Diseño y despliegue de un clúster de bajo presupuesto para el desarrollo de las prácticas de Programación de Sistemas Distribuidos (PSD)

Daniel Quiñones Sánchez
Miguel Romero Martínez

Directores:

Alberto Núñez Covarrubias
Luis Llana Díaz

Curso Académico 2017/2018

Trabajo de fin de Grado en Ingeniería de Computadores
Facultad de Informática
Universidad Complutense de Madrid

Agradecimientos

A mi familia por su fé en mí y su apoyo incondicional.

A mi segunda familia de Computadores, por hacer de esta travesía algo tan divertido, en especial a Miguel, Rome y Álvaro.

Daniel

A Rosalía, por dejarme cometer errores.

A Miguel y Chuvi, por animarme a empezar.

A Laura, Peter y Jake, por aguantarme en exámenes.

Miguel

A Alberto, por creer en este proyecto y ayudarnos a llevarlo adelante.

A Pilar y Vero, por enseñarnos, abrirnos las puertas y aguantarnos.

A Computadores, por su empeño en ayudar a los mayores.

A Sánchez y Andrés por proveernos de cerveza y cafeína.

Daniel && Miguel

Índice general

Índice de figuras VII

Resumen XII

Abstract XIV

1. Introducción 1

1.1. Alcance de este trabajo . . . 1

1.2. Motivación . . . 1

1.3. Objetivos . . . 3

1.4. Planificación . . . 3

1.5. Estructura del documento . . . 5

2. Arquitectura General del Sistema 7

2.1. Esquema General . . . 7

2.2. Clúster . . . 9

2.2.1. Raspberry Pi Modelo B . . . 9

2.2.1.1. Límites térmicos . . . 9

2.2.1.2. Disipadores de calor . . . 10

2.2.2. Switch D-LINK DGS-1008D . . . 11

2.2.2.1. Especificaciones . . . 12

2.2.3. Tarjeta microSD Kingston . . . 12

2.2.4. Adaptador Nanocable USB 3.0 a Gb Ethernet . . . 12

2.2.4.1. Especificaciones . . . 13

2.2.5. Ventiladores . . . 13
2.2.6. Cargador USB . . . 14
2.3. Software Cliente-Servidor . . . 15

3. Topología y configuración del clúster **17**
3.1. Virtualización del sistema . . . 17
3.2. Topología del Clúster . . . 18
3.3. Configuración de la red . . . 19
3.3.1. Configuración de DHCP . . . 20
3.3.2. Configuración del montaje de directorios mediante NFS . . . 22
3.3.2.1. Instalación de NFS . . . 22
3.3.3. Configuración del servidor NFS como un *daemon* del sistema . . . 23
3.4. Configuraciones adicionales . . . 25
3.4.1. Nmap . . . 25
3.4.2. Hostname . . . 25
3.4.3. Forzar el arranque sin HDMI . . . 26
3.4.4. Super User Do . . . 26
3.5. Instalación de OMNeT++, INET y SIMCAN . . . 26
3.5.1. Instalación de OMNeT++ . . . 27
3.5.2. Instalación de INET . . . 27
3.5.3. Instalación de SIMCAN . . . 28
3.5.4. Instalación de un nodo back-end . . . 28
3.6. Listado de ficheros modificados . . . 28

4. Diseño Arquitectónico del Clúster **31**
4.1. Características del modelado . . . 31
4.2. Diseño del Modelo 1 . . . 32

4.2.1. Prototipo del Modelo 1 . . . 32
4.3. Diseño del Modelo 2 . . . 33
4.3.1. Prototipo del Modelo 2 . . . 34
4.4. Diseño del Modelo 2b . . . 35
4.5. Diseño del Modelo 3 . . . 36
4.5.1. Prototipo del Modelo 3 . . . 37

5. Estudio Térmico del Clúster **39**
5.1. Configuración de los Experimentos . . . 39
5.2. Experimento 1 . . . 40
5.3. Experimento 2 . . . 41
5.4. Experimento 3 . . . 43
5.5. Experimento 4 . . . 45
5.6. Conclusiones . . . 45

6. Modelo final **47**
6.1. Diseño del Modelo final . . . 47
6.1.1. Prototipo del Modelo final . . . 48
6.1.2. Modelo final . . . 48
6.2. Experimentos . . . 49
6.2.1. Experimento 1 . . . 49
6.2.2. Experimento 2 . . . 50
6.2.3. Experimento 3 . . . 50
6.3. Conclusiones . . . 51

7. Software para la ejecución remota de prácticas **53**
7.1. Cliente-Servidor . . . 53

7.2. Planificador 55

8. Experimentos de rendimiento **59**

8.1. Configuración de los Experimentos 59

8.1.1. Características del clúster 59

8.1.2. Características del ordenador portátil 59

8.1.3. Carga de trabajo 60

8.2. Experimento 1 60

8.3. Experimento 2 61

8.4. Experimento 3 61

8.5. Experimento 4 62

8.6. Conclusiones 63

9. Presupuesto **65**

9.1. Componentes y Materiales 65

9.2. Resumen de gastos 67

10. Conclusiones y trabajo futuro **69**

10.1. Conclusiones 69

10.2. Trabajo Futuro 70

10.3. Conclusions 70

10.4. Future work 71

A. Aportación individual de cada integrante del proyecto **73**

A.1. Daniel Quiñones 73

A.2. Miguel Romero 74

B. Software desarrollado en el proyecto **75**

Bibliografía 77

Índice de figuras

1.1. Diagrama de Gantt 4

2.1. Estructura de un módulo 8
2.2. Estructura global del clúster 8
2.3. Raspberry Pi Modelo 3 b 9
2.4. Captura Térmica 10
2.5. Disipadores 11
2.6. Switch D-LINK DGS-1008D 11
2.7. MicroSD Kingston 12
2.8. Adaptador Nanocable USB 3.0 a Gb Ethernet 13
2.9. Ventiladores USB 14
2.10. Cargador USB 14

3.1. Sistema de archivos 18
3.2. Arquitectura del clúster 19
3.3. Servidor DHCP 20
3.4. Esquema de Archivos en el front-end 29
3.5. Esquema de Archivos en el back-end 29

4.1. Distintas vistas del Modelo 1 en 3D 32
4.2. Prototipo del Modelo 1 33
4.3. Distintas vistas del Modelo 2 en 3D 34
4.4. Prototipo del Modelo 2 35
4.5. Distintas vistas del Modelo 2b en 3D 36

4.6. Distintas vistas del Modelo 3 en 3D 37
4.7. Prototipo del Modelo 3 38

5.1. Temperatura del Experimento 1 sobre el Modelo 1 40
5.2. Temperatura del Experimento 1 sobre el Modelo 2 40
5.3. Temperatura del Experimento 1 sobre el Modelo 2b 41
5.4. Temperatura del Experimento 1 sobre el Modelo 3 41
5.5. Temperatura del Experimento 2 sobre el Modelo 1 42
5.6. Temperatura del Experimento 2 sobre el Modelo 2 42
5.7. Temperatura del Experimento 2 sobre el Modelo 2b 42
5.8. Temperatura del Experimento 2 sobre el Modelo 3 42
5.9. Temperatura del Experimento 3 sobre el Modelo 1 43
5.10. Temperatura del Experimento 3 sobre el Modelo 2 43
5.11. Temperatura del Experimento 3 sobre el Modelo 2b 44
5.12. Temperatura del Experimento 3 sobre el Modelo 3 44
5.13. Temperatura del Experimento 4 sobre el Modelo 1 45

6.1. Elaboración Modelo final 47
6.2. Distintas vistas del Modelo final en 3D 48
6.3. Montaje Modelo final frontal 49
6.4. Montaje Modelo final 49
6.5. Resultados Prueba 1 50
6.6. Resultados Prueba 2 50
6.7. Resultados Prueba 3 51

7.1. Interfaz Gráfica 54
7.2. Envío completado 55
7.3. Esquema de directorios 55

7.4. Planificador de tareas 56
7.5. Esquema global del proceso 57

8.1. Ejecución de 1 simulación (Portátil Vs 1 Raspberry Pi) 60
8.2. Ejecución de 4 simulaciones (Portátil vs 1 Raspberry Pi) 61
8.3. Ejecución de cuatro simulaciones (Portátil Vs 4 Raspberry Pi) 62
8.4. 12 Simulaciones 62

Resumen

La ciencia se compone de errores, que a su vez
son los pasos hacia la verdad.
Julio Verne

El segmento de la supercomputación sigue avanzando año tras año y, según la célebre lista TOP500[1], son cada vez más los sistemas que mejoran en potencia y eficiencia. Por definición, un clúster de ordenadores es un conjunto de máquinas que trabajan juntas y que, en determinados aspectos, podría contemplarse como un único sistema. Sin embargo, al hablar de clústeres, es fácil imaginar gigantescas instalaciones con grandes servidores que, además de ocupar un enorme espacio, también tienen un consumo de energía sumamente elevado.

La rápida evolución en los sistemas informáticos y en las redes de comunicaciones han permitido que el concepto de clúster se pueda aplicar a proyectos más modestos. Por ello, hemos creado nuestro propio sistema con Raspberry Pi, placas de bajo coste, bien equipadas y de fácil acceso para montar un sistema de cómputo paralelo económico y eficiente, el cuál no sólo es válido en el ámbito de la educación, sino que también puede emplearse para problemas de cómputo actuales.

Este Trabajo de Fin de Grado presenta el diseño completo y montaje de un clúster de bajo coste, modular y escalable, destinado a las prácticas de la asignatura de Programación de Sistemas Distribuidos. Para ello, no sólo se ha prestado especial atención al hardware, sino que también se ha desarrollado un software específico para el entorno en el que será ejecutado. Además, se detalla la construcción de varios modelos en los que se han realizado estudios

[1] https://www.top500.org/

de temperatura y rendimiento, tanto en situaciones estables como forzando al sistema con pruebas de estrés.

Otro de los aspectos relevantes de este trabajo es el análisis de rendimiento realizado, el cual compara nuestro sistema con otros de uso cotidiano, tales como un ordenador portátil. El software implementado consiste en una interfaz gráfica que hace más atractiva la interacción del usuario con el clúster, así como un completo sistema de comunicación cliente-servidor.

