Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Tecnologías de la Información

Implementación de un sistema de eficiencia energética en clusters de computación

Autor:
D. Sergio Delgado Álvarez
Tutor:
D. Iván Santos Tejido

A mi tía Pili
que tanto cariño y apoyo me ha dado.

Resumen

El objetivo de este trabajo se centra en la implementación de un sistema de eficiencia energética sobre un cluster de computación de alto rendimiento. A lo largo de este trabajo se realiza un estudio comparativo de las diferentes opciones en cuanto a sistemas de gestión energética, definiendo las diferentes características de cada uno de ellos. Además de esto se muestra el proceso de puesta en marcha de un cluster de computación basado en Rocks (sistema operativo basado en CentOS) así como de la implementación del sistema de gestión energética elegido.Se detallan una serie de mejoras implementadas en el sistema elegido, las cuales proporcionan nuevas funcionalidades y características. Finalmente se presentan una serie de conclusiones sobre la eficiencia energética en la computación, además de unas lineas de trabajo futuro complementarias a este trabajo.

Agradecimientos

Quisiera agradecer a varias personas la ayuda que me han prestado en la realización de este Trabajo de Fin de Grado. Entre ellas y en primer lugar a mi, a mi tutor, por todos los consejos y directrices que me ha dado a lo largo de este curso.

Me faltarían paginas para agradecer la ayuda de Alberto, siempre dispuesto a ayudarme desinteresadamente. Por ayudarme a escribir mejor y su apoyo durante estos años que son los que me han traído hasta aquí. En parte este trabajo también es tuyo. A Bayon, Anta, Mario, Paredes, Óscar, Príncipe, Dani y como no a mi tocayo Sergio, compañeros que siempre han estado ahí dispuestos a molestar y ayudar tanto como les fuera posible. Muchas gracias.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Introducción y motivación

Con el paso de los años, el avance de la computación trae consigo el aumento de la complejidad de los problemas que esta pretende resolver: modelos más complejos, mayor volumen de datos e información, tiempos de resolución y respuesta limitados, necesidad de cálculos más exactos, etcétera

La computación de alto rendimiento o HPC (de sus siglas en inglés High Performance Computing) surge de esta necesidad de potencia de computo demandada por problemas cada vez más complejos. Esta computación abarca un gran conjunto de tecnologías, como los supercomputadores o los clusters de computación y comunicación, apoyando su funcionamiento en los paradigmas de programación distribuida y paralela. Puede encontrarse más información sobre este tema en el apéndice A.

Desde la década de los 1990, tanto los centros de datos como los centros de computación se han vuelto herramientas hardware muy populares para todas aquellas aplicaciones que requieren un alto grado de cómputo o tratan con mucha información. Esta popularidad trae consigo un consumo energético desmesurado por parte de estas plataformas, lo cual deriva en una gran cantidad de contaminación que sigue incrementando la huella de carbono que estos centros producen. En Noviembre de 2007 la comunidad de la computación de alto rendimiento decidió complementar la famosa lista "Top500" [1] con la lista "Green500" [2], comparando las máquinas a partir de su rendimiento por vatio consumido, tratando así de reconocer de alguna forma a aquellos sistemas que se preocupan por las consecuencias de un consumo energético excesivo.

Los riesgos empresariales derivados de la contaminación son globales e irreversibles. Es por ello que la eficiencia energética es uno de los mayores desafíos que las industrias y empresas afrontan a día de hoy. Tanto es así, que muchas empresas ya buscan formas de reducir su huella de carbono, puesto que la consciencia ambiental de las mismas se esta volviendo un factor muy a tener en cuenta para clientes e inversores.

En 2014 los centros de datos y computación de Estados Unidos consumieron entorno a 70 billones de kilovatios por hora, lo que supone un 1'8 % del consumo eléctrico anual de este país. Estudios recientes realizados por la ETA (Energy Technologies Area) muestran que el consumo eléctrico de estos centros se incrementó en un 4 % entre los años 2010 y 2014. Se estima que el consumo energético seguirá aumentando de la misma manera que en los últimos años, incrementándose otro 4 % entre 2014 y 2020 [3].

Son muchos los factores que influyen a la hora de determinar las necesidades energéticas de

estos centros. Valorando las tendencias actuales, es posible deducir algunos posibles escenarios que determinen el futuro en cuanto a necesidades energéticas se refiere. La figura 1.1 muestra una gráfica con las diferentes estimaciones del consumo energético a lo largo de dos décadas para los diferentes posibles escenarios [3].

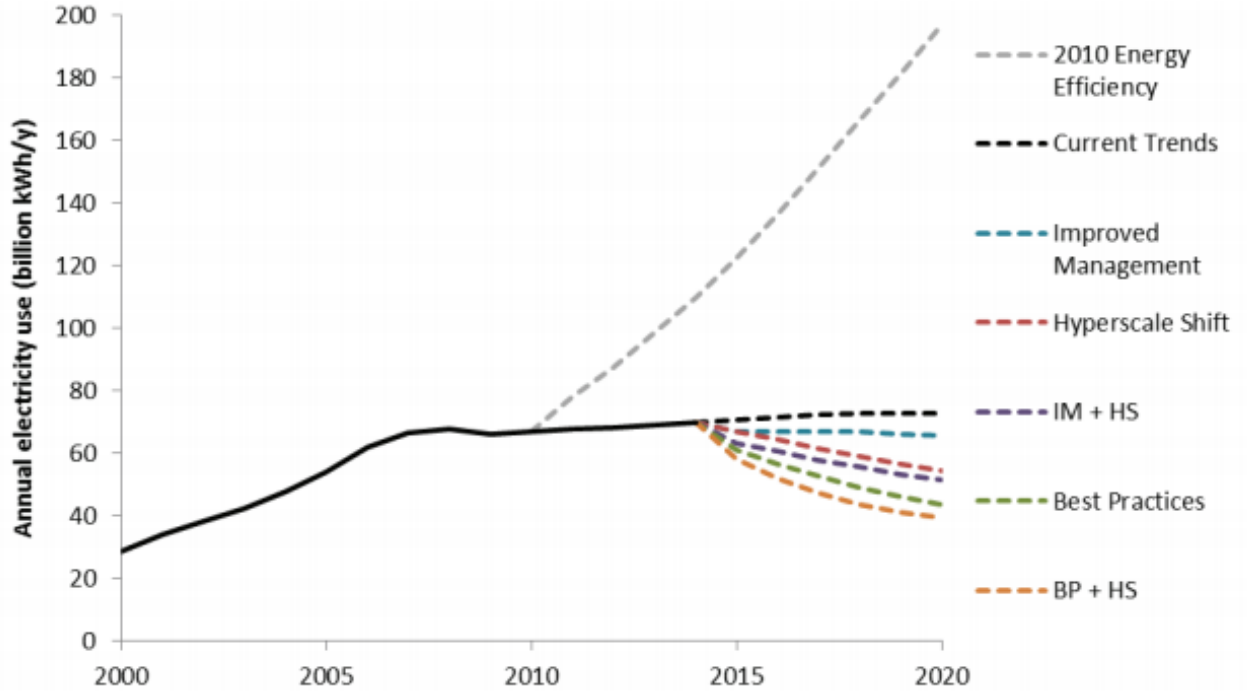

Figura 1.1: Estimación del consumo energético.

- La mejora de la gestión (linea azul) incluye también las mejoras en la eficiencia. Las tendencias actuales se centran en cambios operacionales y tecnológicos aplicados a los componentes que ya se encuentran en los centros de datos y computación.
- Mejores prácticas (linea verde) consiste explotar las mejoras de eficiencia que se pueden obtener mediante la adopción generalizada de las tecnologías y practicas más eficientes para cada sistema.
- El escenario de cambio hiperescalar (linea roja) representa un cambio agresivo en la organización de los centros de datos y de computación reduciendo el numero de centros pequeños y trasladándolos a entidades mayores. Las tendencias actuales ya advierten cierto movimiento en cuanto al uso de servidores más grandes en los centros de datos.

Para proporcionar una visión real del consumo energético que hacen los sistemas actuales utilizaremos como ejemplo el “Sequoia - BlueGene/Q”, el cual se encuentra en la cuarta posición de la lista Top500. Sus características son las siguientes:

- Número de núcleos: 1,572,864
- Rmax (rendimiento máximo alcanzado): 17.173,2 TFlop/s
- Rpeak (rendimiento máximo teórico): 20.132,7 TFlop/s

- Energía: 7.890 kW

El Sequoia consta de un total de 98.304 nodos de computación, donde cada uno de ellos utiliza un procesador A2 de 16 núcleos, por lo tanto, cada uno de los nodos consumirá unos 80 vatios.

Tras observar estos datos, es posible observar la importancia de apagar aquellos nodos que no estén realizando trabajo. Si solo se apagasen el 10% de los recursos del Seqouia se obtendría un ahorro de 789kW, una cantidad nada despreciable de energía.

Si comparamos estas cifras con un ejemplo genérico como un laboratorio similar al laboratorio general de la Escuela de Ingeniería Informática que consta de unas 30 máquinas, las cuales consumen unos 500W cuando están encendidas, obtenemos un total de 15kW. Este consumo comparado con el del Sequoia es prácticamente despreciable, teniendo en cuenta que las maquinas de un laboratorio no están siempre encendidas esperando trabajo el consumo medio de este laboratorio es bastante menor a 15kW.

Por lo tanto, es lógico que si una máquina no se está utilizando conviene mantenerla apagada, puesto que esto proporcionará un ahorro significativo en cuanto al consumo energético. Esta simple comparativa refleja a grandes rasgos el impacto que puede tener la implementación de un sistema de eficiencia energética sobre estos sistemas destinados a la computación de altas prestaciones.

Otro ejemplo es el proyecto “Monica”(Monitorización Integral de Caléndula), desarrollado por la Fundación Centro de Supercomputación de Castilla y León, que consiguió reducir en un 29% el consumo energético de un supercomputador como Caléndula tras cinco meses de su implementación en 2014 [4].

1.2. Objetivos

Los objetivos propuestos para este trabajo se centran en la puesta en marcha de un sistema de gestión energética para un cluster de computación, soportado por el sistema operativo Rocks.

Para ello, se realizarán las siguientes tareas:

- Comprender el funcionamiento de los sistemas gestores de energía: se pretende comprender el funcionamiento y objetivos de los diferentes sistemas gestores de energía, características comunes, técnicas para el ahorro energético, etc.

- Implementar un cluster de computación utilizando el sistema operativo Rocks: se realizará la instalación y configuración de Rocks en diecinueve máquinas, las cuales servirán como cluster para las posteriores instalaciones.

- Instalación y configuración de un sistema gestor de energía: tras un estudio de los diferentes sistemas gestores de energía se seleccionara el que mejor se adapte a nuestras necesidades.

- Implementación de mejoras que añadan funcionalidad al sistema de gestión energética instalado: tras la selección e instalación de un sistema de gestión energética, se tratará de implementar una serie de mejoras que extiendan la funcionalidad de este sistema.

1.3. Recursos utilizados

Para la realización de este trabajo se disponen de los siguientes equipos, todos ellos ubicados, a fecha de redacción, en el laboratorio 1L022 del edificio de Tecnologías de la Información y Telecomunicaciones de la Universidad de Valladolid.

- 15x Intel Pentium 4 640(3.2GHz), 4Gb RAM, 1Tb HDD.
- 4x Intel Pentium D 840(3.2GHz), 4Gb RAM, 1Tb HDD, NVidia GTX650Ti-2GBDDR5.

Estos equipos serán utilizados para la puesta en marcha de un cluster de computación que utilizará el sistema operativo Rocks 6.2, el cual esta basado en CentOS y centra su funcionamiento en una serie de módulos denominados "rolls". Estos módulos añaden soporte y funcionalidad al sistema de manera transparente para el usuario. Puede encontrarse más información acerca de Rocks en el Apéndice B.

Además de todas las máquinas anteriormente mencionadas también se ha utilizado un ordenador portátil para propósito general, el cual ha sido utilizado para tareas de investigación y desarrollo.

- Acer Aspire E1-571, Intel Core i7-3612QM CPU @ 2.10GHz x 8, 5.7Gb RAM, 500Gb HDD.

1.4. Estructura del documento

Esta memoria posee una estructura basada en capítulos, los cuales a su vez se dividen en secciones y subsecciones. En los que se exponen los diferentes conceptos que se consideran necesarios para la total comprensión de este trabajo. Finalmente se pueden encontrar varios apéndices que complementan explicaciones y proporcionan recursos.

El primero de estos capítulos, contiene la introducción al tema que abordamos en esta memoria: la eficiencia energética en clusters de computación. Además de esto detalla los objetivos del trabajo, recursos utilizados para su realización y motivación del mismo.

El segundo capitulo, aborda todo lo concerniente a los sistemas de gestión energética, desde una breve explicación de su funcionalidad, hasta el estado del arte de estos sistemas.

El capitulo tres explica todo lo relacionado con la implementación del sistema de gestión energética Clues en nuestro cluster. Se explica tanto su instalación y configuración, como la implementación de varias mejoras que añaden nuevas funcionalidades al sistema. Además de esto, se describe el proceso de elección, instalación y configuración de un sistema de alimentación ininterrumpida.

Por último, el capitulo cuatro trata las conclusiones deducidas de la realización de este trabajo y el posible trabajo futuro que podría realizarse para ampliar el contenido de este proyecto.

Capítulo 2

Sistemas de gestión de energía

Un sistema de gestión energética es un conjunto de elementos interrelacionados o que interactúan entre sí para establecer y alcanzar una política y unos objetivos energéticos definidos. Los sistemas de gestión energética se basan en el ciclo de mejora continua (la rueda de Deming): Planificar-Ejecutar-Verificar-Actuar [5].

En el ámbito de las tecnologías de la información, un sistema de gestión energético se define como una variedad de aplicaciones software interrelacionadas que auditan, recolectan información, generan reportes y proveen de herramientas que ayudan a la reducción de los costes energéticos y de consumo para un sistema. Esta variedad de aplicaciones software se denomina "Software de gestión energética" (EMS de sus siglas en inglés Energy Management System).

Las funciones principales de estos sistemas se centran en tres aspectos principales:

- Monitorizar recursos: el EMS, es capaz de recopilar información de los diferentes recursos del sistema, ya sea solicitando esta información al mismo recurso o haciendo uso de un software intermedio.

- Monitorizar demanda: la demanda de recursos por parte de los diferentes trabajos que llegan al cluster debe ser monitorizada por el EMS. Este recopila información acerca del número de trabajos en ejecución, en cola y algunas de las características de los mismos, número de procesadores necesarios para su ejecución, usuario que lo lanzó, etc.

- Ajustar los recursos a la demanda: la funcionalidad principal de todo EMS consiste en la asignación de la demanda de trabajo entrante a los diferentes nodos que componen el cluster. Estas asignaciones pueden seguir diferentes estrategias, como el balanceo de carga o la elección de un determinado nodo para un tipo especifico de trabajo.

Es algo común que los sitemas de computación de altas prestaciones utilicen sistemas gestores de colas que se encarguen de la gestión de los recursos, en cuanto a asignación de trabajo y monitorización de las tareas. Estos gestores de colas gestionan información acerca de los recursos del sistema y su estado. Los EMS trabajan de forma conjunta con estos sistemas para recopilar información acerca de los trabajos y los recursos existentes en el sistema. De esta forma pueden tomar las decisiones que permitan alcanzar los objetivos energéticos propuestos.

Las siguientes secciones muestran la filosofía en cuanto a la gestión y asignación de recursos a la demanda para diferentes sistemas de gestión energética.

2.1. EnergySaving Cluster

EnergySaving Cluster es un sistema de gestión energética que imita los objetivos principales de otros proyectos como Rocks-Solid [6] y PowerSaving [7], los cuales se centran en minimizar el consumo eléctrico sin afectar al funcionamiento de los gestores de colas y de esta manera minimizar el coste y hacer los sistemas más rentables. Basa su funcionamiento en el apagado y encendido de nodos del cluster en función de la demanda de recursos para la realización de tareas, pero con una funcionalidad más completa [8].

EnergySaving Cluster está desarrollado como un módulo (Roll) para el sistema Rocks Cluster, junto a la utilización del sistema gestor de colas Sun Grid Engine (SGE).

2.1.1. Diseño y Arquitectura

EnergySaving Cluster centra su arquitectura en un diseño que limita la posibilidad de implementación del sistema a clusters HPC que utilicen NPACI Rocks Cluster como sistema operativo, cuenten con un nodo maestro (front-end) y un módulo SGE como gestor de colas.

La figura 2.1 representa la arquitectura de EnergySaving Cluster, la cual basa su funcionamiento en tres componentes principales: demonios, base de datos y una interfza web [8].

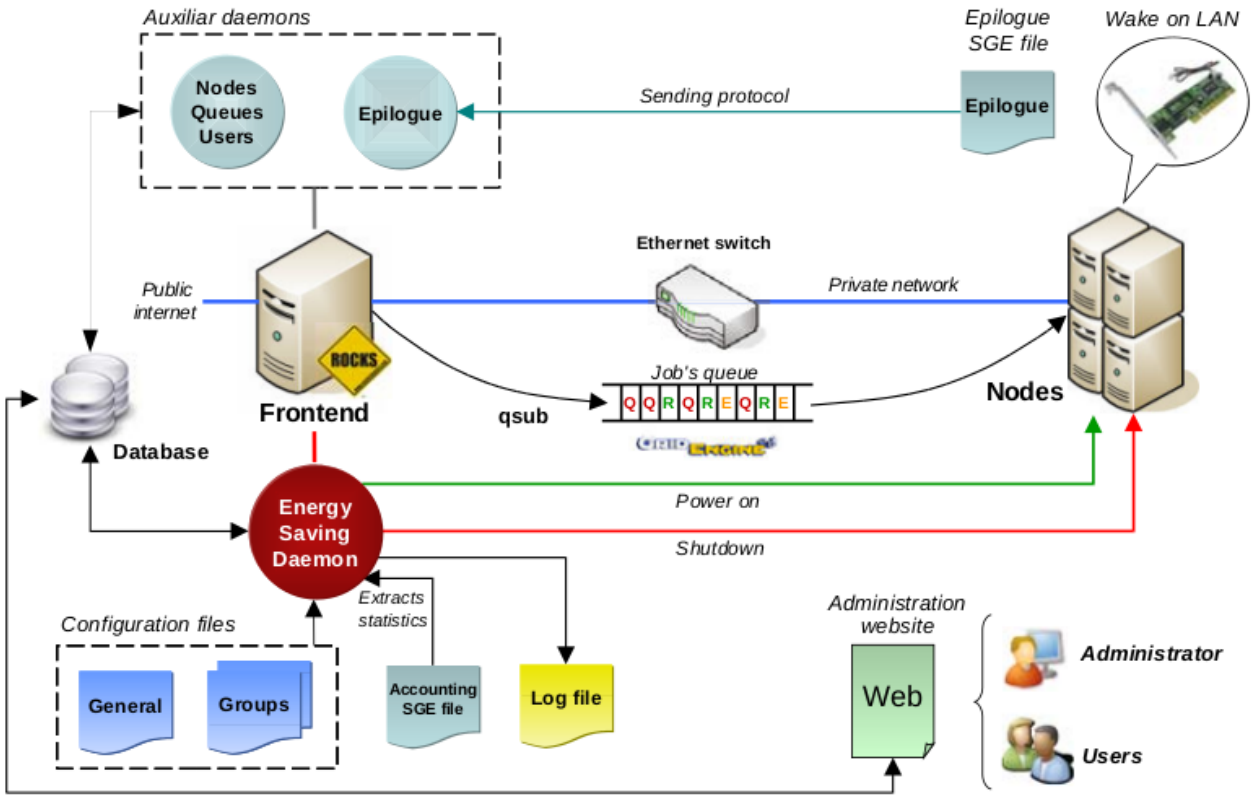

Figura 2.1: Representación de la arquitectura de EnergySaving Cluster.

