Proyecto Fin de Carrera en la empresa Luss Innovation S.L.: Instalación de R en un clúster desplegado en la nube

Pablo Baños López

5 de septiembre de 2009

Índice general

Índice de figuras

A mi madre y a mi padre,
a mis amigos y familia
y a la Universidad de Murcia.

Agradecimientos

A mi jefe, Javier Pérez-Griffo Callejón, por todo lo que ha puesto para que este proyecto saliese adelante, pues sin su ayuda no hubiera sido posible.

A mi tutor en este proyecto, Luis Pedro García González, y a mi profesor, Domingo Jiménez Cánovas, por la ayuda y apoyo que me han dado durante la ejecución del proyecto. También a los profesores y a la Universidad de Murcia por toda la formación que me han proporcionado a lo largo de estos años.

De manera especial, a mi madre, María del Carmen López Lax, por haberme ayudado a creer en mis sueños y a mi padre, Leonardo Baños Nortes, por mantenerme con los pies en el suelo.

Y por supuesto también a todos mis amigos, familiares y compañeros de clase, por el apoyo y los buenos momentos que me han animado a seguir cuando parecía difícil.

Muchas gracias a todos.

Capítulo 1

Introducción

Durante el curso 2008/2009 realicé una estancia de prácticas en la empresa Luss Innovation S.L. que posteriormente alargué para realizar el proyecto fin de carrera en esta empresa durante el verano. La empresa se dedicaba a la administración avanzada de sistemas, configuración de entornos de computación distribuidos, desarrollo de software de monitorización de sistemas, etc, y para ello hacíamos uso de tecnologías como la virtualización [6] y el cloud computing[3], a las que se alude en este documento.

Entre los proyectos que llevaba la empresa, se encontraba la instalación de las aplicaciones científicas en un entorno de computación distribuida. El entorno estadístico R [17], junto con un modulo llamado Rmpi que facilita el aprovechamiento de un entorno distribuido desde el anterior, era una de las aplicaciones científicas a instalar y de esta necesidad de la empresa surge el presente proyecto fin de carrera.

Además del interés de la empresa en disminuir el tiempo necesario para la instalación, preparando y comprobando cada paso del procedimiento de la misma, era necesario también realizar un estudio sobre ciertos parámetros de la instalación, concretamente nos interesaba saber que compilador y que implementación de MPI[16] proporcionarían un mayor rendimiento. Por estas dos condiciones se hacía necesaria la construcción de una maqueta del entorno final, esto es, un entorno de computación distribuido, en la que poder resolver estas dos cuestiones.

Esta maqueta tenía que presentar, o al menos simular, los elementos del entorno final, entre los que se incluyen un sistema operativo Linux, un sistema de ficheros distribuido, un servicio de gestión de cuentas de usuario centralizado y un sistema gestor de colas.

La empresa no contaba con un número de máquinas suficiente como para construir ese entorno en sus instalaciones, por lo que se decidió utilizar los servicios de infraestructura cloud de la plataforma Rightscale. De esta forma la empresa sólo ha pagado por el tiempo que ha utilizado la infraestructura y en función de sus características, al tiempo que se dotaba de escalabilidad

y automatización al despliegue de la maqueta.

En el momento de la entrega de este proyecto, la instalación en el sistema final no se ha producido, pero con independencia de la ejecución final de ese proyecto, el trabajo que se detalla en este documento ha supuesto para la empresa una base sobre la que trabajar en futuros proyectos de características similares.

Como será el caso de muchos proyectos fin de carrera realizados en empresa, podrá parecer que los resultados acádemicos alcanzados son algo escasos, pero se debe tener en cuenta el elevado volumen del trabajo de campo que, por tratarse de un proyecto fin de carrera en empresa, se ha tenido que realizar, y con el que he ganado una valiosa experiencia profesional.

1.1. Objetivos

Los dos objetivos principales planteados para este proyecto son:

1. **La construcción de una maqueta de un entorno de computación distribuido utilizando la infraestructura cloud que ofrece Rightscale.** Esta maqueta debe ser escalable y su despliegue debe estar lo más automatizado posible, con el proposito de facilitar su uso. Además, en ella deben estar configurados los elementos y herramientas que encontramos y utilizamos habitualmente en entornos de computación distribuida.

2. **La instalación en dicho entorno de la aplicación científica R y su módulo Rmpi.** Esta instalación tiene que ser automatizada al máximo, mediante la construcción de un paquete RPM, para simplificar el procedimiento de instalación en el sistema final y para permitir al cliente su ejecución en un futuro si fuera necesario. Relacionados con este objetivo se plantean los siguientes objetivos secundarios:

   - **Evaluación del rendimiento de R en función del compilador utilizado en su instalación.** El utilizar un compilador u otro en la instalación de la aplicación puede afectar al rendimiento de la misma. Los efectos de esta elección deben ser evaluados adecuadamente y tenidos en cuenta en el diseño de la solución final, y para cumplir con esto, realizaremos pruebas de rendimiento de la aplicación instalada utilizando dos compiladores distintos.
   - **Evaluación del rendimiento de R y su módulo Rmpi en función de la interfaz de intercambio de mensajes utilizada.** De la misma forma que la elección del compilador tiene efectos en el tiempo de computación de una aplicación científica, la elección, en la instalación de una aplicación científica, de una interfaz de intercambio de mensajes u otra puede tener efectos

sobre el tiempo de comunicación de la misma al ejecutarse en un entorno de computación distribuida. Por tanto, realizaremos pruebas de rendimiento de la aplicación instalada usando dos implementaciones de MPI distintas, LAM/MPI y OpenMPI,[16] y utilizaremos los resultados que obtengamos para la elección entre ellas en el diseño de una solución final.

1.2. Trabajo realizado y recursos utilizados

El trabajo realizado en la empresa, que analizamos en el tercer capítulo y en los anexos, se desglosa en las siguientes tareas:

1. **Formación en las tecnologías utilizadas.** Esta tarea se completó en 60 horas e incluye la búsqueda y análisis de información referente a los siguientes temas:
   - Virtualización
   - La plataforma cloud de Rightscale
   - Uso Avanzado de Linux
   - Sistemas de Almacenamiento Distribuido
   - Servicios de centralización de usuarios
   - Sistemas gestores de colas
   - Interfaces de intercambio de mensajes MPI
   - Construcción de paquetes RPM
   - Uso de R
2. **Preparación de la maqueta.** Para esta tarea fueron necesarias aproximadamente 80 horas de trabajo. El producto son los scripts analizados en los anexos C, D y E que realizan las tareas de:
   - Configuración automática de red
   - Configuración automática de acceso por SSH y actualización de la contraseña
   - Montaje automático de un disco persistente
   - Configuración automática de un servidor NFS
   - Configuración automática de un cliente NFS
   - Configuración automática de un servidor NIS
   - Configuración automática de un cliente NIS
   - Configuración automática de un cliente NTP
   - Instalación automática de paquetes y configuración automática de variables de entorno

- Instalación automática del gestor de colas Sun Grid Engine
- Instalación y configuración automáticas de LAM/MPI
- Instalación de Open MPI y configuración automática para su uso

3. **Preparación de paquetes RPM para R.** En esta tarea se invirtieron aproximadamente 30 horas y comprende:
   - Compilación de R con los compiladores GNU
   - Compilación de R con los compiladores de Intel
   - Preparación de los paquetes RPM
4. **Despliegue de la maqueta y ejecución de pruebas de rendimiento.** Si bien la ejecución de las pruebas requirió mas tiempo, al automatizarse esta, solamente fue necesario que invirtiera aproximadamente 10 horas.
5. **Documentación del proyecto.** La creación de este documento requirió aproximadamente 50 horas

La inversión de tiempo que he realizado asciende aproximadamente a un total de 230 horas.

Los recursos utilizados han sido

- Un ordenador de sobremesa en la oficina de la empresa y un ordenador portatil en casa para el acceso a la plataforma de Rightscale.
- Horas de cómputo de instancias *m1.small*[1]. Se detalla en la tabla 1.1.

Cuadro 1.1: Gasto en cómputo con instancias *m1.small* en Rightscale

Utilización (horas)	Precio hora	Gasto
652	\$0,10	\$6,52

- GB mes de almacenamiento persistente EBS[2]. Se detalla en la tabla 1.2.

Cuadro 1.2: Gasto en almacenamiento permanente EBS en Rightscale

Utilización (horas)	Precio GB Mes	Gasto
15,08	\$0,10	\$2,26

[1]Se explica en el capítulo 3.

[2]Se explica también en el capítulo 3.

1.3. Contenidos

Este documento está estructurado en cinco capítulos y cinco anexos.

El primer capítulo constituye una introducción del tema que nos ocupa y de los objetivos de este proyecto, así como una presentación de la estructura de este documento.

En el segundo capítulo realizamos un análisis de las tecnologías, herramientas y, en definitiva, elementos disponibles para la configuración de un entorno de computación distribuida.

En el tercer capítulo presentamos los elementos que constituyen la maqueta de un entorno distribuido y escalable, sobre la que trabajaremos y su instalación y configuración.

En el cuarto capítulo tratamos la metodología seguida para las evaluaciones de rendimiento en función de los criterios presentados en el apartado anterior así como los resultados obtenidos.

El quinto capítulo expone las conclusiones que hemos alcanzado y las posibles líneas de trabajo que quedan fuera del alcance de este proyecto, pero relacionadas con él.

El primer anexo es una introducción a la preparación de paquetes RPM. Se expone la preparación de los paquetes para conseguir una instalación automática de la aplicación científica elegida.

El segundo anexo consiste en una guía o manual de utilización de la plataforma de computación en la nube Rightscale.

El tercer anexo presenta la plantilla del nodo maestro en un entorno de la nube y los scripts asociados de preparación del entorno de computación a los que se alude en el tercer capítulo.

El cuarto anexo presenta la plantilla de los nodos esclavo en un entorno de la nube y los scripts asociados de preparación del entorno de computación a los que se alude en el tercer capítulo.

El quinto y último anexo presenta un par de scripts adicionales necesarios para la configuración y utilización del entorno de computación distribuido.

Capítulo 2

Estado del Arte

El esfuerzo computacional requerido para la resolución de los problemas que el científico se encuentra durante la realización de su labor investigadora es tan elevado hoy en día, que la investigación en muchos campos no se puede concebir sin el apoyo de potentes entornos de cómputo. Estos entornos evolucionan a pasos agigantados, siguiendo los avances que se producen en el campo de la informática, tanto a nivel de hardware como de software. Dos de las tecnologías que mayor repercusión están teniendo en los últimos años en la estructura de los entornos de computación, independientemente de la orientación académica, militar o comercial de su uso, son la virtualización[6] y la computación en la nube o cloud[3], y deben ser consideradas adecuadamente para, mediante su uso, dotar al entorno de computación de las ventajas que ofrecen.

Los entornos de cómputo pueden ser muy variados, pueden estar orientadas a la resolución de problemas concretos o ser de próposito general, pueden estar compuestos por una única máquina o por varias, etc. En cualquiera de los casos, no obstante, es tarea del ingeniero en informática su configuración para obtener de ellos el mayor rendimiento posible para el propósito de su uso. Para ello, se deberá prestar atención a cada elemento que conforma el entorno, eligiendo la configuración que mejor se ajuste al uso que se hará del mismo.

En este capítulo analizaremos el contexto en el que se desarrolla el proyecto y las herramientas disponibles para su ejecución.

2.1. Superordenadores y computación distribuida

La gran demanda de capacidad computacional del ser humano para múltiples fines (investigación, medicina, fines militares...), satisfecha por el continuo avance de la informática, tanto a nivel hardware como a nivel software, ha propiciado la aparición desde hace varios años de máquinas con una potencia computacional espectacular a las que se llama comúnmente

superordenadores.

Un superordenador es una máquina de propósito computacional con capacidades de cálculo muy por encima de las de los ordenadores personales. Su desarrollo se ha basado en las tres tecnologías que analizamos a continuación:

- **La tecnología de registros vectoriales de Seymour Cray.**[11] El uso de registros vectoriales permite la ejecución de una operación aritmética sobre múltiples datos en paralelo, lográndose así un alto grado de paralelismo al reducir el tiempo de ejecución de una operación vectorial al de una operación escalar.

- **Los procesadores masivamente paralelos o M.P.P. (Massively Parallel Processors).**[11] Se utilizan un gran número de procesadores coordinados muy estrechamente para la resolución eficiente de un mismo problema.

- **Las tecnologías de computación distribuida**.[11] Se utiliza un clúster de ordenadores: un gran número de computadoras, organizadas de diversas maneras e integradas en una infraestructura distribuida de telecomunicaciones para la resolución de problemas. Así problemas más grandes se resuelven utilizando ordenadores más simples que se encargan de resolver partes del problema más pequeñas, compartiendo recursos.

Las computadoras de registros vectoriales y los M.P.P. presentan dos grandes desventajas:

- El alto coste en su fabricación y en su mantenimiento tiende a limitar su uso a organizaciones con un poder adquisitivo extremadamente alto, esto es, organismos nacionales y supranacionales de fines militares o de investigación. Estas entidades harán uso de estas implementaciones con el objetivo de resolver problemas que requieren una gran capacidad de cálculo, como problemas de criptoanálisis, modelado molecular, predicción meteorológica y estadística, etc.

- Existe una estrecha relación entre la aplicación a ejecutar y sus datos y la máquina a construir. Esto conlleva que, a la hora de conseguir un rendimiento máximo, se deba adoptar un diseño orientado al propósito específico de la máquina, lo cual suele redundar en un peor rendimiento cuando se utiliza con propósito general, esto es, para la resolución de problemas distintos de aquellos a los que se orientó su diseño.

- Si bien nos encontramos con cierta escalabilidad en cuanto a capacidad de cálculo durante el diseño, esta escalabilidad esta limitada por el hecho de que una vez una máquina de estos dos tipos ha sido construida es difícilmente ampliable.

De esta forma, la computación distribuida se ha convertido en los últimos años en una línea en auge, hasta tal punto que de las 500 máquinas con mayor capacidad de cómputo en noviembre de 2008, el 82 % se clasifican como clusters [12]. Esta línea, sin embargo, también presenta desventajas entre las que quizá destaca una falta de estandarización provocada por su carácter abierto y su evolución constante.

2.2. Herramientas de virtualización

El término virtualización describe la separación de un recurso o petición de servicio de la implementación física subyacente de ese servicio. La virtualización es una tecnología que, dado un único recurso físico, nos permite disponer de múltiples recursos virtuales de esa clase más *pequeños*, cada uno independiente del resto. Permite además la agrupación de esos recursos de modo que queda oculta su naturaleza y límites ante quien los utiliza. Se trata de una tecnología bastante antigua, que ya se utilizaba en los años 60 para habilitar la compartición de los mainframes de las empresas, pero que en los últimos años ha recobrado interés por su capacidad para mejorar el aprovechamiento de los recursos y su eficiencia, así como para aumentar la escalabilidad y la flexibilidad en las tareas de administración. El recurso a virtualizar puede ser software o hardware, interesándonos más el segundo caso para el proyecto que nos ocupa. Podemos entonces, con esta tecnología, virtualizar máquinas completas, cada una de las cuales ejecutará su propio sistema operativo. [6]

El particionamiento de una máquina para que soporte la ejecución concurrente de múltiples sistemas operativos presenta varios retos. En primer lugar, las máquinas virtuales deben estar totalmente aisladas, pues no sería aceptable que la ejecución de una afecte al rendimiento de otra. Además, los sistemas operativos soportados deben ser variados, para permitir la ejecución sobre ellos de aplicaciones heterogéneas. Por último, si bien es lógico que se produzca cierta sobrecarga en el rendimiento total de la máquina real debido al uso de la virtualización, se debe cuidar que esta sea lo más reducida posible.

En los CPDs (Centros de Proceso de Datos), los entornos de servidores básicos basados en la arquitectura de Linux y x86, entornos de escalabilidad horizontal, constituyen una tendencia en auge. En estos entornos, el aumento de la potencia de procesamiento por servidor hace que cada vez tenga menos sentido un despliegue de aplicaciones que dedique a cada servicio una única máquina. En este sentido, la virtualización permite:

- aprovechar un mayor porcentaje de la capacidad de procesamiento de cada máquina, permitiendo la asignación de varias aplicaciones o servicios a una máquina sin perder la tolerancia a fallos,

- reducir los costes de hardware al disminuir el número de servidores físicos necesarios y, en consecuencia, el espacio físico necesario y el consumo de energía
- y facilitar las tareas de administración, permitiendo la gestión remota de los servidores virtuales y una asignación de recursos rápida y escalable que se adapte mejor a la demanda existente, pues, por ejemplo, ante un aumento de la demanda de procesamiento de una aplicación se podría automatizar el despliegue de servidores virtuales dedicados.

2.2.1. Paravirtualización y Virtualización completa

Podemos distinguir dos tipos de virtualización en función de la relación entre el sistema operativo del equipo físico anfitrión y el equipo virtual invitado:

- En los sistemas de virtualización completa [6] nos encontramos con una capa de virtualización que es la encargada de gestionar el acceso por parte de los sistemas operativos de los equipos virtuales a los diferentes recursos físicos, permitiendo la coexistencia simultánea de los primeros. La gran ventaja que presentan es que no es necesario realizar modificaciones en un sistema operativo para su ejecución en un equipo virtual invitado. Además, de acuerdo con los resultados de [5], en estos sistemas una instancia invitada está efectivamente protegida contra el mal comportamiento del resto. Por otro lado, es necesaria la simulación de las instrucciones privilegiadas por parte del equipo virtual, lo cual conlleva una mayor sobrecarga a costa del uso de esta tecnología. Otro inconveniente es la imposibilidad de compartición de recursos de forma cooperativa, puesto que cada sistema operativo invitado tendrá la percepción de que tiene dominio total y exclusivo de los recursos a su disposición.
- En los sistemas de paravirtualización [7], los sistemas operativos invitados reconocen la capa de virtualización. Esto permite una gran mejora en el rendimiento, puesto que las instrucciones privilegiadas y el código difícil de virtualizar de los sistemas operativos invitados son sustituidos por invocaciones a funcionalidades de la capa de virtualización, funcionalidades implementadas de manera más eficiente. El principal problema es de portabilidad, ya que un sistema operativo debe sufrir modificaciones de gran calibre para poder actuar como sistema operativo invitado en un sistema con paravirtualización.

2.2.2. VMware

VMware es una línea de productos creada por la filial de EMC Corporation, VMware Inc., la cual proporciona una parte importante del software

de virtualización disponible para ordenadores con arquitectura compatible con x86.[6] Entre las soluciones de esta línea se incluyen los productos VMware ESX Server, VMware Workstation, VMware Server y VMware Player, los cuales pueden funcionar sobre Windows y Linux, y, sobre la plataforma Mac OS X, corriendo sobre procesadores Intel y bajo el nombre de VMware Fusion.

VMware Player

Se trata de un producto gratuito que permite la ejecución de máquinas virtuales elaboradas con otros productos de la línea VMware, pero no su creación, la cual es posible utilizando productos mas avanzados como VMware Workstation.

VMware Server

Este producto era una versión de pago, pero a finales de 2006 fue liberado y desde entonces puede ser descargado y utilizado de forma gratuita. Esta versión ofrece una interfaz de administración intuitiva, que permite, a diferencia de la versión anterior, la creación de máquinas virtuales, ejecutándose sobre el sistema operativo anfitrión como virtualización completa y soportando arquitecturas de 64 bits.

VMware Workstation

VMware Workstation es un producto de virtualización completa comercial que permite la virtualización en todas las plataformas de la arquitectura x86. Se instala como una aplicación estándar que ejecuta las máquinas virtuales, restringiéndose el aprovechamiento de recursos.

VMware ESX Server

Esta versión de pago es un sistema complejo de paravirtualización que se ejecuta como sistema operativo dedicado en exclusiva a la gestión y administración de las máquinas virtuales alojadas, no necesitando por ello un sistema operativo anfitrión extra.

Está pensado para entornos grandes, donde permite la centralización y virtualización de los servidores, no siendo compatible con gran cantidad de dispositivos hardware de uso doméstico y necesitando hardware específico para su funcionamiento.

2.2.3. XEN

Ian Pratt, de la Universidad de Cambridge, desarrolló Xen en 2003 como software de virtualización de código abierto, con objeto de poder ejecutar

instancias de sistemas operativos con todas sus características y de forma totalmente funcional en un equipo sencillo. Más tarde, Ian Pratt fundó la empresa XenSource, Inc. que se dedica a dar soporte a la versión de Xen de código abierto y que también distribuye versiones comerciales de Xen para empresas. En 2007 Citrix Systems compró Xen Source Inc.

En la actualidad disponemos de tres soluciones comerciales orientadas a las necesidades de las empresas y una de distribución libre [6].

XenServer Express Edition

XenServer Express Edition es una versión comercial gratuita con la funcionalidad restringida. El producto incluye un hipervisor así como herramientas de monitorización de máquinas virtuales.

XenServer Standard Edition

Se trata de una versión comercial de pago que ofrece más funcionalidad que la solución Express y se corresponde con la gama media del software comercial de virtualización de Citrix.

XenServer Enterprise Edition

Esta solución constituye la gama alta de la línea de virtualización ofrecida por Citrix, contando con muchas funcionalidades y enfocada a la virtualización en centros de datos.

Xen Source

Xen Source es la versión de código abierto desarrollada a partir de la idea original de Ian Pratt y cuenta con una gran comunidad de desarrolladores y usuarios. Actualmente se encuentra en la versión 3.4, soporta paravirtualización con rendimiento cercano al de una máquina física e incluso permite la migración de máquinas virtuales en tiempo real.

2.3. Computación Cloud

La Computación en la Nube o Cloud Computing consiste en que proveedores ofrezcan recursos o servicios de uso a través de Internet a sus clientes, permitiendo la separación de la implementación de un recurso o servicio informático del entorno del usuario o consumidor del mismo, que lo utiliza a través de la red sin conocer detalles de lo que sucede en la infraestructura del proveedor [3].