Palabras clave: clúster,Raspberry Pi, simulación, HPC, MPI, SIMCAN y arquitectura.

Abstract

I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it.
Bill Gates

The segment of the supercomputing advances year after year and, according to the well-known TOP500 list, there are increasingly more systems that improve their power and efficiency. A computer cluster is a set of machines that work together and can be seen as a single system. However, it's easy to imagine huge facilities with large servers, occupying massive rooms.

At present, the rapid evolution of computer systems and communication networks has allowed the clusters can be applied to much more humble projects. Since we've created a system using Raspberry Pi, a cheap, well-equipped and accessible boards to assemble an economical and efficient parallel computing system, which can be applied in the field of education, but can also solve current computational problems.

This project presents a complete design and assembly of a low cost cluster to perform simulation tasks of the Distributed Systems Programming course. To that end, not only special attention has been paid to the hardware, but a specific software designed for the environment where it will be executed has also been developed. We detail the construction of several models, including a complete study of temperature and performance.

Another relevant aspect of this work is the performance analysis, which compares our system with a regular computer. The software implemented consists on a graphical interface that

makes the interaction with the cluster more attractive, as well as a complete client-server communication system.

Keywords: cluster,Raspberry Pi, simulation, HPC, MPI, SIMCAN and architecture.

Capítulo 1

Introducción

¡Tranquilos, es sólo un nombre! Como la Zona de la Muerte o la Zona sin Retorno. Esos nombres son normales en la Galaxia del Terror.

Profesor Hubert Farnsworth

Este trabajo consiste en el diseño y despliegue de un sistema de cómputo de alto rendimiento y bajo coste, destinado a ejecutar las prácticas de la asignatura de Programación de Sistemas Distribuidos (PSD)[11].

1.1. Alcance de este trabajo

Este trabajo permitirá a los alumnos de la asignatura de PSD ejecutar sus prácticas en el clúster diseñado. Además, servirá como ejemplo de un sistema distribuido real, con el que poder afianzar los conocimientos de la asignatura. Para conseguir este objetivo, disponemos tanto de un sistema hardware, compuesto por el clúster de placas Rasbperry Pi, como del software integrado en la GUI del simulador, capaz de enviar y recibir trabajos.

1.2. Motivación

Durante los últimos años, el número de proyectos enfocados a sistemas de alto rendimiento y bajo coste se ha incrementado de forma considerable. El desarrollo y fácil acceso a hardware de bajo coste, como Raspberry Pi[1], Arduino u ODROID, han facilitado, en gran parte, este crecimiento. Principalmente estos proyectos se han centrado en construir clústers usando la mayor cantidad de dispositivos posible para formar sistemas HPC (del inglés, Sistema de Alto Rendimiento)

En la década anterior, los clústers eran más caros e inaccesibles. Sin embargo, actualmente, es más fácil montar uno, pudiendo reflejar todos los aspectos del desarrollo de éste, centrándonos particularmente en la correcta disposición y distribución de los componentes para mejorar aspectos como la accesibilidad, la correcta refrigeración de los nodos y la mejora de rendimiento en ellos.

Algunos de los proyectos que utilizan hardware de bajo coste destinado a HPC son:

- VMW Research Group Raspberry Pi Cluster. Este sistema dispone de un clúster de 24 nodos realizado con Raspberry Pi 2. En total consta de 96 con 24GB de RAM, dispone de una interfaz de pantalla táctil y 2 adaptadores de Ethernet que controlan la fuente de alimentación. Utiliza DHCP, NFS y Ganglia. El clúster consume alrededor de 92W cuando está bajo carga y 70W cuando está inactivo. Han llegado a conseguir 15.4 GFlops de rendimiento, lo cual significa que en en el año 1993 hubiesen sido los séptimos según la lista TOP500[10].

- Universidad de Southampton. El profesor Simon Cox y su equipo construyeron la supercomputadora Iridis-Pi de 64 procesadores y 1 TB de memoria usando piezas de Lego. Este sistema utiliza MPI (Message Passing Interface) para comunicarse entre nodos a través de Ethernet. El profesor Cox utiliza el plug-in gratuito 'Herramientas de Python para Visual Studio' para desarrollar código para la Raspberry Pi[5]. Su motivación es ver cómo este sistema de bajo coste puede ser un punto de partida para futuros estudiantes que quieran aplicar computación de alto rendimiento y manejo de datos. Este sistema tiene un coste aproximado de 3100 euros.

- David Guill, en su web 'Like Magic Appears' ofrece una guía de construcción de un clúster de 40 nodos y dispone de material audiovisual como guía. David deseaba un clúster que pudiera usar para probar software distribuido, así que construyó su propio sistema, formando parte de su tesis para el MSCE[1] (Master of Science in Computer Engineering) y acabando como proyecto personal.

  Sus objetivos eran construir un modelo que imitase a una supercomputadora, pero que tuviese un hardware alojado en una caja no más grande que una torre de PC, piezas fáciles de reemplazar y gran eficiencia en el uso del espacio y la energía, económicamente eficiente y visualmente agradable. Posee 40 núcleos Broadcom BCM2835 @700MHz, 20GB de RAM, 5TB de disco duro (ampliables a 12TB) y 440GB de SSD. Este sistema tiene un coste aproximado de 2500 euros.

[1] https://engineering.tamu.edu/cse/academics/degrees/ce/ms.html

1.3. Objetivos

El objetivo principal de este trabajo es **la creación de un un clúster de cómputo de alto rendimiento y bajo coste destinado a ejecutar las prácticas de la asignatura de Programación de Sistemas Distribuidos**. Además, se plantean los siguientes objetivos secundarios:

- Diseño arquitectónico del clúster prestando atención a una correcta distribución de cada uno de los elementos hardware que lo forman.
- Estudio de rendimiento y temperatura del hardware bajo situaciones de cómputo masivo.
- Montaje y configuración del clúster.
- Generación de guías de instalación y configuración del sistema.
- Desarrollo de un software para realizar el envío y recepción de trabajos al clúster, así como un planificador de tareas para la distribución de trabajos entre los nodos.

1.4. Planificación

En la figura 1.1 se muestra un diagrama de Gantt con la planificación que se ha llevado a cabo para este trabajo. La planificación está dividida en seis fases, cada una de ellas orientadas a un tipo distinto de tarea:

- Fase 1: Configuración e instalación de servicios y servidores en el entorno real y virtual.
- Fase 2: Diseño, modelado y elaboración de prototipos.
- Fase 3: Diseño, modelado y elaboración del Modelo final en metacrilato.
- Fase 4: Configuración e instalación del Sistema Operativo, servidores y servicios en el clúster.
- Fase 5: Desarrollo del software Cliente-Servidor, así como su interfaz gráfica y el planificador de tareas.
- Fase 6: Test de rendimiento del Modelo final.

El plan de trabajo comienza en septiembre de 2017 y finaliza en Junio de 2018, con una duración total de diez meses en los cuales hemos trabajado paralelamente, tanto en el desarrollo del proyecto, como en la elaboración de este documento.

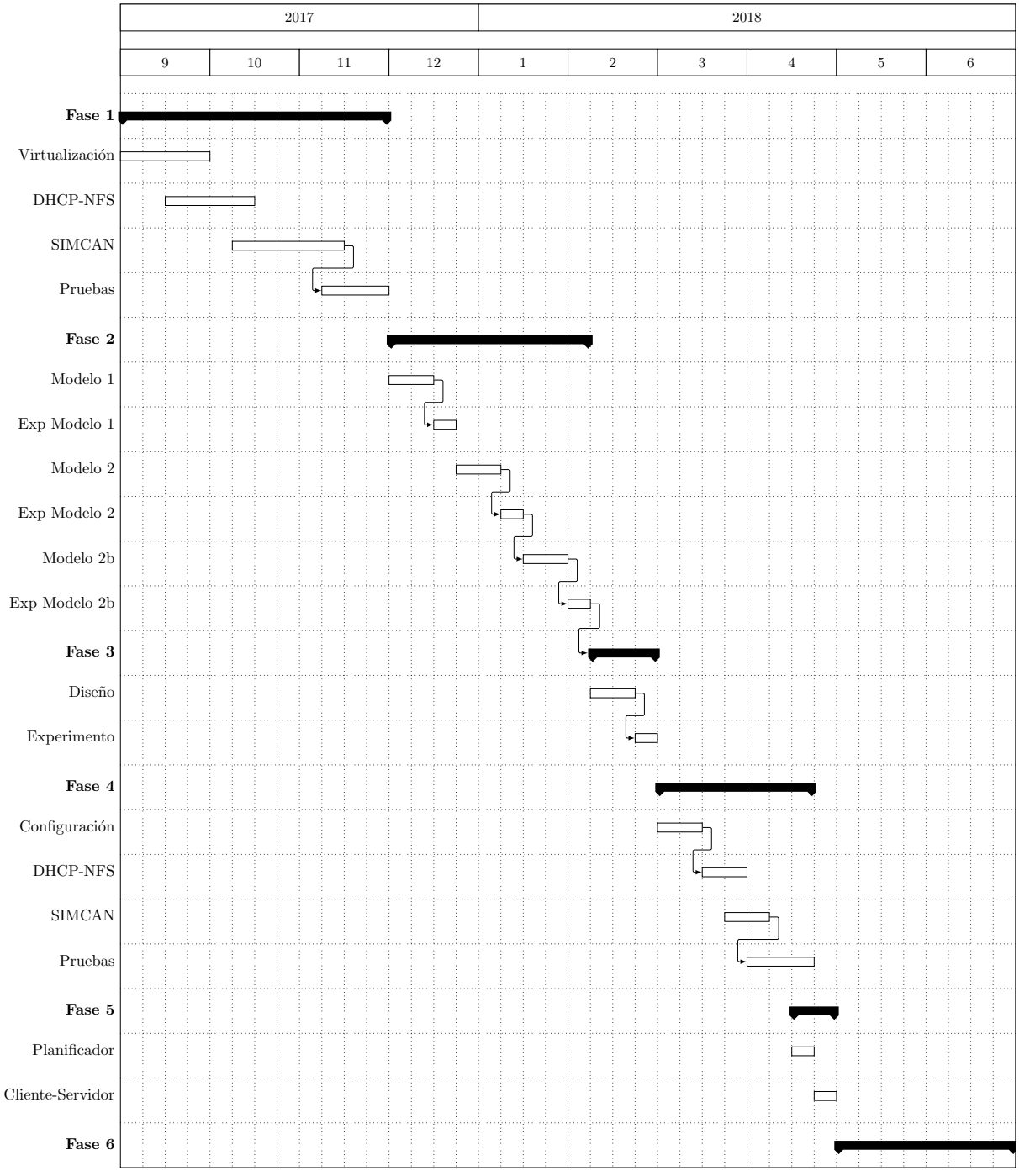

Figura 1.1: Diagrama de Gantt

1.5. Estructura del documento

El presente documento se ha dividido en diez capítulos siguiendo la siguiente estructura:

- En el capítulo 1 se introduce el proyecto. Además, se detallan los objetivos y requisitos de partida, así como la motivación y planificación.
- En el capítulo 2 se presentan los elementos relativos a la Arquitectura general del sistema.
- En el capítulo 3 se detalla la configuración de la red, la virtualización del sistema, el montaje del sistema de archivos NFS y la configuración del sistema.
- En el capítulo 4 se muestra la evolución de todos los diseños que han sido parte del proyecto, adjuntando los modelos en 3D y sus características.
- En el capítulo 5 se detallan los estudios térmicos realizados sobre los diferentes modelos diseñados en el capítulo anterior, adjuntando las gráficas y exponiendo las conclusiones extraídas de dichos experimentos.
- El capítulo 6 presenta el Modelo final, su diseño, modelado en 3D y las pruebas de temperatura que se han realizado sobre él.
- En el capítulo 7 se describe el software responsable de la comunicación entre el clúster y el mundo exterior.
- En el capítulo 8 se detallan los experimentos de rendimiento realizados sobre el Modelo final.
- En el capítulo 9 se desglosa el presupuesto del proyecto.
- En el capítulo 10 se presentan las conclusiones generales y el trabajo futuro.

Capítulo 2

Arquitectura General del Sistema

Bueno, pero aparte del alcantarillado, la sanidad,
la enseñanza, el vino, el orden público, la
irrigación, las carreteras y los baños públicos,
¿qué han hecho los romanos por nosotros?

The Life Of Brian

En este capítulo se presentan cada uno de los elementos que componen el clúster, detallando sus características más importantes y describiendo sus limitaciones más destacadas.

2.1. Esquema General

A lo largo de este documento utilizaremos la siguiente terminología:

- *Nodo*: Máquina física del clúster. Se emplean placas Raspberry Pi 3 model B.
- *Front-end*: Nodo principal, encargado de realizar la comunicación entre el exterior del clúster y el back-end.
- *Back-end*: Nodos encargados de realizar las operaciones de cómputo.

Uno de los objetivos del clúster es que éste sea modular y escalable. Para ello, hemos optado por utilizar una estructura de modelos, los cuales pueden conectarse a través de una red.

En la figura 2.1 se muestra el esquema de un módulo. Por un lado, el cliente, el cual dispone de una GUI para realizar el envío de los archivos para ejecutar una simulación. Por otro lado, el clúster, con un nodo front-end y varios nodos back-end, todos ellos pertenecientes a una misma red de área local. El cliente realizará la conexión directamente con el front-end. Únicamente existirá un front-end en el clúster.

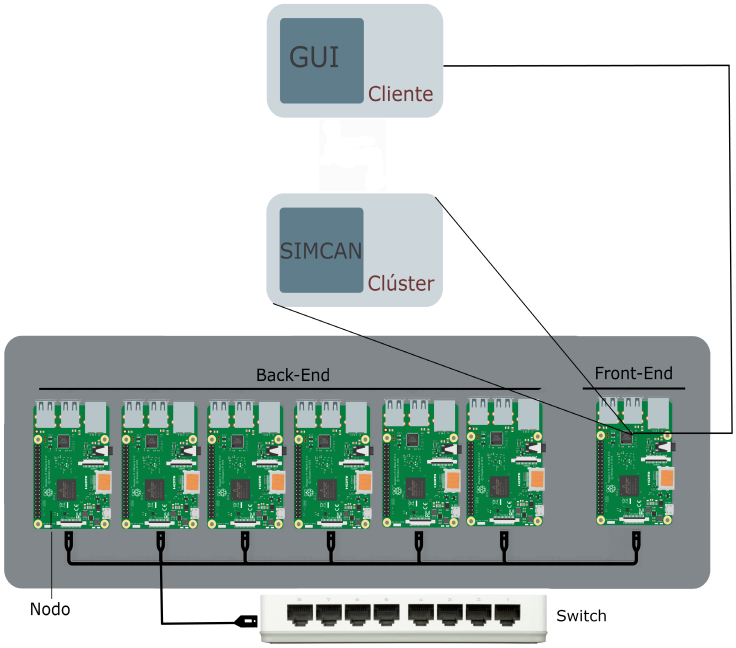

Figura 2.1: Estructura de un módulo

En la figura 2.2 se muestra la estructura global del clúster. Todos los nodos estarán conectados a través de los diferentes switches de cada uno de los módulos.

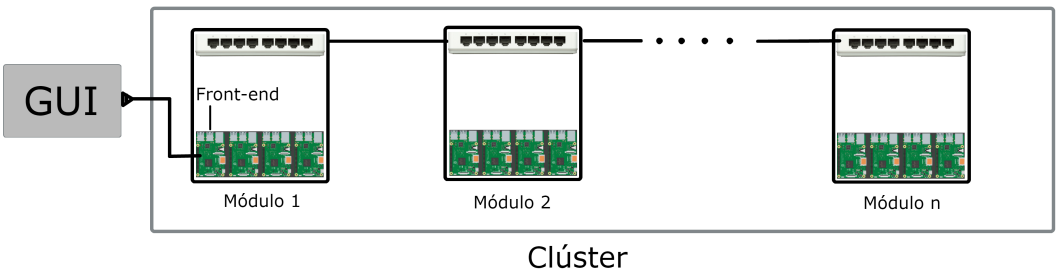

Figura 2.2: Estructura global del clúster

Una vez el cliente ha realizado la comunicación con el front-end, éste distribuye el trabajo entre los nodos que componen el back-end. Cada nodo de cómputo trabaja de forma independiente, consiguiendo así paralelizar la carga de trabajo y aumentando el rendimiento de éste.

2.2. Clúster

En esta sección se detallan los distintos elementos que componen el clúster, cada uno de ellos fueron obtenidos a través del Proyecto de Innovación Docente, en el curso académico 17/18, de la Universidad Complutense titulado *"Diseño y despliegue de un clúster de placas Raspberry Pi 3 para la ejecución de las prácticas de la asignatura PSD"*. Una vez finalizado el proyecto, el sistema implementado quedará a disposición de los alumnos, tanto para ejecutar las prácticas de PSD, como para realizar futuros TFGs. Con el presupuesto obtenido para este proyecto se ha financiado gran parte de material necesario.

2.2.1. Raspberry Pi Modelo B

Se ha elegido la placa Raspberry Pi 3 Modelo B como nodo de cómputo. Entre las características más destacadas podemos remarcar su limitado consumo, tamaño reducido y bajo coste. Además, ésta posee cuatro procesadores a 1.2 GHz así como 1GB RAM, lo cual hace que sea muy apropiada para el diseño de clústeres.

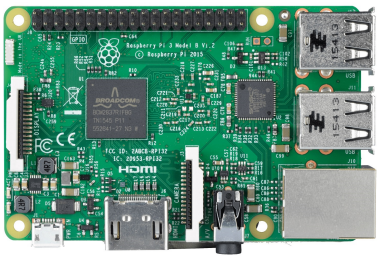

Figura 2.3: Raspberry Pi Modelo 3 B

Las especificaciones de Raspberry Pi 3 modelo B son:

- Chipset Broadcom BCM2387
- 1,2 GHz de cuatro núcleos ARM Cortex-A53
- 1 GB LPDDR2
- Ethernet socket Ethernet 10/100 BaseT
- 802.11 LAN inalámbrica
- HDMI rev 1.3 y 1.4
- USB 4x Conector USB 2.0
- Conector GPIO

2.2.1.1. Límites térmicos

Uno de los mayores problemas de utilizar Raspberry Pi para el proyecto es que ésta carece de elementos de disipación de calor integrados, lo que hace que sea especialmente sensible

a las altas temperaturas. En una situación de carga computacional baja, mantiene unos rangos de temperatura estables, sobre los 40°C. Sin embargo, en condiciones de cómputo masivo, el aumento de la temperatura es elevado, llegando a temperaturas cercanas a los 80°C, condiciones en las cuales, por seguridad, se produce un apagado súbito del sistema. En la figura 2.4 se puede apreciar un procesador con la temperatura crítica de apagado.

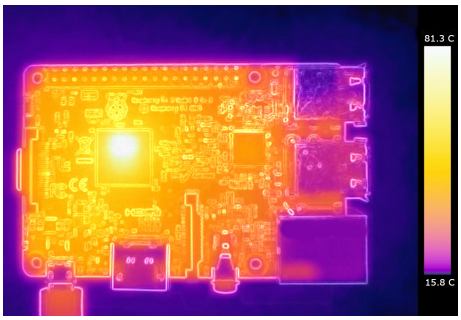

Figura 2.4: Límite térmico de la Raspberry Pi 3 Modelo B

En este caso, los elementos que se ven especialmente afectados son la memoria, el controlador de Ethernet y el procesador.

Generalmente, dentro del clúster existe una gran proximidad entre las placas Rasbperry Pi 3. Aquellas placas que se encuentran el la parte central de la estructura están expuestas a un efecto de *tubo de calor* proveniente de los nodos próximos a ellas. Por esto, es necesario disponer de un buen método de disipación de calor, con el fin de evitar el sobrecalentamiento del sistema. Tanto la disposición de las placas, como otras soluciones aportadas a este problema, se describen ampliamente en el capítulo 4. Adicionalmente, con el fin de aliviar este problema, incluimos algunos disipadores de calor disponibles para este modelo de Raspberry.

