HYDRA: plataforma para despliegue y administración remota de sistemas heterogéneos en red

Ignacio Díez, Cleto Martín, David Villa, Francisco Moya,
Félix Jesús Villanueva, Juan Carlos López

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Ignacio.Diez@alu.uclm.es, {Cleto.Martin, David.Villa, Francisco.Moya, Felix.Villanueva, JuanCarlos.Lopez}@uclm.es

Resumen

El uso de varios sistemas operativos en cada computador es una tendencia cada vez más común en todos los entornos, desde centros de investigación, centros de computación y entornos docentes, hasta medianas y grandes empresas. El gran número de nodos que puede llegar a implicar y la heterogeneidad del software utilizado en este tipo de escenarios hace que los costes de mantenimiento y gestión de los mismos sean muy elevados tanto en recursos como en tiempo.

En este trabajo[1] se presenta un sistema diseñado para gestionar el software de un sistema distribuido heterogéneo, de forma que sea posible automatizar su mantenimiento así como el despliegue de software en los nodos y permitiendo su administración remota.

1. Introducción

En un entorno con varias decenas (incluso centenares) de nodos, instalar el software necesario en cada equipo y mantenerlos actualizados y libres de errores puede llegar a ser una tarea difícil que consume una ingente cantidad de tiempo y recursos. Además, en estos entornos suele ser habitual el uso de software y hardware heterogéneo: distintos tipos de SO y aplicaciones, para diferentes arquitecturas.

[1]Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología, el Centro para el Desarrollo de Tecnología Industrial, la Junta de Comunidades de Castilla-La Mancha y los Fondos Europeos para el Desarrollo Regional a través de los proyectos TEC2008-06553/TEC(DAMA), PAI08-0234-8083(RGrid) y CEN-20091048(Energos).

En este tipo de sistemas y desde el punto de vista de la administración, las tareas de gestión de la configuración y del mantenimiento son mucho más complejas que en los sistemas homogéneos; ya que cada nodo tiene unas características y restricciones intrínsecas que difieren de las del resto.

El sistema HYDRA (*Heterogeneous sYstem Deployment and Remote Administration*) que se describe en este documento pretende dar soporte y automatizar las tareas de mantenimiento de estos escenarios, desde la instalación de un sistema operativo, hasta restaurar la configuración de un nodo, pasando por la aplicación de actualizaciones o parches.

Con el sistema que se propone en este artículo, los administradores pueden reducir el tiempo que dedican a instalar nuevos SSOO y recuperar o solucionar incidencias; y los usuarios podrán decidir sobre los SSOO que desean, así como sus configuraciones personalizadas y las aplicaciones que necesiten.

En la sección 2 del presente trabajo se analizarán algunas iniciativas previas que abordan temas similares y en la sección 3 se plantean los objetivos que se pretenden alcanzar en la solución propuesta. Después, en la sección 4, se muestra la arquitectura del sistema propuesto y se describen sus distintos componentes; y en la sección 5 se detalla el esquema de funcionamiento del mismo. En la sección 6 se presenta un caso de estudio práctico y la aplicación de HYDRA en el ámbito docente. Finalmente, en las secciones 7 y 8 se explican las futuras líneas de trabajo y las conclusiones, respectivamente.

2. Trabajo previo

En [14] se describe un modelo para simplificar el despliegue de servicios mediante *Virtual Appliances*. Estas *appliances* están respaldadas por máquinas virtuales, y sólo se centran en el despliegue de aplicaciones, sin considerar la distribución de SSOO. Por ello, no se tienen en cuenta aspectos como la preparación y configuración de los nodos (particiones, sistema de ficheros, plataforma hardware,etc), necesidades específicas de cada SO para el arranque...

Grid5000 [2] ofrece una infraestructura de grid con herramientas de despliegue pensada para facilitar la investigación en todos los niveles de la pila software. SystemImager [6] permite para instalar imágenes de GNU/Linux en varios nodos, aunque no ofrece la posibilidad de tener distintas imágenes (SO y aplicaciones) en cada nodo. BCFG [4] es una herramienta que automatiza las tareas de gestión de la configuración en entornos heterogéneos, aunque no se encarga del despliegue de imágenes.