2.3.1. Capas de servicios Cloud

Podemos distinguir diferentes capas de servicios Cloud:

- Software como Servicio o *Software as a Service* (SaaS). El recurso o servicio ofrecido es una aplicación completa estándar desarrollada en algunos casos y mantenida por el proveedor mismo y con la que da servicio a multitud de clientes a través de la red, sin necesidad de que estos instalen ningún software adicional. Un ejemplo de este tipo de aplicaciones sería la suite ofimática de Google (GoogleDocs).

- Plataforma como Servicio o *Platform as a Service* (PaaS). El proveedor proporciona como servicio una plataforma que permite el desarrollo de software a través de la red y que elimina la necesidad de que el usuario gestione bases de datos, servidores, redes y herramientas de desarrollo. Para ello, el proveedor debe escalar los recursos en función de los requerimientos de la plataforma para obtener el mejor rendimiento posible. Un ejemplo es la plataforma Google Apps Engine.

- Infraestructura como Servicio o *Infrastructure as a Service* (IaaS), que es la capa de servicios Cloud de la que haremos uso en este proyecto. El proveedor ofrece como servicio la infraestructura, esto es, un entorno de cómputo, donde el cliente paga sólo por lo que usa en función del espacio en disco utilizado, tiempo de CPU, datos transferidos por la red, etc... De esta forma una empresa puede delegar la gestion de sus necesidades de IT, reduciendo el coste de la misma. Además, en muchos casos, estos servicios ofrecen una escalabilidad automática o semiautomática que permite la ampliación de la infraestructura contratada ante picos en la necesidad de cómputo. Un ejemplo de este servicio es el Amazon Web Service (AWS) de Amazon.

2.3.2. Virtualización en el Cloud

La virtualización es esencial en el desarrollo de un entorno de computación cloud [3], especialmente en el caso de servicios IaaS, pues permite la creación de dispositivos virtuales que constituirán el servicio de infraestructura ofrecido. El uso de esta tecnología en la implementación de servicios cloud otorga los siguientes beneficios, que son los resposables directos del proceso de maduración de los servicios cloud que observamos hoy en día:

- Rápida ampliación de recursos para el servicio contratado.

- Gran reducción de los costes de espacio y consumo.

- Administración del sistema centralizada y simplificada.

- Facilidad y agilidad en la creación de entornos en los que realizar pruebas de modificaciones del entorno, sin que estas afecten a la versión del sistema en funcionamiento.
- Protección de una máquina respecto de los fallos de las demás, con independencia de que se ejecuten en el mismo servidor físico.

2.4. El sistema operativo GNU/Linux

Linux es un término genérico utilizado para referirse a sistemas operativos multiproceso y multiusuario, parecidos a Unix y basados en el núcleo Linux [15].

En 1991 Linus Torvalds empezó a trabajar en un nucleo de sistema operativo basándose en el sistema MINIX de Tanembaum que más adelante acabaría constituyendo lo que hoy conocemos como el nucleo o kernel Linux. El proyecto GNU, iniciado en 1983 por Richard Stallman, estaba desarrollando un sistema operativo UNIX completo libre y cuando la primera versión de Linux fue liberada decidieron adoptarla como núcleo.

Hoy en día las distribuciones de Linux se utilizan en diversos ámbitos, desde sistemas embebidos a supercomputadores, donde, en Junio de 2009, 443 de los 500 sistemas del Top 500 (el 88,6 %) ejecutan una distribución Linux. [12]

Algunas de las ventajas que presentan estos sistemas operativos son, su amplia extensión, que son *open-source* y en muchos casos gratuitos, además de que cuentan con una gran comunidad de usuarios que colabora activamente en su desarrollo.

A la hora de diferenciar entre una distribución y otra, encontraremos distribuciones con sistema de paquetes RPM de Red Hat o sistemas de paquetes Debian, así como distribuciones con diferentes organizaciones de ficheros de configuración.

2.4.1. Suse

SUSE Linux es una distribución Linux muy conocida a nivel mundial. Su nombre, SuSE, es un acrónimo del alemán "Software und Systementwicklung", que significa Desarrollo de Sistemas y de Software, y que era parte del nombre de la compañía que lo desarrollo (SuSE Linux). En enero de 2004, la multinacional estadounidense Novell adquirió SuSE LINUX y en 2005 anunció la liberación de la distribución SuSE Linux bajo el nombre openSUSE encargando a la comunidad su desarrollo [15].

SUSE usa el sistemas de paquetes RPM de Red Hat, aunque no guarda mucha relación con esta distribución. Además incluye un programa para instalación y administración del sistema, YaST2, que permite su actualización,

la configuración de la red y el cortafuegos, la administración de usuarios entre otras, integradas en una sola interfaz amigable. La realización de estas tareas mediante el método tradicional de manipulación de ficheros es complicada, pues esta distribución varía mucho del resto y es dificil en ocasiones localizar el/los fichero/s donde cambiar un parámetro. Además el soporte es de pago, por lo que no siempre es una buena opción.

2.4.2. Ubuntu

Ubuntu es una distribución Linux orientada tradicionalmente sobretodo a ordenadores personales, aunque, especialmente en los últimos tiempos con una línea del producto específica, también a servidores. Se trata de una distribución muy extendida a nivel mundial, que se concentra en ofrecer facilidad y libertad de uso, instalación fluida y lanzamientos de nuevas versiones regulares (cada 6 meses). Su desarrollo está dirigido por Canonical Ltd., una empresa que Mark Shuttleworth fundó y financia [15].

Ubuntu soporta oficialmente las arquitecturas de hardware Intel x86, AMD64 y desde abril de 2009, se ofrece también soporte oficial para procesadores ARM. Además, de forma extraoficial, ha sido portada a PowerPC, HP PA-RISC, SPARC, IA-64 y Playstation 3.

Los desarrolladores de Ubuntu basan gran parte de su trabajo en otros proyectos de software libre, especialmente en los de la comunidad Debian. La distribución en sí misma está basada en la distribución Debian GNU/Linux, con la que comparte el sistema de paquetes y gran parte de la organización de ficheros de configuración.

2.4.3. CentOS

CentOS 5 es una distribución del sistema operativo Linux que está basada en Red Hat y es muy utilizada en entornos de computación distribuida donde las herramientas de clustering tienen una mayor importancia por su papel frente al resto de herramientas, principalmente porque CentOS incorpora de forma nativa muchas aplicaciones dedicadas al agrupamiento de servidores [10].

Aunque estas aplicaciones pueden ser instaladas en cualquier distribución Linux, la facilidad introducida en CentOS para su uso ha ocasionado que esta distribución se haya extendido por centros de computación y otros entornos donde se desea mantener agrupaciones de servidores.

Podemos adquirir la distribución CentOS 5 vía web, ftp o torrent de web de CentOS (http://www.centos.org/) y además, podemos encontrar multitud de mirrors de los que descargar este software.

2.5. Sistemas de ficheros distribuidos

Cuando se piensa en el sistema de ficheros que utiliza un cluster, se tiende a imaginar sistemas de ficheros complicados y muy distintos de los utilizados en sistemas de propósito general. Sin embargo, esto no tiene por qué ser así y a la hora de elegir un sistema de ficheros para nuestro entorno de computación distribuida se nos presenta un amplio abanico que comprende desde sistemas de ficheros distribuidos ordinarios a sistemas de ficheros más complejos, como los paralelos. La idoneidad de uno u otro dependerá del uso y la exigencia que se vaya a dar a la opción elegida.

A continuación analizamos las características de un sistema de ficheros distribuido estándar (NFS) y un sistema de ficheros distribuido paralelo (Lustre).[4]

2.5.1. NFS

Network File System o NFS es un protocolo establecido en el nivel de aplicación correspondiente al modelo OSI que se utiliza tradicionalmente como sistema de archivos distribuidos en redes de área local. Desarrollado en 1984 por la empresa Sun Microsystems, es un protocolo muy portable ya que presenta independencia de la máquina, sistema operativo e incluso protocolo de transporte y se incluye por defecto en los sistemas operativos UNIX. Como sistema de ficheros distribuido permite a varios sistemas conectados a una misma red el acceso a ficheros remotos como si se tratara de ficheros locales.

NFS utiliza un esquema cliente-servidor, y esta desarrollado utilizando RPC. Los clientes acceden de forma remota a los datos de los ficheros que se encuentran almacenados en el servidor, necesitando menor espacio de disco, al estar los datos centralizados en un servidor y evitando así la necesidad de replicación de datos. Uno de sus usos habituales es la centralización del espacio dedicado a los datos de usuario dentro de organizaciones.

Las operaciones sobre ficheros en NFS son generalmente de naturaleza síncrona, significando esto que la operación sólo termina cuando se ha completado su trabajo asociado, ya sea la lectura o la escritura física de los datos en disco, lo cual es una forma sencilla de garantizar la integridad de los ficheros.

Hoy en día podemos encontrar tres versiones de NFS en uso:

- La versión 2, que fue desarrollada originalmente sobre UDP, es la más antigua y es soportada por la mayoría de sistemas operativos.

- La versión 3, que soporta arquitecturas de 64 bits, permite, además, el manejo de ficheros de más de 4 GB, escrituras asíncronas que tienen como objetivo la mejora del rendimiento y, en algunas implementaciones, soporte de TCP como protocolo de transporte.

- La versión 4, en la que se ve la influencia de otros sistemas de ficheros como AFS y CIFS, presenta cambios orientados a la mejora ulterior del rendimiento y el soporte de tecnologías de seguridad como Kerberos y ACLs.

La principal ventaja que podemos encontrar en este sistema de ficheros distribuido es su facilidad de instalación y configuración, que deriva de su calidad de estandar entre los sistemas operativos UNIX. Por otro lado, bajo condiciones muy exigentes, el rendimiento que ofrece es mucho inferior al de sistemas de ficheros distribuidos paralelos.

2.5.2. Lustre

Lustre es un sistema de ficheros distribuido paralelo desarrollado y mantenido por Sun Microsystems. Fue concebido para entornos de computación distribuidos y se trata de una opción muy robusta, con alta disponibilidad que además está disponible bajo licencia GNU GPL.

Sin embargo, la mejor característica de este sistema de almacenamiento es, quizás, su capacidad para escalar correctamente desde sistemas con unas decenas de nodos y terabytes de información, hasta grandes entornos de computación distribuida que cuentan con miles de nodos y cuya capacidad de almacenamiento se mide en petabytes. Es por esto por lo que se le considera en ocasiones el mejor sistema de ficheros distribuido paralelo y en abril de 2007, el 50 % de los 30 mayores superordenadores del mundo hacían uso de él.[12]

2.6. Sistemas de gestión de usuarios

En un entorno de computación distribuida todos los nodos deben tener definidas y sincronizadas las cuentas de usuario, pues de lo contrario encontraremos problemas a la hora de ejecutar sobre él un trabajo paralelo. Si el número de nodos es pequeño sería posible realizar una administración de usuarios nodo a nodo, repitiendo en cada uno de ellos las mismas tareas de gestión, pero esta práctica, ademas de requerir tiempo y ser tediosa, es propensa a fallos. Además, en los casos en que el número de nodos es muy grande, esto último se hace imposible en la práctica. La solución es la centralización de las cuentas de usuario del entorno en un servicio, y NIS y LDAP son dos opciones muy extendidas.

2.6.1. NIS

Para solucionar el problema que hemos comentado, Sun Microsystems desarrollo Network Information Service o NIS, que, basicamente, es una base

de datos distribuida que sustituye copias de distintos ficheros de configuración UNIX (/etc/hosts, /etc/passwd, /etc/group, ...) que tiene cada nodo por un único fichero central. [13]

La filosofía de trabajo es bajo el modelo cliente servidor: un equipo, que actúa como servidor, mantiene las copias originales de los ficheros de datos de configuración, llamados mapas, donde se realizan los cambios, y distribuye la información de esa copia entre el resto de equipos, que adoptan el rol de clientes, cuando estos la requieren. Los mapas, con el propósito de aumentar la eficiencia, no se guardan en ASCII, si no en un formato binario llamado DBM.

Es posible que haya mas de un servidor, en cuyo caso solo uno de ellos, el maestro será el poseedor de los mapas, mientras que el resto, los esclavos, se limitaran a responder a los clientes a partir de los mapas proporcionados por el servidor maestro.

NIS se ha convertido en un estandar de facto en sistemas UNIX como servicio de centralización de usuarios por su simplicidad y facilidad de configuración.

2.6.2. LDAP

En 1988 OSI definió el protocolo de servicio de directorios X.500, que propone una organización de las entradas de un directorio en un espacio de nombres jerárquico. Junto con esta organización X.500 también incluye capacidades de búsqueda muy potentes para la consulta de la información almacenada en un directorio, pero tenía un fallo: el protocolo de interacción entre el cliente y el servidor de directorios, Directory Access Protocol (DAP) requería todas las capas del modelo OSI. Esto propició el desarrollo de un protocolo más ligero, que usaba TCP/IP y que terminaría siendo llamado Lightweight Directory Access Protocol, o LDAP. Su tercera y, hasta el momento, última versión, fue propuesta en el RFC 2251. [9]

Los RFC en los que se han definido las distintas versiones de LDAP prestan atención a la definición de un protocolo de red y a la estructura y organización de los directorios, definida por cuatro modelos que comentaremos a continuación:

- El modelo de información de LDAP define los datos que se almacenan en un directorio. La unidad basica de información son las entradas que se organizan jerárquicamente y están compuestas por atributos que describen propiedades del objeto. Los atributos pueden ser uni o multivaluados así como opcionales u obligatorios. Así, en un esquema, podemos definir los atributos que tendrán las entradas de nuestro directorio LDAP.

- El modelo de nombres de LDAP define la organización de los datos y su referencia a ellos en el directorio. La organización de las entradas es

siempre en un árbol invertido y las entradas se referencian utilizando los atributos identificadores de las entradas desde la hoja a la raíz.

- El modelo funcional de LDAP define las operaciones que permiten la consulta y actualización de los datos del directorio:
  - Las operaciones de consulta en el directorio, que nos permiten la realización de búsquedas y extracciones de datos del mismo,
  - Las operaciones de actualización en el directorio, que nos permiten la adición, el borrado y la edición de las entradas en un directorio,
  - Las operaciones de autenticación y control, que permiten la identificación ante el servidor de los clientes que desean hacer uso del servicio con objeto de controlar aspectos de la sesión, acceso a entradas, etc.
- El modelo de seguridad de LDAP define la protección de la información ante accesos no autorizados al directorio.

La configuración de LDAP en un sistema Linux, comparada con la de NIS, es mucho mas complicada y el cluster de este proyecto no va a hacer uso de la gran ventaja que LDAP puede ofrecer como servicio centralizado de usuarios, la capacidad de guardar otra información además de la pertinente a la cuenta de usuario UNIX clásica.

2.7. Sistemas gestores de colas

Conectando una serie de ordenadores en red dispondremos de un sistema que podríamos llamar cluster, pero para aprovechar los recursos invertidos de forma óptima es necesario instalar un software o middleware que distribuya adecuadamente los trabajos a realizar entre los recursos de cluster. A este software se le conoce como software o middleware de cluster o sistema gestor de colas.[4]

2.7.1. PBS, OpenPBS y Torque

Portable Batch System o PBS es un sistema de gestión de recursos y trabajos batch que fue desarrollado para la NASA en los años 90, por la empresa MJR, siguiendo el estándar POSIX 1300.2d Batch Environment Standard. MJR fue adquirida por Veridian, que a su vez fue adquirida por Altair Engineering que distribuye PBS Professional comercialmente y una versión *open source* sin soporte que recibe el nombre de OpenPBS.[8]

Torque es una versión derivada de OpenPBS desarrollada y mantenida activamente por la empresa Cluster Resources Inc.

Por su fiabilidad, PBS y derivados han venido siendo usados en multitud de clusters, convirtiendose por ello en un estándar de facto en Linux.

Estos productos, como sistemas gestores de recursos, aceptan trabajos batch compuestos por un guión Shell que contiene atributos de control. Una vez aceptados los trabajos, los preservan y protegen hasta que están listos para la ejecución. Entonces los ejecutan y proveen al usuario con la salida de los mismos.

El sistema se puede considerar constituido por los siguientes componentes que interactúan en la figura 2.1:

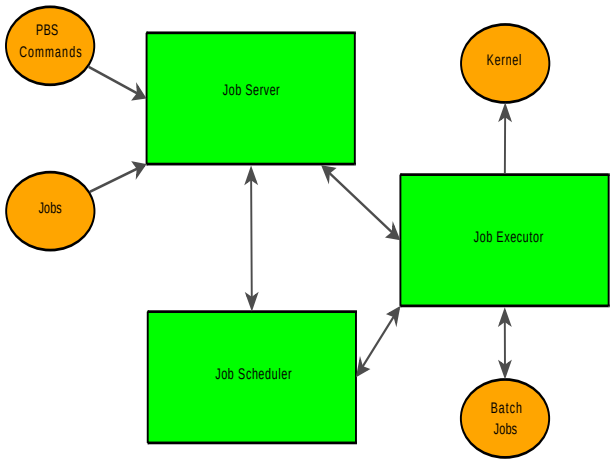

Figura 2.1: Componentes de PBS

- Commands: Este componente proporciona una serie de comandos en línea junto con una interfaz gráfica conformes al estándar POSIX 1003.2d que permiten el envío, monitorización, modificación y borrado de trabajos.
- Job Server: Se trata del componente central del sistema. Es un servidor con el que se comunican los commands y demonios para la obtención de servicios básicos como la creación y envío de un trabajo, su modificación y su ejecución.
- Job Executor: Es un demonio encargado de iniciar la ejecución de los trabajos enviados. Reproducirá para ello la sesión del usuario propietario del trabajo y devolverá, cuando el trabajo acabe, su salida al usuario.
- Job Scheduler: Se trata de un demonio que controla las politicas de ejecución de los trabajos, esto es, determina cuando y donde se ejecutará un determinado trabajo en función de variados criterios como la carga del sistema o el usuario propietario del mismo.

2.7.2. Sun Grid Engine

Sun Grid Engine (SGE) es un sistema de gestor de colas que ha sido desarrollado por Sun Microsystems.

En el año 2000, la empresa Gridware, especializada en productos de gestión de recursos de computación, fue adquirida por Sun con el objetivo de lanzar una versión libre de sus productos para las plataformas Solaris y Linux. En 2001, el código del producto original se libera y se adopta como modelo de desarrollo el modelo open source, lo cual propicia la portabilidad a otras plataformas UNIX. Así nace el sistema gestor de colas Sun Grid Engine (SGE).

Este producto de Sun cuenta con multitud de ventajas. Para empezar ofrece una gran facilidad de administración y de uso, lo cual ha favorecido su extensión en entornos de computación donde muchos usuarios no son ingenieros en informática. Además, en el plano técnico, nos encontramos con un sistema de gestión de colas altamente configurable y escalable, lo cual también ha contribuido a su extensión entre los entornos de computación distribuida mas potentes del mundo.

SGE puede aceptar, planificar y ejecutar un gran número de trabajos como sistema gestor de colas escalable que es. Pero además también puede gestionar y planificar recursos distribuidos tales como procesadores, memoria, almacenamiento secundario e incluso licencias de software.

Estructuralmente, un cluster SGE se compone de un maestro (master host) y uno o varios esclavos (execution hosts). Para proporcionar alta disponibilidad pueden, adicionalmente, configurarse uno o varios maestros de backup que tomarán el control si el maestro fallase. Para trabajar con los requisitos de los trabajos de los usuario, SGE, al igual que otros softwares de clúster, trabaja con guiones Shell, lo cual no impide que pueda trabajar directamente con binarios e incluso trabajos interactivos.

Podemos afirmar, que probablemente SGE es el producto mas completo de este tipo disponible en el mercado, en base a las características que hemos analizado en este apartado. Además, debemos tener en cuenta el gran tamaño de la comunidad de desarrolladores y administradores que lo mantienen y usa, así como el respaldo de una empresa como Sun Microsystems.

2.8. Interfaces de intercambio de mensajes

Las librerías de paso de mensajes o interfaces de intercambio de mensajes permiten la comunicación entre los diferentes nodos de cómputo durante la ejecución de una aplicación, aportando su uso principalmente dos ventajas:

- Facilita la programación de una aplicación paralela por paso de mensajes al permitir al programador tratar la comunicación entre los diferentes procesos a un nivel más alto.

- Permite la independencia de la aplicación con respecto al número y tipo de nodos, no importando donde se ejecutará que tarea y facilitando así su ejecución en un clúster con sistema gestor de colas.

Sin embargo, la lógica de los mensajes en las aplicaciones siempre estará en manos del programador.

A continuación examinamos dos de los estándares más utilizados:

2.8.1. PVM

PVM es una librería de intercambio de mensajes *open source* creada por el laboratorio Oak Ridgre en 1989, que permite la ejecución de aplicaciones paralelas sobre máquinas distribuidas y heterogéneas.[8] Presenta las siguiente características:

- El usuario debe definir un conjunto de nodos que se dedicarán a cómputo.
- El usuario selecciona de entre los anteriores el conjunto de nodos donde se ejecutará la aplicación paralela, pudiendo ser alterado este conjunto durante la ejecución, permitiendo tolerancia a fallos.
- Se pueden aprovechar las características especiales de cada nodo para, por ejemplo, ejecutar determinados cálculos, esto es, se permite un acceso semitransparente al hardware de cada nodo.
- Supone un modelo basado en procesos y pasos de mensajes explícitos.
- Soporte de distintos lenguajes: C, C++ y Fortran.

2.8.2. Standard MPI

Se trata de una interfaz de intercambio de mensajes definida por un comité de vendedores, implementadores y usuarios. que se ha convertido en el estándar de facto del modelo de paso de mensajes en entornos de computación distribuida tanto en el mundo empresarial como en el de la investigación.[16]

En 1994 apareció la primera versión de MPI, que fue seguida, en 1997 por la versión MPI-2 que incorporó funciones de entrada salida paralela entre otras mejoras significativas y que sigue hoy en día vigente. Podemos encontrar bindings de MPI para multitud de lenguajes, aunque los los más difundidos son aquellos para Fortran, C, C++, Java o Python.