Demonios

Existen tres demonios dentro del sistema que realizan funciones como el manejo de la base de datos, recolección de estadísticas, apagado y encendido de los nodos, etc. A continuación se definen de forma detallada la funcionalidad de estos procesos.

Demonio de final de petición. Este demonio es el encargado de recolectar información sobre los trabajos finalizados por el cluster. Para ello, cuando un nodo finaliza una tarea, el demonio de final de petición recibe información desde el modulo de ejecución del SGE (el cual es fundamental para monitorizar y gestionar el cluster y por lo tanto para aplicar las políticas de ahorro energético establecidas). Puesto que la base de datos se encuentra en el nodo maestro, es necesario enviar toda esta información a través de la red interna. Para ello se abre una comunicación TCP entre el nodo que finalizó la tarea y el demonio de final de petición del nodo maestro. De esta forma se actualiza el modulo de ahorro energético de la base de datos con la información recibida, pudiendo así ajustar las políticas de ahorro de forma dinámica.

Demonio para las colas, usuarios y nodos. Este demonio debe asegurarse de que toda la información de los nodos, usuarios y colas que están activos en el SGE está correctamente reflejada en la base de datos. Para ello realiza peticiones al sistema de colas obteniendo información acerca de los nodos y colas del sistema. La obtención de información acerca de los usuarios se realiza a través de peticiones al SO, siendo capaz de comprobar la consistencia de la base de datos. También es responsable de la activación de aquellos nodos que se marquen como no disponibles.

Demonio de activación/desactivación y estadísticas. Este demonio es el más importante del módulo, se encarga del apagado y encendido de los nodos del sistema en función de las necesidades del mismo. Este demonio realiza comparaciones entre una serie de parámetros establecidos por el administrador del sistema y los valores actuales de esos parámetros en la base de datos, realizando acciones de apagado y encendido de nodos en caso de que fuera necesario. Si fuera necesario apagar algunos nodos, el demonio suspenderá todas las colas (de esta manera se previene que se puedan asignar mas trabajos antes de que el nodo se apague), apagará el nodo que haya sido marcado, realizará una actualización de la base de datos y el fichero de log con los nuevos parámetros del sistema y reanuda la cola de trabajos. Las comparaciones con los parámetros no tienen por que ser igual para todos los usuarios del sistema, es posible crear grupos de usuarios con diferentes valores para los parámetros y establecer el orden de comprobación de estas comparaciones. Además de esto, este demonio también se encarga de actualizar el tiempo de espera(tanto de usuarios como colas) de todos los trabajos.

Base de datos

La base de datos contiene toda la información necesaria para administrar el sistema. Almacena información sobre cada nodo del cluster,(como el nombre, el tiempo desde que finalizó la última tarea, un bit de “activo”, marcado si el administrador decide que debe estar siempre encendido, el número de trabajos ejecutados por dicho nodo y un marcador de prioridad), información sobre las colas (como nombre, número de trabajos y las medias de tiempo de espera y ejecución para los trabajos lanzados a cada cola). Además de esto también almacena información sobre los usuarios (nombre, grupo al que pertenecen, número de trabajos lanzados a las colas y tiempo medio de espera y de ejecución para sus trabajos). Finalmente la base de datos también mantiene un historial de acciones en cada nodo (apagados y encendidos), el momento en el que fueron realizadas y la

causa (condición que se ha cumplido para el apagado o encendido del nodo).

Interfaz Web

El modulo posee una interfaz web la cual facilita la administración del sistema de ahorro energético. Algunas de las acciones que permite realizar son:

- Comprobación y modificación de parámetros de ahorro energético.
- Visualización, modificación y borrado grupos de usuarios en el sistema.
- Visualización y modificación de los parámetros que definen la política de apagado y encendido de los nodos.
- Visualización de la base de datos en formato HTML, y la posibilidad de generación de reportes con la información solicitada.
- Monitorización de las operaciones del cluster a través de diagramas.
- Monitorización del consumo y coste económico del cluster.
- Activar el bit de "activo" de los nodos del cluster, así como marcarlos como no disponibles en caso de un fallo.
- Encender, apagar o reiniciar los demonios, resetear los contadores y los temporizadores de la base de datos.

Todos los cambios realizados a través de la interfaz se reflejan en la base de datos y los ficheros de configuración. Los usuarios sin privilegios en el sistema únicamente podrán consultar estadísticas y reportes a través de la interfaz.

2.2. Cherub

Cherub es otro sistema diseñado para realizar la gestión de energía de un cluster. Cherub posee un diseño modular y extensible que resulta adecuado para gestionar tanto el balanceo de carga de un sistema de servidores como la gestión energética de un cluster HPC [9].

2.2.1. Diseño y Arquitectura

Cherub centra su funcionalidad en un único demonio el cual utiliza una serie de scripts de extensión para dar soporte a diferentes gestores de colas. Cherub es capaz de realizar la gestión del apagado y encendido de nodos tanto con IPMI (Intelligent Platform Management Interface) como con WOL (Wake On Lan) y fue diseñado inicialmente para tratar con dos de los grandes gestores de recursos como son PBS/TORQUE (enfocado a sistemas HPC) y Linux Virtual Server(para la gestión del balanceo de carga).

Además de esto, Cherub implementa su propio sistema de estados de los nodos basándose en el estándar ACPI (Advanced Configuration and Power Interface), el cual divide los estados de los nodos

en varios subconjuntos que determinan el nivel de inmersión del nodo en dicho estado (Un nodo en estado “sleep” puede retomar el estado “working” en un breve periodo de tiempo, mientras que un nodo en estado “soft off” requiere un reinicio del sistema para volver al estado activo).

Cherub esta desarrollado en C de manera modular y los scripts de extensión que utiliza están desarrollados en Python.

Componentes de Cherub

Los componentes principales de Cherub son un demonio, que se encargar de la gestión energética del sistema y uno o varios scripts de extensión, los cuales proporcionan las tareas básicas(obtener información de la carga de los nodos, apagar y encender los nodos, etc) para los diferentes gestores de recursos que el sistema pueda utilizar.

Además de esto, la configuración de Cherub se realiza a través de un archivo de configuración en el cual pueden cambiarse parámetros como el tiempo que un nodo debe pasar en un determinado estado antes de poder ser apagado, el tiempo que debe pasar sin completarse una acción antes de suponer un error en dicha tarea, etc.

Estado de los nodos

Cherub define los diferentes estados de los nodos de un cluster basándose en la idea de que es importante obtener un enfoque genérico, el cual permita manejar los estados que puedan definir los diferentes gestores de recursos del sistema. Para ello Cherub define los siguientes estados:

- Unknown: el estado del nodo es desconocido, debido a un error ocurrido mientras se encontraba en otro estado.
- Busy: el nodo se encuentra ejecutando carga de trabajo y no será apagado.
- Online: el nodo no está trabajando, pero esta disponible para recibir carga. Es posible apagar el nodo en caso de que no sea necesario.
- Offline: el nodo está encendido, pero no ha sido registrado por el gestor de recursos del sistema, por lo tanto no puede recibir carga de trabajo.
- Down: el nodo está apagado y por lo tanto no está registrado por el gestor de recursos del sistema.

La transición entre estados se muestra en la figura 2.2 [9]. Las etiquetas de las transiciones están especificadas de la siguiente manera: estado en el que termina el nodo/acción que se desempeña para la transición.

2.3. EECluster

EECluster es un sistema de gestión energética orientado a funcionar con dos gestores de recursos: OGE/SGE y PBS/TORQUE. Basa su funcionamiento en reglas, con el añadido de un conjunto

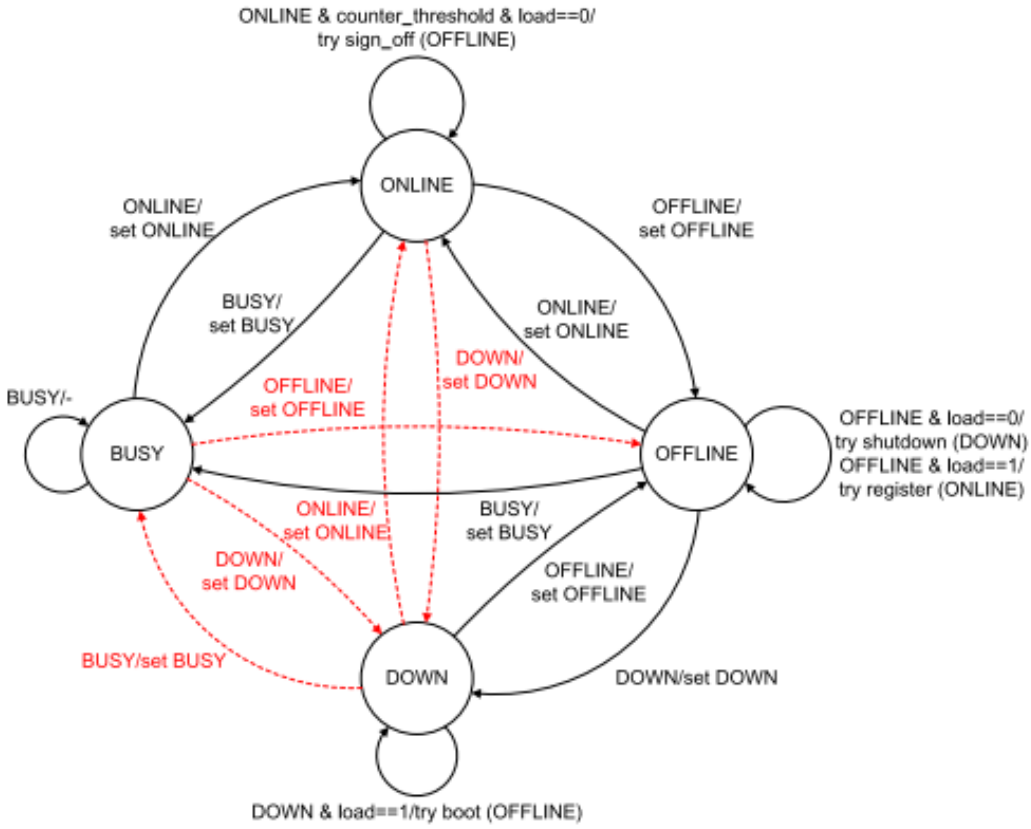

Figura 2.2: Diagrama de transiciones de estados de Cherub.

de parámetros numéricos los cuales se obtienen a través de un algoritmo evolutivo multiobjetivo enfocado a una máquina de aprendizaje [10].

El funcionamiento de EECluster se centra en el apagado y encendido de los nodos de un cluster HPC en función de la carga de trabajo que llega al cluster.

2.3.1. Diseño y Arquitectura

La solución propuesta por EECluster consiste en un servicio y en un panel de administración, complementados con un sistema gestor de bases de datos e implementado sobre un cluster HPC el cual utilice un gestor de colas como OGE/SGE o PBS/TORQUE, utilizando tanto WOL como IPMI para el encendido y apagado de los nodos.

La figura 2.3 muestra una visión de alto nivel de los componentes del sistema EECluster [10]. El ciclo de trabajo de EECluster es el siguiente:

- Sincronización con el sistema.
- Utilización del sistema de aprendizaje para determinar si es necesaria alguna reconfiguración (Determinación del estado de los nodos).
- Selección de los objetivos a reconfigurar.
- Apagado y encendido de los nodos a través de lo módulo de gestión de energía.

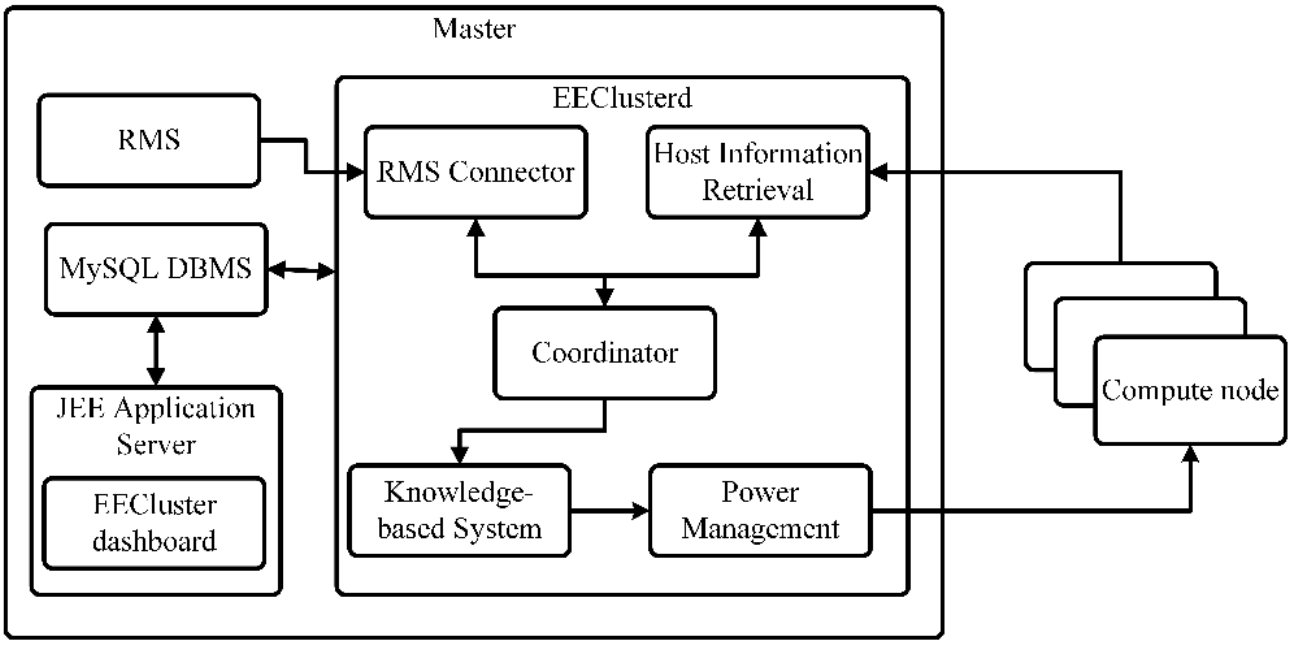

Figura 2.3: Diagrama de componentes de la arquitectura EECluster.

Sincronización

La tarea de sincronización recolecta información del estado del sistema a través del gestor de colas, como por ejemplo el estado de las colas de trabajos (trabajos finalizados, número de trabajos esperando en las colas).

El módulo de sincronización de EECluster también recolecta información de cada uno de los nodos:

- CPUs (modelo, frecuencia, caché) a través de /proc/cpuinfo.
- Memoria RAM a través de /proc/meminfo.
- GPUs (nombre, porcentaje de utilización, temperatura,etc) a través del sistema administración de NVIDIA.
- Intel Xenon Phi (En caso de disponer de ella, modelo, frecuencia, temperatura,etc) a través de micinfo.
- PSU (consumo energético) a través de la interfaz IPMI.
- Dirección MAC a través del protocolo ARP.

Además de esto, recolecta información sobre los usuarios a través de "/etc/passwd".

Estado de los nodos

Para poder identificar la situación de cada uno de los nodos que componen el cluster se han definido una serie de estados entre los que un nodo puede estar. La figura 2.4 define dichos estados y sus transiciones [10].

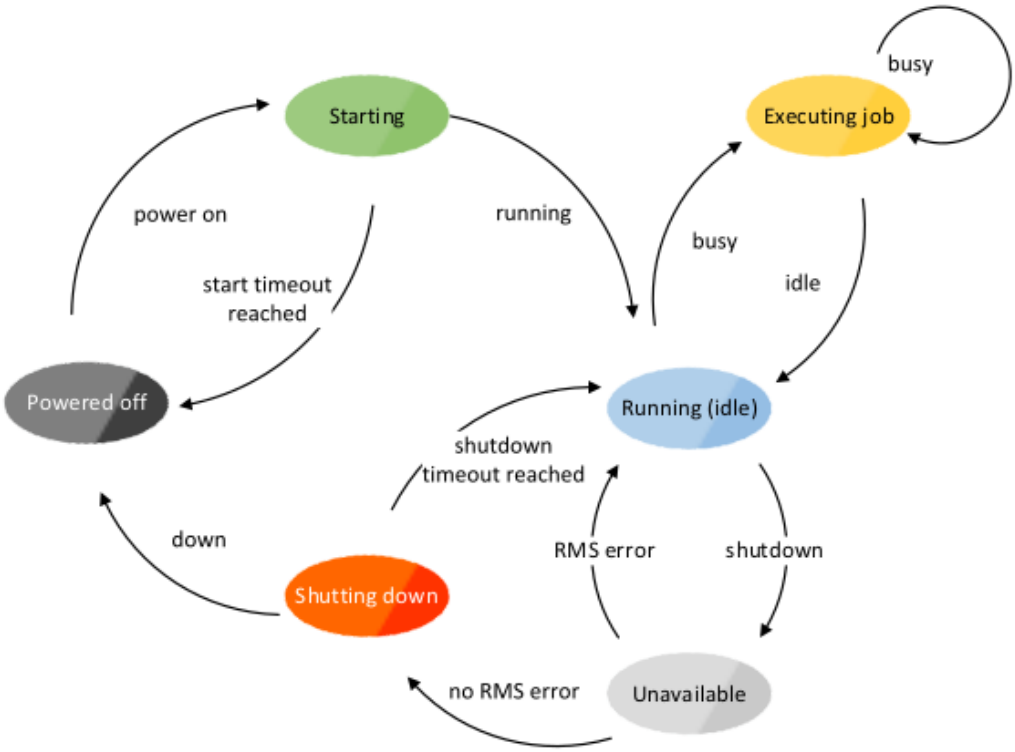

Figura 2.4: Diagrama de estado de los nodos.

Sistema de aprendizaje

El sistema de aprendizaje es una pieza fundamental en la arquitectura del sistema, ya que implementa el mecanismo para la toma de decisiones que determina el número de nodos que deben estar activos en cada momento en el cluster. Las reglas que determinan la toma de decisiones son actualizadas a través de este sistema de aprendizaje junto a un algoritmo evolutivo multiobjetivo [11].

Panel de administración

El panel de administración consiste en una aplicación web la cual muestra el estado actual de del cluster (estado de los nodos, procesos ejecutándose o en cola...etc). También permite el apagado y encendido de los nodos de forma manual y la realización de modificaciones en la configuración del sistema.

