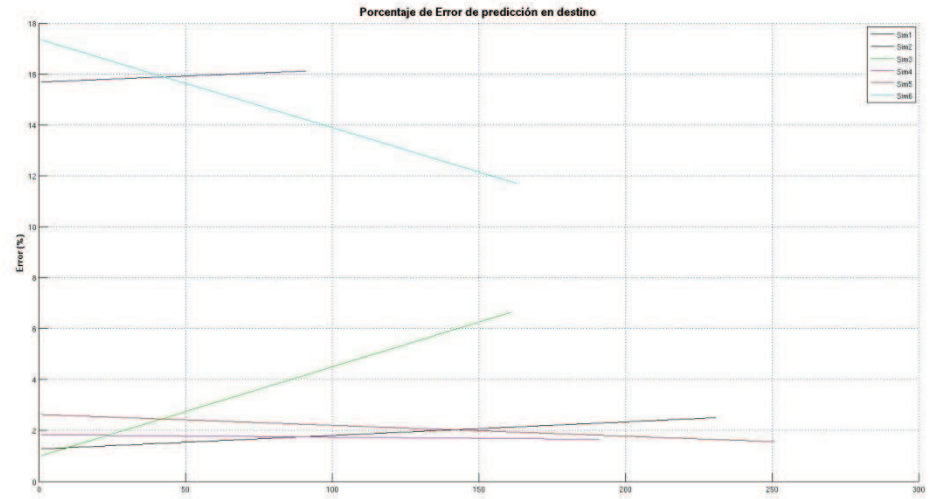


Figura 38: Porcentaje de error en la predicción de carga en el servidor de destino de la migración

5.11. MIGRACIÓN DE MÁQUINAS VIRTUALES EN CALIENTE

Como prueba adicional, se pretende introducir un pequeño estudio sobre la migración de máquinas virtuales en caliente, es decir, sin parada de servicio.

Según las especificaciones de KVM, la migración en caliente que proporciona el hipervisor produce una pequeña parada de servicio que es prácticamente inapreciable para el usuario.

Las pruebas que se van a realizar probarán esta característica de dos maneras diferentes: pérdida de servicio en la red y pérdida de servicio de procesador.

5.11.1. PÉRDIDA DE SERVICIO EN LA RED

Para realizar esta prueba se ha realizado un ping desde una máquina virtual a un servidor remoto mientras que se migraba la máquina virtual de servidor.
El resultado se muestra en el siguiente cuadro, de forma resumida, en el que se aprecia que se transmiten 171 paquetes y se reciben 127 (se pierde el 25%). Este resultado pone de manifiesto que aunque la migración se realice en vivo, habrá ciertas complicaciones en la red y es posible que se pierdan paquetes.

```
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=80 ttl=53 time=240
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=81 ttl=53 time=232
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=82 ttl=53 time=245
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=83 ttl=53 time=252
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=84 ttl=53 time=244
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=85 ttl=53 time=244
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=86 ttl=53 time=242
ms
64 bytes from 209.85.148.105: icmp_req=87 ttl=53 time=253 ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=132 ttl=53 time=33.6
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=133 ttl=53 time=29.1
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=134 ttl=53 time=29.0
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=135 ttl=53 time=29.1
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=136 ttl=53 time=28.9
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=137 ttl=53 time=28.9
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=138 ttl=53 time=28.8
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp_req=139 ttl=53 time=29.1
ms
64 bytes from fra07s07-in-f105.1e100.net (209.85.148.105): icmp req=140 ttl=53 time=36.5
--- www.l.google.com ping statistics ---
171 packets transmitted, 127 received, 25% packet loss, time 184813ms
rtt min/avg/max/mdev = 28.620/121.538/311.832/106.504 ms
```

A partir de esta prueba, en la que se enviaban mensajes de eco ICMP cada segundo, se puede extraer que ha habido una pérdida de conectividad de 45 segundos. Este resultado hace que no sea recomendable migrar máquinas virtuales que requieran un alta interactividad con los usuarios.

5.11.2. PERDIDA DE SERVICIO DE PROCESADOR

Para probar esta característica se ha realizado un programa que se ejecuta en una máquina virtual cada décima de segundo e imprime por pantalla el tiempo en milisegundos mientras se realiza una migración de dicha máquina virtual.
Parte del resultado de esta prueba está en el siguiente cuadro, donde se muestra resaltado que no se imprimen valores durante 3 décimas de segundo pero que esas 3 instrucciones no se han perdido y se ejecutan después en la misma décima, imprimiendo 4 valores correspondientes a la misma décima de segundo.

Este resultado muestra que, a diferencia de la pérdida de servicio en la red, nunca se pierden instrucciones pero sí que se puede obtener un comportamiento ligeramente diferente al deseado durante una pequeña fracción del tiempo que dura la migración de la máquina virtual.

```
1311011308160
1311011308260
1311011308360
1311011308460
1311011308560
1311011308660
1311011308760
1311011308860
1311011308960
1311011309060
1311011309405
1311011309406
1311011309406
1311011309460
1311011309560
1311011309660
1311011309760
1311011309860
1311011309960
1311011310060
1311011310160
1311011310260
1311011310360
1311011310460
```

5.12. CONCLUSIONES

Este capítulo ha presentado las pruebas realizadas sobre el diseño implementado. Estas pruebas están encaminadas a probar todas las funcionalidades implementadas que dotan de capacidad de decisión al sistema de gestión autónoma de máquinas virtuales. A continuación se resumen las ideas principales deducidas a partir de estas pruebas.

- A la vista de las gráficas obtenidas en los casos de prueba, queda demostrado que la carga de los servidores es sustancialmente más relevante que la memoria de los mismos, ya que la memoria de las máquinas virtuales se reserva aunque no se use y por lo tanto un mayor uso de la memoria por parte de la máquina virtual no se verá reflejado en el servidor.
- Aunque la predicción de la carga en los servidores no es todo lo precisa que se desearía, los algoritmos de decisión consiguen resolver los problemas de carga de forma efectiva ya que si se ha fallado en la predicción, las nuevas mediciones consiguen activar de nuevo los mecanismos de migración de máquinas virtuales.
- Como se ha explicado, cuando un servidor tiene una carga muy elevada es menos preciso realizar una predicción de la carga que tendrá cuando se migre una máquina virtual que cuando el servidor tiene una carga menos elevada.
- El estudio realizado sobre los errores en la predicción de carga no exhibe a priori patrones que hagan posible introducir parámetros de corrección de dichos errores, si bien se podrían establecer márgenes de confianza en las predicciones.
- Para realizar una migración de una máquina virtual se necesita más memoria que la que contiene la propia máquina virtual, y pasado un tiempo esa memoria adicional se libera.

Gracias a estos resultados es posible realizar una valoración crítica del trabajo realizado y extraer los puntos en los que hay capacidad de mejora, así como ampliación de la funcionalidad del trabajo. El siguiente capítulo expondrá las conclusiones obtenidas en este Proyecto Final de Carrera y los puntos que quedan por realizar o que es posible mejorar.

6. CONCLUSIONES Y TRABAJO FUTURO

6.1. INTRODUCCIÓN

Una vez realizadas las pruebas sobre el diseño, es necesario interpretar los resultados obtenidos con ellas y cómo se diferencian de los resultados que se esperaban cuando se realizó el diseño.
Adicionalmente, el sistema de gestión autónoma de máquinas virtuales implementado, tiene cierta capacidad de mejora en algunos aspectos, así como la necesidad de probar algunas de sus funciones desarrolladas que no han podido ser probadas por las limitaciones materiales.

6.2. CONCLUSIONES

Las decisiones para la migración de máquinas virtuales se han basado principalmente en la carga de procesador que experimenta el servidor, ya que solamente se tomará la decisión de migrar si el servidor tiene una carga que rebase cierto umbral fijado anteriormente. Esta decisión podría haberse basado en la cantidad de memoria ocupada por el servidor, pero a la vista de las pruebas realizadas la memoria del servidor no varía de forma significativa en periodos cortos de tiempo, ya que las grandes variaciones en memoria se producen cuando una máquina virtual se arranca en el servidor y ésta pasa ocupar toda la memoria que tiene asignada, independientemente de que se esté utilizando o no. Por este motivo, se ha llegado a la conclusión de que no es necesario mantener una monitorización de la memoria del servidor tan estricta como la de la carga. Sin embargo, las pruebas realizadas también muestran que cuando se realiza una migración de una máquina virtual toda la memoria de dicha máquina es transferida al servidor de destino. Esta situación hace que sea recomendable conocer si el servidor de destino tiene libre la cantidad de memoria requerida para albergar la nueva máquina virtual, ya que en caso contrario habrá pérdida de rendimiento causada por el trasiego de páginas de memoria a disco. Por estos motivos, la conclusión a la que se ha llegado es que será **necesario mantener monitorizaciones tanto para la memoria como para el uso de procesador del servidor y que la decisión de iniciar el algoritmo de migración deberá estar basada en ambos parámetros.**
Para esta situación se darán unas recomendaciones a seguir en el apartado *6.3 Trabajo futuro* de este mismo capítulo.