Flissi y Merle [7] describieron un marco de trabajo para unificar herramientas sofwtare de uso común en los entornos de grid como componentes de su sistema, para facilitar el despliegue de aplicaciones y servicios.

Una solución propuesta por Zhang y Zhou[15] describe un escenario en el que los programas están almacenados en un servidor central, y los usuarios los ejecutan bajo demanda desde otros equipos, desligando así el almacenamiento del programa de su ejecución. La administración se vuelve más sencilla, ya que los programas se encuentran ubicados en un sólo equipo, pero éste se convierte en un punto de fallo crítico.

Draper et al. [5] definieron un esquema XML para describir *unidades de instalación*, con la intención de crear un estándar común para que dichas unidades de instalación pudieran ser manejadas por cualquier tecnología de instalación. Su trabajo estaba orientado a paquetes, aplicaciones, plug-ins, etc., sin dar un soporte específico a la instalación de SSOO.

En Kadeploy[9], los nodos tienen un sistema instalado (al que denominan *entorno de referencia*), y varias particiones en el disco duro previamente establecidas. Un *entorno* está constituido por un SO y las aplicaciones que contiene. Cuando un usuario quiere usar los nodos con un *entorno* específico indica en qué partición debe alojarse y Kadeploy realiza el despliegue y reinicia los nodos, que arrancarán esa partición. Una vez que termine de usar los equipos, éstos se vuelven a reiniciar, esta vez al entorno de referencia. Dado que los usuarios deben conocer qué particiones hay y cuáles están disponibles, y Kadeploy no proporciona ninguna solución para automatizar la gestión de esta información, en entornos complejos esto puede llegar a ser problemático.

3. Objetivos

Se persigue conseguir una herramienta que facilite la labor tanto al administrador de sistemas como a los usuarios, en entornos de trabajo donde se utilice software heterogéneo en diferentes máquinas.

Entre los principales objetivos a cubrir se encuentran:

Instalación automática y masiva: La principal finalidad es poder distribuir aplicaciones y SSOO en un número elevado de computadores de forma automática y *completamente desatendida*, de forma que ni los administradores ni los usuarios deban tener en cuenta los detalles de gestión y configuración.

Instalación nativa: La utilización de una máquina virtual impide el acceso a los dispositivos que ésta no emule en su totalidad, lo que restringe, por ejemplo, la programación de *drivers*, o la utilización de la aceleración por hardware de las tarjetas gráficas. Utilizar el SO de forma nativa evita estos inconvenientes, y también se consigue que todo el potencial de la CPU y la memoria queden a disposición del usuario, lo que se traduce en una mejora del rendimiento de la máquina.

Recuperación ante fallos: Mediante el uso de la herramienta de administración del sistema, será muy sencillo restaurar un equipo que se esté defectuoso o que haya sido atacado. Bastará con indicarle al sistema que vuelva a configurar ese equipo desde cero, y esto se hará de forma automática.

Configuración a medida: Cada usuario podrá crear una o varias configuraciones de SO y

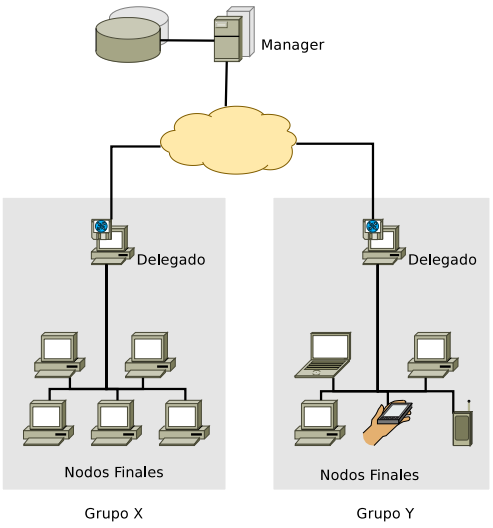

Figura 1: Estructura de HYDRA

aplicaciones, y adaptarlas con los programas y el software necesario acorde a sus necesidades específicas.