2.4. Clues

Clues es un sistema de gestión de energía para clusters de altas prestaciones e infraestructuras cloud, cuya función principal es la de apagar los nodos internos del cluster cuando no están siendo utilizados y, de manera reciproca, encenderlos cuando son necesarios [12].

2.4.1. Diseño y Arquitectura

Clues basa su arquitectura en un diseño completamente modular, cuyo objetivo es facilitar su implementación y adaptación a una gran cantidad de infraestructuras. La Figura 2.5 muestra el resultado de su diseño modular [12].

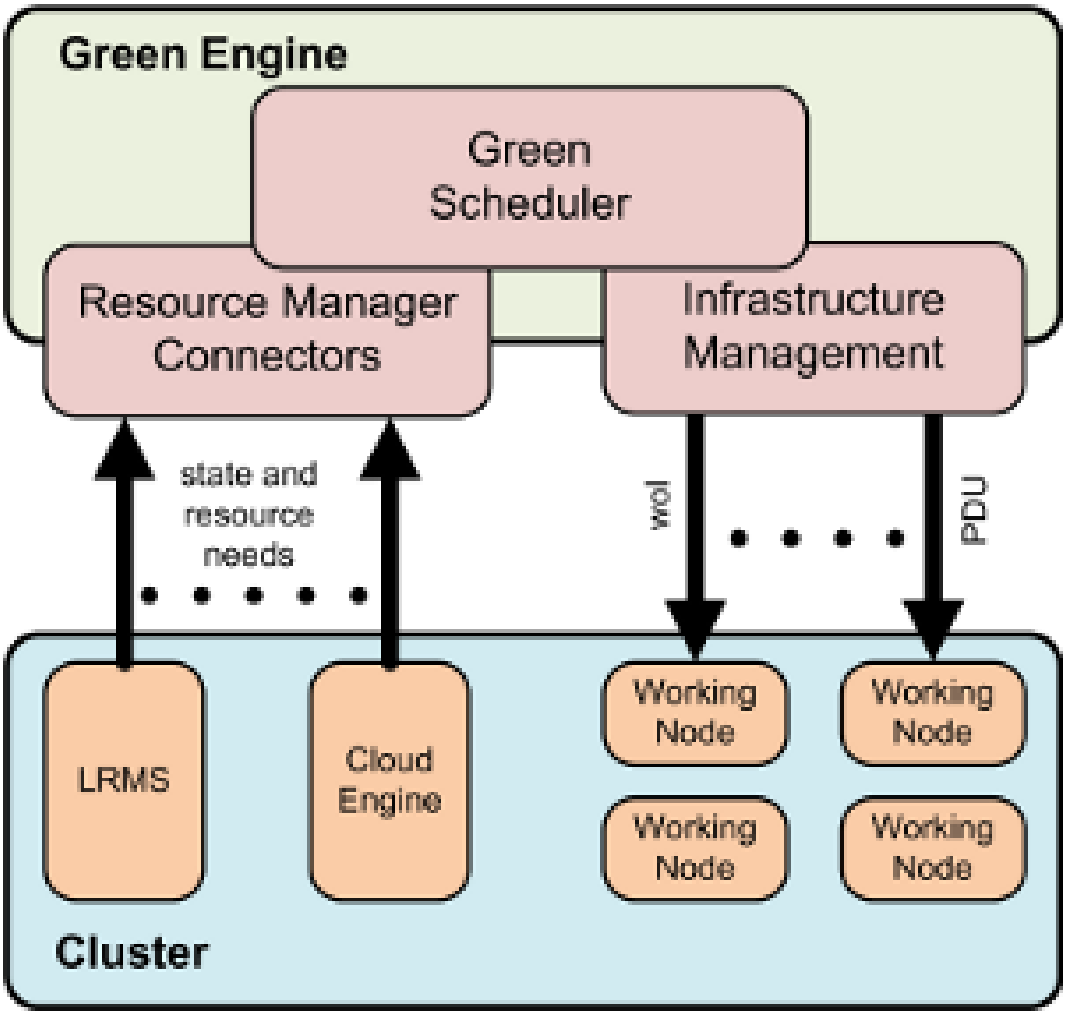

Figura 2.5: Diagrama de la arquitectura de Clues.

La arquitectura de Clues consta de un componente principal, el gestor Clues, que utiliza una serie de conectores con los gestores del cluster y un conjunto de mecanismos de encendido y apagado de los nodos, los cuales son implementados a través de diversos plug-ins.

En cuanto al conjunto de conectores con los middlewares de gestión del cluster, estos tienen dos tareas: la captura de las solicitudes de trabajo y la monitorización del uso de los nodos.

Captura de solicitudes de trabajo

Clues proporciona una variada gama de posibilidades para la captura de solicitudes de trabajo. La primera de ellas basa su funcionamiento en la característica de muchos gestores de clusters de incorporar mecanismos que permitan realizar acciones en el momento de lanzar un trabajo a la cola de su planificador. De esta forma permite integrar el middleware de gestión con Clues siguiendo su propia arquitectura.

Otra posibilidad siempre disponible es la de escribir una aplicación o script que se encargue de sustituir la llamada real. De esta forma integra el middelware con Clues, realizando posteriormente la llamada efectiva a la aplicación real.

Por último, es posible desarrollar planificadores específicos que se comuniquen con Clues a través de su API, pudiendo crear las solicitudes de recursos de forma expresa. Esto proporciona al usuario de un mayor control del funcionamiento, gestión e integración de Clues.

Monitorización

En cuanto a la monitorización de los nodos del cluster, Clues hace uso de aquellos comandos para comprobar el estado de los nodos implementados en cualquier middleware de gestión de clusters. De esta manera Clues obtiene la información necesaria acerca de los nodos del cluster, además del estado de utilización de la infraestructura.

2.4.2. Características

Las principales características de Clues, las cuales le diferencian de otros sistemas de gestión energética destinados a clusters HPC son:

- Desarrollado en python: puesto que Clues ha sido desarrollado completamente en python, su distribución se realiza bajo una licencia Open Source GNU General Public License - 3.0. Permitiendo a los usuarios realizar modificaciones y distribuirlas bajo el mismo tipo de licencia.
- Infraestructura heterogénea: gracias a su sistema de conectores modular, Clues es capaz de gestionar cualquier tipo de infraestructura, ya sea homogénea o heterogénea.
- Multisistema: Clues puede integrarse virtualmente con cualquier gestor de recursos locales (LRMS), ya sean sistemas de colas o gestores de infraestructuras Cloud Computig.
- Políticas de ahorro energético adaptables: es posible establecer políticas de ahorro de energía variables, pudiendo ajustar diversas características como el encendido por exceso, la lista de nodos que no deben entrar en políticas de apagado, etcétera.
- Encendido contenido: debido a la posible sobredemanda en el consumo eléctrico y los picos de consumo de las fuentes de alimentación durante el arranque, Clues realiza un encendido contenido de los nodos, tratando de limitar la incidencia del encendido y apagado de los nodos en el sistema eléctrico general.
- Herramientas de informes y gestión de nodos: Clues proporciona una serie de herramientas para la gestión de nodos, tanto en linea de comando como de interfaz web. Además de esto incorpora una herramienta para la generación de informes sobre el funcionamiento del cluster.

2.5. Estudio comparativo

El sistema de gestión energética elegido para este proyecto ha sido Clues, debido a que las características de este se adaptan perfectamente a los objetivos propuestos. Una de estas características

	EnergySaving Cluster	Cherub	EECluster	Clues
Sistema multiplataforma	No	Sí	Sí	Sí
Herramientas de generación de INFORMES	Sí	Parcialmente implementado	Parcialmente implementado	Sí
Interfaz de gestión/-configuración	Sí	No	Sí	No
Gestores de colas soportados	SGE	PBS/Torque, LVS	OGE/SGE, PBS/-Torque	Torque, SLURM, OpenNebula, SGE, Globus Toolkit y Glite
Protocolos de apagado/encendido soportados	WOL, IPMI	WOL, IPMI	WOL, IPMI	WOL, IPMI, ONE
Diseño modular	No	Sí	No	Sí
Lenguaje de desarrollo	Desarrollado como un Roll de Rocks Cluster	Desarrollado en C, scripts de extensión en Python	Java y Bash	Python

Tabla 2.1: Comparativa de las características de los diferentes EMS.

es, por ejemplo, la arquitectura modular de Clues, que permite añadir nuevas funcionalidades de una manera relativamente sencilla. Además de esto Clues está desarrollado en Python, puesto que se trata de un lenguaje interpretado es sencillo realizar módulos de funcionalidad portables. Otra característica destacable es que es el sistema de gestión energética más documentado (de los presentados en las secciones anteriores), lo que facilita su implementación y configuración.

Por otra parte, Clues también tiene sus carencias comparándolo con algunos de los sistemas de gestión energética que se han presentado en las secciones anteriores. Una de ellas es la falta de un sistema de gestión/configuración con entorno gráfico, lo cual facilita mucho las tareas de administración a la hora de elaborar un plan de gestión energética. Otra de estas carencias, es que las diferentes opciones que Clues proporciona en cuanto a la gestión de los nodos proporcionan una visión de los nodos del cluster como una serie de máquinas idénticas en cuanto a necesidades energéticas, permitiendo, por ejemplo, establecer un tiempo único para el apagado de los nodos.

Capítulo 3

Nuestra implementación

Una vez se ha finalizado la instalación y configuración del cluster (como se muestra en el apéndice B) se pretende implementar el sistema de eficiencia energética Clues expuesto en el capitulo anterior. Además de esto, se realiza una configuración y calibración inicial del sistema sobre el cual trataremos de implementar mejoras que permitan añadir cierta funcionalidad a Clues.

Las mejoras que se exponen en el siguiente capitulo se centraran en cubrir las carencias presentadas en la sección anterior. Para ello, se trataran de implementar las siguientes funciones.

- Panel de configuración: se implementará un panel de configuración accesible a través de la página WordPress del servidor. Este panel permitirá realizar modificaciones en la configuración de Clues.
- Política de gestión de grupos: se pretende implementar las funcionalidades necesarias dentro de Clues, que permitan la definición de grupos de nodos, de forma que estos grupos puedan tener diferentes valores para ciertos parámetros de su configuración.
- Finalmente, se pretende instalar un sistema de alimentación ininterrumpida que permita, en caso de caídas en la red eléctrica, el apagado ordenado de los nodos del cluster así como el reporte a través de email a los usuarios con trabajos pendientes o ejecutándose en el sistema.

3.1. Implementación de Clues

Para la instalación de Clues, se han seguido las instrucciones que se muestran en el GitHub de GRyCAP [13].

3.1.1. Prerrequisitos

Puesto que Clues está escrito en Python, es necesario tener instalada la versión 2.4 o superior del mismo junto con sqlite.

```
$ yum install python python-sqlite
```

Además de esto, es necesario instalar las herramientas de desarrollo de Python (Python developement tools) y un paquete de utilidades para Clues. Este paquete lo podemos encontrar en el GitHub de GRyCAP llamado Clues Python Utils(cpyutils). Para ello:

```
$ yum install python-dev
```

para cpyutils:

```
$ git clone https://github.com/grycap/cpyutils
$ mv cpyutils /opt
$ cd /opt/cpyutils
$ python setup.py install --record rutas-instalacion.txt
```

3.1.2. Librerías opcionales

Una vez se han instalado todas las dependencias, es posible realizar una instalación funcional de Clues en nuestro sistema. También es recomendable instalar algunas librerías opcionales, las cuales permiten aprovechar todas las funcionalidades que Clues ofrece. Es por ello que instalaremos la librería "rrdtool", que permite utilizar la herramienta "clues_report" para generar gráficas de utilización y rendimiento de los diferentes nodos del sistema.

Para su instalación, debemos crear un nuevo archivo de repositorio para yum llamado dag.repo, en el cual introduciremos lo siguiente:

```
[dag]
name=Dag RPM Repository for Red Hat Enterprise Linux
baseurl=http://apt.sw.be/redhat/el$releasever/en/$basearch/dag
gpgcheck=1
gpgkey=http://dag.wieers.com/rpm/packages/RPM-GPG-KEY.dag.txt
enabled=1
```

Una vez hecho esto, realizamos la instalación de la librería:

```
$ yum --enablerepo=rpmforge\* install rrdtool-python rrdtool
```

3.1.3. Instalación de Clues

Ahora que todas las dependencias y librerías opcionales están instaladas, podemos comenzar con la instalación de Clues. Para ello debemos descargar los archivos necesarios desde el repositorio de GitHub de GRyCAP [13]:

```
$ git clone https://github.com/grycap/clues
```

A continuación iniciamos la instalación mediante la siguiente secuencia de comandos:

```
$ mv clues /opt
$ cd /opt/clues
$ python setup.py install --record installed-files.txt
```

Una vez que la instalación haya finalizado, es necesario configurar Clues de manera que este pueda comenzar a funcionar. Para ello es necesario establecer una configuración válida en los ficheros de

configuración de este, ya que en caso de no hacerlo el sistema sera inservible.

3.1.4. Configuración y calibración

La configuración de Clues se realiza a través de un archivo de configuración el cual se encuentra en "/etc/clues2/clues2.cnf". La instalación básica de Clues proporciona una serie de plantillas sobre las que es sencillo realizar una configuración básica.

Algunos de los parámetros más importantes a configurar son:

- LRMS_CLASS: determina el plugin que debe utilizarse para el gestor de recursos utilizado en el cluster(PBS/TORQUE, SGE...etc).
- POWERMANAGER_CLASS: define qué tipo de servicio se utilizará para gestionar el apagado y encendido de los nodos del cluster (WOL, IPMI...etc).
- SCHEDULER_CLASSES: establece qué planificadores de Clues van a utilizarse separados por comas, los cuales serán inicializados en el mismo orden que se ha establecido en el fichero.
- IDLE_TIME: determina el tiempo mínimo (en segundos) que un nodo debe pasar sin recibir carga de trabajo antes de ser considerado como candidato para apagarse.
- DELAY_POWOFF: tiempo que ha de pasar desde que se lanza la orden de apagado a un nodo hasta que este puede considerarse como apagado.
- DELAY_POWON: tiempo que ha de pasar desde que se lanza una orden de encendido a un nodo hasta que pueda considerarse como encendido.
- PERIOD_MONITORING_NODES_FAIL_GRACE: tiempo que ha de pasar tras solicitar información a un nodo y no recibir respuesta antes de considerar error.
- PERIOD_MONITORING_NODES: tiempo entre solicitudes de estado a los nodos.
- EXTRA_NODES_FREE: número de nodos que deberán permanecer encendidos, independientemente del tiempo que pasen sin recibir asignaciones de trabajo.

Algunos de estos parámetros solo serán utilizados en caso de que se hayan asignado correctamente los diferentes planificadores en el parámetro "SCHEDULER_CLASSES". Por ejemplo, el parámetro "EXTRA_NODES_FREE" requiere del planificador "clueslib.schedulers.CLUES_Scheduler_PowOn_Free" para funcionar correctamente. En caso de que no se haya incluido en la configuración el valor de este parámetro es ignorado por el servidor de Clues.

Ejemplo de configuración para SGE

Este ejemplo muestra la configuración inicial de Clues para nuestro cluster, el cual trabaja con un gestor de recursos SGE,y utiliza el protocolo WOL para el apagado y encendido de los nodos. Para realizar esta configuración, se deben modificar algunos parámetros en el fichero de configuración de Clues (/etc/clues2/clues2.cnf). A continuación se muestra la configuración seleccionada para algunos de los parámetros más importantes.

```
[general]
CONFIG_DIR=conf.d
LRMS_CLASS=cluesplugins.sge
POWERMANAGER_CLASS=cluesplugins.wol
MAX_WAIT_POWERON=300
...
[monitoring]
COOLDOWN_SERVED_REQUESTS=300
...
[scheduling]
SCHEDULER_CLASSES=clueslib.schedulers.CLUES_Scheduler_PowOn_Requests,
clueslib.schedulers.CLUES_Scheduler_Reconsider_Jobs,
   clueslib.schedulers.CLUES_Scheduler_PowOff_IDLE
IDLE_TIME=300
RECONSIDER_JOB_TIME=600
EXTRA_SLOTS_FREE=0
EXTRA_NODES_PERIOD=60
```

Una vez hayamos determinado estos parámetros, es necesario configurar los diferentes plugins que vamos a utilizar: tanto el plugin del gestor de recursos (cluesplugins.sge), como el gestor del apagado y encendido de los nodos (cluesplugins.wol). Para ello debemos modificar los diferentes archivos de configuración los cuales se encuentran en "/etc/clues2/conf.d/".
Para "cluesplugins.sge":

```
[SGE]
# Command used to get the hosts in the queue
SGE_QHOST_COMMAND=/opt/gridengine/bin/linux-x64/qhost
# Command to get the configuration of sge
SGE_QCONF_COMMAND=/opt/gridengine/bin/linux-x64/qconf
# Root path for SGE
SGE_ROOT=/opt/gridengine
```

De la misma manera para "cluesplugins.wol":

```
[WOL]
# The file in /etc/hosts format-like that contains the correspondence of the MAC
   addresses to the host names in the LRMS
WOL_HOSTS_FILE=wol.hosts
# The commandlines that must be used to power on and off the nodes using WOL
   (CLUES will subsitute the %%a as the MAC address of the node, using print
   function)
#
# * you can use %%a to substitute the MAC address and %%h to substitute the
   hostname
WOL_CMDLINE_POWON=/sbin/ether-wake -i eth0 %%a
WOL_CMDLINE_POWOFF=ssh %%h 'shutdown -h now'
```

Una vez hemos configurado los diferentes plugins, es necesario definir el archivo “wol.hosts”. Este fichero contendrá el identificador y la dirección MAC de los diferentes nodos del cluster para la gestión del apagado y encendido de los mismos. Para la creación y actualización de este archivo se ha creado un script que automatiza esta tarea, el cual puede encontrarse en el primer apartado del apéndice C.

Configuración de colas SGE

La creación de diferentes colas de trabajo resulta una práctica habitual y de gran utilidad. Muchos de los clusters que se implementan poseen un conjunto de nodos heterogéneo, es decir, nodos con diferentes características. En nuestro caso particular, algunos de los nodos que componen el cluster tienen una tarjeta gráfica “NVidia GTX650Ti-2GBDDR5”. Esta característica hace a estos nodos más eficientes para tareas que requieran un cierto grado de procesamiento gráfico.

Debido a esta heterogeneidad, resulta conveniente definir dos posibles colas de trabajo. De esta manera, la primera de estas colas contendrá a los nodos que poseen tarjeta gráfica y de manera opuesta, la otra cola contendrá todos aquellos nodos que no tengan instalado este componente.

Para definir estas colas utilizaremos como plantilla la configuración de la cola que SGE trae configurada por defecto “all.q”. Guardaremos dicha configuración en dos archivos (1C.q, GPU.q) que luego modificaremos para que incluyan los nodos con un solo core y los nodos con soporte gráfico.

```
root$ qconf -sq > 1C.q
root$ qconf -sq > GPU.q
```

Ahora que tenemos ambos archivos, deberemos modificarlos de tal manera que solo contengan los nodos correspondientes. A continuación se muestra la configuración para la cola “GPU.q”.