Uno de los pilares fundamentales del sistema de gestión autónoma de máquinas virtuales implementado es la predicción de la carga, tanto en el servidor de origen como de destino, realizada previamente a la migración de las máquinas virtuales.
Las gráficas obtenidas en el capítulo anterior, muestran que la predicción de carga no es totalmente precisa ni en el origen ni en el destino de la migración. Lo deseable sería que se cumpliera la predicción de carga en todos los casos ya que en ciertas ocasiones una pequeña desviación en la predicción podría causar problemas de rendimiento en el sistema de gestión autónoma de máquinas virtuales. Afortunadamente, la probabilidad de estar en una de estas situaciones es muy pequeña y en caso de que ocurran, existen otros mecanismos de medición en el sistema que solucionarían el problema.
Analizando los errores cometidos en las predicciones en los casos de prueba realizados, se obtienen **valores que son bastante pequeños y por lo tanto asumibles y válidos**

para considerar dichas predicciones correctas, tanto en origen como en destino, siendo algo menores en el destino.
No obstante, mediante el análisis de los errores cometidos en las predicciones se ha realizado un estudio con objeto de reducir dicho error, para ello se ha pretendido introducir un factor de corrección que compensase el error cometido pero no ha sido posible encontrar patrones de error, motivo por el que no se ha podido introducir dicho factor de corrección.

El otro pilar fundamental del sistema de gestión autónoma de máquinas virtuales es el algoritmo de decisión utilizado para asignar las máquinas virtuales de un servidor con la carga alta a otros servidores del grupo de servidores sin aumentar la carga de estos últimos a valores no permitidos**. El algoritmo implementado, actúa de forma rápida ya que no necesita comprobar todas las posibilidades de colocación de máquinas virtuales para encontrar la solución óptima.**
Otras soluciones consideradas sí necesitaban realizar un alto número de comprobaciones, lo que podía introducir un retardo muy alto para números elevados de servidores en el grupo con varias máquinas virtuales cada uno. Una de estas soluciones consistía en encontrar la mejor situación posible migrando máquinas virtuales desde el servidor con el problema a un único servidor del grupo de servidores. Es decir, no se podían migrar máquinas virtuales a varios servidores diferentes. Para este algoritmo se necesitaban realizar el número de comprobaciones indicado en el siguiente cuadro, siendo *n* el número de máquinas virtuales en el servidor con el problema y *m* el número de servidores del grupo.

$$número\ de\ comprobaciones = (m-1)\frac{n(n-1)+2(n-1)}{2}+2$$

Este algoritmo implica que se realicen todas las comprobaciones, es decir, debe ser ejecutado hasta el final.
Un ejemplo típico con n=10 y m=5 implica realizar 280 comprobaciones para decidir que máquinas virtuales migrar a que servidor. Es importante recordar que todas las máquinas virtuales se tienen que migrar a un único servidor, lo que implica que si se modifica el algoritmo para poder migrar máquinas a varios servidores el número de comprobaciones aumentaría considerablemente.
Aplicando el mismo ejemplo al algoritmo de decisión utilizado en el sistema de gestión autónoma de máquinas virtuales, con 10 máquinas virtuales en el servidor que tiene el problema y 5 servidores en el grupo, será necesario realizar un máximo de 40 comprobaciones, no siendo necesario en la mayoría de los casos llegar a realizar todas las comprobaciones, porque con una alta probabilidad se resolverá antes el problema. El motivo por el que el número de comprobaciones es siete veces menor se debe a que previamente se realiza una ordenación de los servidores del grupo y de las máquinas virtuales, descartando así muchos casos que no es necesario comprobar porque con toda seguridad serán peores que otros ya comprobados. Además, con este algoritmo es posible distribuir la carga entre los servidores del grupo migrando máquinas a todos ellos y no solo a uno.
El siguiente cuadro muestra el número máximo de comprobaciones que hay que realizar en el algoritmo implementado.

$$\boldsymbol{número\ de\ comprobaciones \leq n(m-1)}$$

6.3. TRABAJO FUTURO

El sistema implementado tiene puntos en los que es posible mejorar, así como nuevas funcionalidades que es posible añadir para aumentar la eficiencia del mismo.
Este apartado describirá estos puntos, proponiendo una solución cuando sea posible.

Como se ha mencionado en el apartado de conclusiones de este mismo capítulo, una mejora importante sobre el sistema implementado sería tomar decisiones de migración de máquinas virtuales basándose en la carga y en la memoria del servidor. Para ello, una ocupación excesiva de la memoria del servidor podría al igual que se hace con la carga, arrancar el algoritmo de decisión de migración de máquinas virtuales. Una vez arrancado dicho algoritmo, sería necesario predecir cómo quedará la memoria del servidor de destino una vez terminada la migración de la máquina virtual. Esto se haría de forma análoga a como se hace actualmente con la carga del servidor. La meta de este algoritmo sería intentar mantener tanto la carga del servidor como la memoria por debajo de unos umbrales que serían configurables por el administrador.
Como ampliación y siguiendo con el mismo concepto, sería posible mantener la utilización de la interfaz de red del servidor por debajo de otro umbral.

Otro de los puntos a mejorar es la predicción que se realiza actualmente de la carga futura. Aunque dicha predicción es lo suficientemente acertada, sería interesante reducir el error cometido actualmente, consiguiendo así aumentar la fiabilidad del algoritmo de decisión. Para lograrlo podría ser necesario estudiar el planificador del procesador que se esté utilizando y realizar los cálculos en función de dicho planificador.
Como ampliación de lo anterior y aunque se ha intentado de forma insatisfactoria en este Proyecto Final de Carrera, se debería poder utilizar el sistema de gestión autónoma de máquinas virtuales con servidores con diferente procesador. Este punto no ha sido llevado a cabo debido la imposibilidad de predecir la carga que tendría el servidor de destino de la migración si éste utiliza un procesador diferente al de origen.

Mientras que no sea posible afinar la predicción de carga, sería posible utilizar márgenes de seguridad, tanto en el servidor de origen como en el de destino, que posibilitaran evitar que la carga predicha más el error esté por encima del umbral. Esto tiene como inconveniente que se realiza para todas las predicciones y podría llegar a acumularse un error muy grande. Lo ideal sería aplicarlo solo a aquellos casos en los que la carga predicha esté muy cerca de la carga máxima permitida.

Desde el punto de vista del rendimiento de las máquinas virtuales, sería interesante implementar un mecanismo que impida mover una máquina virtual si ésta ha sido migrada hace poco tiempo, al no ser que sea la única solución a un problema de carga en el servidor.

Por limitaciones en los recursos físicos utilizados no ha sido posible probar la totalidad del funcionamiento del algoritmo de decisión en un grupo formado por más de dos servidores. Por el contrario el intercambio de mensajes si ha podido ser probado con tres servidores.

Así mismo, se pueden realizar pruebas del sistema completo sobre otros sistemas operativos diferentes a Ubuntu Server.

Por último, sería interesante evitar la limitación de uso de la interfaz por defecto, como conexión de red de las máquinas virtuales. La idea sería ser capaz de monitorizar la interfaz de red del servidor que esté usando cada máquina virtual.

6.4. RESUMEN

A continuación se muestran tablas con las ideas principales del apartado de conclusiones y de trabajo futuro.

CONCLUSIONES
Mediante el sistema de gestión autónoma de máquinas virtuales es posible distribuir de forma óptima la carga que generan éstas en un grupo de servidores.
Los errores cometidos en la predicción son lo suficientemente pequeños como para asumir los valores de carga obtenidos en las predicciones.
El algoritmo de decisión implementado es más rápido que otras soluciones ya que no necesita realizar tantas comprobaciones.

Tabla 7: Conclusiones

TRABAJO FUTURO
La decisión de iniciar el algoritmo de migración de máquinas virtuales debe basarse tanto en el uso de procesador como en la ocupación de memoria del servidor.
Mejorar la precisión de la predicción de la carga del servidor de destino y origen.
El algoritmo de decisión deberá ser capaz de decidir entre servidores con procesador diferente.
Utilizar márgenes de seguridad que eviten predecir cargas menores al umbral de carga fijado y que la carga real esté por encima de dicho umbral.
Implementar un mecanismo que impida, siempre que no haya una solución alternativa, mover una máquina virtual si ésta ha sido migrada hace poco tiempo.
Probar el algoritmo de decisión en un grupo de servidores formado por más de dos servidores.
Probar el sistema completo sobre sistemas operativos diferentes a Ubuntu Server.
Monitorizar interfaces de red utilizadas por las máquinas virtuales diferentes a la interfaz por defecto.