Planificación: se deberá permitir algún tipo de planificación del funcionamiento del sistema que permita automatizar la gestión de los despliegues e instalaciones a lo largo del tiempo. De esta forma se reduce la carga de trabajo del personal de administración de sistemas.

4. Arquitectura

En esta sección se describen los componentes principales del sistema, las relaciones entre ellos y su papel dentro del mismo. La figura 1 representa un esquema de la arquitectura del sistema HYDRA.

Manager

El *Manager* es el servicio principal del sistema. Este servicio alberga los siguientes componentes:

- Servicios básicos para las comunicaciones y el despliegue del software.
- Imágenes HYDRA, es decir, ficheros que mantienen un determinado sistema de archivos y que contienen un sistema operativo junto con las aplicaciones, configuraciones y datos personalizados de los usuarios.
- Una base de datos en la que se almacenan las configuraciones de los nodos y los diferentes despliegues.

El *Manager* se encarga de preparar las imágenes para su distribución, enviarlas a los delegados y coordinar la instalación. Tanto el *Manager* como la base de datos se encuentran replicadas para garantizar la disponibilidad ante fallos en el servicio.

Como se verá más adelante, los usuarios contactarán con este servicio para añadir, borrar y reconfigurar el despliegue de las imágenes HYDRA.

Delegado

El *Delegado* actúa como un enlace entre los nodos finales y el *Manager*. Cada delegado es responsable de un subconjunto de los nodos del sistema. Descarga las imágenes que sus nodos van a necesitar y se ocupa de servirlas. También es el responsable de despertar a sus nodos y ofrecerles una configuración básica para que empiecen a funcionar.

La figura del *Delegado* sirve para varios propósitos. En primera instancia, permite descongestionar la red y la carga de trabajo del *Manager*, al hacer que los nodos obtengan sus imágenes desde el *Delegado* en lugar de que todos las soliciten al *Manager*. Por otro lado, sirve de enlace entre los nodos y el *Manager*, ya que es posible que éste último, al ser una máquina *sensible*, esté en una red distinta y protegida a la que los nodos no puedan acceder. Y, finalmente, la jerarquía de responsabilidades que existe entre *Delegado* y *Manager* permite que la organización adapte la infraestructura de HYDRA a su organización interna.

HostInfo e Installer

Son dos servicios que actúan en los nodos finales. El primero de ellos, *HostInfo*, es desplegado automáticamente por el *Manager* a todo nodo que se conecte a la red HYDRA, y se encarga de recabar información sobre el hardware de la máquina: tipo de procesador, cantidad de memoria, estructura de los discos duros, etc.

El servicio *Installer* sólo se despliega en aquellos nodos que van a comenzar el proceso de instalación. Con él se controla y efectúa la instalación de

las imágenes. Se recibe la configuración de la instalación desde el *Manager*, es decir, qué imágenes tiene que instalar y a qué *Delegado* debe pedírselas. Al finalizar, informa al *Delegado* y al *Manager*.

Middleware de comunicaciones

En cualquier aplicación distribuida suele ser necesario el uso de un middleware que permita la intercomunicación de los distintos componentes que integran el sistema. Concretamente, los middlewares orientados a objetos como CORBA [13], Java RMI [11] o ZeroC Ice [10] abstraen al programador del acceso a la red y permiten construir aplicaciones distribuidas manteniendo la filosofía de orientación a objetos.

Para el desarrollo de HYDRA se eligió Ice por ser multi-plataforma y soportar varios lenguajes de programación, dos importantes características para un sistema heterogéneo.