A continuación se muestran las lineas que se han modificado para configurar correctamente la cola GPU.q.

```
qname               GPU.q
hostlist            compute-0-0.local compute-0-1.local compute0-2.local
                          compute-0-3.local
slots               1,[compute-0-2.local=2],[compute-0-1.local=2], \
                    [compute-0-3.local=2],[compute-0-0.local=2]
```

El fichero 1C.q se modifica de forma análoga incluyendo los nodos con un solo core e indicando que esos nodos solo tienen un slot para la ejecución de trabajos.

Ahora que se han modificado los ficheros de configuración, es posible definir las colas a través de estos ficheros.

```
root#$qconf -Aq GPU.q
root#$qconf -Aq 1C.q
```

Una vez definidas, el sistema tendrá un total de tres colas, ya que estas conviven con la que SGE configura por defecto. Para evitar confusiones a la hora de determinar que nodos deben recibir el trabajo deberemos deshabilitar la cola “all.q”

```
root#$qmod -d all.q
```

Una vez hecho esto, las únicas colas disponibles para recibir trabajos serán aquellas que hemos definido.

3.2. Prueba de funcionamiento

Ahora que se ha establecido una configuración valida para Clues, se realizan unas pruebas de funcionamiento iniciales para comprobar que todo funciona como es debido. Para ello, lo primero de todo será activar el servicio de Clues a través de "cluesserver".

```
root#$cluesserver start
```

Tras iniciar el servicio, Clues realizará la primera monitorización tanto de los nodos que forman parte del cluster como de los trabajos que pueda haber en las colas de este. Si observamos el log de Clues (el cual podemos encontrar en "/var/log/clues2/clues2.log") podemos observar los resultados de esta primera monitorización como se muestra en la Figura 3.1.

```
   [CLUES];DEBUG;2017-06-17 12:25:08,009;1497695108.010;monitoring jobs and nodes at the same time
   [CLUES];DEBUG;2017-06-17 12:25:08,078;1497695108.078;
First monitorization of LRMS:
List of nodes:
[NODE "compute-0-9.local"] state: off (since 1497695108), 1/1 (slots) - ID: 001 @1497695108
[NODE "compute-0-16.local"] state: off (since 1497695108), 1/1 (slots) - ID: 002 @1497695108
[NODE "compute-0-3.local"] state: off (since 1497695108), 2/2 (slots) - ID: 003 @1497695108
[NODE "compute-0-2.local"] state: off (since 1497695108), 2/2 (slots) - ID: 004 @1497695108
[NODE "compute-0-12.local"] state: off (since 1497695108), 1/1 (slots) - ID: 005 @1497695108
[NODE "compute-0-7.local"] state: off (since 1497695108), 1/1 (slots) - ID: 006 @1497695108
[NODE "compute-0-1.local"] state: off (since 1497695108), 2/2 (slots) - ID: 007 @1497695108
[NODE "compute-0-5.local"] state: off (since 1497695108), 1/1 (slots) - ID: 008 @1497695108
[NODE "compute-0-14.local"] state: off (since 1497695108), 1/1 (slots) - ID: 009 @1497695108
[NODE "compute-0-13.local"] state: off (since 1497695108), 1/1 (slots) - ID: 010 @1497695108
[NODE "compute-0-10.local"] state: off (since 1497695108), 1/1 (slots) - ID: 011 @1497695108
[NODE "compute-0-0.local"] state: off (since 1497695108), 2/2 (slots) - ID: 012 @1497695108
[NODE "compute-0-15.local"] state: off (since 1497695108), 1/1 (slots) - ID: 013 @1497695108
[NODE "compute-0-4.local"] state: off (since 1497695108), 1/1 (slots) - ID: 014 @1497695108
[NODE "compute-0-11.local"] state: off (since 1497695108), 1/1 (slots) - ID: 015 @1497695108
[NODE "compute-0-8.local"] state: off (since 1497695108), 1/1 (slots) - ID: 016 @1497695108
[NODE "compute-0-6.local"] state: off (since 1497695108), 1/1 (slots) - ID: 017 @1497695108
[NODE "compute-0-17.local"] state: off (since 1497695108), 1/1 (slots) - ID: 018 @1497695108

   [CLUES];DEBUG;2017-06-17 12:25:08,089;1497695108.090;first monitorisation of jobs in LRMS
```

Figura 3.1: Monitorización de nodos y trabajos inicial de Clues.

Como podemos observar, en este momento el cluster se encuentra libre de carga de trabajo y todos sus nodos están apagados. A continuación lanzaremos un trabajo. De esta manera podremos observar, a través del log de Clues, como gestiona el encendido de un nodo que atienda esta petición.

```
root#$qsub tareas/trabajo1.pl
```

Una vez el trabajo es lanzado a la cola, Clues lo detecta y levanta uno de los nodos del cluster para que atienda esta petición, como se muestra en la Figura 3.2.

Por último, debemos comprobar cómo Clues gestiona el apagado de los nodos una vez han estado un tiempo suficientemente largo sin recibir trabajo. El tiempo que hemos definido para esta acción

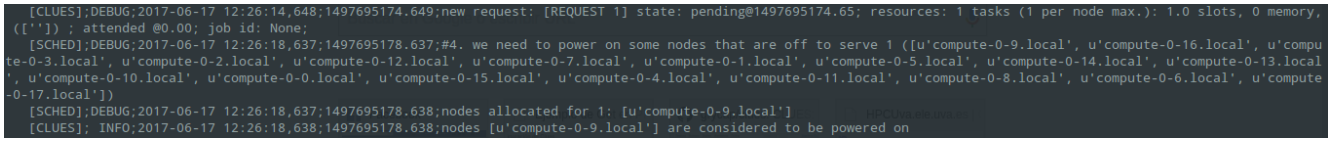

Figura 3.2: Encendido de un nodo para atender una petición de trabajo.

ha sido de trescientos segundos, transcurrido este tiempo podemos observar como Clues gestiona el apagado de este nodo. La Figura 3.3 muestra el log de Clues transcurrido este tiempo.

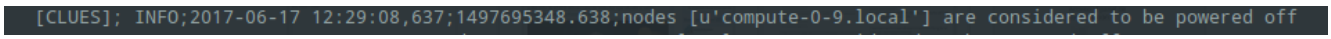

Figura 3.3: Apagado de un nodo tras un tiempo sin recibir trabajo.

Tras la realización de las pruebas iniciales y asegurarse de que todo funciona correctamente, es importante revisar de nuevo la configuración que se ha establecido en el sistema. Esto se debe a que las necesidades de cada cluster dependerán de su carga de trabajo. Resulta conveniente reajustar diferentes parámetros de tal forma que se adapten a las necesidades especificas de cada sistema.

3.3. Implementación de un panel de configuración

Debido a que Clues no proporciona una interfaz de configuración, resulta una tarea tediosa realizar la configuración del sistema para usuarios novatos. Por otra parte, es imprescindible definir una configuración valida para el sistema, ya que en caso contrario no sera posible iniciar el servicio de gestión energética, o este no funcionará de manera correcta.

Es por esto que se ha implementado un panel de configuración accesible vía web, el cual permite configurar los parámetros más determinantes para la gestión energética del sistema, como se muestra en la figura 3.4.

Los parámetros que se consideran válidos para permitir su libre modificación a través del panel de control son: IDLE_TIME, DELAY_POWOFF, DELAY_POWON, PERIOD_MONITORING_NODES y PERIOD_MONITORING_NODES_FAIL_GRACE.

Además de esto, el acceso al panel se ha incorporado en la pagina WordPress que el sistema operativo Rocks trae por defecto, como se muestra en la figura 3.5. Puesto que se trata de un gestor de contenido es necesario acceder al panel de configuración de WordPress, una vez ahí resulta intuitivo añadir un nuevo enlace en el menú. El acceso a este panel está restringido mediante identificación usuario/contraseña. Estas credenciales se autenticaran a través de un pequeño archivo de usuarios y contraseñas oculto.

La figura 3.6 muestra el formulario de acceso al panel de configuración.

Una vez se ha modificado la configuración del sistema, es necesario realizar un reinicio del servicio para que los cambios tengan efecto. Es posible realizar esta acción a través de la interfaz web una vez se han terminado de introducir los cambios, como se muestra en la figura 3.7.

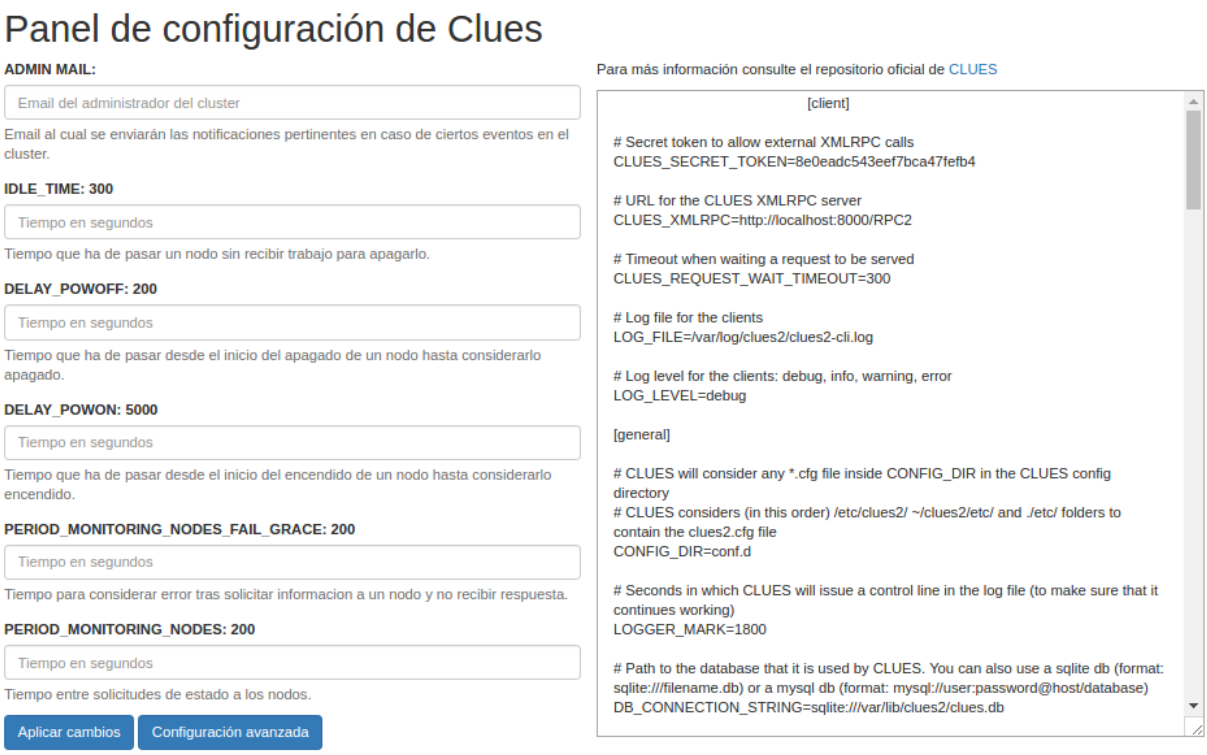

Figura 3.4: Panel de configuración para Clues.

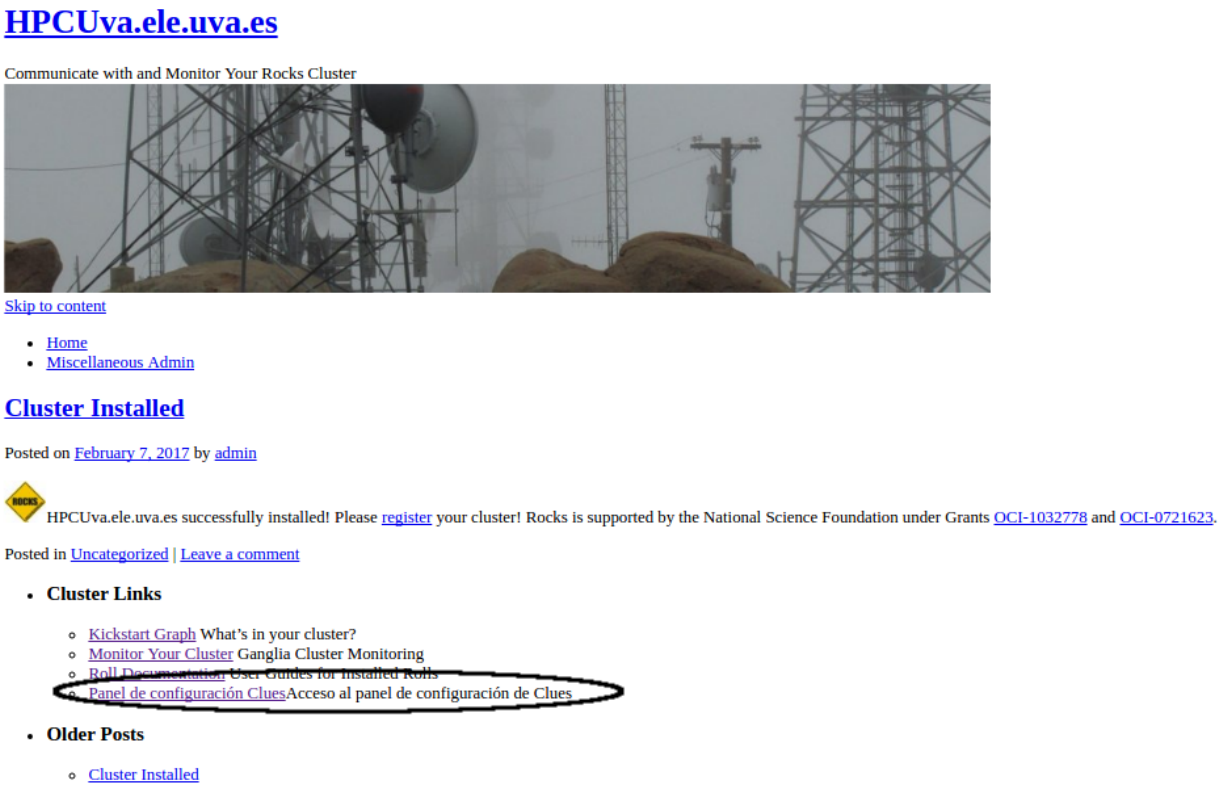

Figura 3.5: Pagina inicial del WordPress de Rocks.

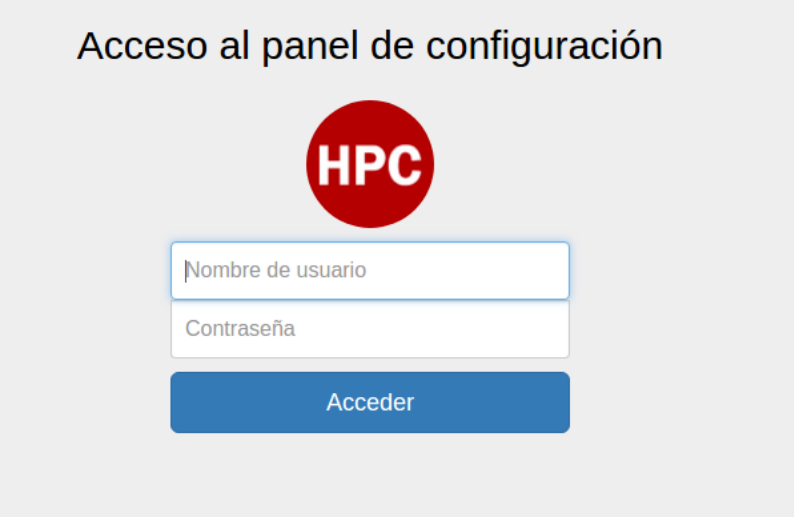

Figura 3.6: Formulario de identificación al panel de control.

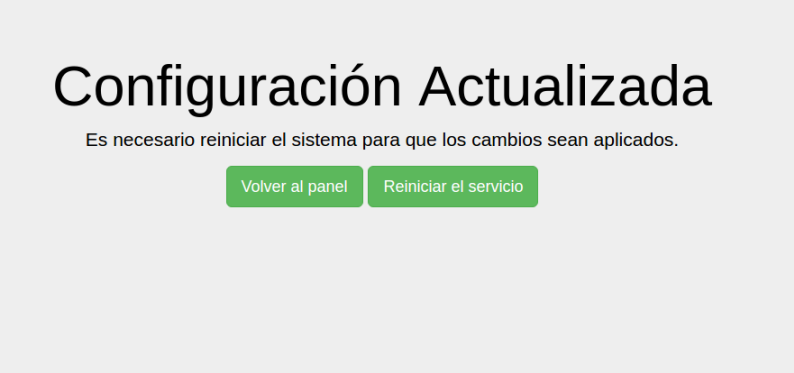

Figura 3.7: Reinicio del servicio a través de interfaz.

Toda la interfaz de configuración ha sido desarrollada con HTML, PHP y Boostrap2 [14], puede consultarse el código en el CD adjunto. Es necesario que todos los archivos contenidos en el subdirectorio “Login” y “PanelPrincipal” estén ubicados en la carpeta desde la que el servidor sirve los archivos, junto con las carpetas “res” y “test”.

3.4. Implementación de una política de gestión de nodos por grupos

Clues ofrece la posibilidad de utilizar una gran variedad de políticas de gestión para el apagado y encendido de los nodos, como por ejemplo, la posibilidad de definir el tiempo que ha de pasar un nodo sin trabajo como para considerarlo prescindible y apagarlo, definir un grupo de nodos que deben permanecer encendidos independientemente de su carga de trabajo, o la definición de un grupo de nodos que no deben encenderse, de manera que queden como un grupo de reserva.

Se ha decidido complementar estas políticas agregando una nueva posibilidad, la cual se basa en la definición de grupos de nodos. Estos grupos se diferencian en el tiempo que han de pasar sus nodos sin recibir carga de trabajo antes de considerar que deben ser apagados. De esta manera, una posible configuración sería, por ejemplo, un grupo de nodos que se apaguen rápidamente una

vez han dejado de recibir carga de trabajo, otro grupo que tarde un periodo de tiempo moderado y finalmente un pequeño grupo de nodos que permanezca encendido durante un largo periodo de tiempo.

Esta definición de grupos permite a los nodos del cluster a adaptarse más agilmente a la variación de carga de trabajo que llega al cluster. Al mismo tiempo, tener un grupo de nodos que permanezcan encendidos un largo periodo de tiempo hace que los usuarios sufran menores tiempos de espera al enviar trabajos al cluster, ya que es muy probable que haya algún nodo disponible para recibir esas tareas.

Esta nueva política combinada con las políticas ofrecidas por Clues puede dar lugar a una política de gestión energética muy versátil, permitiendo a Clues adaptarse de manera más precisa a las necesidades especificas de cada sistema que este gestiona.

3.4.1. Estudio del código de Clues

La implementación de esta mejora conlleva la modificación del software de Clues. Esto es algo sencillo, ya que al estar escrito en Python no será necesario compilar e instalar el software de nuevo en el sistema y bastará con realizar las modificaciones sobre la propia instalación. Antes de realizar las modificaciones necesarias se estudió y comprensión de todo el código de Clues, el cual es accesible a través del Github de GRyCAP [13].