Una implementación del estándar MPI o MPI-2 consiste en una librería que ofrece un conjunto de llamadas que constituyen la interfaz de intercambio de mensajes. A través de estas llamadas tenemos a nuestra disposición cualquier acción que estimemos necesaria para la gestión de las comunicaciones en una aplicación. Podemos clasificar las llamadas en:

- Llamadas para la inicialización, gestión y finalización de las comunicaciones.
- Llamadas para la transferencia de datos entre dos procesos.
- Llamadas para la transferencia de datos entre varios procesos.
- Llamadas de definición de tipos de datos.

Las razones del éxito de MPI, además del gran esfuerzo que se hizo por parte del comité que lo definió, son las bondades de su diseño:

- Portabilidad: MPI puede ser usado en entornos muy diversos, que cuenten con nodos uni o multiprocesador, con heterogeneidad en los nodos, con diferentes tipos de redes, etc
- Buenas prestaciones, cualidad muy valorada en entornos de computación donde la inversión en recursos ha sido valiosa.
- Amplia funcionalidad ofrecida, que facilita la programación de cualquier tipo de intercambio de mensajes.

Otra de las razones que han propiciado su extensión es la existencia de implementaciones *open source*, entre las que destacan:

- MPICH fue la primera implementación del estándar MPI y fue llevada a cabo por el Argonne National Laboratory y la Universidad del Estado de Mississippi. La versión MPICH2 implementa la segunda versión del estándar MPI, MPI-2.
- LAM/MPI es una implementación desarrollada por el Ohio Supercomputing Center y mantenida en la actualidad por la Universidad de Indiana.
- OpenMPI es una versión derivada de LAM/MPI entre otras.

En este proyecto realizaremos una comparación de rendimiento de las últimas dos, que comentamos a continuación:

LAM/MPI

LAM (Local Area Multicomputer) comprende un sistema de desarrollo y un entorno de programación MPI para maquinas heterogéneas en una red. [15]

LAM/MPI, además de una implementación MPI, ofrece control e inicio de procesos basados en demonios, por lo que facilita la operación con trabajos paralelos en un cluster.

La implementación MPI de LAM/MPI puede utilizar tanto memoria compartida como tecnologías de red comunes en un cluster como TCP/IP, Myrinet o Infiniband.

Open MPI

Open MPI es un proyecto tiene por objetivo el desarrollo de la mejor librería de intercambio de mensajes posible. Es utilizada por muchos de los sistemas del la lista TOP500.

OpenMPI representa la fusión de tres conocidas implementaciones de MPI:

- FT-MPI de la Universidad de Tennessee,
- LA-MPI del laboratorio nacional estadounidense Los Alamos y
- LAM/MPI, comentado anteriormente.

con la contribución del equipo PACX-MPI de la Universidad de Stuttgart. Estas cuatro instituciones son los miembros fundadores del equipo de desarrollo de Open MPI.[15]

Las implementaciones de MPI en las que se basa fueron seleccionadas por los desarrolladores por destacar en algun aspecto.

El código de Open MPI se divide en tres grandes modulos:

- OMPI que comprende el código con la implementación MPI,
- ORTE u Open Run-Time Environment, un entorno de ejecución y
- OPAL u Open Portable Access Layer, utilizado por los anteriores.

2.9. Compiladores de C y Fortran

El compilador con que se instala una aplicación científica puede afectar al rendimiento final de la misma. Esto es así debido a que es el compilador el responsable de generar ejecutables que se adapten lo mejor posible a los recursos de la máquina y que los aprovechen al máximo.

La suite de compiladores Intel® Compiler Suite es una opción de pago que ofrece muy buenos resultados sobre las plataformas del fabricante, que están muy extendidas y son las que encontraremos en la mayoría de los entornos de computación distribuida.

Los compiladores de GNU (gcc, g++, gfort, ...) constituyen una opción *open source* que en general producen ejecutables peores en rendimiento que los generados por los compiladores de Intel.

Como objetivo de este proyecto, realizaremos una comparación de los rendimientos ofrecidos por los ejecutables generados por ambos compiladores en la instalación de una aplicación científica.

2.10. El entorno estadístico R

R es un paquete estadístico *open source* que funciona bajo sistemas GNU/Linux entre otros y que ofrece un entorno informático estadístico con herramientas de análisis de datos y generación de gráficas. En este entorno se opera utilizando un lenguaje, que recibe el mismo nombre que el paquete, mediante el cual se hace uso de los distintos módulos estadísticos que la herramienta incluye o que sobre ella se pueden instalar.[17]

Un módulo *open source* que podemos instalar en el entorno ofrecido por R es Rmpi. Este modulo nos permite de forma sencilla utilizar las funcionalidad de la implementacion de MPI que tengamos instalada en nuestro cluster, posibilitando, por ejemplo, la delegación de calculos a otros nodos. Se trata, en definitiva, de un módulo que permite la utilización de la totalidad del cluster desde el entorno de R. [1]

Las razones por las que hemos elegido R con Rmpi como aplicación científica a instalar en nuestro cluster son las siguientes

1. Son gratuitas y están extendidas.
2. Al ser *open source* tenemos a nuestra disposición los fuentes para compilarlos.
3. Podemos elegir el problema a paralelizar y diseñar una solución a medida, sabiendo siempre qué estamos ejecutando y para qué.

Capítulo 3

Presentación del entorno de computación distribuida

3.1. El servicio de computación cloud de Rightscale

Para realizar un estudio adecuado del rendimiento de la aplicación a instalar en base a los compiladores utilizados y las librerías de intercambio de mensajes necesitamos un entorno distribuido fácilmente escalable: queremos hacer pruebas en entornos que tengan un nodo maestro y un número variable de nodos esclavos que podamos variar de forma sencilla.

Para su implementación hemos hecho uso del servicio de computación cloud de pago proporcionado por RightScale. RightScale proporciona un acceso a los servicios de Amazon Web Service construidos sobre la tecnología de virtualización Xen y enriquecido con una amplia gama de herramientas para la configuración a medida de las máquinas virtuales. Utilizando la interfaz web de su plataforma, podemos preparar plantillas para equipos virtuales de diversas características, seleccionar una imagen de entre una amplia gama de sistemas operativos que incluye CentOS e indicar scripts a ejecutar en el arranque del sistema operativo virtual (scripts de arranque) o cuando nosotros pulsemos un botón (scripts operacionales). Así mismo, el entorno nos ofrece la posibilidad de crear unidades de almacenamiento permanentes, utilizando el servicio EBS, en las que almacenar datos que no queramos perder al apagar las máquinas, pues tras su apagado estas se borran. En una de estas unidades realizaremos la instalación de aplicaciones sensibles y el almacenamiento de datos que queramos conservar, en particular los resultados de las pruebas ejecutadas. En cuanto al coste por el despliegue y el uso del entorno, al igual que este último, es escalable y está en función de lo que se usa, como se puede ver en la tabla 3.1.

Realizaremos las pruebas de rendimiento en entornos donde todos los nodos, tanto maestro como esclavos, serán instancias de tipo *m1.small*, esto es, nodos monocore. Realizaremos las pruebas siempre con un nodo maestro

Cuadro 3.1: Características de los tipos de instancias en RightScale

Instancia	Cores	Memoria	Precio Hora
m1.small	1	1,7 GB	$0,10
m1.large	2	7,5 GB	$0,40
m1.xlarge	4	15 GB	$0,80
c1.medium	2	1,7 GB	$0,20
c1.xlarge	8	7 GB	$0,80

y variando el número de nodos esclavo de 1 a 16 en potencias de 2.

3.2. Tecnologías y herramientas utilizadas

Como sistema operativo de las máquinas de nuestro clúster hemos elegido CentOS 5, por estar muy extendido su uso entre los clusters dedicados a la investigación y por estar disponible como imagen para las máquinas de Rightscale bajo el nombre *RightImage CentOS5_0V4_0_1*.

Si bien Lustre es una opción como sistema de ficheros distribuido que ofrece mayor escalabilidad y, en definitiva, responde mejor cuando el número de nodos es muy alto, este no va a ser nuestro caso, pues el entorno sobre el que realizaremos las evaluaciones no superara los 16 nodos y NFS, que es extremadamente sencillo de instalar y configurar, podrá responder adecuadamente ante la demanda de entrada/salida.

Como servicio para la centralización de usuarios hemos escogido NIS porque aunque LDAP ofrece una mayor funcionalidad, no la vamos a utilizar y, por tanto, nos compensa mas configurar NIS por ser el procedimiento mucho más sencillo.

Como sistema gestor de colas hemos elegido Sun Grid Engine, por ser el producto más completo y ser gratuito y fácil de instalar.

También haremos uso de las versiones 1.3.3 y 7.1.2 de las interfaces de intercambio de mensajes OpenMPI y LAM/MPI respectivamente, de los compiladores GNU e Intel para C y Fortran y de las versiones 2.8.0 y 0.5-7, de la herramienta R y su módulo Rmpi respectivamente.

3.3. El nodo maestro

El nodo maestro hace de servidor NIS y NFS, teniendo montada en el directorio /opt la unidad de almacenamiento permanente. También es el maestro para el sistema gestor de colas y desde el se ejecutarán los trabajos con los cuales evaluaremos el rendimiento de R bajo los criterios ya mencionados.

Para su configuración hemos creado una plantilla, que hemos llamado *PFC: Plantilla: Nodo Maestro*, indicando como instancia *m1.small* y como imagen *RightImage CentOS5_0V4_0_1*. A continuación hemos creado unos guiones shell con los que automatizamos una serie de tareas de administración que deben realizarse cuando el maestro arranca o cuando el clúster se reconfigura (cuando se han añadido nuevos nodos, por ejemplo).

Cuando arranca el maestro deben realizarse las siguientes tareas, de las cuales se encargan los siguientes scripts de arranque de la plantilla creada para el nodo maestro:

1. Debe configurarse el fichero /etc/hosts para que contenga el listado de direcciones IP y los nombres de cada máquina del entorno.
2. Debe habilitarse el acceso por SSH.
3. Debe asignar una contraseña al superusuario.
4. Debe montarse en /opt la unidad de almacenamiento permanente.
5. Debe configurarse un servidor NFS para que exporte la carpeta /opt a todos los esclavos.
6. Debe configurarse un servidor NIS al que se conectarán los esclavos para que dispongamos de un directorio de usuarios idéntico en todo el entorno, requisito para la instalación de un sistema de colas.
7. Debe configurarse un cliente NTP, pues es importante que todas las máquinas del entorno estén sincronizadas.

Una vez arrancados todos los esclavos, se deberá ejecutar el script operacional de instalación de Sun Grid Engine y una vez este ha completado su tarea podremos lanzar alguno del resto de scripts operacionales:

- Instalación LAM MPI.
- Configuración de Open MPI.
- Creación de usuarios.

Todos estos scripts se analizan en el anexo C.

3.4. Los nodos esclavos

Los nodos esclavos se configuran como clientes NIS y NFS del nodo maestro. Adoptarán el rol de *execution_host* en la instalación de Sun Grid Engine y ejecutarán los trabajos que se les envíe desde el nodo maestro.

Para su configuración hemos creado una plantilla indicando la misma imagen que para el nodo maestro (*RightImage CentOS5_0V4_0_1*). Al

igual que para el nodo maestro, el tipo de instancia de la plantilla será *m1.small*.

Hemos asociado a la plantilla creada para los nodos esclavo algunos scripts que realizan tareas de configuración en el arranque, pues cuando cada esclavo arranca se debe:

1. Configurar el fichero /etc/hosts para que contenga el listado de direcciones IP y los nombres de cada máquina del entorno.
2. Habilitar el acceso por SSH.
3. Asignar una contraseña al superusuario.
4. Configurar el esclavo como cliente NFS para que importe la carpeta /opt del nodo maestro.
5. Configurar el esclavo como cliente NIS del nodo maestro.

La instalación de SGE, la adición de usuarios, la instalación de la interfaz de intercambio de mensajes LAM/MPI y la configuración de Open MPI en los esclavos se realizan desde el nodo maestro con los scripts operacionales asociados al mismo, por lo que no son necesarios scripts operacionales asociados a la plantilla del esclavo.

Los scripts mencionados en este apartado son analizados en el anexo D.

3.5. Instalación de los compiladores, de Open MPI, de R y de Rmpi

Los compiladores GNU para C y Fortran se instalan automaticamente en el arranque de las instancias.

Puesto que la configuración de los compiladores de Intel no depende del número y tipo de instancias que conforman el cluster hemos optado por instalarlos en la unidad de almacenamiento permanente, montada en el nodo maestro en /opt. De esta forma la instalación solo se realiza una vez.

La instalación de LAM/MPI se realiza utilizando un paquete disponible en el repositorio de Rightscale, lo cual permite la instalación automática y rápida de la versión 7.1.2 de este software. De forma análoga podemos instalar la versión 1.1.1 de Open MPI pero al hacerlo nos hemos encontrado con problemas de funcionamiento, por lo que hemos optado por instalar la última versión de esta interfaz de intercambio a partir de los fuentes, consiguiendo así evitar estos problemas. Puesto que una instalación de esta forma consume mucho tiempo y podría alargar de forma muy significativa el tiempo necesario para el despliegue completo del entorno, hemos decidido instalar Open MPI en el directorio /opt/openmpi, esto es, en el directorio compartido. De esta forma, su instalación solo se ha tenido que realizar

una vez, y el procedimiento necesario para utilizarlo en un cluster recién desplegado se reduce a introducir en un fichero un listado con los nodos del cluster y a configurar unas variables de entorno. La secuencia de comandos utilizada para la instalación de Open MPI es la siguiente:

```
wget http://www.open-mpi.org/software/ompi/v1.3/\
downloads/openmpi-1.3.3.tar.gz
tar xvzf openmpi-1.3.3.tar.gz
cd openmpi-1.3.3
./configure --prefix=/opt/openmpi
make all install
```

La instalación de R y de Rmpi, si bien varía segun el compilador, y, en el caso de Rmpi, también de la interfaz de intercambio de mensajes que queramos evaluar, la instalación de estas herramientas se mantendrá invariable para bastantes ejecuciones, por lo que hemos optado por instalarlas también de manera *permanente* en el directorio /opt.

Para la instalación de R hemos creado dos paquetes rpm, uno construido con los compiladores GNU y el otro con los compiladores de Intel.[1] La instalación de Rmpi se realiza utilizando el comando R CMD INSTALL Rmpi_0.57.tar.gz con únicamente una interfaz de intercambio de mensajes instalada (la que corresponda) y con las variables de entorno referentes al compilador a usar inicializadas (de la misma forma que en el paquete de R compilado con la suite de Intel, explicada en el Anexo). También hemos creado un script de lanzamiento de R, a partir del script de lanzamiento que se instala en el directorio /opt/R/bin, que mediante la carga del perfil de Rmpi, hace innecesaria la carga del módulo Rmpi al inicio de cada sesión de R iniciada con dicho script.

3.6. Despliegue en la nube

Para despegar un cluster en primer lugar necesitaremos instanciar el nodo maestro a partir de la plantilla *PFC: Plantilla: Nodo Maestro* y un nodo esclavo a partir de la plantilla *PFC: Plantilla: Nodo Esclavo*, pues siempre tendremos al menos un nodo maestro y un nodo esclavo. Para incrementar el número de nodos esclavos clonaremos la instancia creada tantas veces como necesitemos. Estos procedimientos de operación sobre la plataforma Rightscale son explicados en el anexo B.

El despliegue deberá hacerse siempre siguiendo estos pasos:

1. En primer lugar arrancaremos el maestro.

[1] El proceso de creación del paquete viene tratado en un anexo

2. Una vez el maestro esté en el estado *operational* arrancaremos los esclavos (podremos arrancar todas las instancias apagadas pulsando el boton *Start all* en la plataforma). [2]

3. Una vez todos los esclavos estan en el estado *operational* ejecutamos el script operacional *PFC: Instalación de SGE* en el maestro.[3]

4. Una vez este script haya terminado su labor instalaremos la librería de intercambio de mensajes LAM/MPI ejecutando el script operacional *PFC: Instalacion LAM MPI* o configuraremos el entorno para la utilización de Open MPI, ejecutando el script operacional *PFC: Configuración Open MPI*. Adicionalmente, deberemos ejecutar el script confianza-ssh.sh, explicado en el anexo E, desde la cuenta de usuario sgeadmin, desde la que realizaremos las pruebas.

[2]Si no esperaramos a que el maestro haya terminado la secuencia de arranque, podría pasar que un esclavo intentara configurar el cliente NFS antes de que el maestro hubiera configurado el servidor.

[3]Si no hubieran terminado todos los esclavos la secuencia de arranque, podría pasar que el script de instalación intentara realizar la instalación en ellos antes de tiempo.

Capítulo 4

Evaluaciones de rendimiento de R y Rmpi

En este capítulo explicamos el procedimiento y metodología seguidos para la evaluación del rendimiento de la herramienta en función del compilador y la implementación de MPI utilizada y presentamos los resultados de la misma.

4.1. Procedimiento de evaluación seguido

Lo primero que debemos hacer antes de iniciar un conjunto de pruebas es configurar el entorno para ellas. Por tanto:

1. Iniciaremos el maestro y los esclavos que necesitemos si no los tenemos levantados,
2. Ejecutaremos el script de instalación de SGE,
3. Instalaremos el paquete de R construido con el compilador a evaluar, si no lo tenemos instalado,
4. Instalaremos la librería de intercambio de mensajes pertinente en todos los nodos y
5. instalaremos el módulo Rmpi, configurando su instalación para que se usen el compilador y la librería de intercambio de mensajes adecuados, si no lo tenemos así instalado.

Para cada configuración del entorno hemos ejecutado cinco veces el programa calculo.R que se encuentra en la carpeta /opt/pruebas, un programa escrito en R que presentamos en el siguiente apartado, volcando la salida de la ejecución i-ésima al fichero s*i*.out de un directorio dedicado a la salida de las pruebas para esa configuración del entorno, con el comando

```
mpirun C /opt/R/bin/Rmpi CMD BATCH /opt/pruebas/calculo.R \
<fichero de salida> --no-save -q
```

para usar la interfaz LAM/MPI y

```
mpirun --hostfile /opt/mpi/openhosts.def /opt/R/bin/Rmpi CMD \
BATCH /opt/pruebas/calculo.R <fichero de salida> --no-save -q
```

para usar la interfaz Open MPI.

Durante la ejecución de las pruebas no hemos hecho uso del gestor de colas, puesto que teníamos la seguridad de tener disponibles todos los nodos del entorno. En un entorno real, esto no siempre será así, y será muy recomendable realizar las ejecuciones a través del gestor de colas instalado, Sun Grid Engine. Para ello, incluiremos el comando a ejecutar en un script, de nombre *comando_ mpi.sh* por ejemplo, y con el comando

```
qsub comando_mpi.sh
```

lo mandaremos al gestor de colas.

4.1.1. Programa utilizado para realizar las mediciones

Al consultar la sección de tutoriales sobre Rmpi en la página del módulo[1], se nos refería al tutorial del Acadia Centre for Mathematical Modelling and Computation. En este tutorial [1] hemos encontrado varias soluciones paralelas en R al problema de la validación cruzada con 10 pliegues, una operación utilizada en estadística para saber como de bueno es un modelo predictivo. Este problema posee la propiedad de ser facilmente paralelizable y por ello lo hemos escogido, pues no queremos estudiar el comportamiento del problema, si no el de la herramienta R.

Entre las implementaciones que hemos encontrado en este tutorial nos hemos decantado por la solución que utiliza el esquema de bolsa de tareas: el nodo maestro genera una lista de tareas y los nodos esclavo, mientras quedan tareas disponibles, ejecutan un bucle en el que piden una tarea al nodo maestro, la realizan y le envían la solución. La razón por la cuál hemos escogido esta implementación es que el tamaño del problema a solucionar no tiene que cumplir ninguna condición para su paralelización al tiempo que el trabajo se reparte uniformemente entre los nodos esclavo y que se adaptará automaticamente al entorno de computación distribuida donde la generemos, no siendo necesario configurar en el código de la implementación el numero de nodos del mismo.