2.2.1.2. Disipadores de calor

Este componente está disponible para tres dispositivos, cada uno con una medida específica para el procesador, la memoria y el controlador Ethernet de una Raspberry Pi. (Ver figura 2.5)

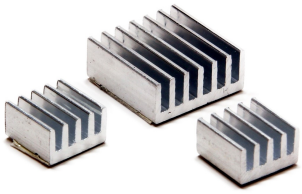

Figura 2.5: Disipadores

Su precio es reducido y, como veremos más adelante, aunque por sí sólos no proporcionan una gran mejora para refrigerar el sistema, son capaces de reducir entre 2°C y 3°C la temperatura de Raspberry Pi. Es importante remarcar que con una buena ventilación ofrecen una mejora significativa.

2.2.2. Switch D-LINK DGS-1008D

El switch es el componente encargado de realizar la interconexión de los nodos dentro de una misma red de área local LAN.

Figura 2.6: Switch D-LINK DGS-1008D

La elección del switch es importante, ya que limita el número de nodos que pueden conectarse a la red del clúster. En cada módulo hay un único switch y varias placas Raspberry Pi. Por ello, decidimos trabajar con un switch de ocho puertos (ver figura 2.6), lo que permite disponer de hasta siete nodos de cómputo, ya que una de las entradas del mismo sirve como enlace entre módulos en caso de que se requiera aumentar el número de módulos en el clúster, haciendo el sistema escalable.

2.2.2.1. Especificaciones

Este modelo permite alcanzar una velocidad de hasta 1000 Mbps. Sin embargo el modelo de Raspberry Pi utilizado está limitado a 100 Mbps. En la versión mas reciente de Raspberry (Modelo B+), este límite sigue vigente.

- 8 Puertos Ethernet 1000/100/10 Mbps
- Velocidad de transferencia: 2000 Mbps full duplex

2.2.3. Tarjeta microSD Kingston

Raspberry Pi no lleva incorporado almacenamiento en disco. Como alternativa, dispone de una ranura microSD en la cual se instala el sistema operativo[16].

La capacidad de las tarjetas es un factor a tener en cuenta, puesto que la instalación del sistema operativo[15] en tarjetas superiores a los 32 Gigabytes (ver figura 2.7) requiere de un software específico. Actualmente, existen incompatibilidades entre las versiones del sistema operativo y el software de instalación para algunas versiones de Raspbian Jessie[18].

Figura 2.7: MicroSD Kingston

Debido a que el sistema completo no supera los 11 Gigabytes, decidimos usar tarjetas de 16 Gigabytes para todos los nodos del clúster. De esta forma, reducimos el coste total, además de evitar problemas de compatibilidad con algunas versiones de Raspbian.

2.2.4. Adaptador Nanocable USB 3.0 a Gb Ethernet

Raspberry 3 Modelo B[6] posee un único puerto Ethernet. Por ello, el front-end necesita un puerto extra para conectarse a la red exterior. Este dispositivo (ver figura 2.8) permite ampliar el número de puertos Ethernet de una Raspberry Pi.

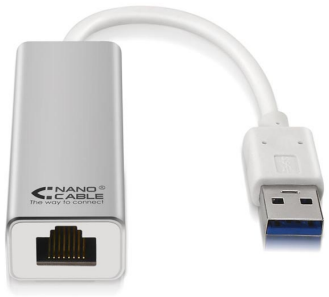

Figura 2.8: Adaptador Nanocable USB 3.0 a Gb Ethernet

2.2.4.1. Especificaciones

Debido a sus características, puede alcanzar una tasa de transferencia de hasta 1000 Mbps. A diferencia del puerto Ethernet instalado en la placa Raspberry Pi, esto permite disponer de una mayor velocidad de cara al front-end, permitiendo unas comunicaciones más fluidas con el exterior del clúster.

- Conexión USB 3.0
- Compatible con IEEE 802.3, 802.3u y 802.3ab
- Chipset RTL8153
- Velocidad de transferencia de datos: 10, 100 y 1000 Mbps
- Detección de crossover y corrección automática

2.2.5. Ventiladores

Además de los disipadores de calor (ver figura 2.4) disponemos de ventiladores (ver figura 2.9). Éstos disponen de una fuente de alimentación USB y tienen un tamaño de 120mm el cuál se adapta perfectamente al número de Raspberry Pi que componen cada módulo.

El uso de ventiladores, y principalmente, la disposición de éstos es esencial para la refrigeración del recipiente. En el capítulo 4 se presenta un estudio, utilizando varios diseños, en la que se refleja en la que se refleja la importancia de los ventiladores dentro del módulo.

Figura 2.9: Ventiladores USB

2.2.6. Cargador USB

La entrada principal de energía de Raspberry Pi es una entrada micro USB, aunque a través de su puerto GPIO puede conectarse a una entrada de $5V$[1]. Sin embargo, esta tensión es insuficiente para el propósito del proyecto, por lo que es necesario mantener una alimentación de 3A[2] para realizar correctamente las tareas de cómputo.

Figura 2.10: Cargador USB

Durante la fase preliminar del diseño, dispusimos de un hub USB conectado a red, con ocho puertos en los cuales conectábamos tanto las Raspberry Pi como los ventiladores. El reparto

[1] Voltios
[2] Amperios

de energía de este dispositivo resultó insuficiente y optamos por dos cargadores de cuatro entradas (ver figura 2.10), que repartían de una manera mas efectiva su conexión a la red eléctrica.

2.3. Software Cliente-Servidor

Como se indicaba en la sección 2.1 la comunicación con el clúster se realiza a través de una GUI disponible para cada cliente. Esta permite seleccionar diferentes ficheros, los cuales tienen un entorno simulado y enviarlos al front-end para que éste gestione su procesamiento. Tras recibir estos ficheros, el back-end realiza en paralelo la simulación de SIMCAN, dividiendo distintas fases de simulación entre los dispositivos disponibles. Una vez concluidas las labores de cómputo por el back-end, el resultado es enviado a cada cliente en base a la información incluida en la GUI. La comunicación entre cliente y front-end se realiza de forma asíncrona, de tal manera que no es necesario mantener la conexión activa entre ambos para completar el trabajo.

Capítulo 3

Topología y configuración del clúster

Zeroth Law. A robot may not injure humanity,
or, by inaction, allow humanity to come to harm.

Isaac Asimov

Al lo largo de este capítulo se detallan los pasos necesarios para la configurar el clúster. Los sistemas que se detallan en este capítulo incluyen, entre otros, la configuración de la red[13], la virtualización del sistema, el sistema de montaje de archivos NFS, la configuración de DHCP[9], OMNeT++, INET y SIMCAN.

3.1. Virtualización del sistema

Debido a la gran cantidad de componentes necesarios para la puesta en producción del clúster, la necesidad de realizar el montaje y desmontaje de forma manual de estos y la dificultad en el acceso y configuración de cada uno de los nodos que lo componen decidimos realizar una virtualización del sistema para aliviar las dificultades descritas. Así, al disponer de un entorno virtual, el cual, replica el clúster real, se reduce el tiempo necesario para la configuración de los distintos servicios y servidores del sistema operativo y sirve, a su vez, como banco de pruebas para el desarrollo de software.

Para ello, utilizamos *VMware workstation 12* como plataforma de software de virtualización y una imagen de Raspbian basada en Debian Stretch, la cual se encuentra disponible en la página oficial de Raspberry Pi[1].

[1] http://downloads.raspberrypi.org/raspbian/images/

Aunque el sistema del clúster parte de una versión diferente de Raspbian, el sistema de carpetas y, sobre todo, la instalación de SIMCAN[2], es similar al del entorno real.

En el repositorio de Github[3] existe una réplica del sistema de carpetas tanto para el servidor como para los nodos esclavo. Cada una de las carpetas y ficheros modificados en el sistema durante la configuración del sistema quedan reflejados en el repositorio, disponiendo así de un listado en forma de árbol de todas las configuraciones necesarias para el correcto funcionamiento de este.

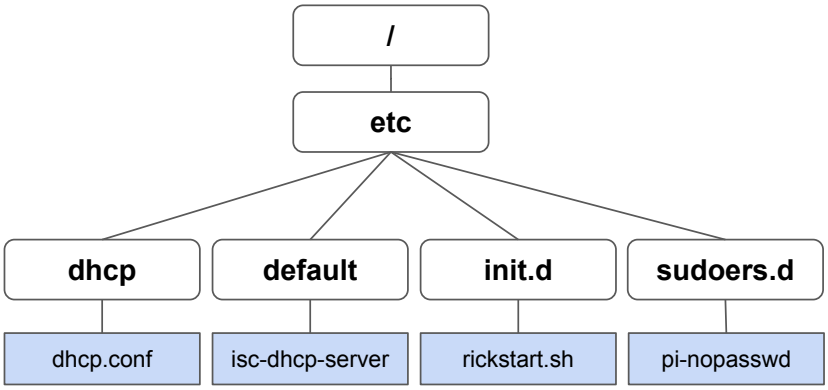

Figura 3.1: Sistema de archivos del repositorio

3.2. Topología del Clúster

El clúster está dividido en dos partes diferenciadas. Por un lado, el front-end, nodo del clúster que tiene instalado el software necesario para instalar y ejecutar SIMCAN, además de ofrecer los servicios de NFS, DHCP y SSH entre otros nodos, como muestra la figura 3.2. Además, este nodo dispone de un punto de comunicación con el exterior, por ello contiene una tarjeta de red adicional.