Tabla 8: Trabajo futuro

7. BIBLIOGRAFÍA

[1]Luis Joyanes Aguilar, **"La Computación en Nube (Cloud Computing): El nuevo paradigma tecnológico para empresas y organizaciones en la Sociedad del Conocimiento"**, ICADE, Revista cuatrimestral de las facultades de Derecho y Ciencias Económicas y Empresariales, nº 76, enero-abril 2009, ISSN: 02 12-7377

[2]Kathryn M. Jones, Esteban González, **"Secretos del VM: Virtualización y drivers"**, Universidad de Costa Rica, Departamento de Ingeniería, San José, Costa Rica, 2008.

[3]Jorge E. López de Vergara, **"Especificación de modelos de información de gestión de red integrada mediante el uso de ontologías y técnicas de representación del conocimiento"**, Tesis Doctoral, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, marzo 2003.

[4]Nancy Samaan, Ahmed Karmouch**, "Towards Autonomic Network Management: an Analysis of Current and Future Research Directions"**, IEEE Communications Surveys & Tutorials, VOL. 11, NO 3, third quarter 2009.

[5]libvirt: The virtualization API, www.libvirt.org, Última visita Febrero 2011.

[6]Jorge E. López de Vergara, Víctor A. Villagrá, Juan I. Asensio, Julio Berrocal, **"Ontologies: Giving Semantics to Network Management Models"**, IEEE Network, special issue on Network Management, Vol. 17, No. 3, May/June 2003. ISSN 0890-8044.

[7]Anatomía de la biblioteca de virtualización libvirt: http://www.ibm.com/developerworks/ssa/linux/library/l-libvirt/

[8]Eaton Corp. www.eaton.com. Última visita Marzo 2011.

[9]VMware, http://www.vmware.com/. Última visita Marzo, 2011.

[10]Guohui Wang, T.S Eugene Ng, **"The Impact of Virtualization on Network Performance of Amazon EC2 Data Center"**, INFOCOM, 2010 Proceedings IEEE, marzo 2010.

[11]KVM: Kernel Based Virtual Machine, http://www.linux-kvm.org/page/Main_Page. Última visita Marzo 2011.

[12]Xen: The Xen virtual machine monitor, http://www.xen.org/files/Marketing/WhatisXen.pdf. Última visita Marzo 2011.

[13]VirtualBox, http://www.virtualbox.org/. Última visita Marzo 2011.

[14]Hyper-V, http://www.microsoft.com/hyper-v-server/en/us/por defecto.aspx

[15]VMotion, http://www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf. Última visita MArzo 2011.

[16]Sara Bouchenak, Noël De Palma, Daniel Hagimont, **"Autonomic Management of Clustered Applications"**, IEEE International Conference on Cluster Computing, Barcelona, 2006, ISBN: 1-4244-0327-8.

[17]Enrique J. García, **"Análisis y Desarrollo de Conceptos Fundamentales de Redes Autónomas"**, Proyecto de Fin de Máster, Universidad Autónoma de Madrid, Noviembre de 2008.

[18]Norman Bobroff, Andrzej Kochut, Kirk Beaty, **"Dynamic Placement of Virtual Machines for Managing SLA Violations"**, Integrated Network Management, 2007. IM 2007 10th IFIP/IEEE International Symposium, ISBN: 1-4244-0798-2.

[19]Zhikui Wang, Xiaoyun Zhu, Pradeep Padala, Sharad Singhal, **"Capacity and Performance Overhead in Dynamic Resource Allocation to Virtual**

Containers”, Integrated Network Management, 2007. IM 2007 10th IFIP/IEEE International Symposium, ISBN: 1-4244-0798-2.

[20]Sigar, http://www.hyperic.com/products/sigar. Última visita Marzo 2011.

[21]xmlBlaster, http://www.xmlblaster.org/. Última visita Marzo 2011.

[22]Virt-manager, http://virt-manager.et.redhat.com/. Última visita Abril 2011.

8. GLOSARIO

API	Application Programming Interface
CHOP	Configuration, Healing, Optimization, Protection
CPU	Central Processing Unit
DNS	Domain Name Server
DRS	Distributed Resource Scheduler
EPS	Escuela Politécnica Superior
FQDN	Fully Qualified Domain Name
GPL	General Public License
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IT	Information Technology
KVM	Kernel Based Virtual Machine
MAC	Media Access Control
MAPE	Monitorización-Análisis-Planificación-Ejecución
MFR	Measure-Forecast-Remap
MOM	Message Oriented Middleware
MV	Máquina Virtual
NAS	Network Attached Storage
NFS	Network File System
RAM	Random Access Memory
SSH	Secure Shell
TCP	Transmission Control Protocol
XML	Extensible Markup Language

9. ANEXO A: CONFIGURACIÓN DE SERVIDORES

Este anexo describe los pasos que hay que realizar para configurar los servidores que vayan a formar parte del grupo.

9.1. CREACIÓN Y CONFIGURACIÓN DEL USUARIO KVMMAN

El usuario que gestionará las máquinas virtuales será *kvmman*. Este usuario no deberá pertenecer al grupo admin ni sudo. Para crearlo se siguen los siguientes pasos.

```
# useradd -d /home/kvmman -m kvmman
# passwd kvmman
```

Editar el archivo /etc/passwd de la siguiente manera:

```
# vi /etc/passwd
```

Editar la línea *kvmman:x:1003:1003::/home/kvmman:* para que quede de la siguiente manera: *kvmman:x:1003:1003::/home/kvmman:/bin/bash*

Será necesario que el usuario *kvmman* pueda ejecutar la herramienta ethtool sin necesidad de introducir contraseña, para ello se modifica el archivo /etc/sudoers

```
# vi /etc/sudoers
```

El archivo /etc/sudoers debe incluir la siguiente línea:

```
kvmman ALL=NOPASSWD:/usr/sbin/ethtool
```

9.2. INSTALACIÓN DE LIBVIRT

Se deben instalar los siguientes paquetes en Ubuntu Server: *kvm* y *libvirt-bin*.

```
# apt-get install kvm libvirt-bin
```

Después de la instalación de los paquetes es necesario añadir el usuario kvmman, usado para gestionar las máquinas virtuales, al grupo *libvirtd.*

```
# adduser kvmman libvirtd
```

9.3. CONFIGURACIÓN DE LAS CONEXIONES SSH

Para un correcto funcionamiento del sistema de gestión autónoma de máquinas virtuales, los servidores que forman el grupo deben poder realizar conexiones ssh entre ellos desde el usuario *kvmman* al usuario *kvmman* sin necesidad de introducir la contraseña de dicho usuario.

Para conseguir esta configuración, ejecutar en cada servidor del grupo la siguiente línea como usuario *kvmman*, dejando todas las opciones por defecto:

```
kvmman@server:~$ ssh-keygen -t rsa
```

Una vez ejecutada la línea anterior, ejecutar la siguiente línea en todos los servidores para cada servidor del grupo, sustituyendo servername por el nombre de cada uno de los servidores del grupo, excepto desde el que se está ejecutando en cada momento.

```
kvmman@server:~$ ssh-copy-id -i ~/.ssh/id_rsa.pub kvmman@servername
```

9.4. CONFIGURACIÓN DE ALMACENAMIENTO COMPARTIDO

Para poder realizar migraciones de máquinas virtuales entre servidores, es necesario que todos los servidores involucrados en la migración tengan acceso a la imagen de la máquina virtual que se va a migrar.
Para ello se debe configurar un almacenamiento compartido donde se almacenen las imágenes de todas las máquinas virtuales y dicho almacenamiento debe ser accesible por todos los servidores del grupo.

Se propone utilizar un almacenamiento compartido NFS cuya configuración se muestra a continuación.

En el servidor que se va a utilizar como almacenamiento de todas las imágenes, instalar el servidor NFS de la siguiente manera:

```
# apt-get install nfs-kernel-server
```

A continuación se necesita configurar el archivo que indica el directorio que va a ser exportado y que otros servidores podrán acceder a dicho directorio. El archivo a editar es /etc/exports.

```
# vi /etc/exports
```

Se debe incluir el siguiente contenido en /etc/exports donde Directorio_a_exportar es la ruta del directorio que contiene las imágenes de las máquinas virtuales y hostname_servidor es el FQDN del servidor que tendrá acceso al almacenamiento compartido.

```
Directorio_a_exportar *hostname_servidor(rw,sync,no_root_squash,no_subtree_check)
```

Se debe añadir en /etc/exports una línea por cada servidor que forma el grupo de servidores. Una vez editado, hay que arrancar el servicio NFS.

```
# service nfs-kernel-server start
```

En los servidores del grupo de servidores, que actuarán como clientes del servidor NFS habrá que instalar el cliente NFS:

```
# apt-get install nfs-common
```

Una vez instalado es necesario montar el directorio del servidor NFS para que el servidor del grupo pueda tener acceso a las imágenes de las máquinas virtuales.

```
# mount hostname_servidorNFS:Directorio_a_exportar punto_de_montaje
```

9.5. LIBRERÍAS DE SIGAR

Para un correcto funcionamiento de la API Sigar, es necesario incluir ciertas librerías en el servidor sobre el que se ejecutará el sistema de gestión autónoma de máquinas virtuales.