Hemos modificado el tamaño del problema, manteniendo el número de variables del mismo y aumentando de 1000 a 400.000 el número de muestras por variable, con objeto de que cada ejecución tarde suficiente como para

[1] http://www.stats.uwo.ca/faculty/yu/Rmpi/

justificar su ejecución en un cluster con 16 nodos de cómputo. Además, hemos introducido en la implementación del tutorial unas líneas dedicadas a la medición del tiempo que se tarda en generar la matriz de datos, en enviar a los nodos esclavos dicha matriz, en que se soluciona el problema y el tiempo de ejecución total. El código final que hemos ejecutado como prueba de rendimiento de cada instalación de la herramienta R queda así:

```
time1 <- Sys.time()
if (mpi.comm.size() < 2) {
    print("More slave processes are required.")
    mpi.quit()
    }

.Last <- function(){
    if (is.loaded("mpi_initialize")){
        if (mpi.comm.size(1) > 0){
            print("Please use mpi.close.Rslaves()
to close slaves.")
            mpi.close.Rslaves()
        }
        print("Please use mpi.quit() to quit R")
        .Call("mpi_finalize")
    }
}

# Function the slaves will call to perform a validation on the
# fold equal to their slave number.
# Assumes: thedata,fold,foldNumber,p
foldslave <- function() {
    # Note the use of the tag for sent messages:
    #     1=ready_for_task, 2=done_task, 3=exiting
    # Note the use of the tag for received messages:
    #     1=task, 2=done_tasks
    junk <- 0

    done <- 0
    while (done != 1) {
        # Signal being ready to receive a new task
        mpi.send.Robj(junk,0,1)

        # Receive a task
        task <- mpi.recv.Robj(mpi.any.source(),mpi.any.tag())
        task_info <- mpi.get.sourcetag()
        tag <- task_info[2]
```

```
        if (tag == 1) {
            foldNumber <- task$foldNumber

            rss <- double(p)
            for (i in 1:p) {
                # produce a linear model on the first
# i variables on
                # training data
                templm <- lm(y~.,data=thedata[fold!=foldNumber
,1:(i+1)])

                # produce predicted data from test data
                yhat <- predict(templm,newdata=
thedata[fold==foldNumber,1:(i+1)])

                # get rss of yhat-y
                localrssresult <- sum((yhat-
thedata[fold==foldNumber,1])^2)
                rss[i] <- localrssresult
                }

            # Send a results message back to the master
            results <- list(result=rss,foldNumber=foldNumber)
            mpi.send.Robj(results,0,2)
            }
        else if (tag == 2) {
            done <- 1
            }
        # We'll just ignore any unknown messages
        }

    mpi.send.Robj(junk,0,3)
    }

# We're in the parent.
# first make some data
n <- 400000 # number of obs
p <- 30 # number of variables

# Create data as a set of n samples of p independent variables,
# make a "random" beta with higher weights in the front.
# Generate y's as y = beta*x + random
x <- matrix(rnorm(n*p),n,p)
```

```
beta <- c(rnorm(p/2,0,5),rnorm(p/2,0,.25))
y <- x %*% beta + rnorm(n,0,20)
thedata <- data.frame(y=y,x=x)

fold <- rep(1:10,length=n)
fold <- sample(fold)

summary(lm(y~x))
time2 <- Sys.time()
# Now, send the data to the slaves
mpi.bcast.Robj2slave(thedata)
mpi.bcast.Robj2slave(fold)
mpi.bcast.Robj2slave(p)

# Send the function to the slaves
mpi.bcast.Robj2slave(foldslave)
time3 <- Sys.time()
# Call the function in all the slaves to get them ready to
# undertake tasks
mpi.bcast.cmd(foldslave())


# Create task list
tasks <- vector('list')
for (i in 1:10) {
    tasks[[i]] <- list(foldNumber=i)
    }

# Create data structure to store the results
rssresult = matrix(0,p,10)

junk <- 0
closed_slaves <- 0
n_slaves <- mpi.comm.size()-1

while (closed_slaves < n_slaves) {
    # Receive a message from a slave
    message <- mpi.recv.Robj(mpi.any.source(),mpi.any.tag())
    message_info <- mpi.get.sourcetag()
    slave_id <- message_info[1]
    tag <- message_info[2]

    if (tag == 1) {
        # slave is ready for a task. Give it the next task,
```

```
# or tell it tasks are done if there are none.
        if (length(tasks) > 0) {
            # Send a task, and then remove it from
# the task list
            mpi.send.Robj(tasks[[1]], slave_id, 1);
            tasks[[1]] <- NULL
            }
        else {
            mpi.send.Robj(junk, slave_id, 2)
            }
        }
    else if (tag == 2) {
        # The message contains results. Do something with
# the results.
        # Store them in the data structure
        foldNumber <- message$foldNumber
        rssresult[,foldNumber] <- message$result
        }
    else if (tag == 3) {
        # A slave has closed down.
        closed_slaves <- closed_slaves + 1
        }
    }
apply(rssresult,1,mean)

time2 - time1 #Tiempo de generación de la matriz
time3 - time2 #Tiempo de su envío a los esclavos
time4 <- Sys.time()
time4 - time3 #Tiempo de solución del problema
time4 - time1 #Tiempo total

mpi.remote.exec(paste("I am",mpi.comm.rank(),
"of",mpi.comm.size()))

mpi.close.Rslaves()
mpi.quit(save="no")
```

4.2. Resultados

En este apartado analizaremos los resultados de las pruebas realizadas, primero comparando los tiempos de ejecución en las instalaciones que han utilizado los compiladores GNU con el de las instalaciones que han utilizado los compiladores Intel, y luego comparando los tiempos de ejecución de las

instalaciones que han utilizado LAM/MPI con el de las instalaciones que han utilizado Open MPI. Estas comparaciones las realizaremos para cada uno de los tiempos medidos:

- el tiempo de generación de la matriz de datos, que se corresponde con un tiempo de inicialización que encontramos en cualquier aplicación paralela y que nos permite analizar el comportamiento de la aplicación cuando se ejecuta un fragmento de código secuencial;
- el tiempo de comunicación de la matriz de datos a los nodos esclavo, que se corresponde con una comunicación del problema del nodo maestro a los nodos esclavo que encontramos en muchas aplicaciones paralelas y que nos permite evaluar el comportamiento de la aplicación cuando los nodos deben comunicarse datos entre ellos;
- el tiempo de computación paralela, que se corresponde con el tiempo de resolución de problema con la colaboración de todos los nodos que encontramos en muchas aplicaciones paralelas y que nos permite analizar el comportamiento de la aplicación cuando se solapan en el código calculo y comunicaciones;
- y el tiempo total, esto es, la suma de los anteriores, que nos da una visión general del comportamiento de la aplicación durante la resolución del problema.

4.2.1. Comparación en función de los compiladores utilizados

Tiempo de generación de la matriz de datos

La figura 4.1 nos muestra una comparativa de los tiempos de generación de la matriz de datos de las ejecuciones del programa de pruebas sobre instalaciones de R que utilizan LAM/MPI como implementación MPI, según se hayan utilizado los compiladores GNU o los de Intel para la instalación de R. La figura 4.2 nos presenta la misma comparativa, pero esta vez con instalaciones de R que utilizan Open MPI.

Como es de esperar, los tiempos se mantienen relativamente constantes respecto al número de nodos esclavo utilizados en la ejecución del programa, pues un solo nodo, el maestro, se encargará de la misma cantidad de trabajo. La pequeña variación que se produce la analizaremos con mas detenimiento cuando comparemos los tiempos de ejecución en función de la implementación MPI utilizada.

Podemos observar es que la instalación de R con los compiladores GNU ha proporcionado tiempos de ejecución secuenciales sustancialmente menores, independientemente de la interfaz de intercambio de mensajes utilizada. Sin embargo, sería posible que este hecho se debiera a que el tamaño del problema

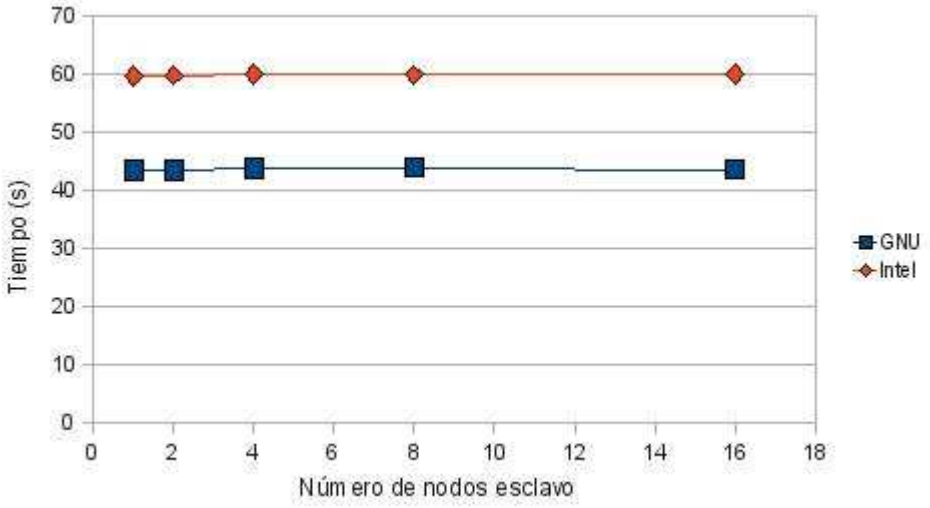

Figura 4.1: Tiempos de generación de la matriz de datos de calculo.R en función de los compiladores, utilizando LAM/MPI

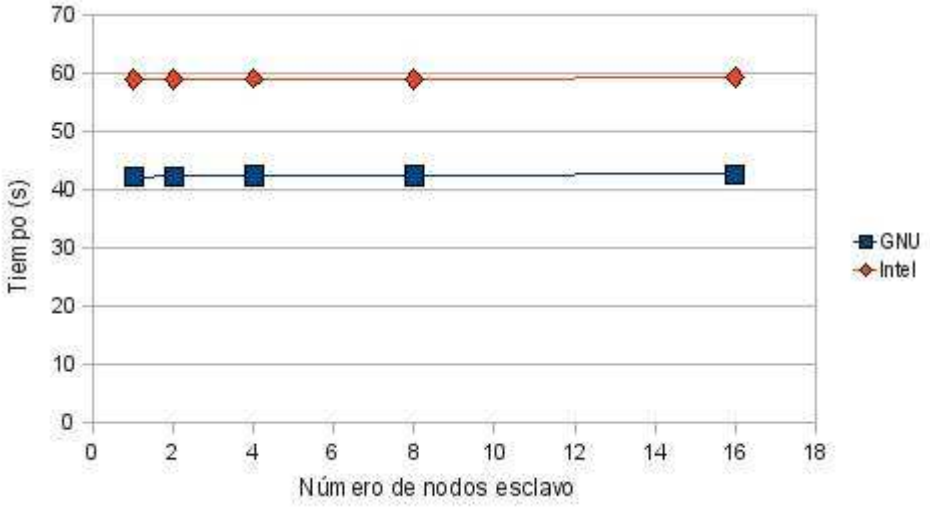

Figura 4.2: Tiempos de generación de la matriz de datos de calculo.R en función de los compiladores, utilizando Open MPI

genere una distribución de los datos del mismo en memoria que beneficie a los tiempos de ejecución de las aplicaciones generadas con los compiladores GNU en detrimento de las generadas con los compiladores Intel.

Tiempo de comunicación de la matriz de datos

La figura 4.3 nos muestra una comparativa de los tiempos comunicación de la matriz de datos de las ejecuciones del programa de pruebas sobre instalaciones de R que utilizan LAM/MPI como implementación MPI, según se hayan utilizado los compiladores GNU o los de Intel para la instalación de R. La figura 4.4 nos presenta la misma comparativa, pero esta vez con instalaciones de R que utilizan Open MPI.

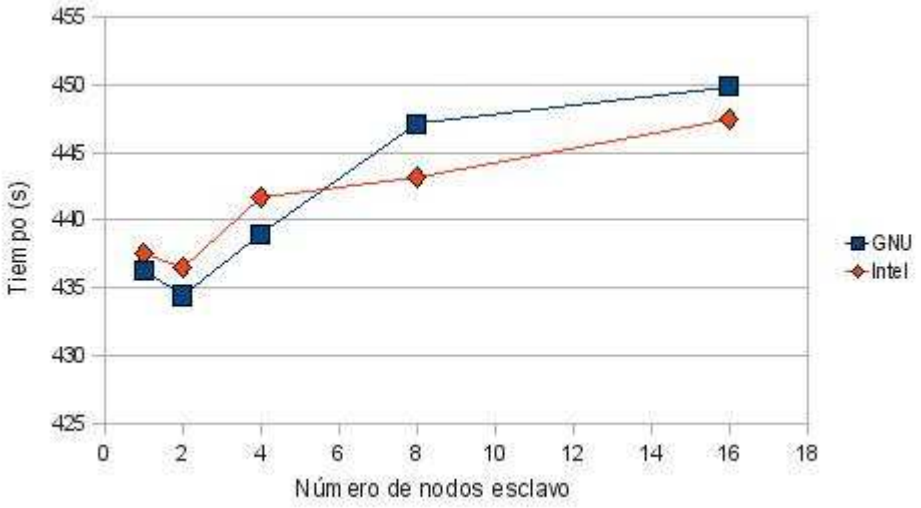

Figura 4.3: Tiempos de comunicación de la matriz de datos de calculo.R en función de los compiladores, utilizando LAM/MPI

A partir de ambas gráficas podemos afirmar una evolución logarítmica del tiempo de las comunicaciones en función del número de nodos esclavo del entorno donde se ha realizado cada prueba, lo cual es posible gracias a un solapamiento de las comunicaciones a los nodos esclavo.

De entre las instalaciones que utilizan LAM/MPI, cuyo comportamiento observamos en la primera gráfica, no podemos determinar cual es la mejor en base a los resultados obtenidos, pues, aunque parece que con 1, 2 y 4 nodos se comporta mejor la instalación construida con los compiladores GNU y con 8 y 16 nodos la construida con los compiladores de Intel, las diferencias de tiempo para un mismo numero de nodos esclavo es bastante pequeña y podría deberse a una carga mayor puntual en el sistema o a una sobrecarga de la red de la infraestructura cloud en el momento de la ejecución.

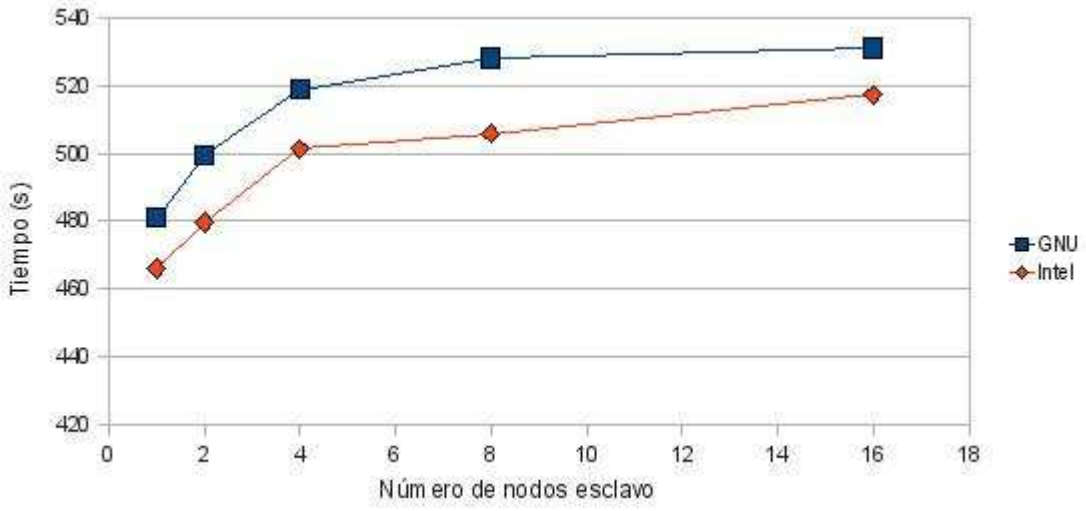

Figura 4.4: Tiempos de comunicación de la matriz de datos de calculo.R en función de los compiladores, utilizando Open MPI

La segunda gráfica nos proporciona resultados más significativos: de las instalaciones que utilizan Open MPI, las construidas con los compiladores de Intel presentan tiempos de comunicación sustancialmente menores que las construidas con los compiladores GNU. Estas diferencias parten de 20 segundos para el caso de un nodo esclavo y van disminuyendo hasta 14 segundos en el caso de dos nodos, lo cual nos lleva a pensar que para un número de nodos suficientemente grande los tiempos de comunicación sobre ambas instalaciones convergerá.

Tiempo de computación paralela

La figura 4.5 nos muestra una comparativa de los tiempos computación paralela de las ejecuciones del programa de pruebas sobre instalaciones de R que utilizan LAM/MPI como implementación MPI, según se hayan utilizado los compiladores GNU o los de Intel para la instalación de R. La figura 4.6 nos presenta la misma comparativa, pero esta vez con instalaciones de R que utilizan Open MPI.

En ambas gráficas podemos observar que el tiempo estudiado disminuye a poco más de la mitad cuando duplicamos el número de nodos esclavo. Esto es lo que esperabamos, pues al duplicar el número de nodos esclavo la carga para cada nodo se reduce a la mitad. El hecho de que sea el nodo maestro el que deba tanto asignar a los nodos esclavo sus tareas como recibir sus resultados, explica que la reducción no sea exactamente a la mitad, pues su carga de trabajo no se reduce.

En cuanto a la comparación que se plantea, los resultados son bastante

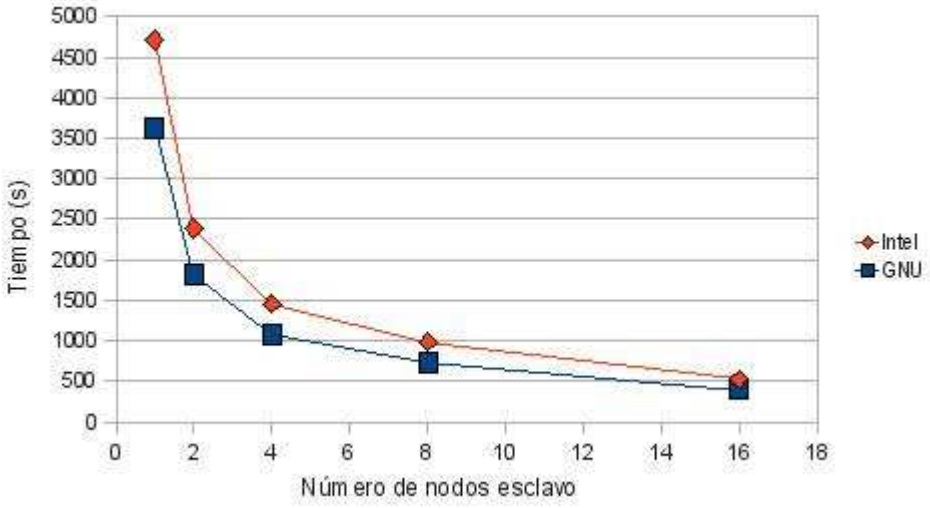

Figura 4.5: Tiempos de computación paralela de calculo.R en función de los compiladores, utilizando LAM/MPI

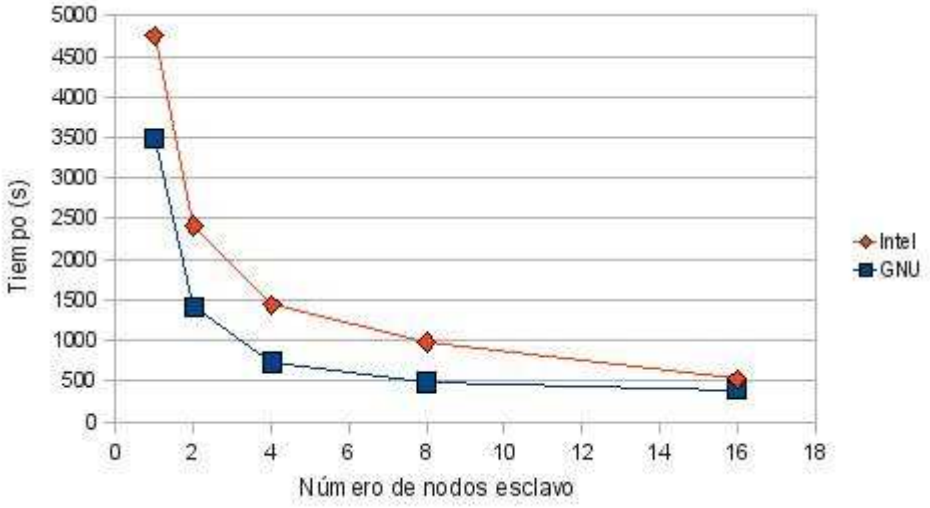

Figura 4.6: Tiempos de computación paralela de calculo.R en función de los compiladores, utilizando Open MPI

claros: al utilizar compiladores GNU obtenemos resultados mejores, aunque la diferencia se va reduciendo conforme aumentamos el número de nodos esclavo.

Tiempo de ejecución total

La figura 4.7 nos muestra una comparativa de los tiempos totales de las ejecuciones del programa de pruebas sobre instalaciones de R que utilizan LAM/MPI como implementación MPI, según se hayan utilizado los compiladores GNU o los de Intel para la instalación de R. La figura 4.8 nos presenta la misma comparativa, pero esta vez con instalaciones de R que utilizan Open MPI.

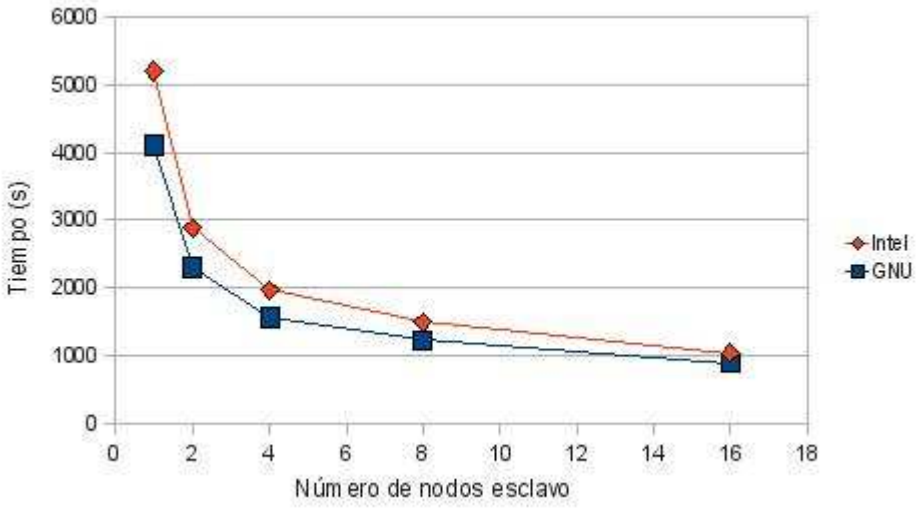

Figura 4.7: Tiempos de ejecución de calculo.R en función de los compiladores, utilizando LAM/MPI

Como estas mediciones constituyen la suma de las anteriores, y de estas, la de tiempo de cómputo paralelo es la que mayor tiempo aporta a la suma, en estas gráficas vemos casi copiadas las del subapartado anterior. En conjunto, y a pesar de que sus tiempos de comunicación puedan ser algo peores, las ejecuciones del programa de pruebas sobre instalaciones de R construidas con los compiladores GNU han proporcionado tiempos menores, aunque la diferencia con los tiempo de ejecución del programa de pruebas sobre instalaciones de R construidas con los compiladores de Intel se reduzca al aumentar el número de nodos.