Además, Ice proporciona múltiples servicios y facilidades para el desarrollo de aplicaciones distribuidas. En concreto, aparte de la *transparencia de localización*, que es esencial en cualquier aplicación distribuida, se hace un uso intensivo de los servicios *IceGrid*, para computación en grid y *IcePatch*, para la distribución de ficheros (tanto aplicaciones de HYDRA como las imágenes de SO).

IcePatch permite ahorrar tiempo y ancho de banda en la actualización de imágenes. Cuando hay una imagen instalada en los nodos y sólo se pretende aplicar algún parche o instalar nuevas aplicaciones, sólo se transmitirán los ficheros que se hayan modificado o añadido. En un sistema como HYDRA, donde los ficheros transmitidos suelen ser pesados (SSOO, aplicaciones, etc.), esta característica es fundamental para obtener un buen desempeño.

5. Esquema de funcionamiento

La figura 2 muestra un esquema de funcionamiento de HYDRA que se describirá a lo largo de esta sección.

En primer lugar, el usuario debe preparar las imágenes en la que se incluirán los SSOO, aplicaciones y demás información conforme a sus necesidades. Estas imágenes pueden crearse utilizando software de gestión de máquinas virtuales como VMWare [2] o VirtualBox [3].

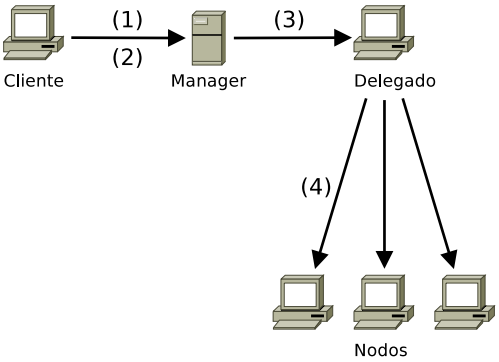

Figura 2: Esquema de funcionamiento HYDRA

Una vez creada, la imagen debe subirse al *Manager* de HYDRA (1). Cuando la imagen se encuentre alojada en el servidor, el usuario debe indicarle al *Manager* la información del *Despliegue* para dicha imagen (2). Un *Despliegue* es la forma de indicarle al sistema:

- Qué imágenes tiene que instalar,
- A qué nodos involucra, y
- Cuándo deben estar disponibles (el calendario).

Para facilitar la labor de crear *despliegues*, se pueden definir grupos de nodos, de forma que desplegar una imagen en un grupo implica instalarla en cada uno de los nodos que lo integran. Estos grupos no tienen porqué ser disjuntos, y cada usuario puede crearlos a su medida.

También es posible indicar ciertas *Restricciones* a la hora de instalar una imagen. En un grid, el hardware de los equipos no tiene porqué ser homogéneo, y puede que haya dispositivos que necesiten *drivers* específicos, incompatibles con los de otros dispositivos. Es el caso, por ejemplo, de las tarjetas de vídeo. Podría entonces prepararse una imagen con los drivers para una tarjeta específica y añadir una restricción al *Despliegue* para que sólo instale esa imagen en los equipos que posean dicha tarjeta. Las restricciones pueden aplicarse a cualquier información que pueda recogerse mediante *HostInfo*.

[2] http://www.vmware.com/

[3] http://www.virtualbox.org/wiki/Documentation

Con todo definido, el sistema preparará la imagen para ser distribuida en los equipos. La preparación de las imágenes depende en gran medida del tipo de SO que contenga. Por ejemplo, para las imágenes de sistemas Unix, es necesario eliminar los enlaces simbólicos para evitar que se traten como el fichero al que enlazan, en lugar de como ficheros especiales.