Dependiendo del sistema operativo utilizado y del procesador del servidor, será necesario incluir una librería u otra.

Para identificar el procesador y el sistema operativo se debe ejecutar en el servidor:

```
$ uname -a
```

Este comando tendrá una salida similar a esta:

```
Linux mami 2.6.32-28-server #55-Ubuntu SMP Mon Jan 10 23:57:16 UTC 2011 x86_64 GNU/Linux
```

Esta salida informa de que se trata de un servidor con sistema operativo Linux y con un procesador x86_64.

Para descargar y ver todas las librerías de Sigar ejecutar:

```
$ wget http://sourceforge.net/projects/sigar/files/sigar/1.6/hyperic-sigar-1.6.4.tar.gz/
$ tar -xzvf hyperic-sigar-1.6.4.tar.gz
$ cd hyperic-sigar-1.6.4/sigar-bin/lib
```

Identificar la librería necesaria según sistema operativo y procesador y copiarla en el directorio del servidor /lib como root.

```
# cp libsigar-amd64-linux.so /lib
```

NOTA: El cuadro anterior muestra la librería válida para la información obtenida con úname –a del ejemplo anterior.

9.6. SERVIDOR XMLBLASTER

En el servidor que se desee dedicar para que realice la transferencia de mensajes entre los servidores del grupo se deben realizar los siguientes pasos.

El directorio de descarga del paquete queda a elección del administrador del servidor, pero deberá ser recordado para el arranque del servicio xmlBlaster.

```
$ wget http://www.xmlblaster.org/xmlBlaster_REL_2_0_0.tgz
$ tar -xzvf xmlBlaster_REL_2_0_0.tgz
```

El servidor xmlBlaster puede ser un servidor del grupo de servidores o un servidor dedicado. Además el sistema operativo puede ser cualquier Linux.

10. ANEXO B: EJECUCIÓN Y PARADA

10.1. EJECUCIÓN Y PARADA DE XMLBLASTER

Para arrancar el servidor xmlBlaster es necesario entrar en el servidor en el que se descargó el paquete xmlBlaster y entrar en la carpeta xmlBlaster que resulta de la descomprimir el paquete descargado. Una vez hecho esto ejecutar:

```
$ Java -jar lib/xmlBlaster.jar
```

Con esto el servidor xmlBlaster ya estará ejecutando.

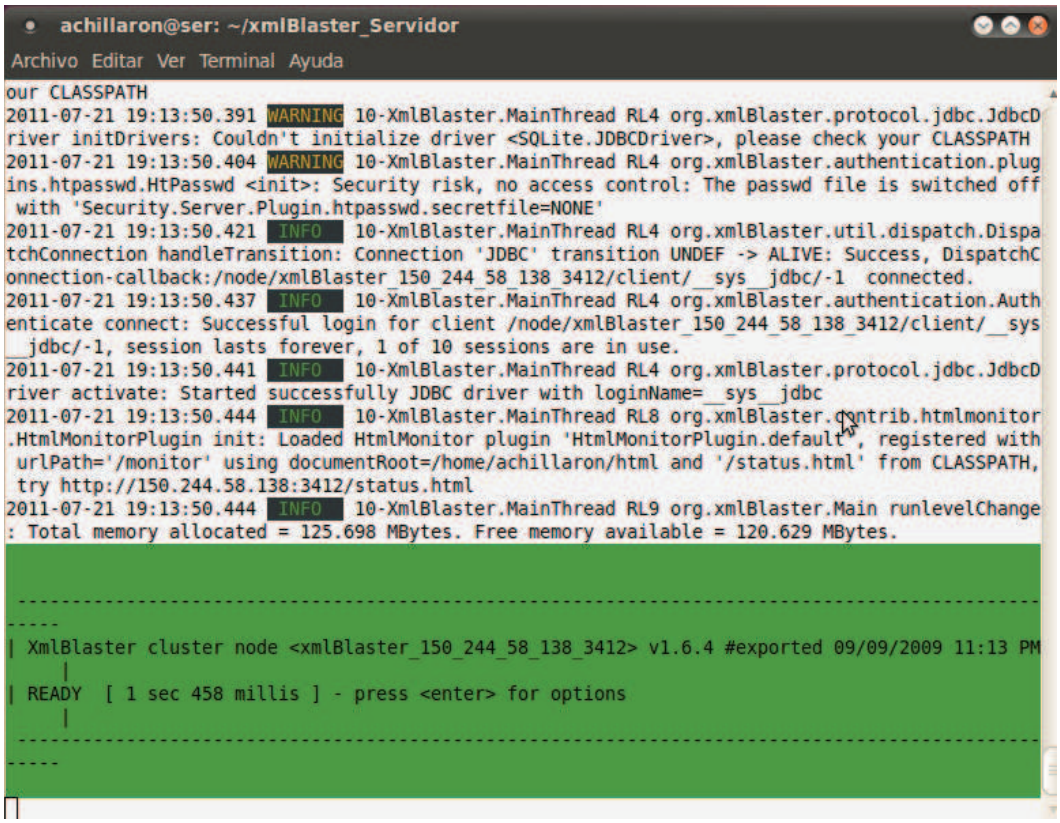


Figura 39: Arrancar servidor xmlBlaster

Para parar la ejecución del servidor xmlBlaster basta con teclear “q” y presionar Intro.

10.2. EJECUCIÓN Y PARADA DEL SISTEMA DE GESTIÓN AUTÓNOMA DE MÁQUINAS VIRTUALES

Copiar el directorio con el sistema de gestión autónoma de máquinas virtuales en la ruta del servidor que se desee. Entrar en el directorio.

Para arrancar el sistema ejecutar:

```
$ Java -classpath lib/xmlBlaster.jar ClaseMain -
dispatch/connection/plugin/socket/hostname nombre_servidor_xmlBlaster
```

Donde *nombre_servidor_xmlBlaster* es el FQDN del servidor xmlBlaster.

Para finalizar el sistema de gestión autónoma de máquinas virtuales presionar Intro y escoger entre una de las opciones presentadas, el siguiente cuadro muestra el mensaje mostrado por el sistema antes de salir del mismo.

```
Puede terminar el programa migrando todas las MVs (1)
o puede terminar sin migrar las MV (2)
Elija como terminar:
```

11. ANEXO C: CLASES Y MÉTODOS IMPLEMENTADOS

El diseño del sistema se ha implementado en código Java. Para ello se han creado varias clases junto con sus métodos asociados. Este anexo muestra un breve resumen de las clases y métodos implementados. Estas clases a su vez hacen uso de las clases de libvirt, Sigar y xmlBlaster, las cuales no se describirán.

	Nombre	Descripción
Clase	*ClaseMain*	Clase desde la que se realiza todo el control del intercambio de mensajes con xmlBlaster. También implementa la mayor parte del algoritmo de decisión.
Métodos	getListaPropServidores	Devuelve un ArrayList con las propiedades recibidas del grupo de servidores.
	getNumeroServidores	Devuelve el número de servidores que hay en el grupo en ese momento.
	getHostName	Devuelve el nombre del servidor.
	pedirInformacionAServidores	Envía un mensaje solicitando el envío de las propiedades a un servidor del grupo. Pone el servidor en modo solucionando problema.
	publicarMVmigrada	Envía un mensaje informando de que se ha terminado de migrar máquinas virtuales.
	suscribirMVmigrada_ACK	Hace que el servidor con el problema se suscriba a los mensajes de asentimiento de información de fin de migración.
	desuscribirMVmigrada_ACK	Deja de estar suscrito a los asentimientos de migración.
	suscribirPedirInformacion_ACK	Se suscribe el servidor a los mensajes de recepción de la información enviada por el resto de servidores del grupo en respuesta a la solicitud realizada por pedirInformacionAServidores.
	desuscribirPedirInformacion_ACK	El servidor deja de estar suscrito al envío de información de servidores.

	convertirServerPropertiesAByteArray	Convierte un objeto Java a byte[]. Convierte el objeto donde se almacenan las propiedades del servidor para enviarlo en un mensaje xmlBlaster.
	convertirBytesAServerProperties	Convierte un byte[] a un objeto Java. Convierte los bytes recibidos en un mensaje xmlBlaster al objeto donde se almacenan las propiedades del servidor.
	borrarListaPropServidores	Vacía las propiedades recibidas.
	ordenarServidores	Ordena los servidores del grupo por puntuación, de mayor a menor.
	reordenarPropiedadesEstimadas	Reordena los servidores por la nueva puntuación después de asignar una nueva máquina virtual.
	comprobarCondicionCarga	Se decide si asignar una máquina virtual a un servidor, comprobando que la carga en el destino no sobrepasará el umbral de carga.
	comprobarCargaFutura	Se comprueba si se ha conseguido reducir la carga del servidor de origen lo suficiente como para que desaparezca el problema.
	decidir	Llama a los métodos de decisión y comprobación de condiciones en el algoritmo de asignación de máquinas virtuales.
	cerrarFicheros	Cierra ficheros de logs
	invertir	Invierte la lista de máquinas virtuales ordenadas por puntuación de menor a mayor.
	reubiarMVs	Ordena las máquinas virtuales de menor a mayor puntuación.
	decidirServidorDestino	Inicia el algoritmo de decisión.
	update	Escucha los mensajes de xmlBlaster y realiza acciones en función del mensaje recibido.