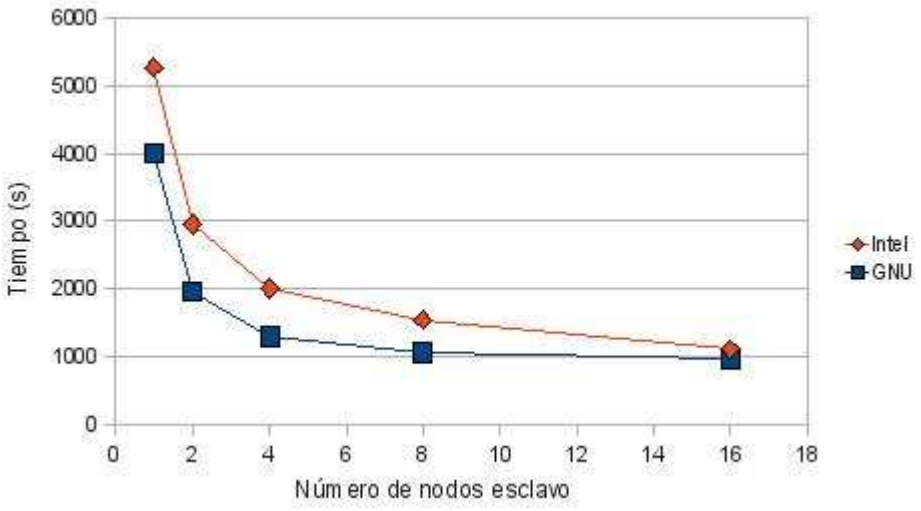

Figura 4.8: Tiempos de ejecución de calculo.R en función de los compiladores, utilizando Open MPI

4.2.2. Comparación en función de la implementación de MPI utilizada

Tiempo de generación de la matriz de datos

La figura 4.9 nos muestra una comparativa de los tiempos de generación de la matriz de datos de las ejecuciones del programa de pruebas sobre instalaciones de R para las que se ha hecho uso de los compiladores GNU, según utilicen LAM/MPI u Open MPI como implementación MPI. La figura 4.10 nos presenta la misma comparativa, pero esta vez con instalaciones de R para las que se ha hecho uso de los compiladores de Intel.

Estas graficas nos permiten analizar que implementación MPI tiene menor impacto en el rendimiento de R durante la ejecución de un código secuencial. Puede llamar la atención que la elección de una implementación MPI u otra tenga efecto en un tiempo dedicado a cómputo secuencial en un único nodo pero esto se debe a que durante la generación de la matriz de datos del problema se realizan operaciones de mantenimiento del entorno MPI. Es por esto que se produce la pequeña varición que observabamos en las figuras 4.1 y 4.2. En base a los resultados obtenidos, podemos afirmar que la implementación Open MPI tiene menor impacto en el rendimiento de R durante la ejecución de un código secuencial.

Tiempo de comunicación de la matriz de datos

La figura 4.11 nos muestra una comparativa de los tiempos de comunicación de la matriz de datos en las ejecuciones del programa de pruebas sobre

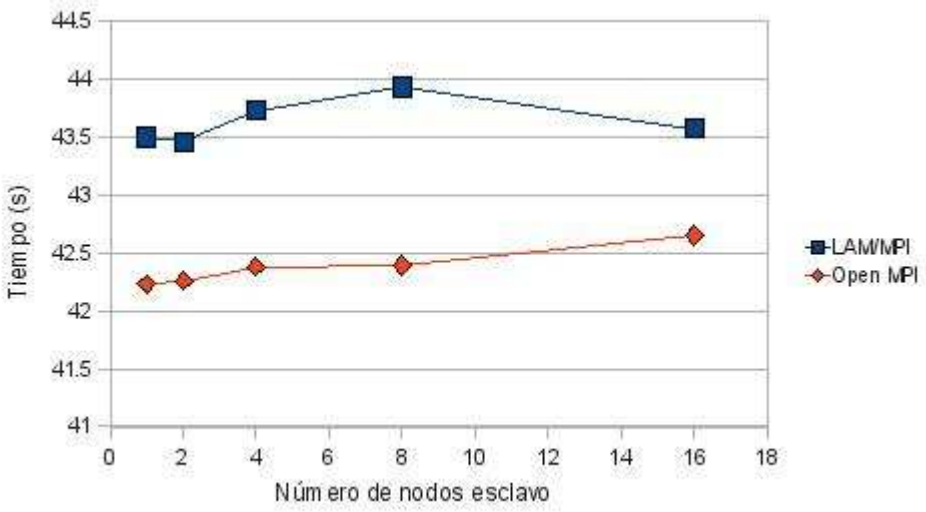

Figura 4.9: Tiempos de generación de la matriz de datos de calculo.R en función de la implementación MPI, utilizando los compiladores GNU

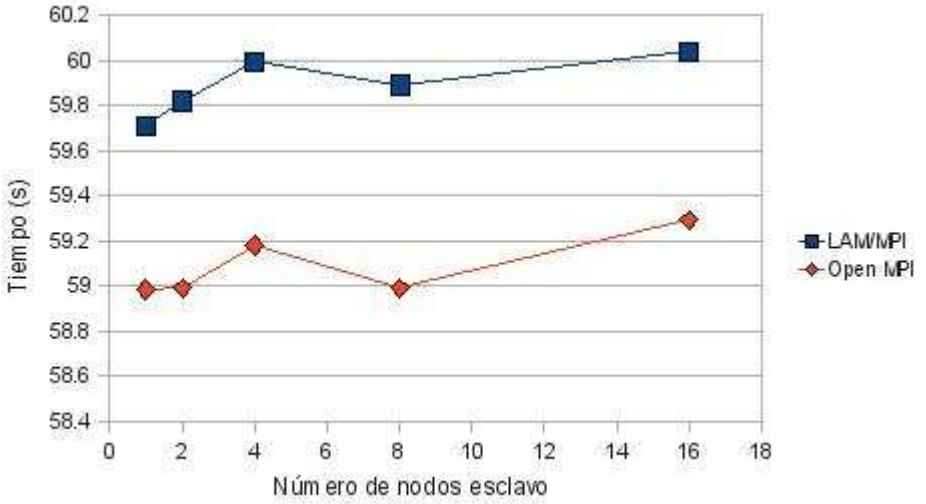

Figura 4.10: Tiempos de generación de la matriz de datos de calculo.R en función de la implementación MPI, utilizando los compiladores de Intel

instalaciones de R para las que se ha hecho uso de los compiladores GNU, según utilicen LAM/MPI u Open MPI como implementación MPI. La figura 4.12 nos presenta la misma comparativa, pero esta vez con instalaciones de R para las que se ha hecho uso de los compiladores de Intel.

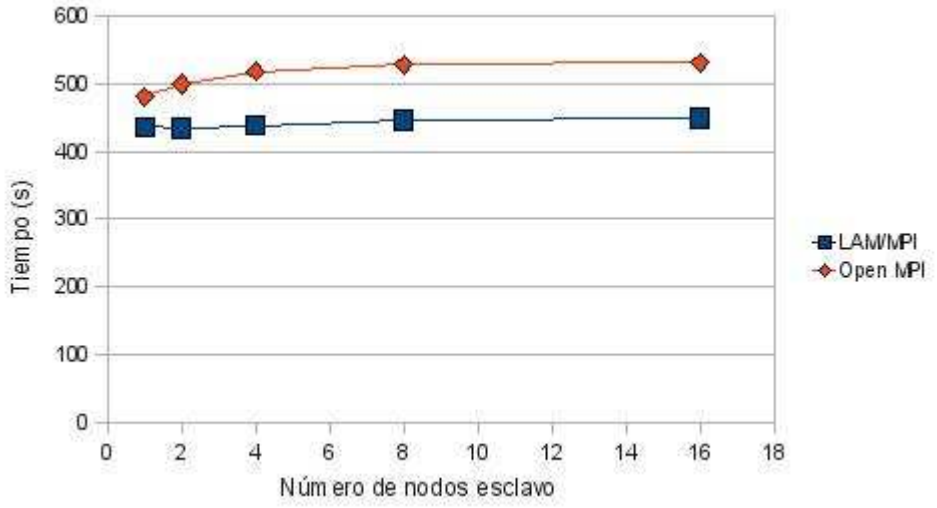

Figura 4.11: Tiempos de comunicación de calculo.R en función de la implementación MPI, utilizando los compiladores GNU

De estos resultados deducimos que el uso de LAM/MPI proporciona mejores tiempos de comunicaciones, que aumentan muy lentamente conforme aumentamos el número de nodos, mientras que el uso de Open MPI proporciona tiempos mayores desde el principio (con un único nodo esclavo) y que aumentan significativamente más rapido, por lo menos al principio.

Tiempo de computación paralela

La figura 4.13 nos muestra una comparativa de los tiempos de computación paralela en las ejecuciones del programa de pruebas sobre instalaciones de R para las que se ha hecho uso de los compiladores GNU, según utilicen LAM/MPI u Open MPI como implementación MPI. La figura 4.14 nos presenta la misma comparativa, pero esta vez con instalaciones de R para las que se ha hecho uso de los compiladores de Intel.

Compilado con los compiladores GNU, R presenta un comportamiento peor en la computación paralela al hacer uso de LAM/MPI que al hacerlo de Open MPI, si bien la diferencia en los tiempos se va haciendo menor conforme aumentamos el número de nodos esclavo y practicamente desparece al llegar a 16.

Al compilarlo con los compiladores de Intel, R presentará comportamientos similares en la computación paralela al utilizar ambas implementaciones

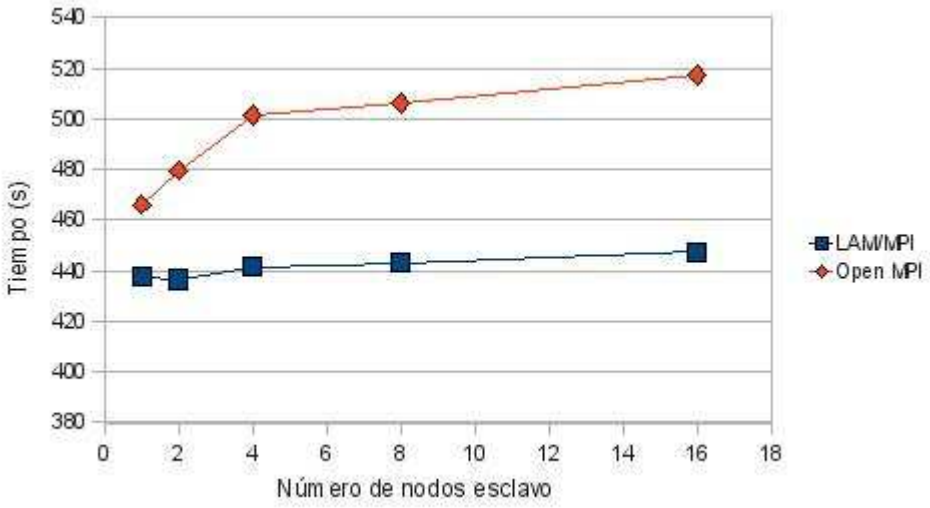

Figura 4.12: Tiempos de comunicación de calculo.R en función de la implementación MPI, utilizando los compiladores de Intel

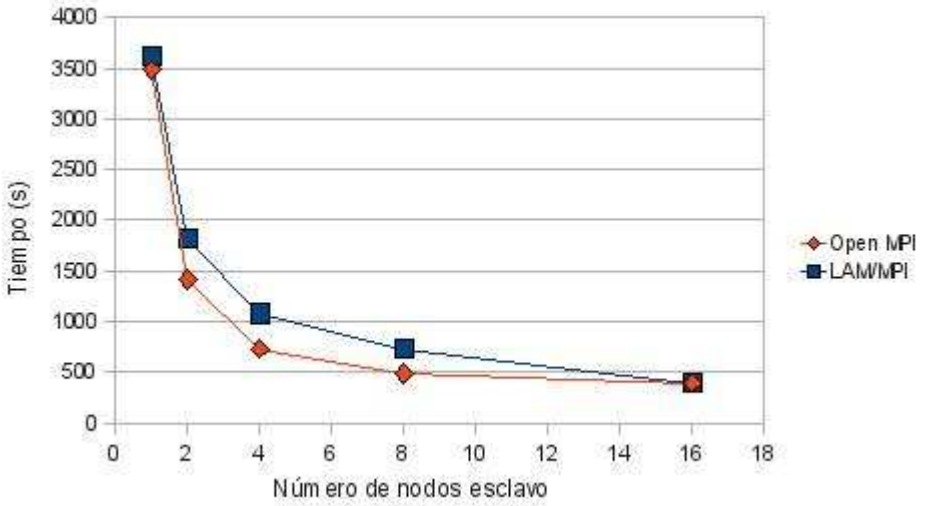

Figura 4.13: Tiempos de computación paralela de calculo.R en función de la implementación MPI, utilizando los compiladores GNU

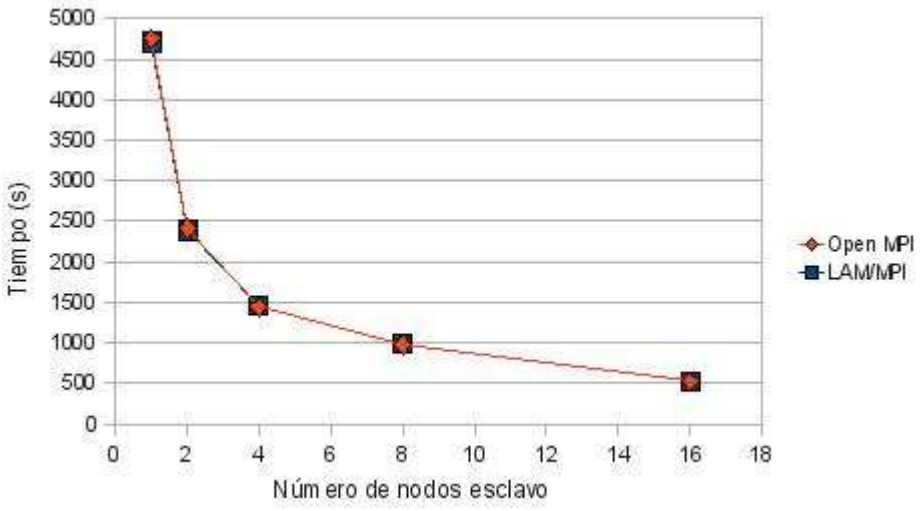

Figura 4.14: Tiempos de computación paralela de calculo.R en función de la implementación MPI, utilizando los compiladores de Intel

de MPI. No obstante, se puede advertir una pequeña diferencia a favor de LAM/MPI que se va reduciendo al aumentar el número de nodos esclavo.

Tiempo de ejecución total

La figura 4.15 nos muestra una comparativa de los tiempos totales de las ejecuciones del programa de pruebas sobre instalaciones de R para las que se ha hecho uso de los compiladores GNU, según utilicen LAM/MPI u Open MPI como implementación MPI. La figura 4.16 nos presenta la misma comparativa, pero esta vez con instalaciones de R para las que se ha hecho uso de los compiladores de Intel.

Compilado con los compiladores GNU, R presenta un comportamiento peor en general al hacer uso de LAM/MPI que al hacerlo de Open MPI, aunque la diferencia en los tiempos se va haciendo menor conforme aumentamos el número de nodos esclavo y al llegar a 16 nodos esclavo la situación se invierte, presentando la instalación que hace uso de LAM/MPI un mejor comportamiento.

Al compilarlo con los compiladores de Intel, R presentará comportamientos ligeramente mejores al utilizar LAM/MPI que al utilizar Open MPI. Al aumentar el número de nodos esclavo, la diferencia de tiempos aumenta.

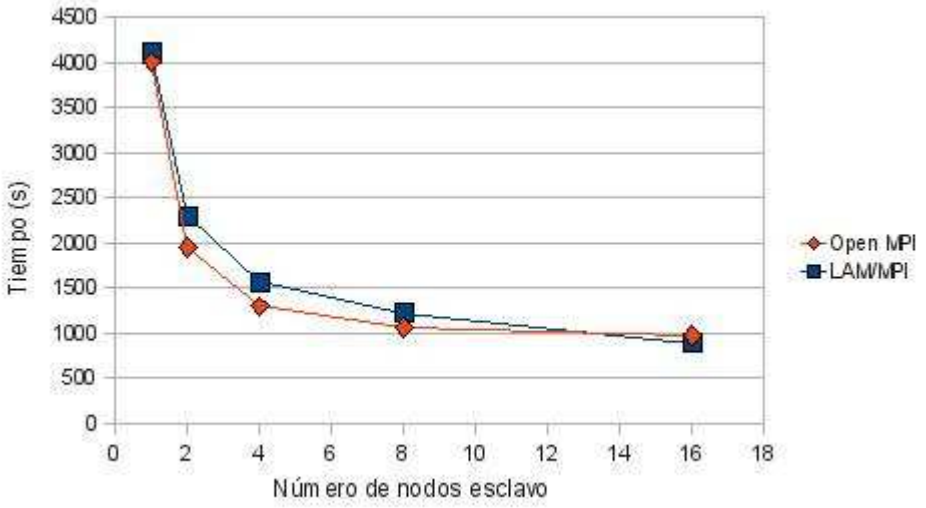

Figura 4.15: Tiempos de ejecución de calculo.R en función de la implementación MPI, utilizando los compiladores GNU

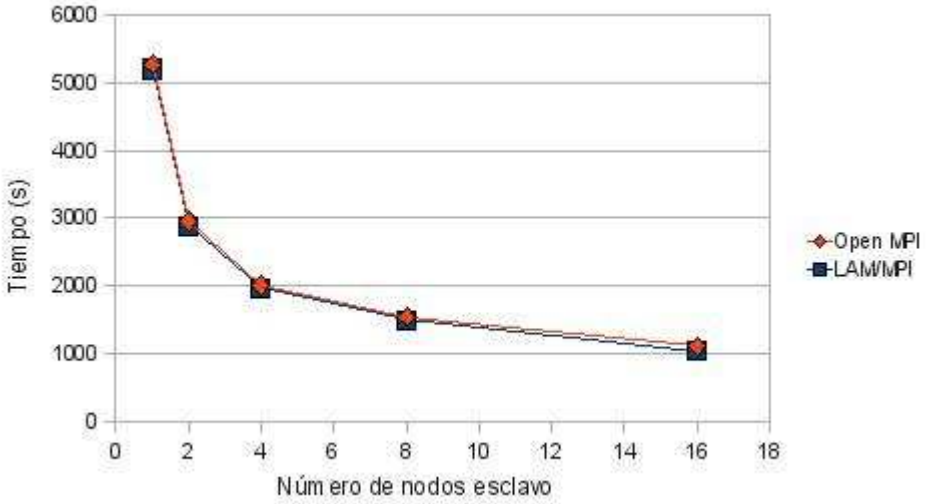

Figura 4.16: Tiempos de ejecución de calculo.R en función de la implementación MPI, utilizando los compiladores de Intel

Capítulo 5

Conclusiones y vías futuras

La utilización de una plataforma cloud para la elaboración de una maqueta de un entorno de computación distribuida ha resultado ser una alternativa viable. En ella hemos podido instalar y configurar todos los elementos de los entornos simulados, automatizando la configuración y despliegue del entorno en gran medida.

La maqueta ha resultado ser un entorno para el estudio del comportamiento de aplicaciones científicas adecuado, al presentar una gran estabilidad que ha dotado de mayor validez a los resultados obtenidos.

De las pruebas realizadas, los mejores resultados han sido obtenidos al realizar la instalación de R utilizando los compiladores GNU. Por ello, no parece recomendable la compra de una licencia para la compilación de aplicaciones científicas con los compiladores de Intel, si van a ser usadas en la plataforma cloud de Rightscale.

La mejor implementación de MPI a usar en la maqueta, sin embargo, dependerá del número de nodos esclavo de la misma, pues según hemos visto en nuestras pruebas, para un número suficientemente grande de nodos esclavo en la maqueta Open MPI puede aportar un mejor rendimiento que LAM/MPI.

No obstante, los resultados de los que hablamos solo son aplicables a las ejecuciones sobre la maqueta, la idoneidad de un conjunto de compiladores o de una implementación de MPI deberá ser comprobada sobre la arquitectura del entorno final.

Como vías de trabajo futuro referentes a la maqueta se plantean las siguientes:

- Automatización y configuración de las alternativas a las herramientas utilizadas consideradas en el segundo capítulo de este documento.
- Utilización de otros tipos de instancias disponibles en Rightscale, de los que hablamos en el tercer capítulo y que se describen en la tabla 3.1.

Como vías de trabajo futuro relacionadas con la instalación de R en este entorno se plantean:

- Realizar pruebas de las instalaciones consideradas variando el tamaño del problema.
- Realizar pruebas de las instalaciones consideradas, haciendo uso de los distintos tipos de instancia que expusimos en el tercer capítulo en la tabla 3.1.

Apéndice A

Creación de RPMs para R

A.1. El fichero .spec

La creación de paquetes rpm va guiada por un fichero .spec que contiene información sobre el software y sobre el proceso de creación e instalación del rpm. Este fichero tiene varias secciones que describimos ayudándonos de uno de los ficheros .spec creados para la instalación de R.

A.1.1. La sección Header o cabecera

```
Summary: Free software environment for statistical computing and
                                                          graphics
Name: R
Version: 2.8.0
Release: 9
Source: http://cran.es.r-project.org/src/base/R-2/R-2.8.0.tar.gz
License: GPL
Group: Development/Tools
BuildRoot: %{_builddir}/%{name}-root
BuildRequires: readline-devel libX11-devel libXt-devel gcc
                                                  gcc-gfortran
Requires: readline >= 5.1, texinfo >= 4.8
```

Esta sección se compone de etiquetas con información sobre el paquete. Algunas de las que se pueden indicar son:

- La etiqueta Summary ofrece una descripción de una línea sobre el software que el paquete instala.
- La etiqueta Name indica el nombre del paquete.
- La etiqueta Version ofrece la version del software del paquete.