Una de las características más destacables de Clues es la utilización de una serie de planificadores destinados a diferentes tareas. Cada uno de estos planificadores determina las acciones que el sistema debe llevar a cabo cuando suceden ciertos eventos, como por ejemplo, la llegada de un nuevo trabajo a la cola. En este caso, el planificador encargado del encendido de los nodos deberá ser capaz de determinar si es necesario encender algún nodo, o por el contrario los recursos disponibles son suficientes para satisfacer la petición.

Puesto que la funcionalidad de nuestra mejora se centra en el tiempo de apagado de los nodos, debemos centrar nuestra atención en el planificador encargado de determinar cuando se debe apagar un nodo. El planificador que necesitamos es el “CLUES_scheduler_PowOff_IDLE” definido en el fichero “schedulers.py”. Este planificador comprueba de manera periódica si alguno de los nodos del cluster ha permanecido en modo ocioso un tiempo igual o mayor al definido en la configuración, de tal manera que deba ser apagado. Además de esto, este planificador realiza la lectura de los diferentes parámetros de configuración necesarios, para la toma de decisiones.

Será necesario por lo tanto añadir algunos parámetros extra a la configuración de Clues. La lectura del archivo de configuración se realiza a través de una serie de funciones definidas en el archivo “config.py” que forma parte de la biblioteca CpyUtils (creada por GRyCAP), de la cual Clues hace uso. Esta biblioteca, al igual que Clues, es accesible a través del GitHub de GRyCAP [15].

3.4.2. Modificaciones sobre el código de Clues

Una vez comprendido el funcionamiento de Clues, es momento de comenzar con la modificación del código.

Archivo de configuración

Lo primero que haremos será determinar que parámetros será necesario añadir al archivo de configuración de Clues. Puesto que definiremos tres grupos de nodos diferentes (siguiendo el ejemplo de configuración que se comentaba anteriormente), añadiremos tres pares de nuevos parámetros. Estos tres pares se definirán de la siguiente manera.

- GRUPO X: contendrá la lista de nodos pertenecientes al grupo "X" separados por comas.
- IDLE_TIME_GROUP X: este parámetro definirá el tiempo que debe permanecer cualquiera de los nodos del grupo "X" sin recibir trabajo antes de ser apagado.

De tal manera que el aspecto del archivo de configuración de Clues sea algo similar a esto.

```
# Time (in seconds) to consider that a node that has remain idle can be powered
   off
IDLE_TIME=60
# Seconds that must pass before considering a node to be powered off (e.g. it has
   just been powered off and it is idle again)
COOLDOWN_NODES=30

GROUP1=compute-0-0.local,compute-0-1.local,compute-0-2.local,
       compute-0-3,compute-0-4,compute-0-5
GROUP2=compute-0-6,compute-0-7,compute-0-8,compute-0-9,
       compute-0-10,compute-0-11,compute-0-12
GROUP3=compute-0-13,compute-0-14,compute-0-15,compute-0-16,compute-0-17
IDLE_TIME_GROUP1=25
IDLE_TIME_GROUP2=50
IDLE_TIME_GROUP3=75
```

Planificador del apagado de nodos

Puesto que el planificador "CLUES_scheduler_PowOff_IDLE" se encarga del apagado de los nodos cuando su tiempo sin recibir trabajo supera el definido por el administrador, deberemos ampliar esas funciones de tal manera que antes de comprobar si el nodo supera dicho tiempo determine si este nodo pertenece a alguno de los grupos que se han definido. En caso de que así sea, la condición que determina si supera el tiempo será diferente para cada uno de los grupos, puesto que se define un tiempo para cada uno de ellos. En caso de que el nodo no pertenezca a ninguno de los grupos, la planificación seguirá siendo la definida por defecto.

Lo primero, por lo tanto, será hacer que el planificador lea los nuevos parámetros de configuración. Para ello, deberemos modificar la llamada que utiliza para recoger los valores del fichero de configuración, añadiendo en los parámetros de la función las etiquetas de todos los campos que vamos a leer.

```
cpyutils.config.read_config("scheduling",
          {
              "COOLDOWN_NODES": 0,
```

```
            "IDLE_TIME": 1800,
            "GROUP1": "",
            "IDLE_TIME_GROUP1": 1800,
            "GROUP2": "",
            "IDLE_TIME_GROUP2": 1800,
            "GROUP3": "",
            "IDLE_TIME_GROUP3": 1800
            },
        self)
```

Esta lectura de los parámetros de configuración se realiza en el momento que se inicializa el planificador, de tal manera que sus valores no se alteran a no ser que se reinicie el servicio habiendo modificado el fichero de configuración.

Una vez hecho esto, debemos modificar la forma en la que el planificador selecciona los nodos como candidatos para ser apagados. Debe comprobarse para cada uno de los nodos si pertenece a alguno de los grupos definidos y en caso de pertenecer a alguno de ellos, comprobar si su tiempo sin recibir trabajo es mayor o igual que el definido para ese grupo antes de seleccionarlo como candidato para ser apagado. En primer lugar definiremos estos tiempos para cada uno de los grupos al igual que se hace con el tiempo general.

```
eps_time_idle = now - self.IDLE_TIME
eps_time_idle_group1 = now - self.IDLE_TIME_GROUP1
eps_time_idle_group2 = now - self.IDLE_TIME_GROUP2
eps_time_idle_group3 = now - self.IDLE_TIME_GROUP3
```

Por último, bastará con modificar ligeramente el bucle que comprueba el tiempo de cada uno de los nodos. De tal manera que tenga en cuenta la posible pertenencia de un nodo a un grupo. Deberemos agregar tres nuevas condiciones que comprueben si un nodo pertenece a alguno de los tres grupos. A continuación se muestra un fragmento del código que determina si el nodo pertenece al “Group1”, esto se realiza de forma similar para los demás grupos.

```
for n_id, node in nodes.items():
    for string in str(self.GROUP1).split(","):
        if string == node.name:
            if (node.enabled) and (node.timestamp_poweredon < eps_time_cooldown)
                and (node.state in [ Node.IDLE ]) and
            (node.timestamp_state eps_time_idle_group1):
                candidates_off.append(n_id)
                break
```

Tras estas modificaciones, el sistema Clues es capaz de dar soporte a la gestión de nodos a través de grupos estáticamente definidos.

Es posible instalar Clues junto con las mejoras implementadas a través de los archivos contenidos en la carpeta “Clues” del CD siguiendo las instrucciones de instalación disponibles en el GitHub de GRyCAP [13].

3.4.3. Panel de configuración

Como se ha comentado anteriormente, Clues no ofrece una interfaz gráfica de configuración, lo que hace de este proceso una tarea tediosa. Es por ello que se ha implementado un nuevo panel de configuración enfocado a configurar las nuevas funcionalidades que se han añadido al sistema, de tal manera que resulte sencillo de configurar.

El panel de configuración que se muestra en la figura 3.8, contiene campos que permiten configurar los diferentes tiempos de espera para los diferentes grupos. Además de esto, permite seleccionar una de las diferentes colas de trabajo que hay definidas en el sistema. De esta forma solo se permitirá gestionar los nodos que forman parte de esa cola de trabajo. Estos nodos están definidos de forma estática en el archivo "groupPanel.php" contenido en el CD que se adjunta con esta memoria.

Figura 3.8: Panel de configuración para los grupos de nodos.

Cada uno de los nodos puede tomar uno de cuatro posibles valores para su configuración: deshabilitado (el nodo no será utilizado), "grupo1", "grupo2" y "grupo3". En caso de no seleccionar ninguna de estas posibilidades el nodo no pertenecerá a ningún grupo y será gestionado por Clues de la forma habitual.

El acceso a este panel de configuración se realiza a través del botón "configuración avanzada" del panel principal de configuración que se muestra en la figura 3.4.

Toda la interfaz de gestión de grupos ha sido desarrollada con HTML, PHP y Boostrap2 [14], puede consultarse el código en el CD adjunto. Es necesario que todos los archivos contenidos en el subdirectorio "PanelGrupos" estén ubicados en la carpeta desde la que el servidor sirve los archivos, junto con las carpetas "res" y "test".

3.5. Sistema de alimentación ininterrumpida

La proliferación de los clusters de ordenadores durante los últimos años ha provocado el desarrollo del estudio de sistemas destinados a garantizar la calidad y continuidad del suministro eléctrico. Una

de las alternativas más utilizadas para garantizar esta continuidad del suministro eléctrico son los sistemas de alimentación ininterrumpida (SAI), los cuales con capaces de mantener el suministro durante un periodo limitado de tiempo [16].

El objetivo de estos dispositivos es el de detectar perturbaciones en la red eléctrica que puedan perjudicar al funcionamiento de los de los equipos alimentados. En el peor de los casos son capaces de apagar o reiniciar los equipos hasta que los valores de suministro vuelvan a ser normales. Es algo común que estos SAI estén complementados con un pequeño proceso software que realiza ciertas acciones en caso de detectar algún problema en el suministro.

3.5.1. Elección e instalación de una SAI

Los SAI, dependiendo de la tecnología de que utilicen se clasifican en dos categorías:

- SAI rotativos: están compuestos por un motor de inducción, un volante de inercia y un alternador. Su funcionamiento se basa en que ante una caída de tensión, el volante actúa como almacenador de energía, amortiguando las variaciones de la red eléctrica con respecto a los valores normales. El diseño de este sistema es de por si ineficiente, ya que en ausencia de perturbaciones en el suministro consume una cantidad de energía innecesaria debido a la perdidas del motor de inducción, el volante y el alternador.

- SAI estáticos: a diferencia de los SAI rotativos, no contienen elementos en movimiento, sino que están compuestos de elementos electrónicos de potencia y baterías que se encargan de entregar el suministro en caso de una perturbación eléctrica. El principal problema de estos sistemas, es que en caso de caída de la red eléctrica la capacidad de suministro depende de la capacidad de las baterías, y por lo tanto, es limitado.

En nuestro caso, disponemos de un cluster de dieciocho máquinas, en el que no es necesario mantener una disponibilidad total. Es por ello que el SAI que mejor se adapta a las necesidades de nuestro cluster es el SAI estático. En caso de detectar una caída en la corriente sería suficiente reportar a los usuarios con la notificación del incidente y el identificador de todos sus trabajos afectados por este. Para ello será necesario un proceso que realice todas estas funciones en caso de detectar una caída.

3.5.2. Procesos complementarios para la SAI

Una vez se ha decidido qué tipo de SAI se adapta mejor a las necesidades de nuestro cluster, es necesario generar dos procesos que se encargaran de la realización de las acciones pertinentes en caso de una caída en el suministro eléctrico.

Proceso de escucha

El primero de estos procesos será responsable de realizar una escucha de los mensajes proporcionados por el SAI. De esta manera se detectará la señal en caso de que exista algún problema con la alimentación. En caso de recibir esta señal deberá encargarse de poner en marcha el proceso de gestión de fallos en el suministro energético.

Debido a que finalmente no fue posible adquirir un SAI, no se realizó la implementación del demonio de escucha, puesto que no es posible comprobar su correcto funcionamiento sin el dispositivo.

Proceso de gestión de fallos

El proceso de gestión de fallos en el suministro estará a cargo de detener todos aquellos trabajos que se encuentren en ejecución en el momento que se reciba una señal de fallo por parte del SAI. Además de esto, deberá notificar del incidente tanto a los usuarios a los que pertenecen los trabajos en ejecución como a aquellos que tenían trabajos esperando en la cola. La notificación se realizará vía mail. Una vez todos los usuarios hayan sido notificados, el proceso procederá a realizar un apagado ordenado de los diferentes nodos activos del cluster.

Para ello se ha definido un script cuyo código puede consultarse en el apartado "notifySAI" del Apéndice C (código contenido en CD). Este script recorre la lista de trabajos del cluster, recogiendo la información necesaria de cada uno de ellos, como: el mail del usuario, el identificador del trabajo, la fecha y hora en la que se agrego los que se exponen a continuación. Una vez haya recopilado esta información creará un archivo de log en cada uno de los directorios de trabajo de los usuarios afectados. Este archivo de log será enviado vía mail a los usuarios. El formato de este log puede verse en la figura 3.9.

```
------------26/06/2017 17:13------------
[root]187, trabajo1.pl, 06/19/2017 17:35:08, t,  /root,  /root
[root]188, trabajo2.pl, 06/19/2017 17:36:10, t, /root, /root/programas
```

Figura 3.9: Fichero de log para incidentes SAI.

La primera linea representa la fecha del incidente. Cada una de las entradas que aparecen debajo de esta linea representan los trabajos afectados, cuyos campos son:

- Usuario: usuario al que pertenece el trabajo afectado.
- ID: identificador del trabajo.
- Nombre: Nombre que el usuario ha dado a la tarea.
- Fecha y hora: fecha y hora en la que el trabajo fue agregado a la cola.
- Estado: estado del trabajo en el momento del incidente.
- Directorio: directorio en el cual puede encontrar el log de notificación.
- Directorio de trabajo: directorio de trabajo desde el cual se lanzo el trabajo.

Capítulo 4

Conclusiones y Trabajo futuro

4.1. Conclusiones

Llegados a este punto y tras la realización de este trabajo, es sencillo realizar algunas afirmaciones sobre los sistemas de computación y la eficiencia energética de en estos.

La computación es una de las herramientas más utilizadas e imprescindibles en prácticamente todos los campos de la ciencia. Al mismo tiempo, su crecimiento y evolución en los últimos años hacen de ella una herramienta atemporal puesto que su desarrollo esta a la par de los avances científicos más importantes, es por esto que a día de hoy, es imposible imaginarse un futuro que no haga uso de la computación.

Por otra parte, el uso de esta herramienta acarrea un consumo energético muy elevado, haciendo de ella una herramienta costosa y poco eficiente energéticamente. Este elevado consumo provoca un coste tanto económico como medioambiental.

Es este coste derivado del consumo energético lo que provoca que las compañías traten de buscar soluciones al problema del desmesurado consumo de la computación. Una de las soluciones más extendidas es la implementación de los sistemas de gestión energética, estos sistemas tratan de establecer una política que permita alcanzar unos objetivos de consumo preestablecidos.

Estos sistemas de gestión son un buen primer paso para alcanzar la eficiencia energética en la computación. Ejemplos como la plataforma de virtualización Kefren o el cluster paralelo ODIN, que implementaron uno de estos sistemas (Clues) consiguieron tras cinco meses una reducción de su consumo de un cuarenta y un cincuenta por ciento respectivamente. Son ejemplos perfectos para demostrar la necesidad de la búsqueda de soluciones ante un problema inevitable pero mitigable.

Por último, es obvio que la computación seguirá avanzando y evolucionando con el paso de los años. Según las tendencias actuales, el consumo energético seguirá aumentando conforme esta evolución se haga patente. Por ello es necesario que estos sistemas de gestión energética evolucionen con la misma rapidez, mejorando sus estrategias y funcionalidades haciendo de estos sistemas una herramienta más útil y eficiente.

4.2. Trabajo futuro

Debido a que la realización de este trabajo tiene unos limites temporales, hay muchas implementaciones y ampliaciones que no han podido ser más que un planteamiento o una idea en la que sería apropiado trabajar en un futuro. A continuación se exponen posibles lineas de trabajo futuro que amplían el contenido de este trabajo.

4.2.1. Contenido web dinámico

Es algo habitual que el contenido de los portales y paginas web no sea un contenido estático, sino que se recoja la información de una base de datos, o se genere de una determinada forma.

Debido al tiempo limitado para la realización del trabajo, información contenida en la interfaz web "Gestión de grupos" está definida de forma estática. Sería conveniente modificar esta interfaz de tal manera que las colas de trabajo y los diferentes nodos que forman parte de estas fuesen recogidos de forma dinámica, por ejemplo a través de código PHP, o incluso definiendo una pequeña base de datos para todos estos datos de los que las interfaces web hacen uso.

4.2.2. Instalación de un sistema SAI

Debido a la imposibilidad de obtener un sistema de alimentación ininterrumpida, no fue posible realizar una instalación hardware real en el cluster. A pesar de esto, se implementó el proceso que debería desencadenar la recepción de una señal de alerta por parte de este sistema. Sería una práctica interesante la obtención e instalación de uno de estos sistemas, ya que se trata de una practica habitual en todo sistema de computación que se precie.

Además de esta instalación se debería desarrollar un nuevo proceso de escucha, que detectase cualquier señal de alerta que el SAI provocase. Por otra parte el desarrollo de estos procesos no es imprescindible, ya que existe software encargado de tratar con estas señales de alerta. El problema fundamental de este software es que las acciones que este realiza se limitan al apagado de los nodos y en ocasiones una notificación a través de un email al administrador. Es por esto que se decidió implementar un proceso que pudiera realizar un protocolo de apagado de emergencia que incluya la detención de todos los trabajos, la generación de un log y la notificación a los usuarios afectados.

4.2.3. Nuevas mejoras para Clues

La implementación de mejoras sobre Clues es un tema bastante extenso, es posible implementar prácticamente cualquier funcionalidad que otro sistema de gestión energética implemente. Algunas mejoras que podría ser interesante implementar sobre Clues son las siguientes:

- Valorar la eficiencia de los nodos: algunos sistemas de gestión energética utilizan medidas de rendimiento de los diferentes nodos que componen el cluster para determinar cual de estos nodos es mejor encender para servir una petición de trabajo. Esta mejora aporta cierta mejora en el rendimiento del sistema, lo que lo convierte en un sistema más productivo y eficaz.
- Interfaz web: a pesar de que las primeras versiones de Clues permitieran la gestión del sistema a través de una interfaz web, las últimas versiones no implementan esta funcionalidad, lo que

afecta a la configuración convirtiéndola en una tarea tediosa y delicada (deben modificarse los archivos directamente). Es por esto que la implementación de una interfaz web completa sería una buena mejora para Clues.

Estas son algunas de las mejoras que se han planteado durante la realización de este trabajo, las cuales aportarían una mejora en cuanto a la implementación de Clues en un cluster genérico. Puesto que Clues esta diseñado de manera modular, sería algo sencillo seguir aumentando sus funcionalidades haciendo que se adapte a las necesidades de cualquier sistema de computación en el que se instale.

4.2.4. Implementación en un sistema real

Tras la realización de este trabajo, se ha decidido implementar Clues como sistema de gestión energética en los clusters del Multiscale Materials Modeling Group (MMM) de la Universidad de Valladolid [17]. Para ello se utilizaran muchos de los conocimientos adquiridos durante la realización de este trabajo.

El trabajo consistirá en implementar Clues junto con las mejoras que se han desarrollado. Pudiendo implementarse alguna mejor adicional dependiendo de las características del sistema.

4.2.5. Comunicación con GRyCAP

Durante la realización de este trabajo se ha mantenido contacto con la organización GRyCAP (desarrolladora de Clues y CpyUtils). Tras el interés mostrado por las diferentes mejoras sobre su sistema, se comunicará todo lo desarrollado a fin de que pueda ser implementado de manera oficial en Clues.