Tabla 9: CalseMain

	Nombre	**Descripción**
Clase	*InfoCPU*	Contiene métodos dedicados a monitorizar la carga del

		servidor.
Métodos	imprimirInfoCPU	Muestra por pantalla las características de la CPU.
	mvHipervisor	Crea tantos controladores de máquina virtual como máquinas virtuales haya en el servidor e inicia la monitorización de dichas MVs.
	shutdownControladores	Para la monitorización de las máquinas virtuales.
	cargaCPU	Obtiene la carga del servidor mediante Sigar y controla el tiempo de penalización cuando no se resuelve el problema.
	monitorizarCarga	Se monitoriza la carga del servidor, analizando si los valores de carga medidos están por encima del umbral de carga, en cuyo caso aumentan el contador y se lanza una petición de información a servidores.
	solicitarMigracion	Lanza el algoritmo de decisión y gestiona el resultado del mismo. Una vez se terminan de migrar la máquinas informa al resto de servidores de que ha terminado.
	conectarHipervisorRemoto	Se crea un objeto de conexión del servidor de destino para realizar la migración.
	migrar	Migra las máquinas virtuales. Actualiza la información de máquinas migradas y pertenecientes.
	calcularNumeroDeCpus	Obtiene el número de procesadores del servidor mediante Sigar.
	run	Método que se ejecuta cada un cierto intervalo de tiempo fijo y llama a los métodos de monitorización.
	registrarCarga	Escribe el fichero de log de carga del servidor.
	cerrarFicheros	Cierra los ficheros de log.
	actualizarCargaCPU	Crea el objeto executor que ejecutará el método run.
	shutdown	Finaliza la ejecución del método run.
	actualizarProperties	Guarda en el fichero properties la carga del servidor.

	sacarProperty	Saca del fichero properties la carga del servidor.

Tabla 10: InfoCPU

	Nombre	Descripción
Clase	*Gestion*	Clase que contiene métodos que sirven de intercambio de información entre las clases del sistema. Esta clase se encarga de arrancar la mayoría de monitorizaciones del sistema.
Métodos	run	Método que se ejecuta cada un cierto intervalo de tiempo fijo y monitoriza el numero de máquinas virtuales activas e inactivas.
	xmlDefinido	Se añaden a una lista las máquinas virtuales que han sido definidas en el servidor.
	generalManager	Inicia la monitorización de las máquinas virtuales activas e inactivas, la carga del servidor y el uso de memoria. Crea la información con las máquinas virtuales definidas en el servidor.
	getState	Devuelve el estado en el que se encuentra el gestor, ejecutando o no ejecutando.
	pararGestor	Para la monitorización de máquinas virtuales activas e inactivas, carga de servidor y uso de memoria.
	cerrarFicheros	Da la orden de cerrar los ficheros de log.
	actualizarProperties	Actualiza el fichero de properties.
	sacarProperty	Obtiene información del fichero de properties.
	servidorDestino	Llama al método decidirServidorDestino de ClaseMain.
	propServidoresPreparadas	Llama al método solicitarMigracion de la clase InfoCpu
	ActualizarMV	Para la monitorización de las máquinas virtuales y hace una actualización de las máquinas virtuales del servidor.
	problemaSolucionado	Actualiza la información sobre

		si se está solucionando el problema del servidor o si se ha conseguido solucionar si ha existido previamente.
	solucionadoProblema	Actualiza la información sobre si se está solucionando el problema.

Tabla 11: Gestion

	Nombre	Descripción
Clase	*HostNameStorage*	Clase que contiene métodos dedicados al almacenamiento de los nombres de los servidores del grupo de servidores.
Métodos	getHostName	Obtiene el nombre del servidor.
	insertadHostName	Añade un nuevo servidor a la lista de servidores siempre y cuando este nombre no estuviera antes en dicha lista.
	borrarHostName	Elimina un nombre de la lista de nombres del grupo de servidores.
	sacarHostName	Obtiene un nombre de la lista de nombres.
	numeroDeServidores	Devuelve el número de servidores que hay en el grupo de servidores.

Tabla 12: HostNameStorage

	Nombre	Descripción
Clase	*ServerProperties*	Clase destinada al registro de las propiedades del servidor.
Métodos	imprimirServerProperties	Muestra por pantalla el estado actual del servidor: carga, número de máquinas virtuales, etc.
	getHostName	Devuelve el nombre del servidor.
	borrarPropsInterfaces	Elimina las propiedades de las interfaces de red que existían previamente.
	insertarPropsInterfaz	Añade una nueva propiedad de interfaz porque aparece una nueva máquina virtual.
	dominiosActivosInt	Devuelve el número de máquinas virtuales activas.
	dominiosInactivosInt	Devuelve el número de máquinas virtuales inactivas.
	cargaCPUDouble	Devuelve la carga del servidor.

	totalCPUsInt	Devuelve el número de procesadores del servidor.
	memoriaTotalLong	Devuelve la cantidad de memoria que tiene el servidor.
	memoriaUsadaLong	Devuelve la cantidad de memoria usada por el servidor.
	percUsoMemoriaDouble	Devuelve el porcentaje de uso de memoria del servidor.

Tabla 13: ServerProperties

	Nombre	Descripción
Clase	*DecisonAlgoritmo*	Clase que almacena el resultado del algoritmo de migración.
Métodos	add	Añade a la lista de máquinas que se van a migrar una máquina virtual junto con su servidor de destino.
	get	Se obtiene de la lista de máquinas virtuales a migrar una máquina junto con su servidor.
	size	Obtiene el tamaño de la lista de máquinas para migrar.

Tabla 14: DecisionAlgoritmo

	Nombre	Descripción
Clase	*ParMiracion*	Clase que representa un objeto formado por el servidor de destino y la máquina virtual que se va a migrar.
Métodos	setServDestino	Fija e servidor de destino.
	setIndMV	Fija la máquina virtual.
	getServDestino	Obtiene el servidor de destino.
	getIndMV	Obtiene la máquina virtual.

Tabla 15: ParMigracion

	Nombre	Descripción
Clase	*ControlarMV*	Clase que contiene los métodos necesarios para la monitorización de las máquinas virtuales.
Métodos	getDomain	Devuelve un objeto libvirt de tipo Domain.
	migrarMV	Realiza la migración de una máquina virtual.
	getDomainName	Devuelve el nombre de la máquina virtual.
	suspenderDominio	Deja una máquina virtual en estado suspendido.
	reanudarDominio	Saca una máquina virtual del

		estado suspendido.
	parametrosDominio	Muestra por pantalla parámetros de interés de la máquina virtual.
	informacionDominio	Obtiene el número de procesadores de la máquina virtual.
	calcularCargaMV	Calcula el uso de procesador que hace la máquina virtual.
	run	Método que se ejecuta a intervalos fijos de tiempo y llama al método calcularCargaMV.
	monitorizarCargaMV	Crea el executor para monitorizar la carga de la MV y crea el objeto para monitorizar la interfaz de red de la máquina virtual.
	nombreDeInterfaz	Obtiene si la interfaz es bridge o por defecto.
	configuracionInterfaces	Obtiene la configuración de los bridges del servidor.
	interfazDelBridge	Obtiene el nombre del bridge si la máquina virtual está usando un bridge.
	crearMonitorInterfaz	Crea un objeto para monitorizar la interfaz de red.
	getInUtilization	Obtiene la utilización del downlink de la interfaz de red.
	getOutUtilization	Obtiene la utilización del uplink de la interfaz de red.
	getPuntuacion	Obtiene la puntuación de la máquina virtual en función del valor de sus parámetros.
	shutdown	Para la monitorización de la máquina virtual y de la interfaz de red asociada a esta.
	getXMLDesc	Obtiene el archivo XML de definición de la máquina virtual.
	undefine	Borra una máquina virtual del registro del hipervisor.