- La etiqueta Release ofrece la version del paquete, ya que podemos tener diferentes versiones del paquete para la misma versión del software. Es importante incrementar en 1 el valor de esta etiqueta cada vez que modifiquemos el paquete.
- La etiqueta Source ofrece la localización del código fuente del paquete, normalmente un tarball. Se puede indicar directamente el nombre del fichero, en cuyo caso la ruta es relativa al directorio SOURCES de construcción de paquetes (/usr/src/redhat/SOURCES en el caso de CentOS), o se puede indicar una URL, con objeto de que un tercero que quiera construir el paquete a traves de el fichero .spec pueda localizar las fuentes. En este ultimo caso, no obstante, sera necesario descargar el paquete y situarlo en el directorio SOURCES.
- La etiqueta License ofrece información sobre la licencia.
- La etiqueta Group proporciona la categoría del paquete, ayudando asi a los usuarios a clasificarlo.
- La etiqueta BuildRoot especifica el directorio de construcción del paquete, que será utilizado en otras secciones. En nuestro caso hacemos uso de dos macros:

  `%{_builddir}`

  que hace referencia al directorio de construcción (esto es, a su propio valor) y

  `%{name}`

  que hace referencia al nombre del software, especificado por la etiqueta Name anteriormente. El valor original de esta etiqueta se corresponde con el directorio BUILD de la estructura de construcción de paquetes (/usr/src/packages/BUILD en el caso de Suse Linux Enterprise 10).
- Las etiquetas Requires y BuildRequires indican de que paquetes depende el que se va a instalar para su funcionamiento y construcción respectivamente, esto es, que paquetes son necesarios para que el software a instalar funcione correctamente tras la instalación y cuales para poder construirlo correctamente. Se pueden declarar tantas etiquetas de dependencias como se desee y se puede tambien declarar varias dependencias en una sola etiqueta separandolas por comas. Cada dependencia tendra la forma *capability_name operador versión*, donde *capability_name* es lo que ofrece el paquete del que el nuestro depende (bien el nombre, bien el valor de la etiqueta Provides), *operador* es un operador de comparacion (==,=,>=,<=,<ó >) y *version* es el numero de versión del software del paquete del que el nuestro depende, siendo los últimos dos parámetros opcionales.

A.1.2. La sección description

```
%description
R is an integrated suite of software facilities for data manipulation,
calculation and graphical display. It includes
    * an effective data handling and storage facility,
    * a suite of operators for calculations on arrays, in particular
matrices,
    * a large, coherent, integrated collection of intermediate tools
for data analysis,
    * graphical facilities for data analysis and display either
on-screen or on hardcopy, and
    * a well-developed, simple and effective programming language which
includes conditionals, loops, user-defined recursive functions and
input and output facilities.
```

Esta sección ofrece una descripción mas en detalle y de varias lineas del software a instalar.

A.1.3. Las secciones de construcción: prep, build, install y clean

Estas secciones indican a modo de script como preparar la instalacion, configurar e instalar el software y como limpiar los directorios de construccion. Estan compuestas por comandos de script bash y por macros.

```
%prep
%setup -q
```

La seccion de preparación es la encargada de dejar todo listo para los siguientes pasos. Normalmente se rellena con la macro *%setup -q* que descomprime el fuente indicado y se situa en el directorio.

```
%build
./configure --enable-R-shlib --prefix=/opt/R
make
```

Esta es la seccion de configuración del paquete. Aquí se deben ejecutarlos scripts configure y make, que configuran la aplicación y el instalador.

```
%install
rm -rf $RPM_BUILD_ROOT
make install DESTDIR=$RPM_BUILD_ROOT
```

En la sección install se ejecutan los comandos de instalación del software. Es buena idea limpiar el directorio de construccion previamente.

```
%clean
rm -rf $RPM_BUILD_ROOT
```

La sección clean especifica los comandos a ejecutar para limpiar el directorio de instalación.

A.1.4. La seccion files

```
%files -f /opt/RpmForR/filelist.txt
```

Las secciones del apartado anterior instalaban el software, pero a la hora de construir un paquete binario, debemos especificar la lista de ficheros que debe contener el paquete, esto es, que debe instalar.

Al incluir en esta lista un directorio se incluyen todos los ficheros dentro de el. Los ficheros de documentación deberían precederse de *%doc*. Cuando se dispone de directorios con ficheros unicamente de documentación se puede indicar con *%docdir* <*directorio*> que el directorio es de documentación, si bien el contenido del mismo no se considerar incluido en la lista (deberá incluirse con otras lineas).

Si alguno de los ficheros que el instalador situa en el directorio de construcción no esta listado, alguno de los de lista no existe o si hay duplicados en la lista entonces el constructor de paquetes fallará y no se construirá el paquete. Si la lista de ficheros es muy larga, como es el caso de nuestros RPMs, puede convenir incluirla mediante un fichero aparte, indicandolo con los parametros -f <fichero lista>tras el nombre de la sección.

A.2. Construcción de un paquete binario para R compilado con los compiladores GNU

El primer paso a la hora de enfrentarse a la construcción de un rpm binario es instalar manualmente el software. Para ello descargamos la ultima versión estable de R y la descomprimimos. La mayoría de valores de la sección cabecera y la sección descripción los conocemos ya y de la instalación manual obtendremos el resto de datos necesarios para completar el fichero .spec. Este proceso de instalación será el tipico *./configure; make; make install*, aunque habrá que especificar algunos parámetros.

Al script configure debemos pasarle los parámetros adecuados. Queremos habilitar la compilación de librerías compartidas, por lo que debemos pasarle el parametro *–enable-R-shlib*. Si no indicamos un prefijo de instalación, R se instalará en subdirectorios de /usr/local. Esto no es lo que queremos, pues queremos instalar la aplicación en /opt/R. Le pasaremos por tanto también la opcion *–prefix=/opt/R*. Estos son los parametros del script configure en la sección *%build*.

De la salida en consola de *./configure* podemos extraer las dependencias del mismo. Para que *./configure* no falle es necesario que tengamos instalados readline-devel, libX11-devel, libXt-devel, gcc y gcc-gfortran. Con estos datos podemos elaborar la etiqueta *BuildRequires*.

El comando *make* no necesitará ningún parametro, pero *make install* sí lo hará, pues debemos indicarle que debe instalar la aplicación en el directorio de construcción con el argumento *DESTDIR=directorio*. Para la instalación manual, en lugar del directorio /usr/src/redhat/BUILD que es el que utiliza el constructor de paquetes, hacemos uso de un directorio en nuestra carpeta personal. Es importante sin embargo, que esté vacío antes de la instalación para que la elaboración de la lista de ficheros a instalar que tendremos que abordar a continuación no se nos complique. El parametro de make install en la sección *%install* será *DESTDIR=$RPM_BUILD_ROOT*, con el cual indicamos que situe la instalación en el directorio de construcción.

Una vez instalado, pasamos a la elaboración de la lista de ficheros. Para ello realizamos un listado recursivo del directorio de instalación (con el comando find por ejemplo) y redirigimos la salida a un fichero. En este fichero debemos sustituir la ruta absoluta de los ficheros por la que deberían tener tras la instalación del paquete binario, esto es, un fichero con la ruta *directorio de construccion/ruta relativa del fichero* debería ser listado como /opt/R/*ruta relativa del fichero*. Esta sustitución la podemos automatizar haciendo uso del editor de textos para línea de comandos *sed*. Ademas debemos indicar los ficheros de documentación mediante etiquetas *%doc* o mediante *%docdir* seguido del nombre de su directorio.

Llegados a este punto podemos completar totalmente el fichero .spec y ejecutar

```
rpmbuild -ba R2.8.0-0.spec
```

para construir los paquetes de fuentes y binario para R. Si todo va bien, estos los encontraremos en los subdirectorios SRPMS y RPMS de /usr/src/redhat respectivamente.

A.3. Construcción de un paquete binario para R compilado con los compiladores Intel

El proceso para construir el paquete R compilado con los compiladores de Intel es muy parecido: podemos tomar como punto de partida el fichero .spec generado para la construcción del paquete compilando con los compiladores GNU y realizar las modificaciones oportunas. En cualquier caso, sigue siendo necesario realizar una instalación manual para descubrir estas modificaciones.

Para poder utilizar los compiladores de C y Fortran de intel, icc e ifort respectivamente, debemos ejecutar en el shell los comandos de configuración

de variables

```
. /opt/intel/Compiler/11.1/046/bin/iccvars.sh ia32
. /opt/intel/Compiler/11.1/046/bin/ifortvars.sh ia32
```

por lo que será necesario incluirlos en las secciones *%build* y *%install*. Además debemos indicarle al script *./configure* que queremos que use los compiladores de Intel, lo cual lo hacemos estableciendo los variables adecuados en las variables CC, CXX y F77 del fichero config.sites. Por ultimo, también será necesario inicializar la variable *LD_LIBRARY_PATH* al valor del directorio que contiene la librería libimf.so para que el instalador pueda encontrarla. Nuestras secciones *%build* y *%install* quedan así:

```
%build
. /opt/intel/Compiler/11.1/046/bin/iccvars.sh ia32
. /opt/intel/Compiler/11.1/046/bin/ifortvars.sh ia32
export LD_LIBRARY_PATH=/opt/intel/Compiler/11.1/046/lib/ia32/
echo "CC=icc" >./config.site
echo "CXX=icc" >>./config.site
echo "F77=ifort" >>./config.site
echo "FC=ifort" >>./config.site
./configure --enable-R-shlib --prefix=/opt/R
make
%install
rm -rf $RPM_BUILD_ROOT
. /opt/intel/Compiler/11.1/046/bin/iccvars.sh ia32
. /opt/intel/Compiler/11.1/046/bin/ifortvars.sh ia32
export LD_LIBRARY_PATH=/opt/intel/Compiler/11.1/046/lib/ia32/
make install DESTDIR=$RPM_BUILD_ROOT
```

Debemos ademas modificar la sección cabecera para eliminar las dependencias con gcc, g++ y gfortran y sustituirlas con *intel-cproc111046e* e *intel-cprof111046e*, que se corresponden con las versiones instaladas en nuestro entorno de los compiladores de C y Fortran de Intel, quedando esta sección así:

```
Summary: Free software environment for statistical computing and graphics
Name: R
Version: 2.8.0
Release: 10
Source: http://cran.es.r-project.org/src/base/R-2/R-2.8.0.tar.gz
License: GPL
Group: Development/Tools
BuildRoot: %{_builddir}/%{name}-root
BuildRequires: readline-devel  libX11-devel libXt-devel libstdc++
intel-cproc111046e intel-cprof111046e
Requires: readline >= 5.1, texinfo >= 4.8
```

Con estas modificaciones realizadas en el fichero guardado como Ricc2.8.0.0-0.spec, podemos ejecutar *rpmbuild -ba Ricc2.8.0-0.spec* para construir los paquetes de fuentes y binario para R.

Apéndice B

Guía de operación de la plataforma Rightscale

Este anexo constituye una guía de utilización de la plataforma Rightscale y esta organizada en función de los tres elementos de la misma que se utilizan en este proyecto.

B.1. Plantillas

Las plantillas nos permiten configurar máquinas tipo, lo cual en un entorno donde casi todos los nodos tienen una configuración similar es muy útil. Para crear una plantilla, elegiremos la opción *New* del submenú *ServerTemplates* del menú *Design*.

La plataforma nos presentará un formulario, el de la figura B.1, en el que introduciremos un nombre y una descripción para la plantilla, así como el tipo de instancia y la imagen que queremos que se cargue al arrancar una instancia de esta plantilla.

Podemos ver las plantillas que hemos creado, como en la figura B.2, pulsando la opción *View all* del submenú *ServerTemplates* del menú *Design*. Desde ahí podemos lanzar servidores configurados con esas plantillas y eliminarlas. Pulsando en el nombre de la plantilla accedemos a una ventana con varias pestañas:

- En la pestaña Info encontramos información de la plantilla y podemos editarla.
- En la pestaña Cloud, mostrada en la figura B.3, se nos muestra entre otras cosas el tipo de instancia y ahí podemos cambiarlo.
- La pestaña Script, mostrada en la figura B.4 nos permite ver los scripts de arranque y operacionales configurados para la plantilla, así como añadir nuevos o eliminarlos.

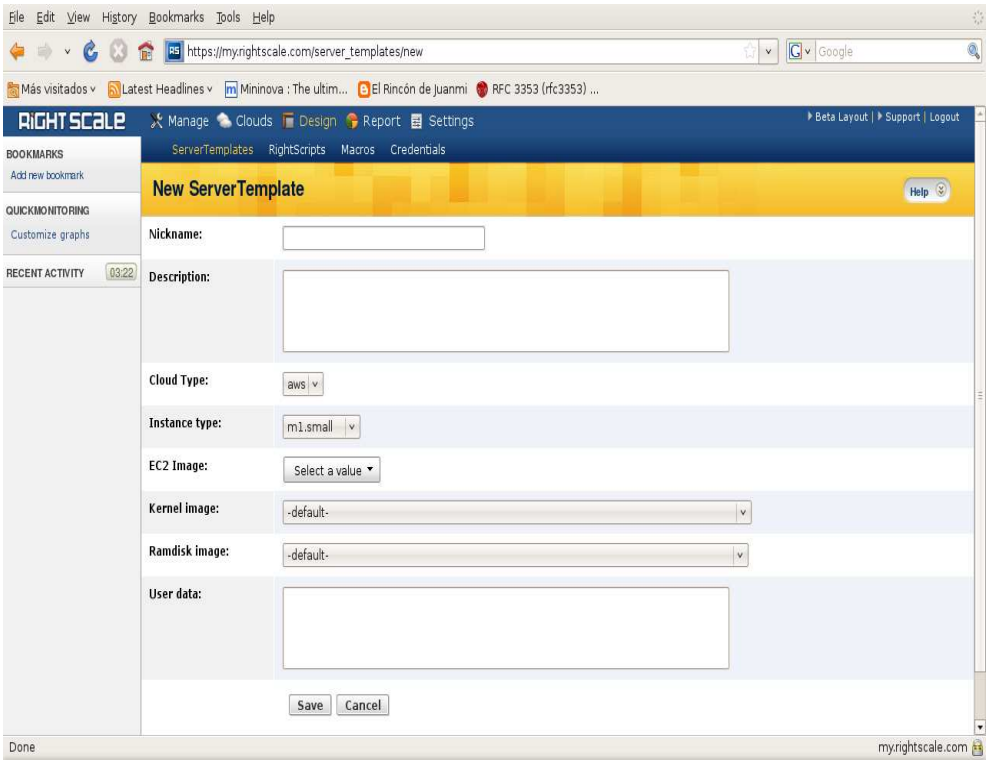

Figura B.1: Creación de una plantilla en Rightscale

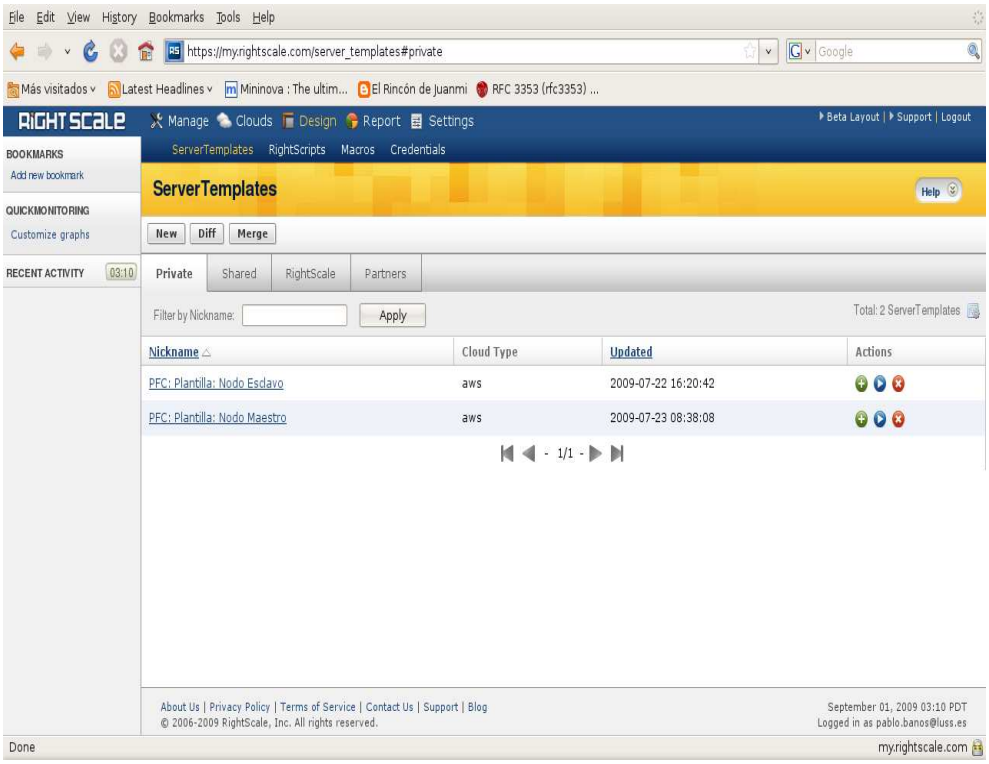

Figura B.2: Nuestras plantillas en Rightscale

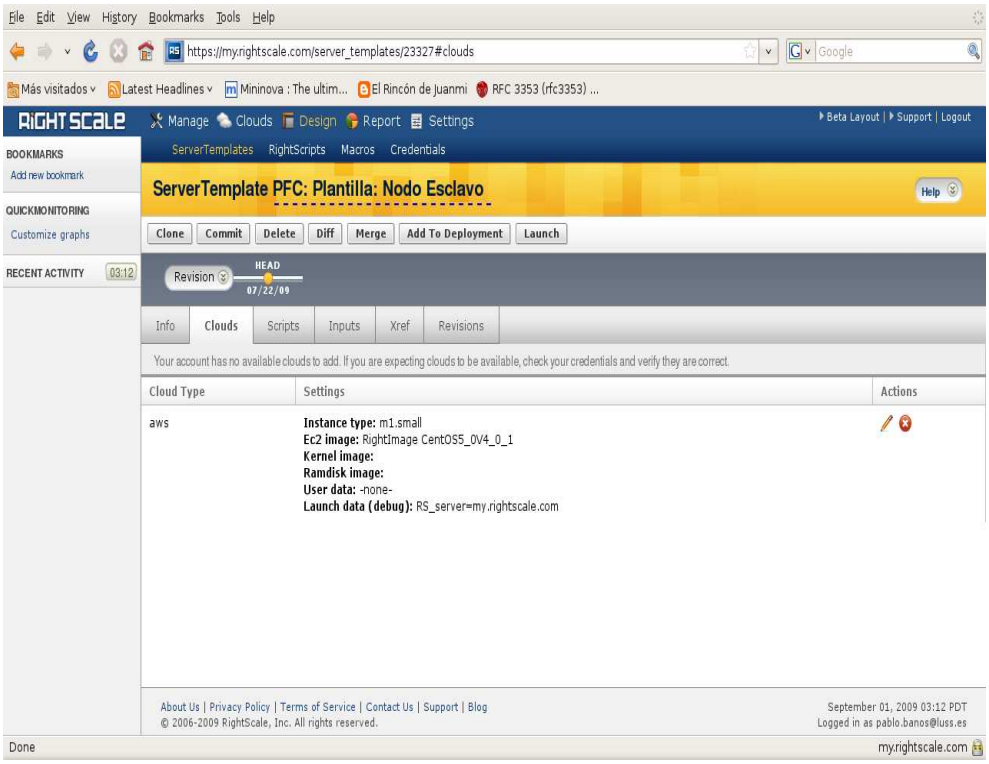

Figura B.3: Pestaña Cloud de una plantilla

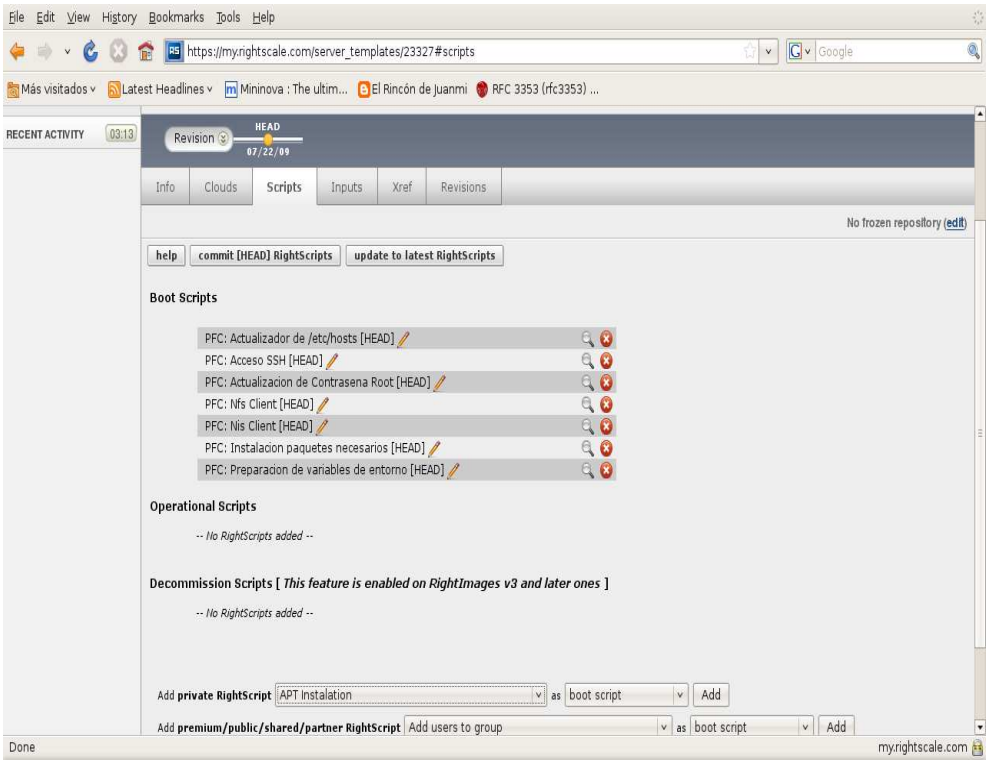

Figura B.4: Pestaña Scripts de una plantilla

B.2. Scripts

Los scripts nos permiten, desde el entorno, automatizar tareas en las instancias de nuestro entorno en la nube.

Para crear un Script elegiremos la opción *New* del submenú *Rightscripts* del menú *Design*, que nos presentará un formulario como el de la figura B.5. Rellenaremos sus campos, solo siendo importante destacar que, al pulsar el botón *Identify*, la plataforma es capaz de detectar las variables que utilicemos y ofrecernos su inicialización cuando lancemos un script.