Apéndice A

Computación de alto rendimiento

La computación de alto rendimiento abarca un gran conjunto de tecnologías, como los supercomputadores o los clusters de computación y comunicación, apoyando su funcionamiento en los paradigmas de programación distribuida y paralela, a través de los cuales, otorga dos puntos de vista para la resolución de problemas:

- **Programación distribuida:** utiliza un gran número de computadoras, junto a una infraestructura de telecomunicación distribuida. Proporciona un uso compartido de los recursos, los cuales pueden estar situados en lugares geográficos distintos y pertenecer a distintos dominios de administración
- **Programación paralela:** es una técnica de programación basada en la ejecución concurrente de procesos, ya sea en un único computador con uno o varios procesadores o en un cluster de computación. Existe un amplio abanico de posibilidades con respecto a la arquitectura, desde sistemas de memoria compartida hasta clusters de computadores comunicados a través de una red local Ethernet.

La computación de alto rendimiento ha facilitado la resolución de muchos problemas computacionales con una complejidad elevada, pero esto no siempre ha sido posible. Durante los primeros años desde lo que conocemos como computadoras modernas se consideraba que, por ejemplo, un servidor debía ser una única máquina potente capaz de dar servicio a un gran número de peticiones. Según este modelo, un aumento de la demanda en las peticiones supone la necesidad de cambio del servidor por una máquina más potente, quedando esta obsoleta con un alto coste asociado a este cambio.

Este modelo quedo obsoleto cuando en 1994 nació el proyecto Beowulf de la mano de Donald Becker y Thomas Sterling, el cual logro un alto nivel de rendimiento en el procesamiento poniendo a trabajar varias computadoras en paralelo comunicadas a través de una red Ethernet. Dando lugar al primer cluster de computación.

A.1. Clusters de computación

Se define el termino clúster de computación como un conjunto de ordenadores conectados entre sí a través de una red de alta velocidad, con un comportamiento que simula una única máquina real.

Las prestaciones ofrecidas por estas estructuras de computación son: alto rendimiento; gracias a la capacidad conjunta de computo que ofrece la conexión entre equipos, alta disponibilidad; ya que si uno de los equipos se ve afectado por un fallo otro equipo del cluster puede reemplazarlo, escalabilidad incremental; debido a que solo es necesaria la conexión de nuevos equipos para aumentar la potencia del cluster, balanceo de carga; dado que los trabajos se distribuyen entre los diferentes equipos que componen el sistema [18, 19].

A.1.1. Clasificación de los clusters

Con el paso del tiempo el uso de los clusters se ha extendido de tal manera que el termino clúster ha tomado diferentes connotaciones con respecto al uso que se hace de ellos. Es por esto que se puede realizar una pequeña clasificación de los clusters en función del trabajo para el que sean utilizados. Es algo común, a pesar de esta clasificación, que clúster de un determinado tipo presente también características propias de los demás. Esto es debido a que las características de las arquitecturas hardware y software son muy similares para todos los tipos de clúster [20].

Alto rendimiento (HPC)

Los clusters de alto rendimiento son los más extendidos debido a su utilidad en campos como la física o la industria aeroespacial, en tareas cada vez más complejas como simulaciones y grandes cálculos las cuales requieren de un sistema con una gran potencia de cálculo. Estos clusters se caracterizan por ejecutar tareas que requieren una gran capacidad de computo, o grandes cantidades de memoria. El transcurso de estas tareas puede comprometer los recursos del cluster durante largos periodos de tiempo.

Alta disponibilidad (HAC)

Se denomina cluster de alta disponibilidad a aquellos que brindan una alta disponibilidad y confiabilidad, utilizando para ello software de detección de fallos y sistemas de recuperación ante estos. Por otra parte, el diseño hardware de estos clusters suele orientarse a sistemas sin un punto único de fallo. Este tipo de clusters a menudo se utilizan como clusters de IT comerciales, permitiendo a las compañías ofrecer servicios a sus clientes con una alta disponibilidad.

Alta eficiencia (HTC)

El objetivo de estos clusters es el de ejecutar el mayor número de tareas en el menor tiempo posible, entendiendo las tareas como trabajos individuales con independencia entre sus datos, es por ello que la comunicación entre los nodos no supone un gran problema.

Otra posible clasificación dentro de los clusters podría ser su clasificación por el sistema de procesamiento, existen dos grandes sistemas de procesamiento, como son el procesamiento distribuido, y el procesamiento en paralelo.

A.1.2. Componentes de un cluster

Los clusters se componen de varios componentes tanto hardware como software, donde cada uno de ellos tiene una funcionalidad diferenciada y bien definida. A continuación se exponen los componentes principales de un cluster de computación.

Nodos

En el ámbito de la computación a través de clusters, se define nodo como un centro de computo al cual se le pueden asignar tareas. Estos nodos pueden ser simples ordenadores con un único procesador, sistemas multiprocesador o estaciones de trabajo(computador de altas prestaciones dedicado a tareas técnicas y científicas).

Existen dos tipos de nodos dentro de un cluster diferenciados en la forma en la que realizan las tareas que se solicitan.

- Nodos dedicados: Estos nodos no disponen de periféricos de entrada como teclado, ratón o monitor, ya que su uso se centra exclusivamente en la realización de las tareas que le son asignadas.
- Nodos no dedicados: En este caso los nodos sí disponen de teclado ratón y monitor. En este caso se busca aprovechar los ciclos de reloj que el usuario del nodo no utiliza para asignarle trabajo.

Almacenamiento

Debido al avance en el diseño e implementación de los clusters de computación dentro de las empresas y compañías, el almacenamiento de datos en estos sistemas se ha vuelto un aspecto a tener en cuenta, llegando a ser el diseño de este igual o más complejo que el resto de los componentes del cluster [21].

En la mayor parte de los casos los clusters requieren de un sistema de almacenamiento compartido que permita a los diferentes nodos acceder a los mismos datos e información. De esta forma todos los nodos pueden leer y escribir sobre los mismos archivos, un sistema de archivos compartido es por lo tanto una condición casi indispensable para cualquier cluster.

La Figura A.1 muestra el esquema básico de un almacenamiento en un cluster, sobre este esquema se explicaran los posibles diseños y soluciones para el almacenamiento en un cluster.

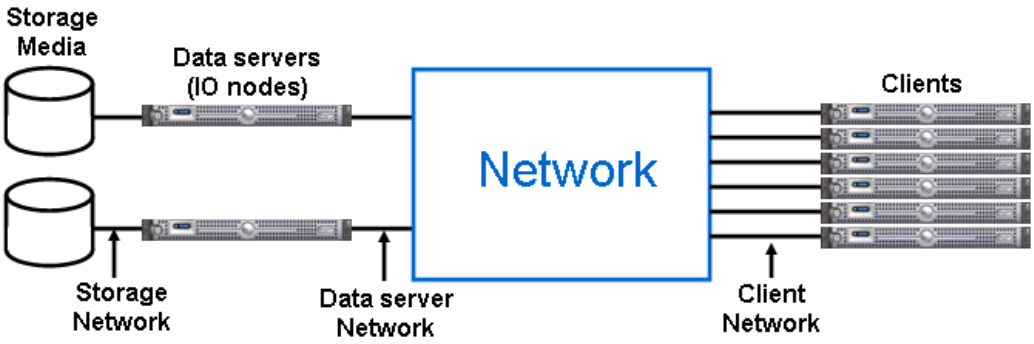

Figura A.1: Esquema básico de almacenamiento de un cluster.

Observando el esquema de izquierda a derecha, se puede ver como el almacenamiento esta conectado a los servidores de datos a través de una red de almacenamiento, la función principal de estos servidores de datos es la de proporcionar los sistemas de archivos a los clientes. Estos servidores se comunican con los clientes a través de la red principal, la cual sirve como punto de encuentro entre las redes de los clientes y de los servidores de datos.

Este esquema representa un caso extremo con una gran variedad de componentes, en la mayor parte de los casos el sistema de almacenamiento suele componerse de un subgrupo de estos.

Una de las soluciones más recurridas en cuanto a almacenamiento de un cluster HPC es NFS, el cual es el único estándar en cuanto a sistemas de ficheros compartidos se refiere. La figura A.2 muestra un esquema de almacenamiento compartido NFS.

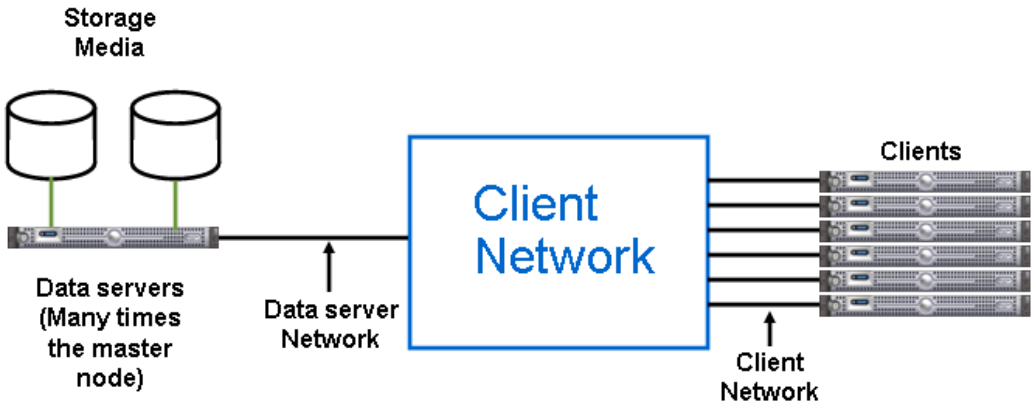

Figura A.2: Esquema de un sistema de almacenamiento NFS.

La figura A.2 muestra un único servidor de datos para todo el sistema de archivos NFS(denominado "filer head"), el cual será el encargado de servir las peticiones de los clientes. Es común que en ciertas implementaciones sea el nodo maestro el que desempeña la labor de servidor de datos, estando este directamente conectado con la red de los clientes.

Por otra parte, NFS tiene ciertas limitaciones. La primera de ellas es que al tener un único servidor de datos es necesario que todas las peticiones de los clientes pasen por este servidor, provocando un cuello de botella en este punto. Esta situación se ve mitigada por los patrones de entrada salida que

implementan la mayoría de las aplicaciones paralelas. Otra limitación importante es que NFC tiene ciertas dificultades para permitir la escritura simultanea de un fichero por parte de múltiples clientes.

Existen ciertos casos donde las limitaciones de NFS hacen necesaria la implementación de una nueva solución. En casos como:

- Clusters de gran tamaño (Gran número de nodos).
- Clusters con una alta demanda de entrada salida compartida (habitualmente paralela).
- Necesidad de un sistema con alta escalabilidad en cuanto a prestaciones y capacidad.

En estos casos, en los cuales NFS no es suficiente, se requieren sistemas de ficheros paralelos y distribuidos los cuales permitan a los diferentes nodos del cluster acceder de forma remota a la misma información, y a su vez poder leer y escribir de manera simultanea dicha información. La implementación de estos sistemas trae consigo una alta complejidad añadida, ya que la solución implementada depende del cluster, se requiere de un procedimiento de análisis y testeo que determine que componentes incluirá la nueva solución

Middleware

Un middleware es un software que actúa como una interfaz entre el sistema operativo y las aplicaciones de forma que puedan comunicarse e interactuar. El uso de un middleware en un cluster lo provee de:

- Una interfaz única de acceso al sistema o SSI (Single System Image), la cual genera la impresión al usuario de que interactúa con un único ordenador.
- Herramientas para la optimización y mantenimiento del sistema (migración de procesos, checkpoint-restart, balanceo de carga...etc).
- Escalabilidad: permitiendo la detección de nuevos equipos conectados al cluster para ser utilizados.

Gestores de colas

Los gestores de colas son sistemas que permiten controlar y planificar la ejecución de las tareas almacenadas en las colas de trabajo. Su finalidad es la de optimizar el uso de los recursos minimizando su uso y maximizando su rendimiento. Algunos de los gestores de colas más extendidos son:

- Sun Grid Engine (SGE): SGE es un gestor de colas desarrollado por Sun Microsystems. Algunas de sus características principales son la reserva de recursos anticipada, utiliza matrices de trabajo y recursos, permite la gestión de trabajos en paralelo y es compatible con sistemas de BigData [22] [23].
- TORQUE/PBS: Torque es un gestor de colas derivado de NQS. Permite la gestión de trabajos en paralelo, siendo uno de los gestores más utilizados en la actualidad [24].
- OpenNebula: OpenNebula es un gestor de colas enfocado a la computación en la nube, siendo uno de los mas utilizados para este tipo de sistemas. Posee mecanismos de estandarización e interoperabilidad lo que lo hacen un sistema muy flexible y versátil [25].

A.2. Supercomputadores

Un supercomputador o superordenador se caracteriza por tener una inmensa capacidad de calculo de operaciones, que sobrepasa ampliamente a las capacidades de un ordenador de propósito general. Esta capacidad de computo se debe a que estas máquinas tienen un número masivo de procesadores, ya sean distribuidos y conectados a través de una red, o conectados físicamente.

Estos superordenadores suelen especializarse en operaciones paralelas y su jerarquía de memoria esta minuciosamente diseñada para las CPU mantengan trabajo constante, a diferencia de los ordenadores de propósito general que pierden grandes cantidades de ciclos de CPU con operaciones de entrada/salida.

Muchas disciplinas científicas serían impensables a día de hoy sin lo que un supercomputador representa como herramienta, como por ejemplo la biología, la climatología o cualquier estudio que requiera simulaciones con grandes cantidades de datos y variables.

A.2.1. Historia

El término supercomputador se introduce en la época de 1960, con el supercomputador conocido como “Atlas” instalado en la universidad de Manchester y diseñado por Seymour Cray que trabajaba para le empresa Control Data Corporation.

Las prestaciones de estos primeros supercomputadores, los cuales solo utilizaban unos cuantos procesadores era muy inferior a las prestaciones ofrecidas por las máquinas modernas en caso de “Atlas”, algunas de sus características son [26]:

- Tamaño de palabra de 48 bits.
- Un espacio de direcciones de 2 millones de palabras.
- Velocidad de 0.6 FLOPS.
- 0.096 Mbytes de memoria RAM.

A pesar de que a día de hoy una máquina con estas características suena impensable, estos primeros supercomputadores pusieron las bases para muchos conceptos que a día de hoy se siguen utilizando, como por ejemplo el sistema ”extracode” que incorporaba el Atlas es el predecesor de lo que a día de hoy conocemos como firmware.

A.2.2. Arquitectura

Esta sección centra la atención en los dos modelos arquitectónicos más utilizados y de los cuales derivan la mayor parte de todos los diseños de los supercomputadores modernos [27].

Paralelismo masivo centralizado

El paralelismo masivo centralizado surge en la década de 1980 cuando la demanda de potencia de cálculo comienza a aumentar. La tendencia de utilizar un gran número de procesadores comenzó ha hacerse patente en muchos de los sistemas que diseñados, todos estos utilizando memoria y sistemas de archivos distribuidos, ya que la memoria compartida no permitía escalar hasta un gran número de procesadores.

Este enfoque es similar a las arquitecturas utilizadas para los clusters de computación, ya que se implementa la conexión de un gran número de nodos a través de una red privada de alta velocidad. Estos nodos están controlados por un middelware, el cual permite ver el sistema como una única unidad de computo.

El problema que surge de este diseño es que al conectar un gran número de nodos semi independientes la velocidad y flexibilidad de las interconexiones se vuelve un punto de interés critico. Los diseños modernos abordan este problema desde diferentes aproximaciones, por ejemplo: El Thiane-2 (que ocupa la segunda posición en el top 500) utiliza una red propia de alta velocidad basada en el sistema InfinitiBand QDR (la cual proporciona un gran ancho de banda y una baja latencia) llama TH Express [28], por otra parte otros sistemas como el BlueGene utiliza una red tridimensional de toroides interconectados con una red auxiliar.

Paralelismo masivo distribuido

El modelo de paralelismo masivo distribuido basa su arquitectura en la conexión de un número masivo de sistemas de computación los cuales se encuentran en diferentes dominios administrativos a través de la red. Se trata de un enfoque ”oportunista”, el cual utiliza los recursos del sistema siempre que estos estén disponibles. Aunque este sistema ha abordado tareas paralelas de manera satisfactoria, simulaciones que exigen una gran capacidad de computo han quedado fuera de su alcance, surgiendo los sistemas distribuidos casi oportunistas.

La supercomputación casi oportunista tiene como objetivo el proporcionar una mayor calidad de servicio en el intercambio de recursos. Permitiendo la ejecución de aplicaciones exigentes. Esto se debe a un sistema de asignación de recursos tolerante a fallos apoyado en un sistema de paso de mensajes.

A.2.3. Aplicaciones

Los sistemas de supercomputación han tenido gran repercusión en un gran número de áreas de estudio como por ejemplo la climatología y la física. Siendo utilizados tanto para el procesamiento de grandes cantidades de información como para la realización de exigentes simulaciones con un gran número de datos. La siguiente tabla resume las tendencias en las aplicaciones que la supercomputación a tenido a lo largo de los años.

Decada	Usos y aplicaciones
1970	Previsión meteorológica e investigación aeronáutica
1980	Análisis probabilistico, modelos para la protección contra la radiación
1990	Decodificación por fuerza bruta
2000	Simulación 3D de pruebas nucleares
2010	Simulaciones moleculares dinámicas

Tabla A.1: Aplicaciones de la supercomputación a lo largo de los años.

Apéndice B

Instalación de Rock Cluster

Rocks Cluster es una distribución Linux para clusters HPC basada inicialmente en la distribución Red Hat Linux, sin embargo las versiones más modernas están basadas en la distribución CentOS (a partir de la versión 4.3). Una de las ventajas de utilizar Rocks para crear y administrar un cluster es que a menudo la creación y administración de estos resulta costosa, y la necesidad de añadir y administrar el software a medida que el cluster se extiende se vuelve una tarea compleja. Rocks Cluster provee al administrador un mecanismo de abstracción el cual permite controlar esta complejidad [29].

La instalación de Rocks se realiza a través de un instalador Anaconda modificado, que simplifica la instalación 'en masa' del sistema. Rocks incluye un gran número de herramientas que no forman parte del sistema CentOS pero son componentes integrales del sistema permitiendo convertir un grupo de ordenadores en un cluster. La instalación es fácilmente personalizable a través de Rolls"(paquetes software que extienden el sistema), los cuales se agregan y configuran durante la instalación del sistema, algunos ejemplos de estos Rolls son: SGE roll, Ganglia roll, Java roll...etc.

B.1. Prerrequisitos del sistema

B.1.1. Requisitos hardware mínimos

Nodo FrontEnd

- Espacio en el disco: 30GB.
- 1GB de memoria RAM.
- 2 puertos Ethernet.
- Orden de BIOS boot: CD, Hard Disk.

Nodo de computación

- Espacio en el disco: 30GB.
- 1GB de memoria RAM.

- 1 puerto Ethernet.
- Orden de BIOS boot: CD, PXE (Network Boot), Hard Disk.

B.2. Configuración previa

Antes de comenzar con la instalación conviene revisar y modificar la configuración de los equipos para así evitar posibles problemas durante el proceso de instalación.