Tabla 16: ControlarMV

	Nombre	Descripción
Clase	*EstadisticasInterfaz*	Clase que contiene métodos para la monitorización de la interfaz de red de la máquina virtual.
Métodos	run	Método que se ejecuta de forma periódica para calcular la

		velocidad de la interfaz y la utilización.
	ethtool	Método para obtener la velocidad de la interfaz de red.
	velocidadInterfaz	Método que obtiene la velocidad de la interfaz.
	monitorizarInterfaz	Método que crea el executor que ejecuta a intervalos fijos de tiempo el método run.
	nombreInterfazPor defecto	Método que obtiene el nombre de la interfaz por defecto.
	utilización	Método que obtiene mediante Sigar la utilización de la interfaz de red física.
	getInUtilization	Obtiene la utilización de la interfaz de red en el downlink.
	getOutUtilization	Obtiene la utilización de la interfaz de red en el uplink.
	shutdown	Método para parar la monitorización de la interfaz de red.

Tabla 17: EstadisticasInterfaz

	Nombre	Descripción
Clase	*PropiedadesInterfaz*	Clase que almacena la utilización del downlink y del uplink.

Tabla 18: PropiedadesInterfaz

	Nombre	Descripción
Clase	*InfoMemoria*	Clase que contiene métodos orientados a monitorizar la memoria del servidor.
Métodos	imprimirInfo	Imprime por pantalla información de interés sobre la memoria del servidor.
	enBytes	Transforma el tamaño de la memoria de kbytes a bytes.
	memoriaUsada	Crea el executor que ejecuta el método run a intervalos de tiempo fijos.
	run	Obtiene los parámetros relevantes de la memoria.
	registrarMemoria	Escribe el archivo de log de la memoria.
	cerrarFicheros	Cierra el archivo de log.
	shutDown	Método para terminar de monitorizar la memoria.
	actualizarProperties	Método de actualización de

		archivo properties de la memoria.
	sacarProperty	Método para obtener una property del archivo properties de la memoria.

Tabla 19: InfoMemoria

	Nombre	Descripción
Clase	*ConectarHipervisor*	Clase con métodos para realizar consultas al hipervisor de un servidor.
Métodos	getConnect	Devuelve el objeto connect de libvirt que representa la conexión a un hipervisor.
	getDomain	Devuelve un objeto domain que representa un máquina virtual, buscándolo por su nombre.
	getDomainID	Devuelve un objeto domain que representa un máquina virtual, buscándolo por su id.
	getNumDomActivos	Obtiene el número de máquinas virtuales activas en el hipervisor.
	setNumDomInactivos	Obtiene el número de máquinas virtuales inactivas en el hipervisor.
	getNumDomInactivos	Obtiene el número de máquinas virtuales inactivas en el hipervisor.
	setNumDomActivos	Obtiene el número de máquinas virtuales activas en el hipervisor.
	getHostName	Devuelve el nombre del servidor.
	listDefinedDomains	Devuelve el id de las máquinas virtuales inactivas.
	listDomains	Devuelve el id de las máquinas virtuales activas.
	getDominioPorID	Devuelve un objeto domain buscado por su id.

Tabla 20: ConectarHypervisor

12. PLIEGO DE CONDICIONES

El presente proyecto presenta el diseño y la implementación de un sistema de gestión autónoma de máquinas virtuales cuyo principal objetivo consiste en mejorar la eficiencia con la que se utilizan los recursos físicos disponibles sin perjudicar la calidad del servicio ofrecido, obteniendo así una reducción de los costes.
El sistema implementado es capaz de distribuir las máquinas virtuales entre un grupo de servidores de forma que se satisfagan en todos los servidores del grupo unas condiciones de carga escogidas por el administrador del sistema.

12.1. ENTREGABLES

- *Archivos de configuración*. Se entregarán los archivos de configuración de los servidores necesarios para ejecutar el sistema de gestión autónoma de servidores.

- *Software xmlBlaster*. Se entregarán los archivos necesarios para ejecutar un servidor xmlBlaster.

- *Software desarrollado*. Archivos de ejecución del sistema de gestión autónoma de máquinas virtuales desarrollado.

- *Documento de diseño y pruebas*. Comprende los documentos que definen la arquitectura diseñada y las pruebas realizadas sobre la implantación.

12.2. CONDICIONES DE DESARROLLO

12.2.1. RECURSOS HARDWARE

- Dos servidores con 8 procesadores 2.33GHz y 8GB de memoria RAM. Utilizados para la realización de pruebas y la validación del sistema.

- Equipo de desarrollo HP Pavilion Elite HPE con procesador Intel® Core™ i7-2600 3.4GHZ, memoria RAM 8GB y disco duro 1 TB. Utilizado para el desarrollo del software y la realización de pruebas.

- Equipo para elaboración de la documentación. Portatil con procesador Mobile AMD Sempron 1.79GHz y 2GB de memoria RAM.

12.2.2. RECURSOS SOFTWARE

- Sistema operativo Ubuntu Server 10.04 utilizado en los servidores de pruebas.

- Sistema operativo Ubuntu Desktop 10.04 utilizado en el equipo de desarrollo.

- Sistema operativo Windows 7 utilizado en el equipo de elaboración de la documentación.

- Microsoft Office Word 2007 utilizado para la elaboración de la documentación.
- Microsoft Office Visio 2007 utilizado para la elaboración de figuras y diagramas de la documentación.

- NetBeans IDE 6.9.1 utilizado para el desarrollo del sistema.

- xmlBlaster utilizado para el intercambio de mensajes entre servidores.

- Libvirt utilizado para la obtención de información referente a las máquinas virtuales de los servidores.

- Sigar utilizado para la obtención de información referente al servidor.

- Hipervisor KVM.

- Matlab utilizado para la generación de gráficas.

13. PRESUPUESTO

El presupuesto se compone de las siguientes partes:

- Presupuesto de ejecución material.
- Gastos generales y Beneficio industrial.
- Honorarios por la redacción y dirección del proyecto.
- Presupuesto total.

El Presupuesto de ejecución material y los Gastos generales y beneficio industrial constituyen el Presupuesto por ejecución por contrata que, junto con los Honorarios por la redacción y dirección del proyecto, integran el Presupuesto total.

Todas las cantidades aparecen expresadas en euros.

13.1. PRESUPUESTO DE EJECUCIÓN MATERIAL

El Presupuesto de ejecución material consta de Costes de mano de obra y Costes de recursos materiales. No se incluirán los Honorarios de dirección del proyecto que serán considerados aparte.

13.1.1. DESCOMPOSICIÓN EN TAREAS

Para facilitar la comprensión de estos costes se realizará una división en tareas del trabajo realizado para el desarrollo de este Proyecto Fin de Carrera. Así mismo, se representarán, a través de un diagrama de Gantt, las relaciones de precedencia temporal entre estas actividades.

El proyecto consta de las siguientes tareas:

- **Tarea 1:**

  Objetivo: Estudio del estado del arte de la Gestión Autónoma de Redes y los sistemas de virtualización. Posteriormente se centrará el estudio en la gestión autónoma aplicada a la virtualización.

  Duración: 4 meses.

  Esfuerzo: Ingeniero superior, 1.5 personas-mes.

- **Tarea 2:**

  Objetivo: Preparación del entorno. Instalación de los sistemas operativos. Instalación del entorno de desarrollo. Estudio y búsqueda de APIs y librerías necesarias para el desarrollo.

Duración: 3.5 meses.

Esfuerzo: Ingeniero superior, 1.5 personas-mes.

- **Tarea 3:**

  Objetivo: Diseño del sistema. Toma de decisiones en el diseño, se decidirá seguir una estrategia de desarrollo. Se diseñan los algoritmos a implementar.

  Duración: 5.5 meses.

  Esfuerzo: Ingeniero superior, 1.5 personas-mes.

- **Tarea 4:**

  Objetivo: Implementación, desarrollo del diseño. Desarrollar en código Java los diseños teóricos.

  Duración: 6 meses.

  Esfuerzo: Ingeniero superior, 4 personas-mes.

- **Tarea 5:**

  Objetivo: Pruebas funcionales. Probar que el funcionamiento del sistema se corresponda con el diseño.

  Duración: 0.25 meses.

  Esfuerzo: Ingeniero superior, 0.12 personas-mes.

- **Tarea 6:**

  Objetivo: Reprogramación, corrección de errores. Corregir los errores encontrados en las pruebas.

  Duración: 1 mes.

  Esfuerzo: Ingeniero superior, 0.5 personas-mes.

- **Tarea 7:**

  Objetivo: Pruebas funcionales. Validación de la corrección de errores.

  Duración: Ingeniero superior, 0.25 meses.

  Esfuerzo: 0.12 personas-mes.

- **Tarea 8:**

  Objetivo: Estudio del rendimiento del sistema y obtención de gráficas que lo certifiquen.

  Duración: 1 mes.

  Esfuerzo: Ingeniero superior, 0.5 personas-mes.