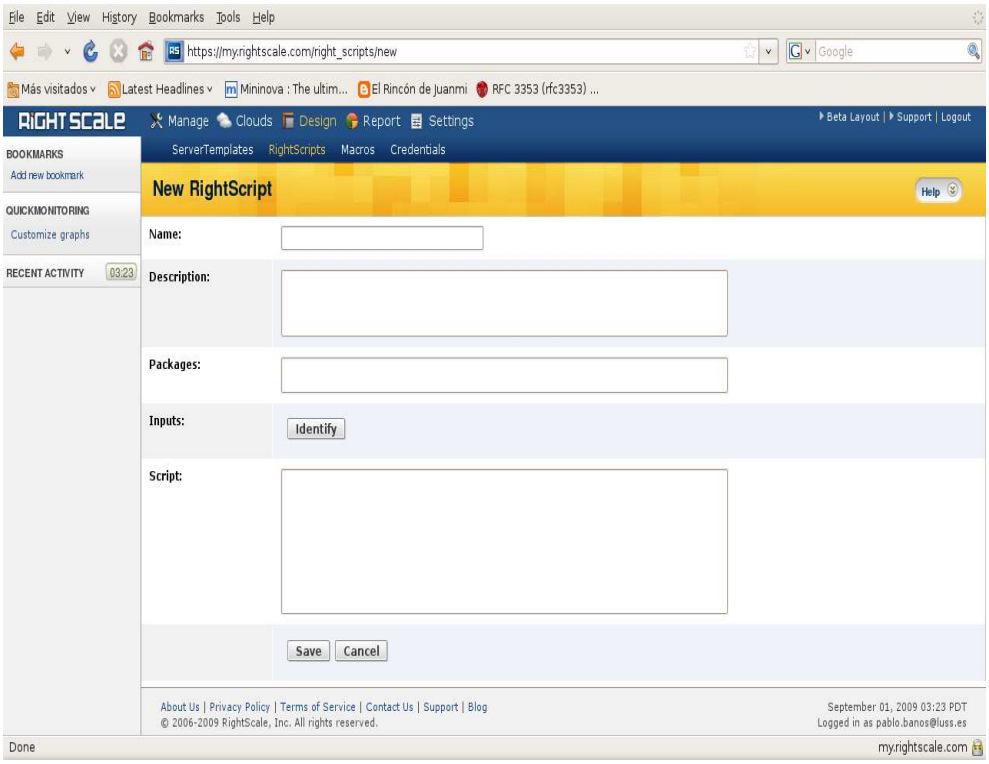

Figura B.5: Creación de un script en Rightscale

Si pulsamos la opción *View all* del submenú *Rightscripts* del menú *Design* veremos todos los scripts creados por nosotros, siendo posible su eliminación desde esa pantalla. Pulsando sobre uno de ellos podemos seguir configurándolo, disponiendo para ello de varias pestañas, entre ellas:

- En la pestaña Script, mostrada en la figura B.6, podemos editar su contenido.
- En la pestaña Attachment, mostrada en la figura B.7, podemos añadir ficheros adjuntos para su utilización en el script.[1]

[1] La variable *$ATTACH_DIR* en el interior de un script de Rightscale estará inicializada

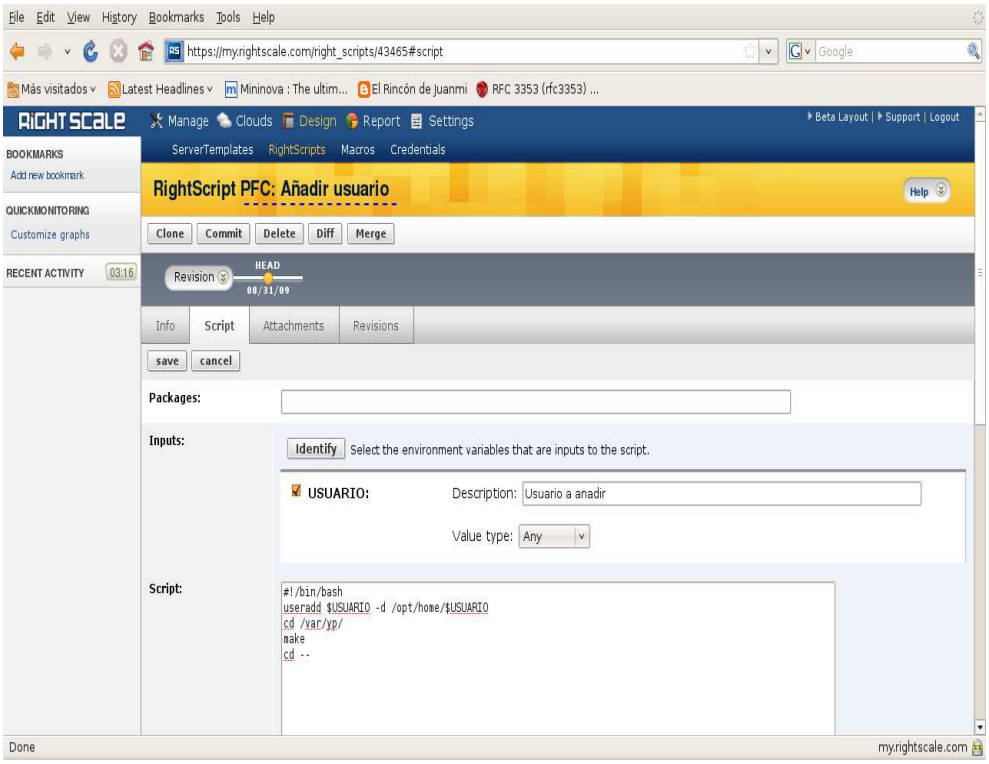

Figura B.6: Pestaña Script de un script

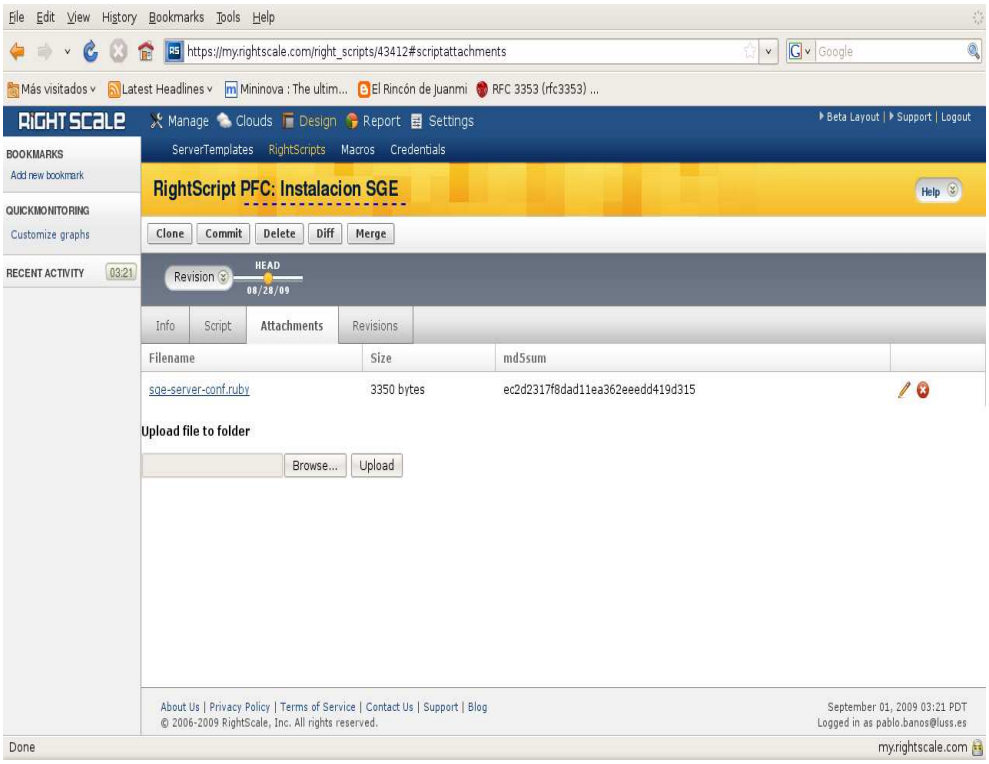

Figura B.7: Pestaña Attachments de un script

B.3. Instancias

Podemos ver las instancias que hemos creado pulsando la opción *View all* del submenú *Deployments* del menú *Manage* y, luego, haciendo clic en el entorno o *deployment* donde las hayamos situado, en nuestro caso en *Default*. Desde ahí, en una ventana similar a la de la figura B.8, podemos lanzar las instancias y pararlas.

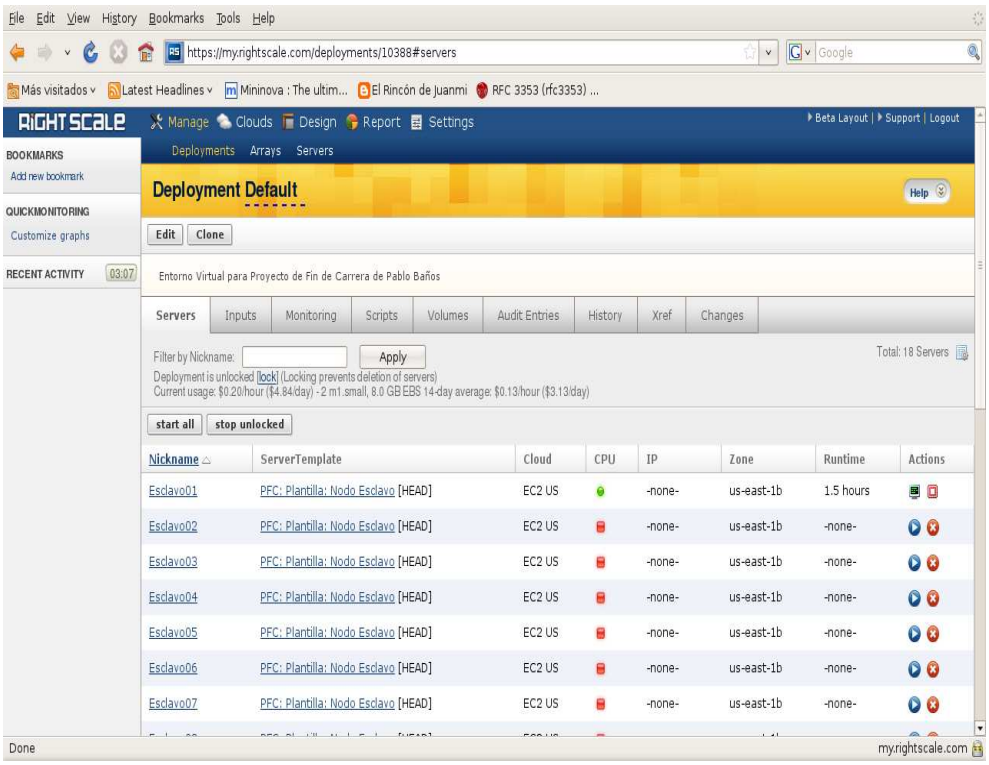

Figura B.8: Las instancias de nuestro entorno en Rightscale

Si hacemos clic en el nombre de una llegamos a una ventana con varias pestañas entre las que destacan:

- La pestaña Info, mostrada en la figura B.9, en la que podemos obtener un nombre DNS asociado a la dirección IP externa de la instancia, y
- La pestaña Scripts, mostrada en la figura B.10 donde podemos ejecutar scripts sobre la instancia.

al directorio en el que la plataforma situe los ficheros adjuntos de este modo.

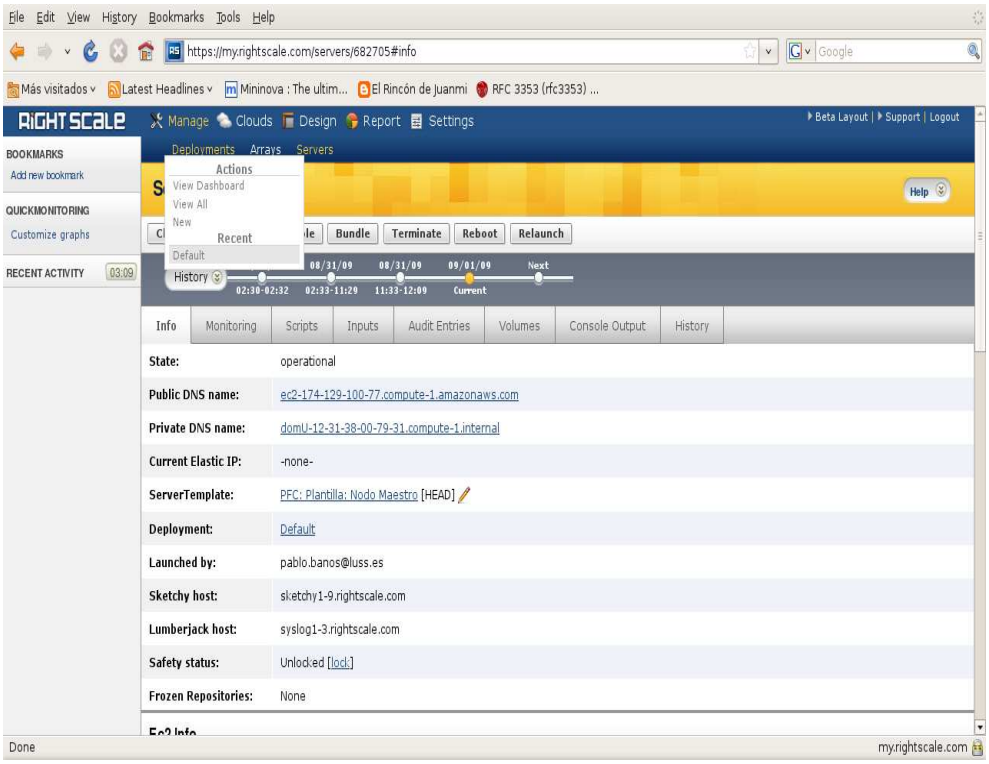

Figura B.9: La pestaña Info de una instancia

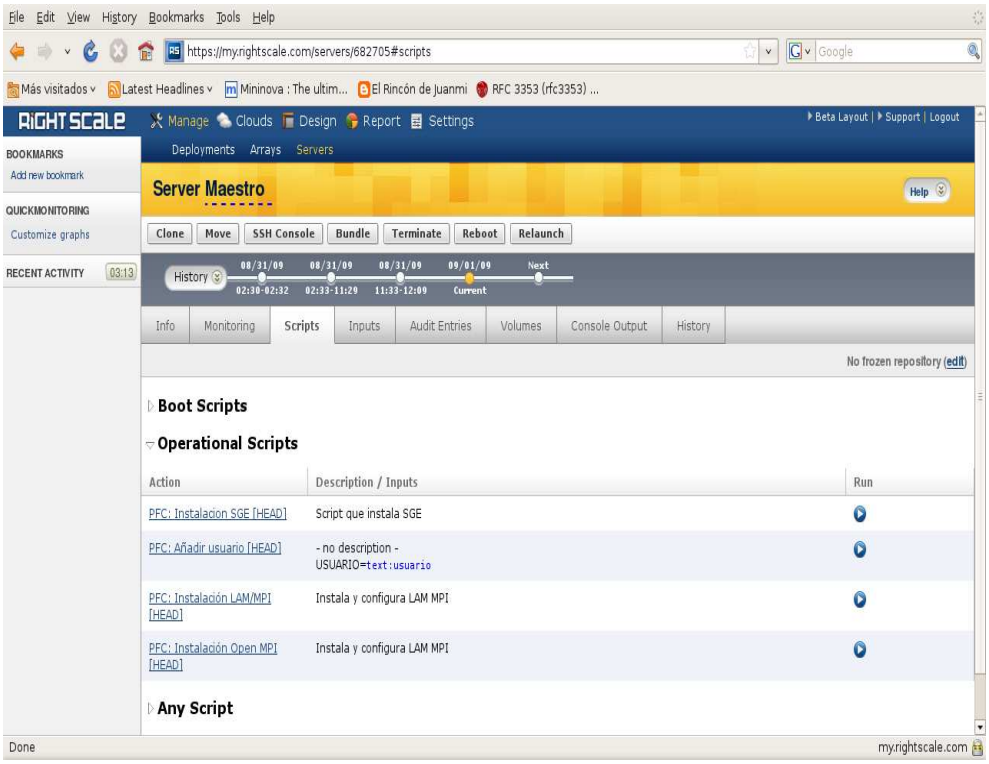

Figura B.10: La pestaña Scripts de una instancia

Apéndice C

Scripts asociados a la plantilla *PFC: Plantilla Nodo Maestro*

C.1. Scripts de arranque

C.1.1. PFC: Actualizador de /etc/hosts

Este script configura el fichero /etc/hosts de la instancia. Recibe como parámetro TYPE el rol que adoptará en el cluster, esto es, si será un nodo maestro, recibiendo el parámetro *master* o un nodo esclavo, recibiendo el parámetro *slave*. Cabe destacar que este script registra la instancia en una base de datos mantenida por el script php hosts.php que se analiza en el anexo E, y del cual este script obtiene una relacion de nombres y direcciones IP de los nodos del cluster.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#

# Recoge variables de session
source "/var/spool/ec2/meta-data.sh"
alias mv="mv -f"

if [ ! -e "/usr/bin/links" ]; then
   rpm -ivh $ATTACH_DIR/elinks-0.11.1-5.1.0.1.el5.i386.rpm
fi

# Registra Maquina
echo "Registro de Maquina"
```

```
echo "Configuring Typo:$TYPE IP: $EC2_LOCAL_IPV4)"
links --source http://www.luss.es/pfc/hosts.php?type=$TYPE
\&ip=$EC2_LOCAL_IPV4
links --source http://www.luss.es/pfc/hosts.php?type=$TYPE
\&ip=$EC2_LOCAL_IPV4

# Obtiene Fichero /etc/hosts
echo "Actualizacion de /etc/hosts"
links --source http://www.luss.es/pfc/hosts.php  >> /tmp/hosts
mv /tmp/hosts /etc/hosts

# Cambiamos el Nombre del Hosts
echo "Cambio de Hostname"
HOSTNAME=‘cat /etc/hosts | grep $EC2_LOCAL_IPV4 | awk  \
’{ print $2 }’‘
echo "HOSTNAME=$HOSTNAME" > /etc/sysconfig/network
echo "NETWORKING=yes" >>  /etc/sysconfig/network
/bin/hostname $HOSTNAME
```

C.1.2. PFC: Acceso SSH

Este script configura la instancia para que pueda ser accedida desde el exterior de la nube por SSH.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#

cat <<EOF> /etc/ssh/sshd_config
Protocol 2
SyslogFacility AUTHPRIV
PermitRootLogin yes
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
UsePAM yes
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE
LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE
LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL
```

```
X11Forwarding yes
ClientAliveInterval 60
ClientAliveCountMax 240
Subsystem sftp /usr/libexec/openssh/sftp-server
UseDNS  no
PermitRootLogin without-password
EOF
service sshd restart
```

C.1.3. PFC: Actualizacion de Contrasena Root

Este script establece una contraseña para el superusuario.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
if [ ! -e "/usr/bin/expect" ]; then
   rpm -ivh $ATTACH_DIR/expect-5.43.0-5.1.i386.rpm
fi

/usr/bin/expect $ATTACH_DIR/root_password_change.expect
rm -f $ATTACH_DIR/root_password_change.expect
```

Este script hace uso del siguiente script expect para conseguir la redirección de la entrada/salida al cambiar la contraseña, pues ocurre, al usar muchos comandos relacionados con funciones de seguridad, que los intentos de redirección en guiones escritos en Bash son eludidos:

```
#!/usr/bin/expect
spawn passwd
expect "assword:"
send "Hola123\r"
expect "assword:"
send "Hola123\r"
expect eof
```

C.1.4. PFC: Montaje de Disco Persitente de Nodo Maestro

Este script monta el disco de almacenamiento permanente en el directorio /opt y tralada el directorio /home a él.

```
#!/bin/bash
#
```

```
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#

# Monta disco Persistente en Nodo Maestro
mount /dev/sdj1 /opt/

# Crea enlace simbolico a nuevo Home
rm -r -f /home
ln -s /opt/home  /home
```

C.1.5. PFC: Nfs Server

Este script configura la exportación NFS del directorio /opt.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#

cat <<EOF> /etc/exports
/opt 10.0.0.0/255.0.0.0(rw,sync,no_root_squash)
EOF
service portmap restart
service nfs restart
```

C.1.6. PFC: Nis Server

Este script configura el nodo maestro como servidor NIS.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
alias mv="mv -f"

yum -y install ypserv
ypdomainname pfc.luss.es
cat /etc/sysconfig/network | grep -v NISDOMAIN > /tmp/network
echo "NISDOMAIN=pfc.luss.es" >> /tmp/network
```

```
mv /tmp/network /etc/sysconfig/network

cat <<EOF> /etc/yp.conf
# /etc/yp.conf - ypbind configuration file
# Valid entries are
#
# domain NISDOMAIN server HOSTNAME
# Use server HOSTNAME for the domain NISDOMAIN.
#
# domain NISDOMAIN broadcast
# Use  broadcast  on  the local net for domain NISDOMAIN
#
# domain NISDOMAIN slp
# Query local SLP server for ypserver supporting NISDOMAIN
#
# ypserver HOSTNAME
# Use server HOSTNAME for the  local  domain.  The
# IP-address of server must be listed in /etc/hosts.
#
ypserver 127.0.0.1
# broadcast
# If no server for the default domain is specified or
# none of them is rechable, try a broadcast call to
# find a server.
#
EOF

service yppasswdd start
service ypservers start
service ypserv start

sed -i "s/is_correct=F/is_correct=T/" /usr/lib/yp/ypinit
/usr/lib/yp/ypinit -m

service ypbind start
service ypxfrd start

cd /var/yp/
make
cd --
ypmatch root -x
```

C.1.7. PFC: Ntp Client

Este script configura la instancia como cliente ntp.