B.2.1. Conexión de los equipos y acceso a la red

Debido a que Roks asigna de manera automática el puerto ”eth0” del FrontEnd a la red interna del cluster y el puerto ”eth1” como puerto de acceso a la red externa, es importante revisar dichas conexiones de tal manera que se asemejen a las conexiones de la figura B.1, ya que podrían asignarse de manera incorrecta, lo que supondría la necesidad de reinstalar el sistema.

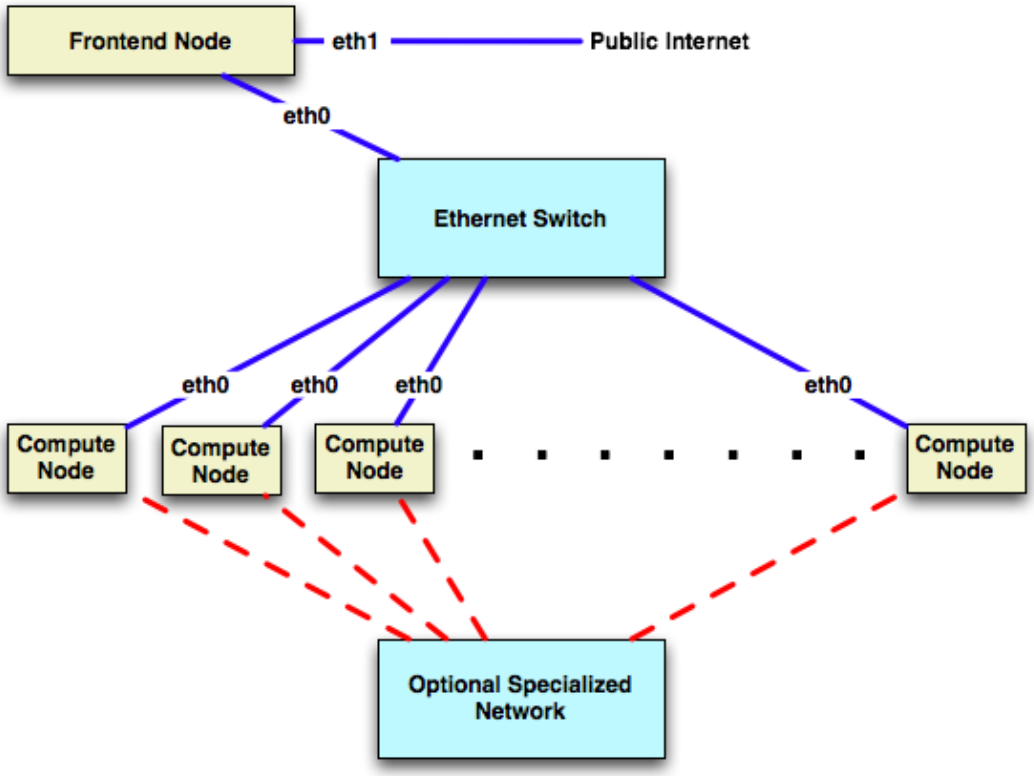

Figura B.1: Esquema de conexiones para un cluster Rocks.

B.2.2. Wake on Lan

En caso de que se requiera administrar el apagado y encendido de los nodos es necesario habilitar en cada uno de ellos Wake on Lan, de esta forma podremos administrar dichos nodos a través del envío de un ”magic packet” desde el FrontEnd.

Para ello debemos acceder a la BIOS de cada uno de los equipos y a través del menú ”Power/ACPI” habilitar la funcionalidad de WOL, como se muestra en la figura B.2.

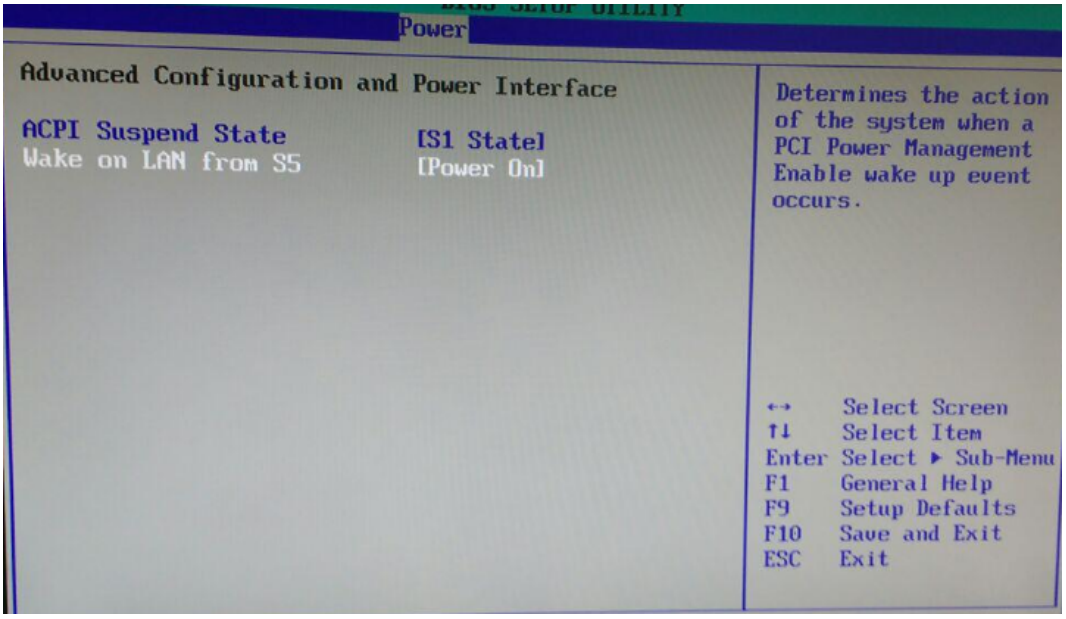

Figura B.2: Activación de WOL a través de la BIOS.

B.2.3. PXE (Network Boot)

Para la instalación de los nodos de computación del cluster, es necesario que estos puedan bootearse a través de la red, es por ello que es necesario activar la opción PXE en la BIOs de los diferentes nodos.

Accedemos al menú de la BIOS de cada uno de los equipos, accediendo al submenú ”Boot”, donde debemos deshabilitar la opción ”Silent Boot” como se muestra en la figura B.3. A continuación simplemente activamos la opción de ’PXE boot to Lan”.

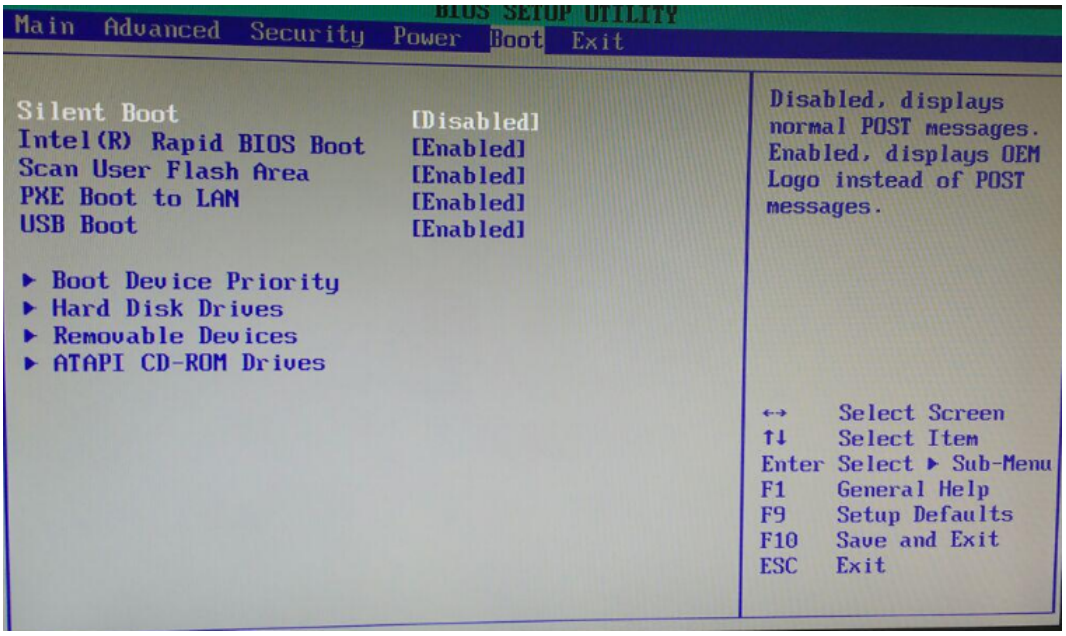

Figura B.3: Activacion de PXE y desactivación de SilentBoot en la BIOS.

Una vez hecho esto debemos asegurarnos de que la secuencia de arranque del equipo comience con PXE (o bien como segunda opción tras CD/DVD).

B.2.4. Hyper-Threading

En caso de que algunos de los nodos que formen parte del cluster dispongan de un procesador de un solo núcleo es importante asegurarse de que este no tenga activado el modo de hyper-threading (el cual aprovecha el procesamiento multi-hilo para simular que el procesador tiene dos núcleos).

Esto se debe a que un procesador simulando tener dos núcleos puede causar problemas a la hora de asignar trabajos a los diferentes equipos. Para deshabilitarlo basta con acceder al menú principal de la BIOS y desactivar la opción como se muestra en la figura B.4.

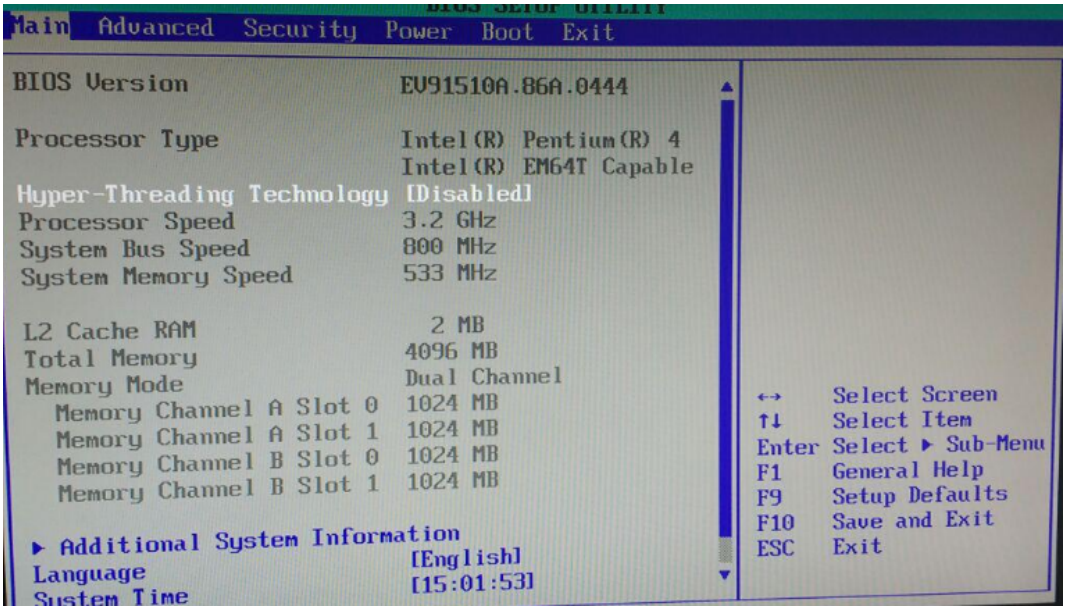

Figura B.4: Desactivación del Hyper-Threading a través de la BIOS.

B.3. Instalación del FrontEnd

Una vez configurados todos los equipos debemos encender aquel que desempeñará el rol de FrontEnd del cluster, para ello antes deberemos haber introducido el CD con la imagen de Rocks Cluster para proceder a su instalación. Una vez el equipo se haya encendido aparecerá una pantalla como la de la figura B.5.

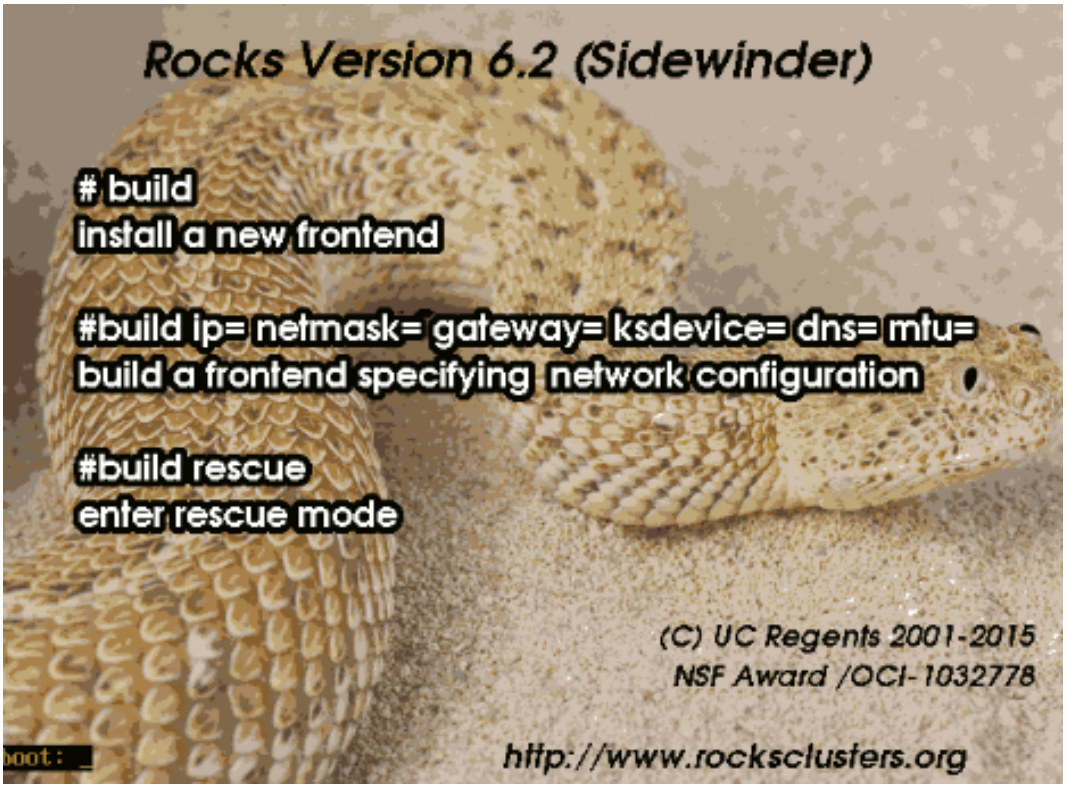

Figura B.5: Panatalla de inicio de la instalación de Rocks.

Para comenzar con la instalación del ForntEnd escribimos:

```
build ksmservice=eth1 asknetwork
```

A continuación se muestran dos pantallas, en las cuales se configuran los parámetros de red para el equipo, la configuración varía dependiendo de cada sistema, en nuestro caso las figuras B.6 y B.7 muestran la configuración utilizada.

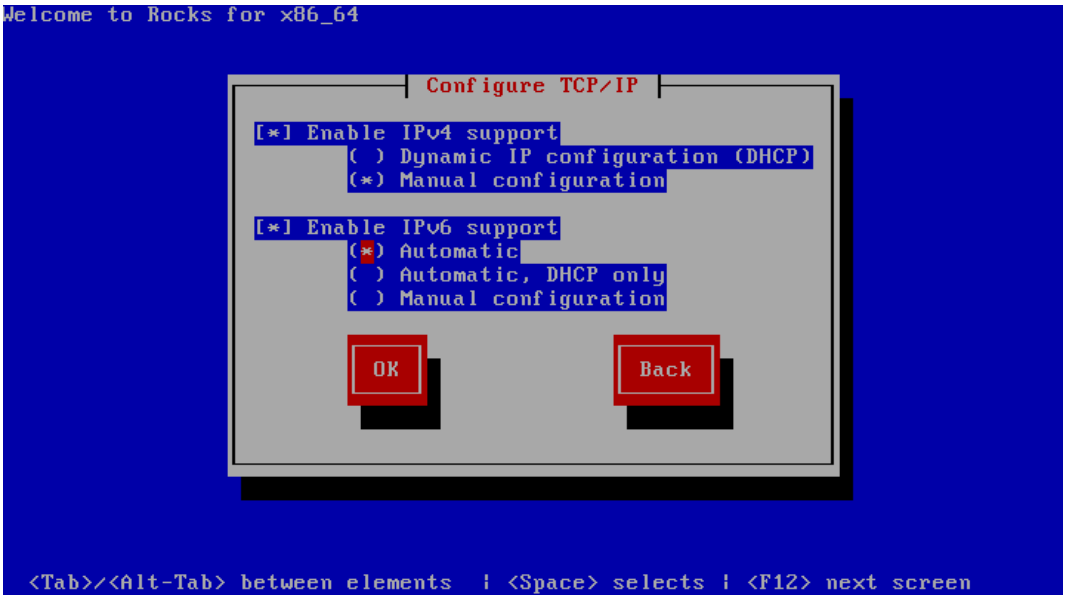

Figura B.6: Primera pantalla de configuración de red para Rocks.

Seleccionamos configuración manual para Ipv4 y automática para Ipv6, de esta manera nos per-

mitirá configurar de manera manual la IP pública del equipo a través del menú que se muestra en la figura B.7.

Figura B.7: Configuración de IP pública para la instalación de Rocks.

Una vez hemos configurado los datos de red y el proceso de carga finaliza se muestra una pantalla en entorno gráfico mediante la cual configuraremos la instalación del nodo frontal, como la que se muestra en la figura B.8.

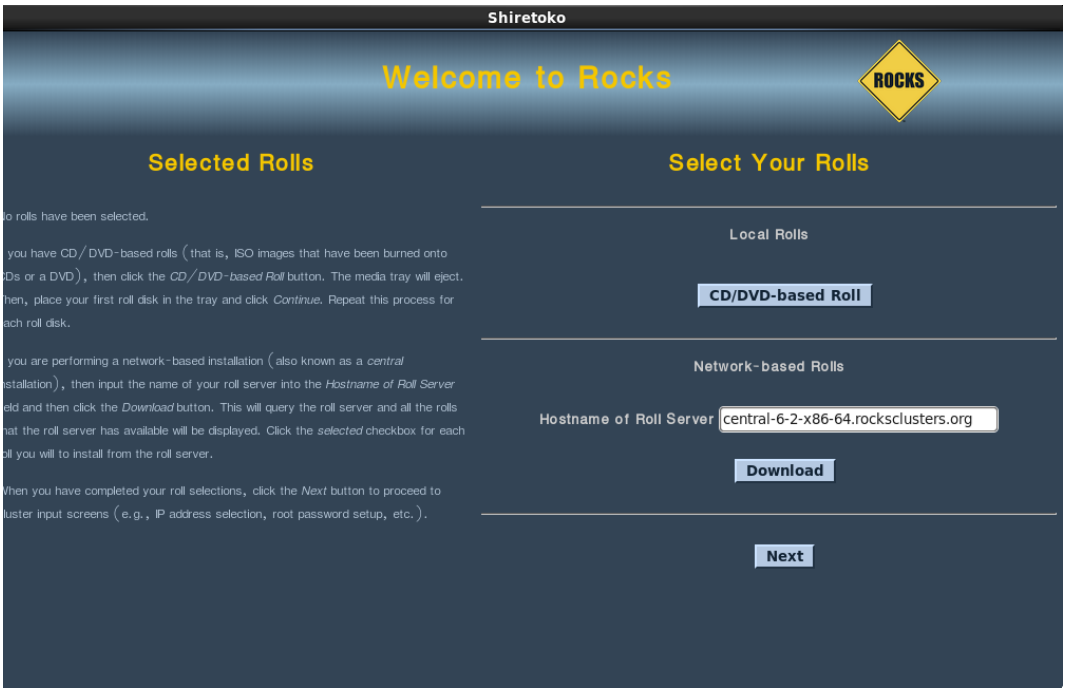

Figura B.8: Formulario de selección de rolls para la instalción de Rocks.