- **Tarea 9:**

  Objetivo: Redacción de la documentación asociada al Proyecto Final de Carrera.

  Duración: 1.5 meses.

  Esfuerzo: Ingeniero superior, 0.75 personas-mes. Administrativo, 0.5 personas-mes.

La *Figura 40* muestra un diagrama de Gantt con las relaciones de dependencia entre las distintas tareas.

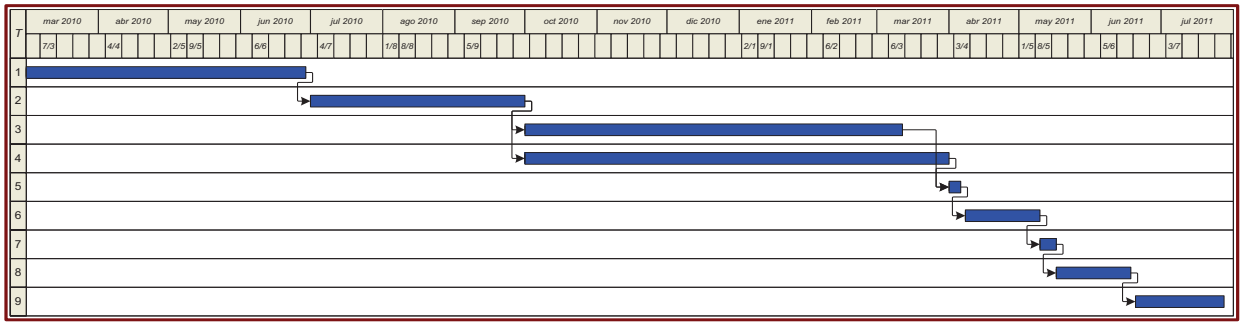


Figura 40: Diagrama de Gantt del Proyecto

Con esto, se obtiene un plazo de ejecución del proyecto de 17 meses.

13.1.2. COSTES DE MANO DE OBRA

Para la realización del proyecto se requieren los siguientes perfiles profesionales:

- Un Ingeniero Superior de Telecomunicación, encargado del planteamiento, desarrollo e implementación del trabajo técnico.

- Un Administrativo, encargado de la redacción, presentación y encuadernación del proyecto.

La estimación de los costes se realiza en base a los siguientes datos:

- Cotizaciones según el Régimen General de la Seguridad Social. El ingeniero pertenece al grupo 1 y el administrativo al grupo 7.

- Jornada laboral de 8 horas/día y 21 días laborales/mes.

Con estos datos y la distribución del trabajo en tareas mostrado, se tiene lo siguiente:

Costes salariales		
Concepto	**Grupo 1**	**Grupo 7**
Base cotizable máxima anual	38761,20	38761,20
Contingencias comunes (23.6%)	9147,64	3540,00
Desempleo, F.G.S y Formación profesional (7.5%)	2907,09	1125,00
Accidentes de trabajo y enfermedades profesionales	3230,10	1250,00
Coste de la seguridad social	15284,83	5915,00
Salario bruto anual	40000,00	15000,00
Coste salarial anual	55284,83	20915,00
Coste salarial por hora	27,42	10,37
Número de horas	1762,32	84,00
Coste total	**48322,81**	**871,08**

Tabla 21: Costes salariales

Mano de obra	
Concepto	**Coste**
Ingeniero superior de telecomunicación	48322,81
Administrativo	871,08
Coste total	**49193,89**

Tabla 22: Costes mano de obra

13.1.3. COSTE DE LOS RECURSOS MATERIALES

En la siguiente tabla constan los costes de los recursos materiales empleados, considerando un periodo de amortización para el hardware y el software de 3 años.

En primer lugar se indicarán los costes totales y a continuación se imputarán las cuantías correspondientes a la amortización de los recursos durante su periodo de utilización en el desarrollo del proyecto.

Recursos hardware			
Concepto	**Coste total**	**Meses**	**Coste real**
Equipo de desarrollo	1500	13	541,67
Servidor 8 procesadores (x2)	6000	13	2166,67
Equipo de generación de documentación	750	2	41,67
Total recursos hardware			**2750,01**

Tabla 23: Gastos recursos hardware

Recursos software			
Concepto	**Coste**	**Meses**	**Coste real**
Sistema operativo Ubuntu Server 10.4 64 bits (x2)	0,00	13	0,00
Sistema operativo Ubuntu Desktop 10.04 32 bits	0,00	13	0,00
Sistema operativo Windows 7	119,99	2	6,67
Hipervisor KVM	0,00	13	0,00
API Libvirt	0,00	13	0,00
Middleware xmlBlaster	0,00	13	0,00

API Sigar	0,00	13	0,00
Herramienta de desarrollo Java Netbeans	0,00	13	0,00
Herramienta de generación de gráficas Matlab	2000,00	1	55,55
Microsoft Word 2007	379,00	2	21,05
Microsoft Visio 2007	725,00	2	40,27
Total recursos software			**123,54**

Tabla 24: Gastos recursos software

Recursos materiales	
Concepto	**Coste**
Recursos hardware	2750,01
Recursos software	123,54
Consumibles, material fungible y de oficina	200,00
Total recursos materiales	**3073,55**

Tabla 25: Gastos recursos materiales

13.1.4. COSTE TOTAL DE LOS RECURSOS

La suma de los costes por mano de obra y de los costes por recursos materiales es lo que constituye el *Presupuesto de ejecución material (P.E.M)*.

Presupuesto de ejecución material	
Concepto	**Coste**
Coste mano de obra	49193,89
Coste recursos materiales	3073,55
Total	**52267,44**

Tabla 26: Presupuesto de ejecución material

13.2. GASTOS GENERALES Y BENEFICIO INDUSTRIAL

Bajo Gastos generales se incluyen todos aquellos gastos derivados de la utilización de instalaciones, cargas fiduciarias, amortizaciones, gastos fiscales, etc. Con esto, el Presupuesto de Ejecución por contrata queda como sigue:

Presupuesto de ejecución por contrata	
Concepto	**Coste**
Presupuesto de ejecución material	52267,44
Gastos generales (16% del P.E.M)	8362,79
Beneficio industrial (6% del P.E.M)	3136,04
Total presupuesto de ejecución por contrata	**63766,27**

Tabla 27: Presupuesto de ejecución por contrata

13.3. HONORARIOS POR REDACCIÓN Y DIRECCIÓN DEL PROYECTO

Los Honorarios que recomienda aplicar el Colegio Oficial de Ingenieros de Telecomunicación, tanto para la redacción como para la dirección del proyecto son los asociados a Trabajos tarifados por tiempo empleado, con un valor de un 5.6%.

13.4. PRESUPUESTO TOTAL

Finalmente, sumando todas las imputaciones anteriores, y aplicando el 18% de IVA, se obtiene el presupuesto total.

Presupuesto total	
Concepto	**Coste**
Presupuesto de ejecución por contrata	63766,27
Honorarios por dirección	3570,91
Honorarios por redacción	3570,91
Subtotal	70908,09
I.V.A (18%)	12763,46
Presupuesto total	**83671,55**

Tabla 28: Presupuesto total

El presupuesto total del proyecto asciende a OCHENTA Y TRES MIL SEISCIENTOS SESENTA Y UN Euros CON CINCUENTA Y CINCO céntimos.

Madrid, septiembre de 2011

14. ANEXO D: DOCUMENTACIÓN ACREDITATIVA DE MÉRITOS

Este Proyecto Final de Carrera ha dado lugar a la elaboración de un poster que ha sido evaluado por pares y aceptado para ser expuesto en el congreso:

5th International DMTF Academic Alliance Workshop on System and Virtualization Management: Standards and the Cloud

A continuación se muestra el email de notificación de aceptación:

```
Dear authors,

We are happy to inform you that your poster abstract submitted to SVM'11
has been accepted for the workshop.  Your abstract will also be included
in the Journal of Network and Systems Management (JNSM) report for this
event.

The reviews of your poster are attached below. We hope that you find them
helpful when creating your poster for the camera ready submission.
Please consider the reviewers' comments carefully, as well as the
instructions found on the SVM'11 Web site, when preparing the camera-
ready copy of your poster, which is required by September 20, 2011.

  http://dmtf.org/svm11/posters

To submit your camera ready poster, please upload a .pdf version of the
poster to your original EasyChair submission:
  https://www.easychair.org/conferences/?conf=svm11postersession


Registration and travel information is available:

  http://dmtf.org/svm11/travel
  http://dmtf.org/svm11/registration

Should you have any questions please do not hesitate to contact the
workshop organizers at:

         svm11@dmtf.org

Thank you again for submission and support of the DMTF Academic Alliance
Workshop, SVM'11.

Best regards,

SVM'11 Program Committee Chair
http://dmtf.org/svm11
```

```
----------------------- REVIEW 1 ---------------------
PAPER: 5
TITLE: Design and implementation of an autonomic management system for
virtual machines in a cloud environment
AUTHORS: Alejandro Chillarón and Jorge E. López De Vergara

There are a number of approaches for addressing the problem that the
authors are addressing.
It seems that the difference is that the authors are proposing a peer-to-
peer approach.   There isn't a lot of work on this so this could end up
being interesting in that it uses a technique not seen a lot cloud
resource management.


----------------------- REVIEW 2 ---------------------
PAPER: 5
TITLE: Design and implementation of an autonomic management system for
virtual machines in a cloud environment
AUTHORS: Alejandro Chillarón and Jorge E. López De Vergara

The poster describes a novel approach to finding an optimal place to run
a Virtual Machine. It seems to match well with the charter of SVM and I
recommend accepting it.
```