[2] http://www.simcansimulator.com/
[3] https://github.com/migurome/RickAndMortys

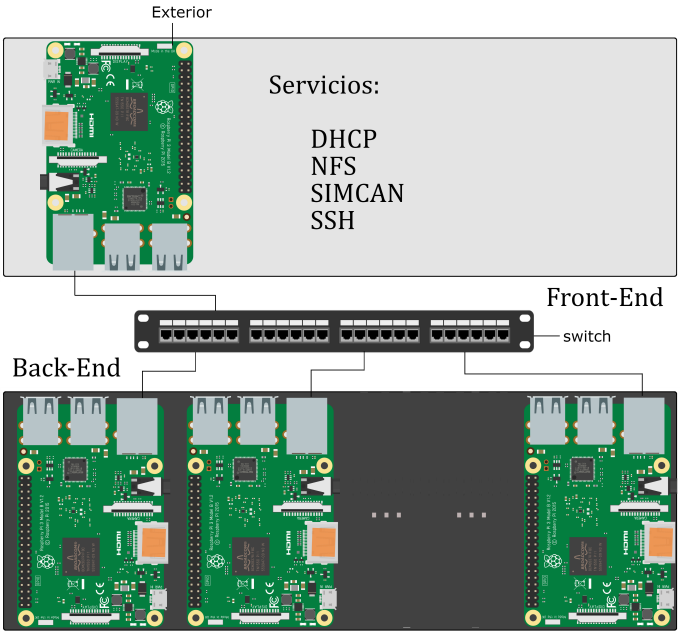

Figura 3.2: Arquitectura del clúster

Por otro lado, en el back-end, disponemos de varios nodos que contienen únicamente del sistema Raspbian Jessie, así como de bibliotecas necesarias para realizar las operaciones de cómputo.

3.3. Configuración de la red

La configuración de la red se realiza mediante DHCP, de forma que se asignan automáticamente las direcciones IP a los nodos del clúster. De esta forma evitaremos duplicidad de direcciones y, si es necesario, añadir nuevos nodos, los cuales obtendrán sus parámetros de acceso a la red de forma automática, haciendo el sistema escalable.

La red en la cuál se ha configurado el clúster es la 169.254.12.0/24, el front-end tiene asignada la dirección 169.254.12.1/24, y ejerce de servidor sobre el resto de nodos para encargarse del reparto de direcciones, tal y como se muestra en la figura 3.3.

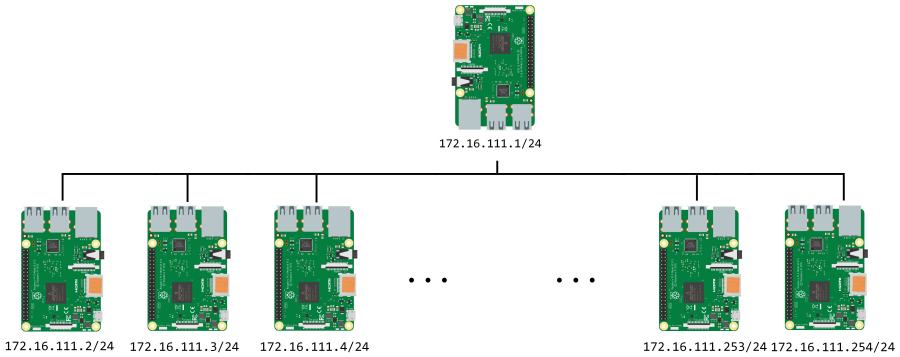

Figura 3.3: Servidor DHCP

El parámetro *default-lease-time* del fichero `/etc/dhcp/dhcpd.conf` determina el tiempo de concesión de cada dirección IP que el servidor asigna a cada nodo. Se ha establecido este paramétro en su máximo valor (7776000, expresado en segundos, el cual equivale a 90 días).

```
...
default-lease-time   7776000;
...
```

Los servicios ofrecidos por el nodo maestro de cara al *front-end* están disponibles a través de su segunda tarjeta de red.

3.3.1. Configuración de DHCP

En este apartado definimos la red mediante los pasos necesarios a seguir para la correcta instalación y configuración de DHCP.

Para instalar el dhcp-server se deben ejecutar los siguientes comandos:

```
1 sudo apt-get update
2 sudo apt-get install isc-dhcp-server
```

Además es necesario editar el fichero `/etc/default/isc-dchp-server` para añadir nuestra red

```
3 sudo nano /etc/default/isc-dchp-server
4 INTERFACES = "eth0"
```

A continuación es necesario editar fichero `/etc/network/interfaces` para configurar nuestra dirección IP, así como la máscara de subred y la red que utilizaremos

```
5 nano /etc/network/interfaces
6 source-directory /etc/network/interfaces.d

auto eth0
iface lo inet loopback
iface eth0 inet static

address 169.254.12.1
netmask 255.255.255.0
network 169.254.12.0

allow-hotplug wlan0
iface wlan0 inet manual
  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
  wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Para establecer la dirección MAC del servidor y su rango de red es necesario incluir la siguiente configuración al final del fichero`/etc/dhcp/dhcpd.conf` para almacenar la información de configuración de red requerida por los clientes.

```
subnet 169.254.12.0 netmask 255.255.255.0 {
  range 169.254.12.2 169.254.12.254;
}

host rick{
  hardware ethernet b8:27:eb:b3:36:02;
}
```

A continuación, ejecutamos los siguientes comandos para realizar la activación de la interfaz eth0 y reiniciar el sistema.

```
7 sudo ifconfig eth0 up
8 sudo reboot
```

Es necesario realizar el arranque del servicio DHCP de forma manual utilizando los siguientes comandos:

```
9 sudo /usr/sbin/dhcpd
```

3.3.2. Configuración del montaje de directorios mediante NFS

Como se ha destacado anteriormente, el front-end es el único nodo que dispone de una versión de SIMCAN instalada. Con esto, entre otros problemas, se evitan incompatibilidades entre versiones, y es mas sencillo realizar actualizaciones. Mediante NFS, el front-end comparte su carpeta `/home/` durante el arranque del sistema. De esta forma, el resto de nodos que componen el back-end realizan el montaje de este directorio compartido en red en su propio directorio `/home/`, creando así un único punto de acceso compartido en red donde se puede acceder a los ficheros ejecutables de SIMCAN sin la necesidad de tener una instalación en cada nodo. El front-end se encarga de compartir los ficheros *.ned* y poner a disposición del resto de nodos los ejecutables.

Hay que mantener un estricto control de los permisos de cada uno de los nodos que componen el back-end. Tanto a nivel interno como de cara al servidor, es necesario que todos los nodos de la red tengan un mismo usuario común, para conseguir una correcta sincronización.

3.3.2.1. Instalación de NFS

En este apartado se detallan cuales son los pasos necesarios para la correcta instalación y configuración de *NFS*.

En primer lugar es necesario instalar el paquete `nfs-kernel-server` en servidor, para ello ejecutamos los siguiente comandos:

```
1 sudo apt-get update
2 sudo apt-get install nfs-kernel-server
```

Para incluir los permisos que NFS importa sobre el directorio es necesario modificar el fichero de configuración `/etc/exports` en servidor. En primer lugar es necesario incluir la ruta

del directorio que se quiere compartir, a continuación la dirección o rango de direcciones sobre las que se va a compartir el directorio, finalmente los permisos que se establece a los cliente sobre el mismo.

```
Ruta de carpeta / Rango de direcciones destino / Permisos
/home/pi 169.254.12.0/24(rw,sync,no_subtree_check,no_root_squash)
```

A continuación es necesario ejecutar los cambios realizados en el servidor, para ello ejecutamos los siguientes comandos:

```
5 sudo exportfs -a
6 sudo /etc/init.d/rpcbind restart
7 sudo /etc/init.d/nfs-kernel-server restart
```

Finalmente, ejecutamos el siguiente comando en el back-end para realizar el montaje del directorio de forma manual:

```
8 sudo mount -t nfs 169.254.12.1:/home/pi /home/pi
```

3.3.3. Configuración del servidor NFS como un *daemon* del sistema

Para evitar tener que realizar el arranque del servidor de forma manual, es recomendable crear un *daemon* e incluirlo en el directorio `/etc/init.d`. De forma que éste se ejecute con el arranque del sistema automáticamente, tanto en el front-end, como en los nodos del back-end.

Para que el script[3] sea considerado un daemon del sistema es necesario aplicar el formato del siguiente cuadro. Incluimos los comandos necesarios para realizar la exportación del sistema de archivos compartidos por NFS al arranque del sistema.

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:    M.Romero
# Required-start: $syslog
# Required-stop:  $syslog
# Default-Start:  2 3 4 5
# Default-Stop:   0 1 6
# Short-Description:  Inicialización de servicios nfs
# Description:
### END INIT INFO
sudo exportfs -a
sudo /etc/init.d/rpcbind restart
sudo /etc/init.d/nfs-kernel-server restart
```

El siguiente script pertenece al back-end, éste realiza el montaje del `/home/pi` compartido por el front-end de forma automática durante el arranque.

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:    M.Romero
# Required-start: $syslog
# Required-stop:  $syslog
# Default-Start:  2 3 4 5
# Default-Stop:   0 1 6
# Short-Description:  Montaje de NFS
# Description:
### END INIT INFO
sudo mount -t nfs 169.254.12.1:/home/pi /home/pi
```

Una vez generados ambos scripts es necesario realizar los pasos que se muestran en el siguiente cuadro para que éstos pasen a ser un *daemon* del sistema. La creación de enlaces simbólicos y situación de los ficheros en `/etc/rcrunlevel.d` correspondiente se realiza de forma automática:

```
1 chmod +x nombre_del_script.sh
2 cp nombre_del_script.sh /etc/init.d/
3 cd /etc/init.d
4 update-rc.d nombre_del_script.sh defaults
```

3.4. Configuraciones adicionales

En esta sección se resumen algunas configuraciones adicionales para afianzar la robustez del sistema.

3.4.1. Nmap

La herramienta *Nmap*[4] nos permite monitorizar el estado de la red, de esta manera, desde el front-end se puede obtener un listado de direcciones IP activas en el sistema en tiempo real. Esta herramienta únicamente ha de ser instalada en el front-end. A continuación mostramos los comandos necesarios para realizar la instalación:

```
1 sudo apt-get update
2 sudo apt-get install nmap
```

Es recomendable crear un script que permita lanzar el escaneo de redes activas de forma automática. El contenido del siguiente cuadro contiene el comando de *Nmap* necesario para realizar dicho el escaneo.