```
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
yum -y install ntp
service ntpd restart
```

C.1.8. PFC: Instalacion paquetes necesarios

Este script instala paquetes utilizados por R y Rmpi.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
yum install -y  compat-libstdc++-33 libX11-devel \
libXt-devel texinfo readline-devel
```

C.1.9. PFC: Preparacion de variables de entorno

Este script configura la instancia para que el directorio donde se encuentran los ejecutables de R este en el PATH en cualquier cuenta.

```
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
echo "PATH=$PATH:/opt/R/bin" >>/etc/profile
echo "export PATH" >>/etc/profile
echo "PATH=$PATH:/opt/R/bin" >>/root/.bash_profile
echo "export PATH" >>/root/.bash_profile
```

C.2. Scripts operacionales

C.2.1. PFC: Instalacion SGE

Este script realiza la instalación del gestor de colas Sun Grid Engine, actualizando previamente el fichero /etc/hosts.

```
#!/bin/bash#
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
alias rm="rm"
# Actualizamos Listado de Maquinas
links --source http://www.luss.es/pfc/hosts.php  >> /tmp/hosts
mv /tmp/hosts /etc/hosts

# Comprobamos a ver cual de esas maquinas estan en pie
cat /etc/hosts | grep -v "#" | grep -v localhost | awk '{ print $3 }' \
> /tmp/hosts
fping -a -f /tmp/hosts > /tmp/up
rm -f /tmp/hosts

# Escribimos fichero de configuracion cluster.conf
cat <<EOF> /tmp/cluster.conf
<configuration client="Pablo">
        <cluster name="um" type="sge">
                <info code="ES" state="Murcia" location="Luss"
email="" organization="Luss" unit="PFC"/>

                <software name="sge" root="/opt/sge"
qmaster_port="6444" qexecd_port="6445"/>

                <admin user="sgeadmin" email="test@luss.es"/>

                <queues>
                        <queue name="rapida" options=""/>
                        <queue name="batch" options=""/>
                </queues>
                <hosts>
                        <master hostname="master" domain="pfc.luss.es"
type="static"/>
                        <submit hostname="master" domain="pfc.luss.es"
type="static"/>
EOF

cat /tmp/up | grep slave | awk '{ printf "\t\t\t<slave
hostname=\"%s\" domain=\"pfc.luss.es\"
type=\"static\"/>\n",$1 }' >> /tmp/cluster.conf
```

```
cat <<EOF>> /tmp/cluster.conf
                </hosts>
        </cluster>
</configuration>
EOF


# Script que sirve para poder intercambiar claves ssh sin problemas
cat <<EOF> /tmp/ssh_copy_id.expect
#!/usr/bin/expect -f

set user [lrange \$argv 0 0]
set password [lrange \$argv 1 1]
set host [lrange \$argv 2 2]

spawn ssh-copy-id -i /\$user/.ssh/id_rsa.pub \$user@\$host
expect {
"#" { }
"$ " { }
"assword: " {
send "\$password\n"
expect {
"# " { }
"$ " { }
}
}
}
EOF
chmod 777 /tmp/ssh_copy_id.expect

# Empezamos con el intercambio de llaves
rm -f /root/.ssh/id_rsa
ssh-keygen -N "" -t rsa -f /root/.ssh/id_rsa
rm -f /root/.ssh/known_hosts
ssh-keyscan -t rsa -f /tmp/up > /root/.ssh/known_hosts

HOSTS=‘cat /tmp/up‘
for HOST in $HOSTS
do
ssh-keyscan -t rsa $HOST
echo "-----------------------"
echo $HOST
echo "......................."
/tmp/ssh_copy_id.expect root Hola123 $HOST
```

```
   echo "-----------------------"
   ssh-keyscan -t rsa $HOST.pfc.luss.es >> /root/.ssh/known_hosts
done

# Procedemos con la instalacion del cluster

USUARIO=‘cat /tmp/cluster.conf | grep admin | awk -F \
\" ’{ print $2 }’‘
GRUPO=‘cat /tmp/cluster.conf | grep cluster | grep \
name | awk -F \" ’{ print $2 }’‘

export SGE_ROOT=‘cat /tmp/cluster.conf | grep software \
| awk -F \" ’{ print $4 }’‘

#creacion de usuario
useradd $USUARIO -d /opt/home/$USUARIO
echo -e "Hola123\nHola123" | passwd $USUARIO
cd /var/yp/
make
cd --

#creacion del directorio de instalacion
chown -R $USUARIO $SGE_ROOT

#creacion de fichero de configuracion para la
#instalacion automatica del maestro sge
ruby $ATTACH_DIR/sge-server-conf.ruby > $SGE_ROOT/auto_qmaster.conf

#cambio de propietario y permisos de los ficheros de instalacion
find $SGE_ROOT -exec chown $USUARIO ’{}’ \; -exec chmod g+w ’{}’ \;

#registro de puertos en /etc/services
cat /etc/services| egrep -v "(sge_qmaster|sge_execd)" \
 >/tmp/services
echo -e "sge_qmaster\t6444/tcp\t#SGE Master" >>/tmp/services
echo -e "sge_execd\t6445/tcp\t#SGE Execution" >>/tmp/services
mv /tmp/services /etc/services
rm -r -f  /opt/sge/um/
rm /etc/profile.d/sge-settings.sh
#

#instalacion
cd $SGE_ROOT
$SGE_ROOT/inst_sge -m -x -auto $SGE_ROOT/auto_qmaster.conf
```

```
#carga de variables de sge al inicio de cada sesion
cp $SGE_ROOT/$GRUPO/common/settings.sh \
/etc/profile.d/sge-settings.sh
```

Este script se apoya en el siguiente script ruby, sge-server-conf.ruby, para crear el fichero de configuración de la instalación de Sun Grid Engine:

```
#!/usr/bin/ruby
#####################################################################
#                                                                   #
# Code obtained from oreilly at                                     #
# http://www.onlamp.com/pub/a/onlamp/2004/08/12/ruby_e4x.html       #
#                                                                   #
#####################################################################

require 'rexml/document'
class NodeWrapper
   def method_missing( name, *args )
       name = name.to_s
       if ( name =~ /^_/ )
           name.gsub!( /^_/, "" )
           return @node.attributes[ name ].to_s
       else
           out = NodeListWrapper.new()
           @node.each_element( name ) { |elem|
               out.push( NodeWrapper.new( elem ) )
           }
           return out
       end
   end

   def initialize( node )
       @node = node
   end

   def to_s() @node.to_s; end

   def to_i() @node.to_i; end

end

class NodeListWrapper < Array
   def method_missing( name, *args )
```

```
        name = name.to_s
        self[0].send( name, args )
    end
end

#############################################################
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
xml = NodeWrapper.new( REXML::Document.new(
File.open( '/tmp/cluster.conf' ) ) )

exec_host = ""

xml.configuration.cluster.hosts.slave.each { |temp|
exec_host << temp._hostname+" "
}
exec_host =  exec_host.chop
puts "SGE_ROOT=\"#{xml.configuration.cluster.software._root}\" "
puts "SGE_QMASTER_PORT=\"#{xml.configuration.cluster.software.
                                              _qmaster_port}\""
puts
 "QMASTER_SPOOL_DIR=\"#{xml.configuration.cluster.software._root}
                      /#{xml.configuration.cluster._name}/spool\""
puts "SGE_EXECD_PORT=\"#{xml.configuration.cluster.software.
                                              _qexecd_port}\""
puts "EXECD_SPOOL_DIR=\"#{xml.configuration.cluster.software._root}
               /#{xml.configuration.cluster._name}/execd/spool\""
puts "CELL_NAME=\"#{xml.configuration.cluster._name}\""
puts "ADMIN_USER=\"#{xml.configuration.cluster.admin._user}\""
puts "GID_RANGE=\"20000-20100\""
puts "SPOOLING_METHOD=\"berkeleydb\""
puts "DB_SPOOLING_SERVER=\"none\""
puts "DB_SPOOLING_DIR=\"#{xml.configuration.cluster.software._root}
                     /#{xml.configuration.cluster._name}/dbspool\""
puts "PAR_EXECD_INST_COUNT=\"20\""
puts "ADMIN_HOST_LIST=\"#{xml.configuration.cluster.
                            hosts.master._hostname}\""
puts "SUBMIT_HOST_LIST=\"#{xml.configuration.cluster.
                            hosts.submit._hostname}\""
puts "EXEC_HOST_LIST=\"#{exec_host}\""
puts "EXECD_SPOOL_DIR_LOCAL=\"\""
```

```
puts "HOSTNAME_RESOLVING=\"true\""
puts "SHELL_NAME=\"ssh\""
puts "COPY_COMMAND=\"scp\""
puts "DEFAULT_DOMAIN=\"#{xml.configuration.cluster.
                          hosts.master._domain}\""
puts "ADMIN_MAIL=\"#{xml.configuration.cluster.
                                admin._email}\""
puts "ADD_TO_RC=\"true\""
puts "SET_FILE_PERMS=\"true\""
puts "RESCHEDULE_JOBS=\"wait\""
puts "SCHEDD_CONF=\"1\""
puts "SHADOW_HOST=\"\""
puts "EXEC_HOST_LIST_RM=\"#{exec_host}\""
puts "REMOVE_RC=\"false\""
puts "WINDOWS_SUPPORT=\"false\""
puts "WIN_ADMIN_NAME=\"Administrator\""
puts "WIN_DOMAIN_ACCESS=\"false\""
puts "CSP_RECREATE=\"true\""
puts "CSP_COPY_CERTS=\"false\""
puts "CSP_COUNTRY_CODE=\"ES\""
puts "CSP_STATE=\"Spain\""
puts "CSP_LOCATION=\"Building\""
puts "CSP_ORGA=\"UMU\""
puts "CSP_ORGA_UNIT=\"ATICA\""
puts "CSP_MAIL_ADDRESS=\"#{xml.configuration.cluster.
                                  admin._email}\""
```

C.2.2. PFC: Añadir usuario

Este sencillo script añade un usuario y actualiza los mapas del servicio NIS.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
useradd $USUARIO -d /opt/home/$USUARIO
cd /var/yp/
make
cd --
```

C.2.3. PFC: Instalación LAM/MPI

Este script instala los paquetes de LAM/MPI y lo configura.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
echo "master.pfc.luss.es" >/opt/mpi/lamhosts.def
cat /tmp/up | grep -v master | awk '{print $0 " cpu=2"}' \
                                  >>/opt/mpi/lamhosts.def
HOSTS=`cat /tmp/up`
for HOST in $HOSTS
do
   ssh $HOST yum install -y lam lam-libs lam-devel
   ssh $HOST cp /opt/mpi/lamhosts.def /etc/lam/lam-bhost.def
done
```

C.2.4. PFC: Configuración de Open MPI

Este script configura el entorno para la utilización de Open MPI.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
cat /tmp/up |awk '{print $0}' >/opt/mpi/openhosts.def

cat /opt/home/sgeadmin/.bashrc | grep -v "PATH=/opt/openmpi/bin" \
        | grep -v "LD_LIBRARY_PATH=/opt/openmpi/lib" >/tmp/bashrc
cat /tmp/bashrc >/opt/home/sgeadmin/.bashrc
echo "export PATH=/opt/openmpi/bin:$PATH" \
                                     >>/opt/home/sgeadmin/.bashrc
echo "export LD_LIBRARY_PATH=/opt/openmpi/lib:$LD_LIBRARY_PATH" \
                                     >>/opt/home/sgeadmin/.bashrc
```

Apéndice D

Scripts asociados a la plantilla *PFC: Plantilla Nodo Esclavo*

D.1. Scripts de arranque

D.1.1. PFC: Actualizador de /etc/hosts

Se trata del mismo script analizado en el anexo anterior.

D.1.2. PFC: Acceso SSH

Se trata del mismo script analizado en el anexo anterior.

D.1.3. PFC: Actualización de Contraseña Root

Se trata del mismo script analizado en el anexo anterior.

D.1.4. PFC: Nfs Client

Este script configura la instancia para que importe por NFS el directorio /opt del nodo maestro y sitúa su directorio home en él.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
service portmap start
echo "master.pfc.luss.es:/opt  /opt  nfs  defaults 1 2"
                                          >> /etc/fstab
mount -a
rm -r -f /home
```

```
ln -s /opt/home /home
```

D.1.5. PFC: Nis Client

Este script configura la instancia como cliente NIS del nodo maestro.

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
alias mv="mv -f"

ypdomainname pfc.luss.es
cat /etc/sysconfig/network | grep -v NISDOMAIN > /tmp/network
echo "NISDOMAIN=pfc.luss.es" >> /tmp/network
mv /tmp/network /etc/sysconfig/network

cat <<EOF> /etc/yp.conf
# /etc/yp.conf - ypbind configuration file
# Valid entries are
#
# domain NISDOMAIN server HOSTNAME
# Use server HOSTNAME for the domain NISDOMAIN.
#
# domain NISDOMAIN broadcast
# Use  broadcast  on  the local net for domain NISDOMAIN
#
# domain NISDOMAIN slp
# Query local SLP server for ypserver supporting NISDOMAIN
#
# ypserver HOSTNAME
# Use server HOSTNAME for the  local  domain.  The
# IP-address of server must be listed in /etc/hosts.
#
ypserver master.pfc.luss.es
# broadcast
# If no server for the default domain is specified or
# none of them is rechable, try a broadcast call to
# find a server.
#
EOF
echo "+::::::" >> /etc/passwd
```

```
cat "+:::" >> /etc/group
sed -i "s/files/files nis/" /etc/nsswitch.conf
service ypbind start
```

D.1.6. PFC: Ntp Client

Este script, que configura la instancia como cliente ntp, ya ha sido comentado en el anexo anterior.

D.1.7. PFC: Instalacion paquetes necesarios

Este script, que instala paquetes utilizados por R y Rmpi, ya ha sido comentado en el anexo anterior.

D.1.8. PFC: Preparacion de variables de entorno

Este script, que configura la instancia para que el directorio donde se encuentran los ejecutables de R este en el PATH en cualquier cuenta de usuario, ya ha sido comentado en el anexo anterior.

Apéndice E

Otros scripts

E.1. hosts.php

En nuestro entorno el nodo maestro recibe el nombre master y el nodo esclavo i-ésimo el nombre slave*i*, todos ellos en el dominio pfc.luss.es. La configuración de un DNS para la gestión de los nombres del dominio, además de no ser sencilla, presentaría problemas de actualización, pues la extensión de las actualizaciones DNS puede requerir bastante tiempo. De esta forma tendríamos problemas cada vez que reconfiguraramos el cluster.

Por estas razones, en lugar de configurar un servicio DNS, hemos preferido simularlo. Este script, alojado en www.luss.es/pfc, mantiene una base de datos sqlite, y permite el registro en ella de un nodo, que debe indicar el tipo de nodo (*master* o *slave*) y dirección IP. Al invocarlo sin estos parametros devuelve un fichero /etc/hosts en el que asigna el nombre master al nodo maestro del cluster y el nombre slave*i* al i-ésimo nodo esclavo que se registró. Utilizando este script podremos mantener la información de dominio en todos los nodos y forzar su actualización cuando lo estimemos necesario[1].

```
<?php
$db= new SQLiteDatabase('/var/www/pfc/hosts.db',
                                    0666, $error);
if (!$db) {
    $error = (file_exists('/var/www/pfc/hosts.db'))
      ? "Impossible to open, check permissions" :
        "Impossible to create, check permissions";
    die($error);
}

if(isset($_GET['type']) && isset($_GET['ip']))
{
```

[1]Cuando se instala SGE, por ejemplo.

```
$q = $db->query("SELECT * FROM hosts WHERE ip='".
        $_GET['ip']."'", SQLITE_ASSOC, $query_error);
if ($query_error)
die("Error: $query_error");
if (!$q)
die("Impossible to execute query.");
if($_GET['type']=='master')
{
$q2 = $db->query("SELECT * FROM hosts WHERE
         type='master'", SQLITE_ASSOC, $query_error);
if ($query_error)
die("Error: $query_error");
if (!$q2)
die("Impossible to execute query.");
if($q->numRows()==0)
if($q2->numRows()==0)
$query="INSERT INTO hosts VALUES
                    ('master','".$_GET['ip']."')";
else
$query="UPDATE hosts SET
          ip='".$_GET['ip']."' WHERE type='master'";
else
if($q2->numRows()==0)
$query="UPDATE hosts SET type='master'
                        WHERE ip='".$_GET['ip']."'";
else
{
$row = $q2->fetch();
if($row['ip']!=$_GET['ip'])
{
$db->query("DELETE FROM hosts
                              WHERE type='master'",
                      SQLITE_ASSOC, $query_error);
if ($query_error)
die("Error: $query_error");
$query="UPDATE hosts SET type='master'
                         WHERE ip='".$_GET['ip']."'";
}
else
exit("No es necesario hacer cambios");
}
}
else
if($q->numRows()==0)
```

```
$query="INSERT INTO hosts VALUES ('slave',
                                    '".$_GET['ip']."')";
else
{
$row = $q->fetch();
if($row['type']=='master')
$query="UPDATE hosts SET type='slave'
                            WHERE ip='".$_GET['ip']."'";
else
exit("No es necesario hacer cambios");
}
$db->query($query,SQLITE_ASSOC, $query_error);
if ($query_error)
die("Error: $query_error");
echo "Cambios introducidos correctamente";
}
else
{
$query = $db->query("SELECT * FROM hosts",
                  SQLITE_ASSOC, $query_error);
if ($query_error)
die("Error: $query_error");
if (!$query)
die("Impossible to execute query.");
?>##################################################
# Fichero generado automaticamente #
##################################################
127.0.0.1 localhost localhost.localdomain
<?php
$n=0;
while ($row = $query->fetch())
{
if($row['type']=='slave')
{
$n++;
$name='slave'.$n;
}
else
$name='master';
print($row['ip']."\t".$name."\t".
                    $name.".pfc.luss.es\n");
}
}
?>
```

E.2. confianza-ssh.sh

A la hora de utilizar las interfaces de intercambio de mensajes cuyo rendimiento estudiamos, nos encontramos con que debemos hacerlo desde una cuenta de usuario sin privilegios: en el caso de LAM/MPI porque no nos permite ejecutar *lamboot* para lanzar el entorno con la cuenta de superusuario y, en el caso de Open MPI porque la configuración necesaria la realizamos para la cuenta de usuario *sgeadmin*[2]. Necesitamos de todas formas tener configurado entre el nodo maestro y los nodos esclavo el acceso por SSH sin necesidad de contraseña. Esto último es lo que, para el usuario que lo ejecuta, realiza este script:

```
#!/bin/bash
#
# Copyright (c) 2009 Pablo Banos Lopez,
# Javier Perez-Griffo Callejon,
# All Rights Reserved Worldwide.
#
# Script que sirve para poder intercambiar claves ssh sin problemas
cat <<EOF> /tmp/ssh_copy_id-opt.expect
#!/usr/bin/expect -f

set user [lrange \$argv 0 0]
set password [lrange \$argv 1 1]
set host [lrange \$argv 2 2]

spawn ssh-copy-id -i /opt/home/\$user/.ssh/id_rsa.pub \$user@\$host
expect {
"#" { }
"$ " { }
"assword: " {
send "\$password\n"
expect {
"# " { }
"$ " { }
}
}
}
```

[2]El usuario root es el único que no tiene el directorio home compartido por lo que realizar cambios en su fichero *.bashrc* hubiera sido más complicado. Hemos elegido el usuario sgeadmin por ser un usuario ya existente y porque así eliminamos la necesidad de crear uno

```
EOF
chmod 777 /tmp/ssh_copy_id-opt.expect

# Empezamos con el intercambio de llaves
rm -f $HOME/.ssh/id_rsa
ssh-keygen -N "" -t rsa -f $HOME/.ssh/id_rsa
rm -f $HOME/.ssh/known_hosts
ssh-keyscan -t rsa -f /tmp/up > $HOME/.ssh/known_hosts

HOSTS=‘cat /tmp/up‘
for HOST in $HOSTS
do
ssh-keyscan -t rsa $HOST
echo "-----------------------"
echo $HOST
echo "......................."
/tmp/ssh_copy_id-opt.expect $USER Hola123 $HOST
echo "-----------------------"
ssh-keyscan -t rsa $HOST.pfc.luss.es >> $HOME/.ssh/known_hosts
done
```

Bibliografía

[1] Acadia Centre for Mathematical Modelling and Computation: *Rmpi Tutorial*, http://math.acadiau.ca/ACMMaC/Rmpi/, accedida en 2009

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield: *Xen and the Art of Virtualization*, SOSP'03, Octubre 19-22 Bolton Landing, New York, USA, 2003

[3] Hernán Barrios Verdugo, Cristián Lucero Fuentes, Arturo Veras Olivo: *Computación en la Nube*, Universidad Técnica Federico de Santa María, 2009

[4] Iván Couto Vivas: *Sistema de computación masiva en Sun Grid*, Universitat Politècnica de Catalunya, 2009

[5] Todd Deshane, Demetrios Dimatos, Gary Hamilton, Madhujith Hapuarachchi, Wenjin Hu, Michael McCabe, Jeanna Neefe Matthews: *Performance Isolation of a Misbehaving Virtual Machine with Xen, VMware and Solaris Containers*, Clarkson University, 2006

[6] David González Aragón: *Trabajo Final de Carrera: Desarrollo de una plataforma de virtualización*, Universidad Politécnica de Cataluña, 2008

[7] Novell: *Virtualización en el centro de datos*, Informe técnico - Centro de datos

[8] Javier Panadero Martínez: *Entorno de desarrollo para clusters*, Universitat Autònoma de Barcelona, 2008

[9] Salvador Ramírez Flandes: *Implementación de un Sistema de Directorios LDAP para la Universidad de Concepción*, Universidad de la Concepción, 2001

[10] Pablo Sanz Mercado: *Instalación de CentOS 5*, Universidad Autónoma de Madrid, Accedido en 2009

[11] Miguel Toledano Ortega: *Proyecto Fin de Carrera: Adaptación a Grid Computing del módulo de cálculo de disponibilidad de satélites de un sistema de navegación*, Universidad Pontificia de Comillas, Madrid, 2007

[12] TOP 500, www.top500.org accedida en Julio de 2009

[13] Alejandro Mauricio Valdés Jiménez: *Integrando NIS y NFS*, http://deb.utalca.cl/public/imagenes/nisnfs.pdf, accedido en 2009

[14] VMware Inc.: *Virtualization Overview*, VMware White Paper, Porter Drive Palo Alto CA USA, 2006

[15] Wikimedia Foundation, Inc.: *Wikipedia.org, The Free Encyclopedia*, www.wikipedia.org, accedida en 2009

[16] Manuel Zaforas: *Implementación de un algoritmo evolutivo basado en MOS*, Universidad Autónoma de Madrid, 2008

[17] Universidad de Zaragoza *Información sobre R*, http://osluz.unizar.es/proyectos/r, accedida en 2009