Cuando el sistema reciba la orden de comenzar la instalación, el *Manager* procederá a inspeccionar su base de datos para decidir qué imágenes debe distribuir a qué delegados (3). Esta orden puede programarse para ejecutarse todos los días a una hora en la que no se utilicen los equipos; por ejemplo, por las noches. A continuación, se describe el proceso de instalación (4), completamente automático. Consta de los siguientes pasos:

- Despertar a los nodos que forman parte del despliegue. Para ello, puede utilizarse tecnologías como *Wake on Lan* [1].
- Iniciar los nodos con una imagen mínima utilizando, por ejemplo, PXE [12].
- Desplegar *HostInfo* e *Installer* en los nodos finales.
- Utilizando la información de *HostInfo* se configuran los nodos y se comprueban las restricciones del *Despliegue*.
- Utilizando *Installer* se crean las tablas de particiones y formatean convenientemente los discos en base a la información del *Despliegue*. Además, se instalan las imágenes en las particiones pertinentes.
- Instalar un gestor de arranque, por ejemplo, (GRUB) [8]. Esto permitirá mostrar un menú al usuario al iniciar la máquina que permita seleccionar elegir qué SO arrancar de entre todos los que se han instalado. De esta forma, los usuarios podrán utilizar los diferentes SSOO planificados.
- Marcar cada nodo como *actualizado*, para que no se distribuya la imagen mínima inicial.
- Apagar los nodos.

Después de todo ello, los nodos estarán listos para ser usados por los usuarios finales. Todos los SSOO y aplicaciones estarán instaladas y los usuarios podrán utilizarlos libremente.

6. Caso de estudio

En la Escuela Superior de Informática de Ciudad Real existen 9 laboratorios docentes, con un total de 184 nodos, en los que se utilizan hasta 35 imágenes de SO distintas para las asignaturas que se imparten.

El administrador estableció un sistema basado en máquinas virtuales, en el que los equipos se inician con un SO pre-instalado, donde el usuario (normalmente, el alumno) elige la imagen que quiere arrancar en función de la asignatura que corresponda a ese momento. La imagen seleccionada se ejecuta en una máquina virtual VMWare. Las imágenes pueden crearlas los profesores y éstos las envían a los administradores para que sean añadidas al sistema.

Sin embargo, este enfoque tiene varias desventajas, sobre todo para el usuario final que, al estar trabajando sobre una máquina virtual, el uso de la memoria y CPU son limitados. Gran parte de estos recursos se dedican al software de virtualización. Además, tampoco tienen acceso directo a la máquina, ya que sólo pueden acceder a los dispositivos que VMWare sea capaz de emular.

Para el administrador también tiene desventajas. Si la gestión del SO residente en cada una de las máquinas no está automatizada ni centralizada, los cambios y actualizaciones en dicho SO pueden ser costosos en términos de tiempo y recursos.

Como solución, HYDRA se planteó como una alternativa al sistema existente, ya que ofrecía ventajas significativas tanto a niveles de administración como de usuario. En este punto debemos distinguir 3 tipos de usuarios:

- El administrador del sistema, encargado de instalar, configurar y mantener HYDRA.
- Los profesores, que utilizan HYDRA para instalar sus imágenes en los ordenadores de las aulas donde van a impartir clase. Las labores de mantenimiento del administrador se reducen al mínimo ya que los profesores pueden añadir, eliminar y actualizar las imágenes *directamente* (sin intermediarios).

- Los alumnos, que son los usuarios finales, y que trabajarán sobre las imágenes instaladas (aunque no interactúan directamente con HYDRA).

Para utilizar HYDRA, se habilitó un *Delegado* cada laboratorio, en cada uno de los cuales hay una media de 20 ordenadores. Para tal fin se eligió el PC reservado a los profesores. Para evitar problemas de compatibilidad y no interferir con el sistema actual hasta la completa aceptación del producto, el SO actualmente instalado en los equipos puede distribuirse como una imagen más de HYDRA.

Aunque el sistema no se ha implantado de forma oficial, se han hecho pruebas de integración en un entorno controlado, instalando 3 imágenes distintas de GNU/Linux, correspondientes a otras tantas asignaturas. Una de ellas es una imagen mínima, con un SO básico, y las otras dos contienen sistemas más completos, con entorno gráfico e incluso soporte para aceleración 3D.