```
#!/bin/bash
nmap 169.254.12.2-254/24 -n -sP | grep report | awk '{print $5}'
```

3.4.2. Hostname

A fin de diferenciar los distintos nodos de la red, a través de su *shell*, es aconsejable establecer el *hostname* de cada uno de ellos. Para ello es necesario modificar los ficheros `/etc/hostname` y `/etc/hosts` en cada nodo, tanto del back-end como del front-end.

Dentro del fichero `/etc/hosts` modificamos el parámetro *127.0.0.1 localhosts* y lo sustituimos por el nombre que le queramos asignar a cada nodo:

```
127.0.0.1    localhost
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::1      ip6-allrouters
127.0.0.1    client_x
```

[4]https://nmap.org/book/man.html

A continuación incluimos el nombre asignado en el fichero `/etc/hostname` para finalizar la configuración:

```
client_x
```

Es necesario reiniciar la máquina para que se apliquen correctamente los cambios.

3.4.3. Forzar el arranque sin HDMI

En ocasiones, *Raspbian Jessie* no realiza la carga del sistema operativo si el conector HDMI no está acoplado en la Raspberry. Para forzar el arranque del sistema es necesario modificar el archivo `/boot/config.txt` e incluir el siguiente contenido:

```
#Arranque sin HDMI
hdi_force_hotplug = 1
```

3.4.4. Super User Do

Por defecto, el usuario *pi* tiene habilitado el uso del comando *sudo*, el cuál permite realizar acciones de super-usuario sin que se requiera la validación del password de éste. Para reforzar la seguridad del sistema es conveniente eliminar estos permisos especiales de *pi*. Para ello es necesario eliminar el fichero *010_pi-nopasswd* del directorio `/etc/sudoers.d` de cada nodo del back-end, tal y como se muestra en el siguiente cuadro:

```
sudo rm /etc/sudoers.d/010_pi-nopasswd
```

3.5. Instalación de OMNeT++, INET y SIMCAN

Para instalar el software SIMCAN es necesario realizar la instalación previa framework OMNeT++[5] en su versión 4.6. Además del framework INET 3.6.3[6], que implementa modelos de código abierto OMNeT++ para redes cableadas, inalámbricas y móviles.

[5] https://www.omnetpp.org/
[6] https://inet.omnetpp.org/Introduction.html

Debido a la baja potencia de la Raspberry, ésta no es capaz de lanzar la aplicación de forma gráfica. Por ello todas las instalaciones han de realizarse a través del terminal. Esto afecta principalmente a la instalación de INET, ya que las principales guías de instalación disponibles en las webs oficiales parten del entorno gráfico de OMNeT++.

En los siguientes apartados se detallan los pasos necesarios para instalar el software necesario para utilizar SIMCAN.

3.5.1. Instalación de OMNeT++

En este apartado se resumen los pasos a seguir para realizar la instalación de OMNeT++.

Inicialmente se descargará el *tar.gz* de OMNeT++ 4.6. Es necesario copiar los archivos *.tar* de OMNeT++ e INET en `/home/usuario/` y descomprimirlos. Posteriormente, desde el directorio `/home/usuario` se deben ejecutar los siguientes comandos:

```
1 sudo apt-get update
2 sudo apt-get install build-essential gcc g++ bison flex perl  tcl-dev tk-
  dev libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0-0
   openmpi-bin libopenmpi-dev libpcap-dev
3 sudo apt-get install gnome-color-chooser
4 cd ompetpp-4.6
5 . setenv
6 ./configure
7 make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
  paralelizar el compilado del proyecto
```

3.5.2. Instalación de INET

En este apartado se resumen los pasos a seguir para realizar la instalación de INET. Es necesario disponer de una instalación de OMNeT++, descrita en el apartado anterior, para la correcta instalación de INET.

Crear un directorio nuevo en `/omnetpp-4.6/` llamado `/proyects`, copiar en el directorio INET descomprimido. A continuación se deben ejecutar los siguientes comandos:

```
8 sudo apt-get install libavcodec-dev libavformat-dev
9 make makefiles
10  make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
  paralelizar el compilado del proyecto
```

3.5.3. Instalación de SIMCAN

Finalmente se describen los pasos necesarios para la instalación de SIMCAN.
Es necesario copiar el directorio de `/simcan` a `/projects` y ejecutar los siguientes comandos:

```
11  export omnetpp_root=$HOME/pi/omnetpp-4.6
12  export INET_HOME=$omnetpp_root/projects/inet
13  export SIMCAN_HOME=$omnetpp_root/projects/simcan
14  export LD_LIBRARY_PATH=$omnetpp_root/lib:$LD_LIBRARY_PATH
15  export PATH=omnetpp_root/bin:$PATH
16  make makefiles
17  make
# Opcionalmente podemos utilizar el comando make -j 'numero de cores' para
  paralelizar el compilado del proyecto
```

3.5.4. Instalación de un nodo back-end

Es necesario que cada nodo del back-end disponga de los siguientes paquetes instalados para poder realizar las labores de cómputo.