Seleccionamos ”CD/DVD-based Roll” para instalar aquellos rolls que necesitemos, una vez seleccionada la opción el sistema muestra una lista de todos los rolls disponibles, de los cuales seleccio-

namos: area51, base, os, kernel, hpc, ganglia, sge, web-server, python, java y perl.

Posteriormente los pasos a seguir son los típicos para cualquier instalación como son: Formulario de información del cluster(nombre, localidad donde se encuentra, organización responsable...etc), configuración de los servidores DNS y NTP, finalmente se solicita una contraseña para el usuario ”root”. Una vez hemos terminado todos los pasos debemos esperar hasta que termine el proceso de instalación. Si todo ha terminado correctamente el equipo se reiniciará con el sistema instalado y listo para ser utilizado, en la siguiente sección se explica como realizar la instalación del sistema para los diferentes nodos del cluster.

B.4. Instalación de los nodos

Una vez hemos instalado el sistema en el nodo frontal debemos añadir nodos a nuestro cluster, para ello abrimos una terminal y escribimos el siguiente comando:

```
$insert-ethers
```

Aparecerá un menú como el que se muestra en la figura B.9 en el cual debemos seleccionar la opción ”Compute”.

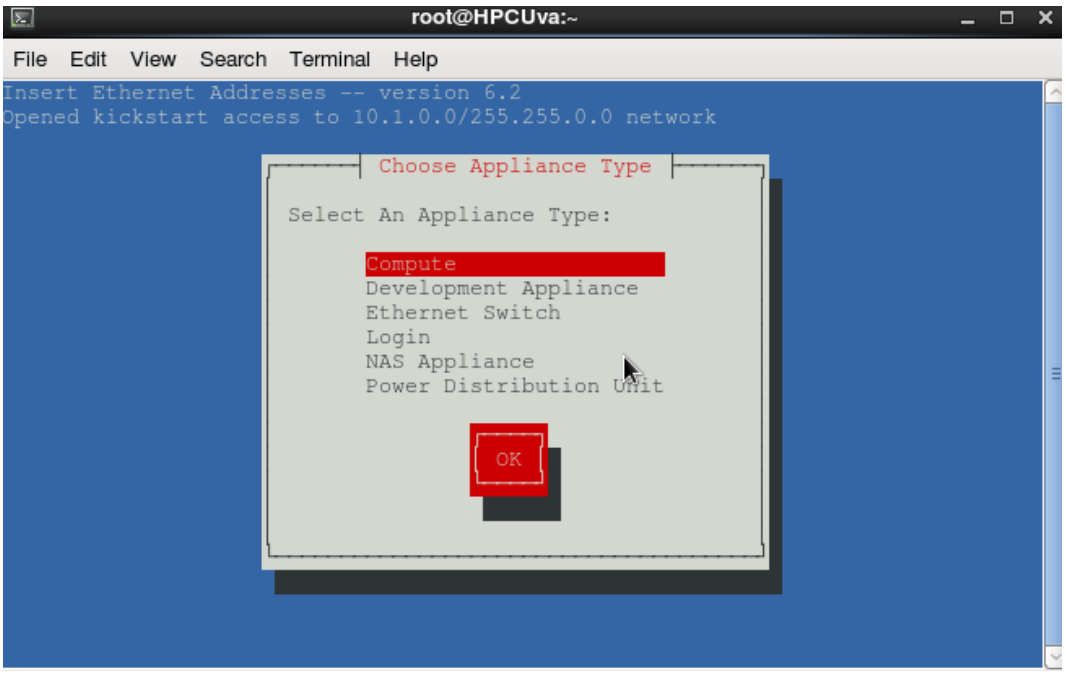

Figura B.9: Menú principal para la instalación de nodos.

Una vez hecho esto, debemos encender los equipos de la red que vamos a añadir al cluster de uno en uno, estos serán detectados por la herramienta, la cual se encargará de realizar la instalación en cada uno de los nodos. Es importante esperar a que cada nodo detectado se marque con un asterisco antes de encender el siguiente(como se muestra en la figura B.10, de esta manera nos aseguramos que la numeración de cada uno de los nodos sea la que deseamos.

Cuando todos los nodos hayan sido marcados solo falta esperar a que termine la instalación en cada uno de ellos, transcurrido ese tiempo podemos listar los nodos que hemos añadido al cluster(como

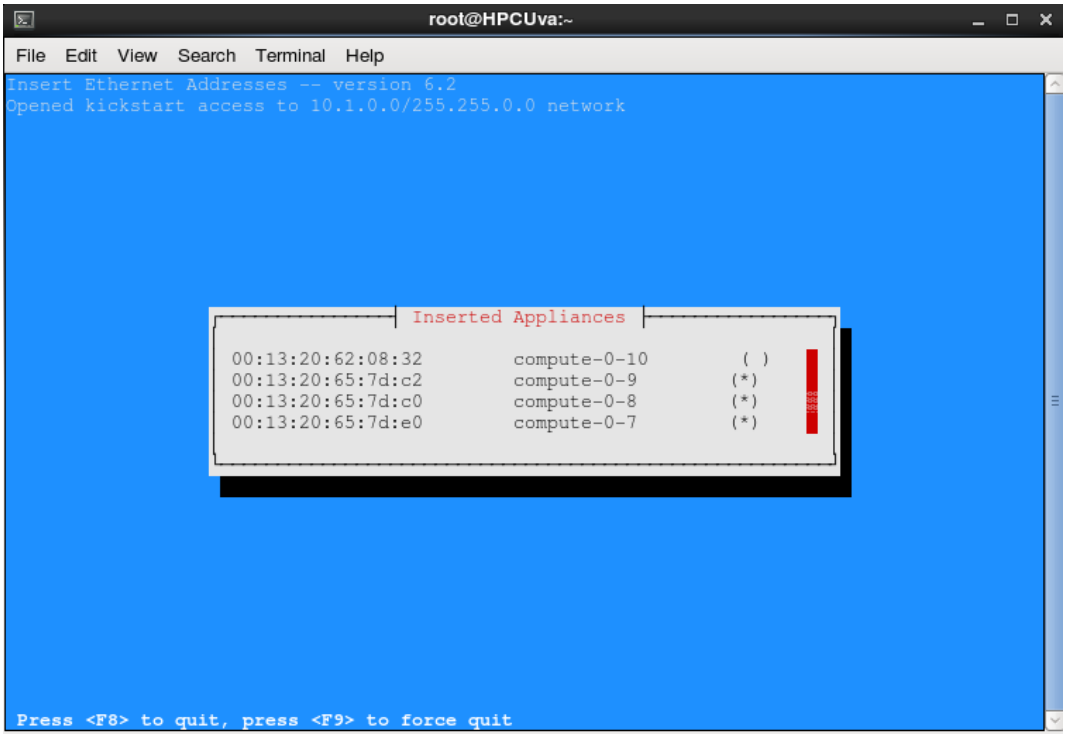

Figura B.10: Detección de nodos por parte de la herramienta insert-ethers.

se muestra en la figura B.11) escribiendo en una terminal:

```
$rocks list host
```

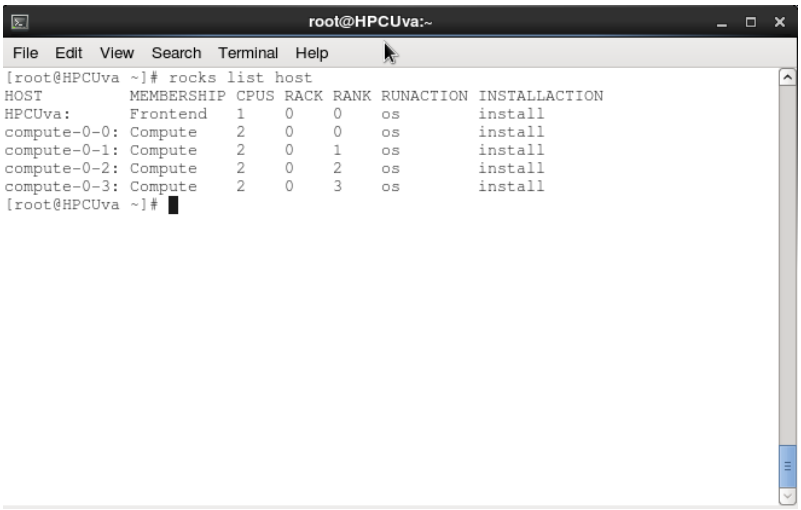

Figura B.11: Listado de los nodos que forman parte del cluster.

Apéndice C

Código

C.1. Script: getNodeMacs

```
#!/bin/bash
nodes=$(rocks list host | wc -l)          #Nmero de nodos del cluster
>/etc/ethers
>/etc/clues2/conf.d/wol.hosts             #Borramos el contenido del fichero donde almacenaremos los nodos y sus
    direcciones MAC
for count in $(seq 0 $((nodes-3)) )
do
        mac=$(rocks run host compute-0-$count ifconfig | grep HWaddr | cut -f 3 -d "r")
        echo $mac compute-0-$count.local >> /etc/ethers
        echo $mac compute-0-$count.local >> /etc/clues2/conf.d/wol.hosts
done
```

C.2. Script: wakeNodes

```
#!/bin/bash
nodes=$(rocks list host | wc -l)
for count in $(seq 0 $((nodes-3)) )
do
        mac=$(cat /etc/ethers | tail -n +$((count+1)) | head -n 1)
        ether-wake -i eth0 $mac
done
```

C.3. Script: shutdownNodes

```
#!/bin/bash
num_nodos=$(rocks list host| wc -l)
for count in $(seq 0 $((num_nodos-3)) )
do
        ssh compute-0-$count 'shutdown -h now'
done
```

C.4. Script: notifySAI

```
#!/bin/bash
#Recuperamos el nombre del cluster
hostname=$(hostname | cut -f 1 -d ".")

#Generamos el cuerpo del mail.
readonly MAILTEXT="El cluster $hostname ha sufrido un problema con el suministro elctrico. Se adjunta un fichero
    de log con los trabajos afectados por este incidente. Puede consultar este archivo en su directorio de
    trabajo del sistema."

#Definimos el contador para controlar que lineas de la salida de
#"qstat" cogemos.
count=0
DIA=`date +"%d/%m/%Y"`
HORA=`date +"%H:%M"`
echo ----------$DIA $HORA---------- >> /tmp/jobsInterrupt.log

while read -r line
do
    if [ $count -ge 2 ]
    then
        index=$((count -2))
        #Si son lineas donde aparece la info de los trabajos,
        #recogemos los datos que nos interesan
        jobID=$(echo $line | cut -f 1 -d " ")
        user=$(echo $line | cut -f 4 -d " ")
        jobState=$(echo $line | cut -f 5 -d " ")
        jobDate=$(echo $line | cut -f 6 -d " ")
        jobHour=$(echo $line | cut -f 7 -d " ")

        #Lanzamos el comando "qstat -j" con el id del trabajo
        mail=$(qstat -j $jobID | grep mail_list | cut -f 2 -d ":")
        workDir=$(qstat -j $jobID | grep sge_o_home| cut -f 2 -d ":")
        mailList[$index]=$mail
        workPaths[$index]=$workDir
        echo [$user]$jobID", "$jobDate $jobHour", "$jobState", "$workDir >> /tmp/jobsInterrupt.log
    fi
    #Incrementamos el contador
    count=$((count + 1))
done < <(qstat)

sortedMails=($(echo "${mailList[@]}" | tr ' ' '\n' | sort -u | tr '\n' ' '))
sortedWorkPaths=($(echo "${workPaths[@]}" | tr ' ' '\n' | sort -u | tr '\n' ' '))
#Se realiza el envio de mails, para ello comprobamos que no se ha notificado ya a ese usuario
#mirando si su email esta en la lista de enviados.
count=0

while [ "x${sortedMails[count]}" != "x" ]
do
    #Generar y enviar mail a los diferentes usuarios.
```

```
    echo $MAILTEXT | mail -s "Sistema automtico de  notificacines  SAI" -a /tmp/jobsInterrupt.log
        "${sortedMails[count]}"
    count=$((count + 1))

done

count=0

while [  "x${sortedWorkPaths[count]}" != "x" ]
do
    #Copiar los archivos de log al directorio de trabajo de los usuarios
    cp /tmp/jobsInterrupt.log ${sortedWorkPaths[count]}
    count=$((count + 1))
done

#Finalmente se notifica al administrador del cluster a  travs  de su email de  administracin .

#Apagamos los nodos del cluster
shutdownNodes

#Apagamos el nodo maestro
shutdown -h now
```

Apéndice D

Contenido del CD

- /
  - **memoria.pdf**: Memoria del trabajo.
  - **Scripts/**
    - **notifySAI**: Código del script de notificación para el SAI.
    - **getNodeMacs**: Script de recolección de direcciones MAC de los nodos.
    - **wakeNodes**: Script destinado a encender todos los nodos del cluster.
    - **shutdownNodes**: Script que automatiza el apagado de todos los nodos.
  - **Web/**
    - **Login/**
      - **login.php**: Código del formulario de login para el acceso al panel de configuración.
      - **loginCSS.css**: CSS del formulario de login
      - **autenticate.php**: Código PHP que gestiona la autenticación de las credenciales de usuario.
    - **PanelPrincipal/**
      - **panel.php**: Código del formulario de configuración de Clues.
      - **changesSaved.php**: Código de la función de escritura del archivo de configuración de Clues.
    - **PanelGrupos/**
      - **groupPanel.php**: Código del formulario de configuración de grupos.
      - **changesSavedGropu.php**: Código de la función de escritura del fichero de configuración de Clues para el formulario de grupos.
    - **res/**: Recursos de los diferentes formularios (Imágenes).
    - **test/**: Archivos de ejemplo sobre los que los formularios realizan las modificaciones.
  - **Clues/**: Archivos necesarios para la instalación de Clues junto con las mejoras implementadas.

Bibliografía

[1] "The top500 supercomputers." https://www.top500.org/lists/, 2005. [Online; acccesed 25-May-2017].

[2] "The green500 supercomputers." https://www.top500.org/green500/lists/, 2007. [Online; acccesed 25-May-2017].

[3] E. T. Area, "United states data center energy usage report." https://eta.lbl.gov/sites/all/files/publications/lbnl-1005775_v2.pdf, 2016.

[4] DiarioDeLéon, "Monica decide ahorrar." http://www.diariodeleon.es/noticias/innova/monica-decide-ahorrar_860664.html, 2017.

[5] T. G. Grid, "Glossary." https://www.thegreengrid.org/en/resources/glossary, 2017.

[6] "Rocks-solid: Extension to rocks cluster that make your cluster more solid!." https://code.google.com/archive/p/rocks-solid/, 2007-2010.

[7] G. E. project, "Grid engine project. powersaving." http://wiki.gridengine.info/wiki/index.php/PowerSaving, 2015.

[8] M. F. Dolz, J. C. Fernandez, R. Mayo, and E. S. Quintana-Ortí, "EnergySaving Cluster Roll: Power Saving System for Clusters," in *Lecture Notes in Computer Science*, vol. 5974, pp. 162–173, May 2010.

[9] S. Kiertscher, J. Zinke, and B. Schnor, "Cherub: power consumption aware cluster resource management," *Cluster computing*, vol. 16, no. 1, pp. 55–63, 2013.

[10] L. Sánchez, J. Ranilla, and A. Cocaña-Fernández, "Eecluster: An energy-efficient tool for managing hpc clusters," *Annals of Multicore and GPU Programming*, vol. 2, no. 1, pp. 15–24, 2015.

[11] A. Cocaña-Fernández, J. Ranilla, and L. Sánchez, "Energy-efficient allocation of computing node slots in hpc clusters through parameter learning and hybrid genetic fuzzy system modeling," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 1163–1174, 2015.

[12] GRyCAP-UPV, "Clues - cluster energy saving (for hpc and cloud computing)." http://www.grycap.upv.es/clues/es/arquitectura.php, 2016.

[13] GRyCAP-UPV, "Clues: an energy management system for hpc clusters and cloud infrastructures.." https://github.com/grycap/clues/, 2017. [Online; acccesed 17-May-2017].

[14] "Boostrap: Sleek, intuitive, and powerful front-end framework for faster and easier web development." http://getbootstrap.com/2.3.2/, 2017.

[15] GRyCAP-UPV, "Clues python utils - utils and general classes that spin off from clues." https://github.com/grycap/cpyutils, 2017. [Online; acccesed 19-Jun-2017].

[16] D. d. Campo Criado, "Estudio comparativo de alternativas de alimentación ininterrumpida para un cluster de ordenadores," 2008.

[17] "Multiscale materials modeling group de la universidad de valladolid." https://www.ele.uva.es/~mm. [Online; acccesed 25-Jun-2017].

[18] Wikipedia, "Clúster (informática)." https://es.wikipedia.org/wiki/Cl%C3%BAster_(inform%C3%A1tica)#Clasificaci.C3.B3n_de_los_cl.C3.BAsters, 2017.

[19] S. Nesmachnow, "Cluster fing: Arquitectura y aplicaciones." https://www.fing.edu.uy/cluster/grupo/cluster_arquitectura_y_aplicaciones.pdf, 2008.

[20] Wikipedia, "High-throughput computing — Wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=High-throughput%20computing&oldid=770995460, 2017. [Online; accessed 20-Feb-2017].

[21] D. T. Center, "Hpc cluster storage." http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/hpc-cluster-storage, 2012.

[22] G. Engine, "Open grid scheduler." http://gridscheduler.sourceforge.net/, 2017. [Online; accessed 22-June-2017].

[23] Univa, "Univa products." http://www.univa.com/products, 2017. [Online; accessed 22-June-2017].

[24] A. Computing, "Torque resource manager." ://www.adaptivecomputing.com/products/open-source/torque/, 2017.

[25] Wikipedia, "OpenNebula — Wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=OpenNebula&oldid=777660045, 2017. [Online; accessed 22-June-2017].

[26] S. H. Lavington, "Manchester computer architectures, 1948-75," *IEEE Annals of the History of Computing*, vol. 15, no. 3, pp. 44–54, 1993.

[27] Wikipedia, "Supercomputer — Wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Supercomputer&oldid=778325988, 2017. [Online; accessed 5-Apr-2017].

[28] Z. Pang, M. Xie, J. Zhang, Y. Zheng, G. Wang, D. Dong, and G. Suo, "The th express high performance interconnect networks," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 357–366, 2014.

[29] U. of California, "Base users guide- rocks cluster." http://central6.rocksclusters.org/roll-documentation/base/6.1.1/, 2014.