A continuación se anexa el poster que se ha elaborado para el congreso:

Design and implementation of an autonomic management system for virtual machines in a cloud environment

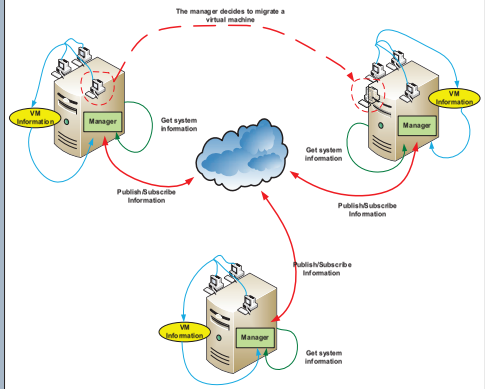
Alejandro Chillarón Oliva, Jorge E. López de Vergara Méndez
High Performance Computing and Networking Research Group, Universidad Autónoma de Madrid, Spain

Abstract

- **Cloud Computing:** Accessing the computing resources through the network
- **Virtual Machines** are deployed in this type of infrastructures
- It is necessary to provide the necessary computing power when needed
- This autonomic system manages these virtual machines
  - All servers monitor their performance
  - This information is shared among all servers
  - If one server is overloaded, it tries to migrate virtual machines
- The system follows the Monitor-Analysis-Plan-Execute loop
- The system fulfills the self-CHOP rules

Overview

Server Group: Servers that can host the same virtual machines
- Each server runs an autonomic system
- The autonomic system collects information about itself
- The servers share the information with the server group
- The autonomic system makes a decision about migrating a VM
  - If the server is overloaded
- Migrating a VM must not generate overload in other server



Materials

- Hypervisor: KVM
- Virtual Machines API: libvirt
- Server information: springsource HYPERIC SIGAR
- Messages exchange: xmlBlaster.org Open Source for MOM (message oriented middleware)

Design

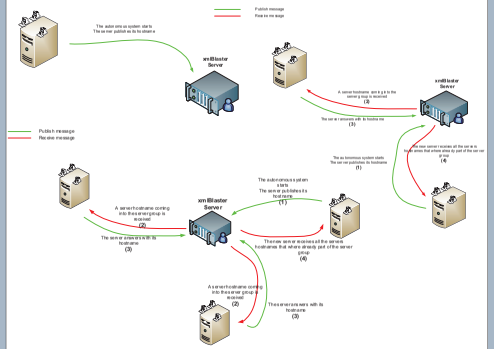
MONITORING PHYSICAL SERVERS PARAMETERS
Active Virtual Machines
Inactive Virtual Machines
Server Load
Number of CPUs
Used Memory
Total Memory
Used Memory Percentage
Network Interface usage

SYSTEM THRESHOLDS
Maximum-load
Maximum-load-time
Waiting-time

MONITORING VIRTUAL MACHINES PARAMETERS
Load generated by the virtual machine in the server
Network Interface usage

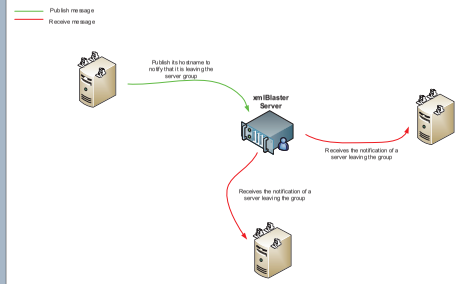
Joining the server group

- If a server joins the server group
  - It publishes its name to the server group members
  - The rest of servers in the group include the new server in the list of active servers



Leaving the server group

- If a server leaves the server group
  - It publishes a message with its hostname
  - The rest of the server in the group must erase it from the list of active servers



Virtual Machines Reallocation Algorithm

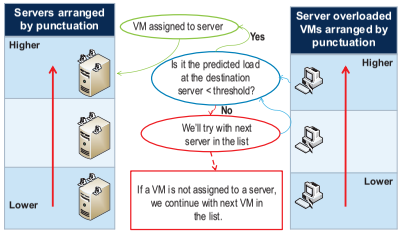
- If a server is overloaded it initiates a process to migrate virtual machines to other servers in the server group.
- First, servers are scored according to the monitoring parameters

Server properties	Server punctuation
•Hostname: Server1 •Active VMs: 1 •Inactive VMs: 4 •CPU load: 0.04 •Number of CPUs: 8 •Total memory: 8 GB •Used memory percentage: 0.6	HIGHER 2.247
•Hostname: Server 2 •Active VMs:3 •Inactive VMs: 3 •CPU load: 0.2 •Number of CPUs: 8 •Total memory: 8 GB •Used memory percentage: 0.5	1.994
•Hostname: Server 3 •Active VMs: 5 •Inactive VMs: 1 •CPU load: 0.55 •Number of CPUs:8 •Total memory: 8GB •Used memory percentage: 0.65	LOWER 1.602

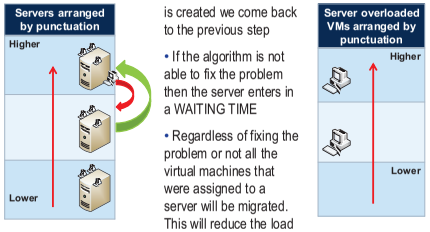
- After this, the overloaded server manager assigns a score to the virtual machines running on it

VMs of the overloaded server	VMs Punctuation
• Load generated by the VM on the server: 0.4 • Downlink utilization: 0.03 • Uplink utilization: 0.01	HIGHER 0.287
• Load generated by the VM on the server: 0.2 • Downlink utilization: 0.03 • Uplink utilization: 0.01	0.147
• Load generated by the VM on the server: 0.05 • Downlink utilization: 0.03 • Uplink utilization: 0.01	LOWER 0.042

- The virtual machines with higher punctuation are reassigned to the servers with higher punctuation when possible.



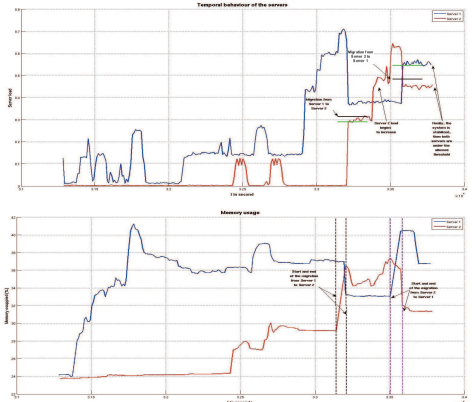
- After a virtual machine is reassigned to a server, the scores in the server list are recalculated



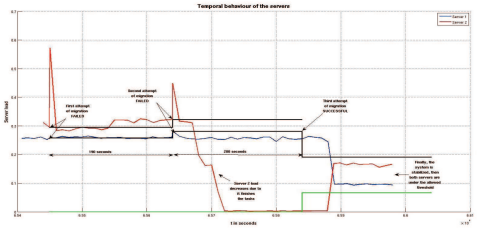
- After the new server list is created we come back to the previous step
- If the algorithm is not able to fix the problem then the server enters in a WAITING TIME
- Regardless of fixing the problem or not all the virtual machines that were assigned to a server will be migrated. This will reduce the load on the overloaded server

Results

Test 1:
- Load Threshold: 60 %
- Consecutive load measurements over the load threshold: 2



Test 2:
- Load Threshold: 20 %
- Initial waiting time: 190 seconds



Conclusions

- Migrating decisions are based on server load
- Decisions could also be based on memory, but:
  - It doesn't vary as quick as the load
  - It increases reasonably when a virtual machine is started or powered off
- All the memory is transferred to the new server when a virtual machine is migrated
  - It is necessary to know if there is enough memory at the destination to host the new VM
- Load prediction is not totally precise
  - Errors are small and therefore they are acceptable
- Decision algorithm is quicker than others

Contact information

UAM High Performance Computing and Networking

Escuela Politécnica Superior
Francisco Tomás y Valiente, 11
28049 Madrid, Spain

Web: www.hpcn.es
Tel: +34 91 497 22 46
E-mail: jorge.lopez_vergara@uam.es
E-mail: alejandro.chillaron@estudiante.uam.es