7. Trabajo futuro

Actualmente, el sistema es capaz de instalar distintas imágenes de sistemas GNU/Linux en formato ext3. Se está trabajando ya en dar soporte a otros SSOO, en concreto Windows y MacOS X. Es posible ya copiar los ficheros de la imagen Windows a los ordenadores destino, y el trabajo actual se centra en el proceso de arranque.

También se pretende integrar nuevos dispositivos con prestaciones más modestas que las de un PC; por ejemplo, las nuevas generaciones de móviles, PDAs, etc. De hecho, existe una versión del middleware Ice para este tipo de dispositivos (llamado IceE) que facilita en gran medida su integración con HYDRA.

8. Conclusiones

Se ha presentado un sistema para distribuir e instalar sistemas operativos y aplicaciones en varias máquinas de forma automática, distribuida y desatendida, de fácil gestión y configuración. Después de la primera instalación el sistema es administrado directamente por los usuarios, interviniendo un administrador sólo en caso de que se produzca algún tipo de fallo.

Los usuarios pueden preparar sus propias imágenes con los programas y la configuración que necesiten, sin tener que depender de otros usuarios o del administrador del sistema.

Puede especificarse un calendario que defina qué SO se necesitan cada día, y programar el sistema para actualizar los nodos de forma automática. También es posible instalar las imágenes en un nodo «en vivo», en caso de que se detecte un funcionamiento incorrecto o se necesite actualizar los equipos una vez haya comenzado la clase.

Referencias

[1] AMD. Magic Packet Technology. Technical report, Advanced Micro Devices, Inc., 1995.

[2] Franck Cappello, Eddy Caron, Michel Daydé, Frédéric Desprez, Yvon Jégou, Pascale Primet, Emmanuel Jeannot, Stephane Lanteri, Julien Leduc, Noredine Melab, Guillaume Mornet, Raymond Namyst, Benjamin Quetier, and Olivier Richard. Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, pages 8 pp.+, 2005.

[3] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23(3):187 – 200, 2000.

[4] N. Desai, A. Lusk, R. Bradshaw, and R. Evard. Bcfg: a configuration management tool for heterogeneous environments. In *Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on*, pages 500–503, 2003.

[5] Christine Draper, Randy George, and Marcello Vitaletti. Installable Unit Deployment Descriptor for Autonomic Solution Management. *Database and Expert Systems Applications, International Workshop on*, 0:742–746, 2004.

[6] Brian Elliott Finley. Va systemimager. In *ALS'00: Proceedings of the 4th annual Linux Showcase & Conference*, pages 11–11, Berkeley, CA, USA, 2000. USENIX Association.

[7] A. Flissi and P. Merle. A generic deployment framework for grid computing and distributed applications. In *Proceedings of the International Conference on Grid Computing, High Performance and Distributed Applications (GADA'06)*, pages 1402–1411, nov 2006.

[8] FSF. Grand unified bootloader. http://www.gnu.org/software/grub/.

[9] Y. Georgiou, J. Leduc, B. Videau, J. Peyrard, and O. Richard. A tool for environment deployment in clusters and light grids. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 8 pp., april 2006.

[10] Michi Henning and Mark Spruiell. *Distributed Programming with Ice(Version 3.3.1b)*, 2009.

[11] Sun Microsystems Inc. *Java Remote Method Invocation (Java RMI)*, 2006.

[12] Intel. Preboot Execution Environment (PXE) Specification. Technical report, Intel Corp., 1999.

[13] Object Management Group. *The Common Object Request Broker: Architecture and Specification*, 3.0 edition, 2002.

[14] Changhua Sun, Le He, Qingbo Wang, and Ruth Willenborg. Simplifying service deployment with virtual appliances. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, volume 2, pages 265 –272, july 2008.

[15] Yaoxue Zhang and Yuezhi Zhou. 4VP: A Novel Meta OS Approach for Streaming Programs in Ubiquitous Computing. *Advanced Information Networking and Applications, International Conference on*, 0:394–403, 2007.