```
1 sudo apt-get update
2 sudo apt-get install build-essential gcc g++ bison flex perl  tcl-dev tk-
  dev libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0-0
   openmpi-bin libopenmpi-dev libpcap-dev
```

3.6. Listado de ficheros modificados

En las figuras 3.4 y 3.5 se muestra un resumen de los ficheros incluidos (Verde), modificados (Azul) y eliminados (Rojo) para la configuración del sistema. En particular, los ficheros modificados en el *front-end* se muestran en la figura 3.4:

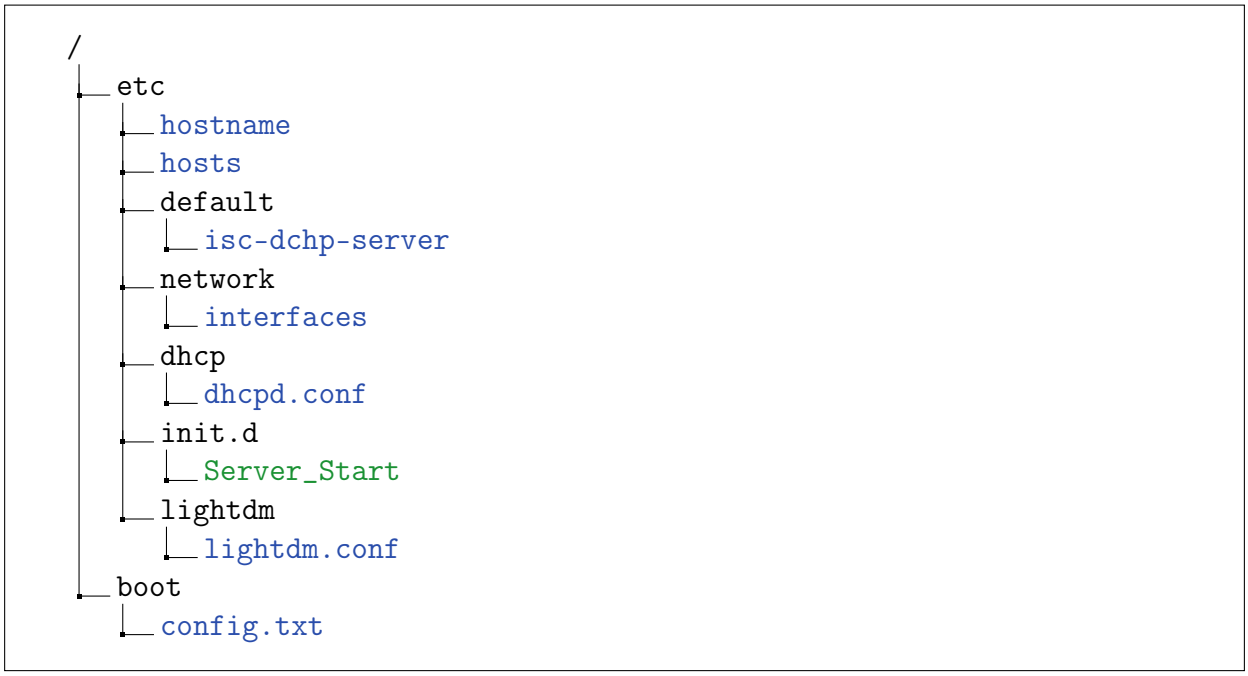

Figura 3.4: Esquema de Archivos en el front-end

De forma similar, los ficheros modificados en el *back-end* se muestran en la figura 3.5

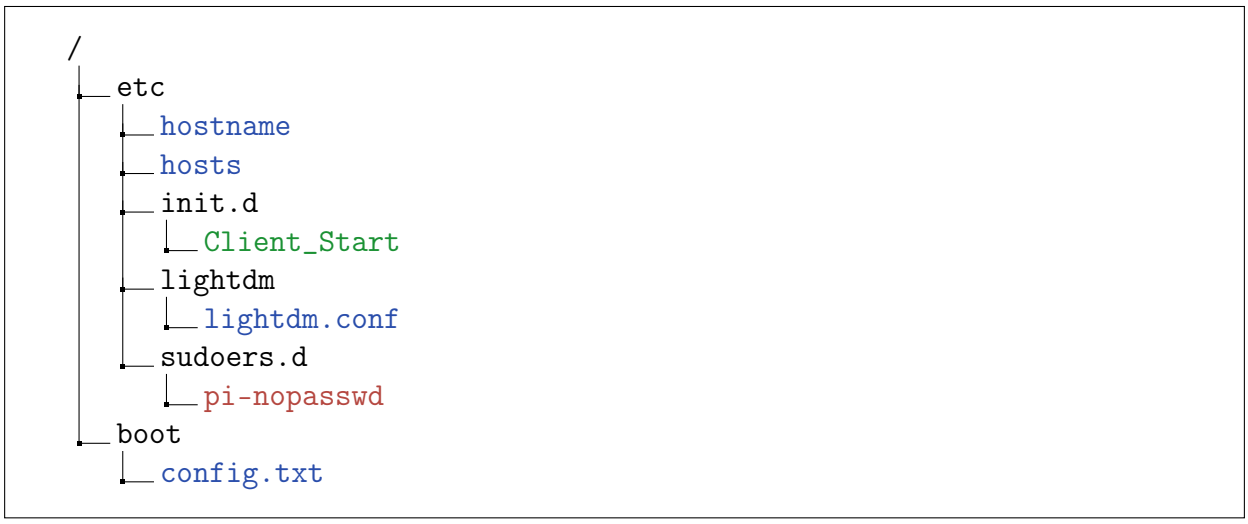

Figura 3.5: Esquema de Archivos en el back-end

Capítulo 4

Diseño Arquitectónico del Clúster

Las historias de imaginación tienden a molestar
a aquellos que no la tienen.

Terry Pratchett

En este capítulo se muestra el diseño de los distintos modelos que se han generado para el proyecto. Cada modelo presenta la colocación de los componentes dentro del clúster, así como la orientación de los ventiladores. Para cada diseño se presenta un diseño en 3D, creado con el programa OpenScad[1], así como una captura del modelo elaborado en cartón.

4.1. Características del modelado

Cada modelo consta de varios componentes, el switch de ocho puertos (representado en Azul), un clúster de Raspberry Pi formado por siete nodos (representado en Verde), la base de enchufes (representada en Rojo) y los ventiladores (representados en Gris).

En cuanto a la disposición de los ventiladores, para cada uno de los modelos, se han realizado diferentes configuraciones, todas ellas destinadas a obtener la corriente de aire más eficiente y la mejor tasa de temperatura cuando los nodos están ejecutando cómputo masivo.

[1] http://www.openscad.org/

4.2. Diseño del Modelo 1

En este modelo la distribución de aire se hace con los ventiladores opuestos entre sí, creando una corriente de aire alrededor de las placas de Raspberry Pi que se encuentra frente a ellos.

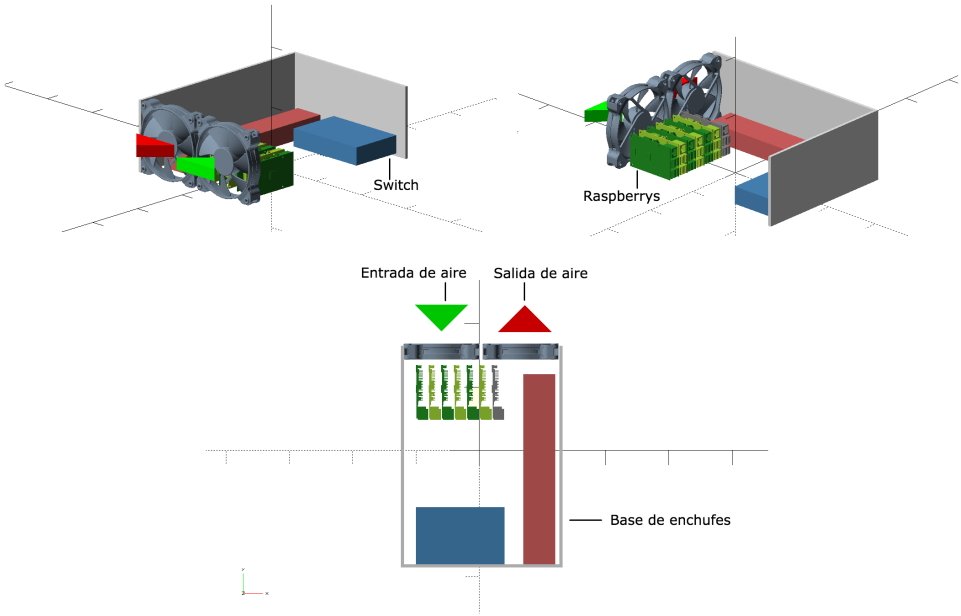

Figura 4.1: Distintas vistas del Modelo 1 en 3D

La figura 4.1 muestra la vista trasera, delantera y alzada del modelo. La prioridad es obtener una buena distribución de aire, así que no se ha tenido en cuenta la facilidad para acceder a los componentes.

Para poder acceder a las tarjetas SD es necesario mover toda la placa de Raspberry Pi, lo que hace que sea complicado el acceso. Una solución alternativa sería ofrecer la opción de desacoplar uno de los ventiladores para facilitar el acceso.

4.2.1. Prototipo del Modelo 1

Este modelo tiene un tamaño de 22x15x35 cm (ancho, alto, fondo). Podemos observar que los componentes disponen de un espacio muy limitado y que la poca flexibilidad de los cables dificulta el cierre del contenedor. En la captura no se han incluido los cables RJ45 que conectan las Raspberry Pi con el switch.

Figura 4.2: Prototipo del Modelo 1

Las figura 4.2 muestra el resultado del diseño descrito en el apartado anterior, donde se puede observar la distribución de los componentes dentro de la carcasa y el cierre del que dispone.

4.3. Diseño del Modelo 2

Para resolver los problemas de accesibilidad, este modelo dispone de una apertura lateral que expone las tarjetas SD de cada Raspberry Pi, evitando abrir el módulo para extraer cada tarjeta. Además, el contenedor está dividido en dos piezas, las cuales se acoplan entre sí para formar un cubo, facilitando la apertura y acceso al mismo.

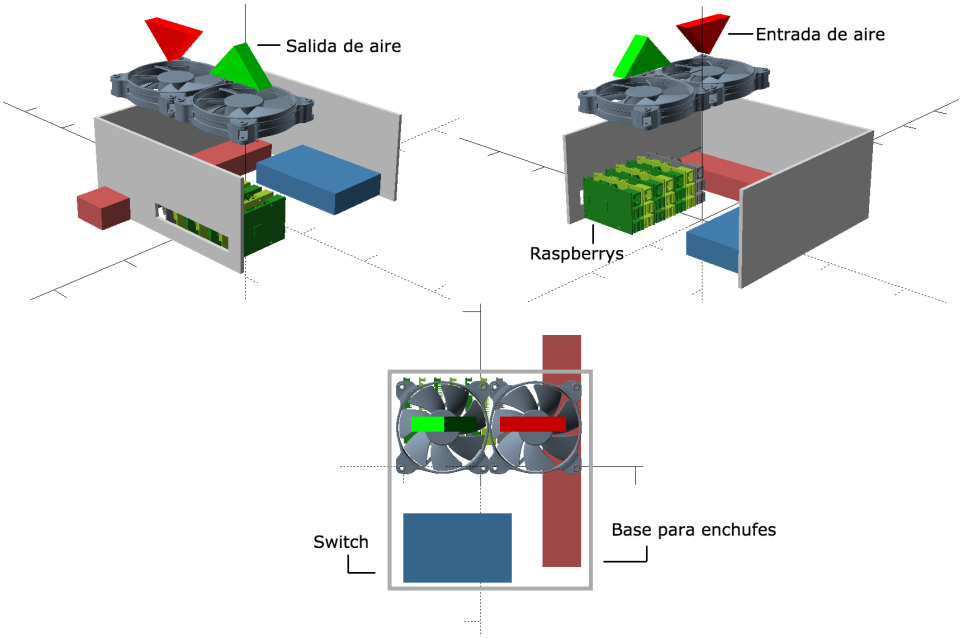

Figura 4.3: Distintas vistas del Modelo 2 en 3D

Como se muestra en la figura 4.3, la ventilación se realiza desde la parte superior, creando, como en el caso anterior, una corriente de aire debida al emplazamiento de cada ventilador. En la vista frontal, se puede observar, la apertura centrada sobre las placas Raspberry Pi.

Este modelo es más compacto que el anterior, parte de la base de enchufes queda en el exterior. Sin embargo, existe un mayor espacio entre los ventiladores y las Raspberry Pi. La disposición de los cables de alimentación es mejor que la del Modelo 1 ya que, junto con el acceso lateral, prácticamente no es necesario manipular ninguno de los componentes.

4.3.1. Prototipo del Modelo 2

Este modelo tiene un tamaño de 18x30x30 cm (ancho, alto, fondo). En la figura 4.4 se puede observar el sistema de acople en dos piezas que abren el contenedor. Como se ha comentado anteriormente, parte de la base de enchufes queda en el exterior, esto supone una mejora ya que permite el encendido y apagado del clúster sin que sea necesario abrir todo el módulo para ello.

Como resultado de todas estas mejoras tenemos un modelo mucho más manejable que el Modelo 1, de tamaño más reducido y con mejoras estructurales que facilitan la manipulación.

Figura 4.4: Prototipo del Modelo 2

4.4. Diseño del Modelo 2b

Este modelo presenta unas modificaciones estructurales internas sobre el módulo. Tras hacer un estudio previo sobre corrientes de aire, decidimos realizar algunas modificaciones sobre éste para conseguir mejorar la eficiencia del flujo de aire dentro del contenedor.

Para esto, se incluye un nuevo panel interior, distribuyendo el contenedor en dos cámaras, una que contiene las placas de Raspberry Pi y otra que contiene el Switch. La corriente de aire se concentra en la primera cámara. Con esto generamos un flujo que entra en el contenedor a través de la entrada lateral, en la que están expuestas las Raspberry Pi, pasando por cada uno de los procesadores y saliendo por la parte superior en la que ambos ventiladores expulsan el aire.

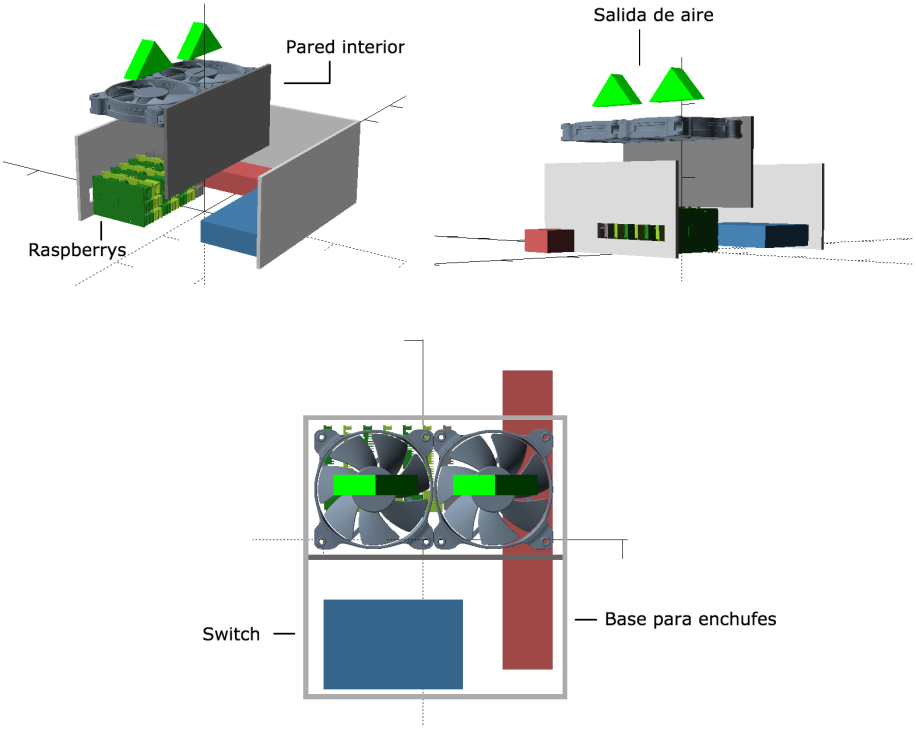

Figura 4.5: Distintas vistas del Modelo 2b en 3D

La figura 4.5 muestra la separación en cámaras y distribución de los ventiladores. Con estos cambios esperamos tener mejores resultados para refrigerar el módulo.

4.5. Diseño del Modelo 3

El Modelo 3 ha sido diseñado utilizando la arquitectura de túnel de viento. Ésta consiste en suministrar una corriente de aire de entrada sobre las partes que más tienden a calentarse, en este caso, las placas de Raspberry Pi, aplicando una corriente de succión con un ventilador de salida, que crea un flujo de aire horizontal.

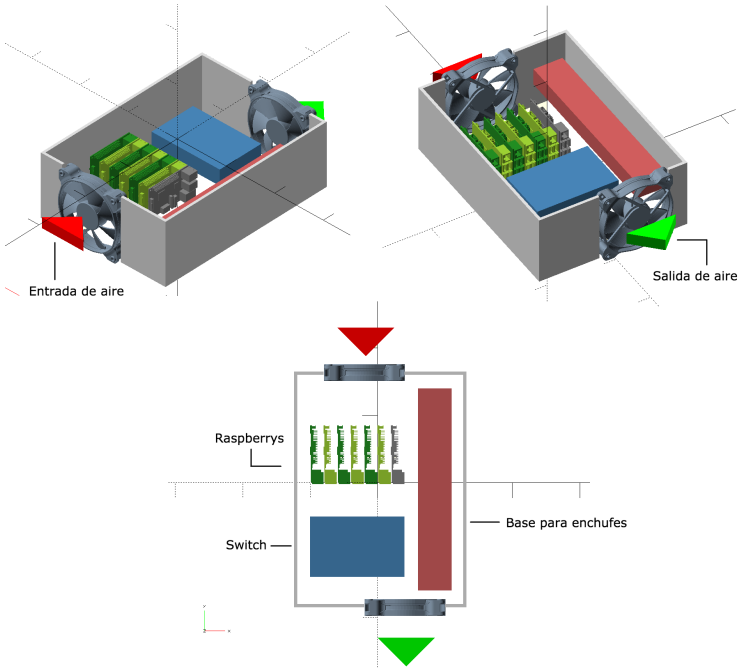

Figura 4.6: Distintas vistas del Modelo 3 en 3D

Debido a las altas temperaturas presentadas por el Modelo 1, decidimos aplicar una corriente de aire directa sobre las Raspberry Pi y sacarla del sistema para evitar bucles de aire calientes a través del segundo ventilador, que está situado al lado del switch, tal y como se muestra en la figura 4.6.

4.5.1. Prototipo del Modelo 3

Tras el desarrollo de los modelos anteriores, hemos creado un modelo híbrido entre el Modelo 1 y Modelo 2. Del primero aprovechamos su base y el ventilador del que ya disponíamos. Del segundo, hemos decidido que en vez de aplicar la corriente de succión en la superior, le aplicaremos un flujo de aire que recorra todo el contenedor a través del túnel de viento ya mencionado. Mientras los anteriores diseños se centraban en las placas de Raspberry Pi, este modelo proporciona una corriente tanto al Switch como a las placas.

Figura 4.7: Prototipo del Modelo 3

Capítulo 5

Estudio Térmico del Clúster

Tendrá todo el dinero del mundo, pero hay algo que nunca podrá comprar... un dinosaurio.

Homer J. Simpson

En este capítulo se muestra un estudio térmico realizado sobre los diferentes modelos diseñados en el capítulo 4.

5.1. Configuración de los Experimentos

Para llevar a cabo el estudio se han realizado un total de cuatro experimentos, todos ellos con las mismas características para cada modelo, variando únicamente el número de nodos que trabajan de forma simultánea en cada caso.
De esta manera, teniendo en cuenta los resultados obtenidos y las características de cada diseño creado en el capítulo 4, podemos determinar qué modelo es el más idóneo para el montaje del clúster.

Además de las pruebas generales, se han realizado una serie de pruebas específicas para cada modelo. El objetivo de éstas es poder determinar la mejor configuración en cuanto a la disposición de los ventiladores dentro del contenedor.

Para la ejecución de cada experimento se ha generado un script que obtiene información de la temperatura del procesador ofrecida directamente por el sistema operativo. El siguiente cuadro muestra el contenido de éste:

```
#!/bin/bash
while test 0 -eq 0
do
   sleep $1
   temp=$(/opt/vc/bin/vcgencmd measure_temp | cut -c 6-9)
   name=$(cat /home/hostname)
   echo "$name $temp"
done
exit 0
```

5.2. Experimento 1

En este experimento se utiliza únicamente hay una Raspberry Pi, la cual emplea sus 4 cores para realizar el cómputo correspondiente. El tiempo total de cada prueba es de tres horas. El periodo de muestreo es de diez segundos. En las gráficas siguientes se muestra el tiempo total expresado en segundos sobre el eje X. La temperatura ambiente oscila entre los 14°C y 18°C. Ninguna de las Raspberry Pi utilizadas dispone de un disipador instalado sobre el procesador.

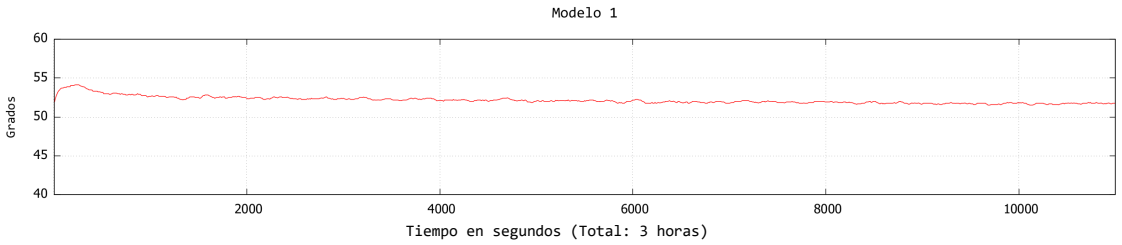

Figura 5.1: Temperatura del Experimento 1 sobre el Modelo 1

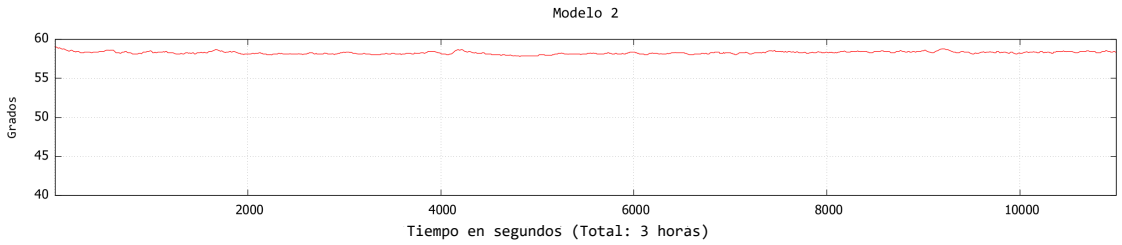

Figura 5.2: Temperatura del Experimento 1 sobre el Modelo 2

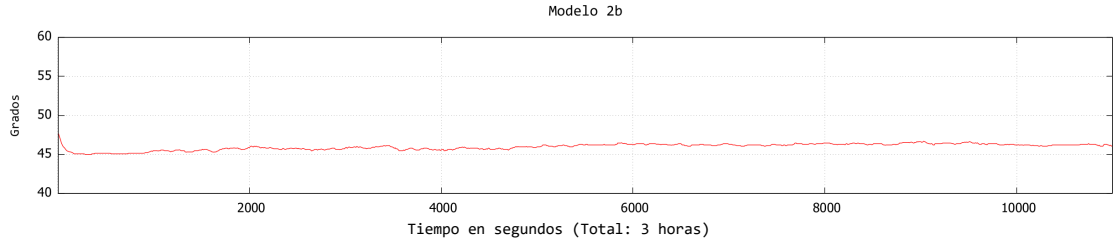

Figura 5.3: Temperatura del Experimento 1 sobre el Modelo 2b

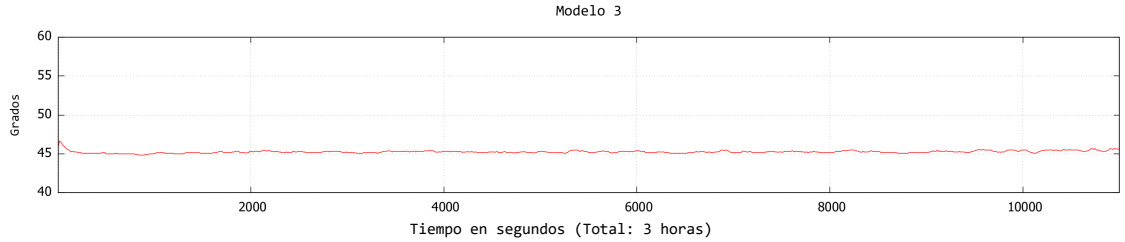

Figura 5.4: Temperatura del Experimento 1 sobre el Modelo 3

Analizando los resultado obtenidos se puede comprobar que, tanto el Modelo 2b, como el Modelo 3, son los que obtienen los mejores resultados. Así el sistema se mantiene estable durante las tres horas de ejecución del experimento, con muy poca variación entre sus temperaturas máxima y mínima. El Modelo 2 registra unos valores cercanos a los 60°C, siendo el peor modelo en cuestiones térmicas.

5.3. Experimento 2

En esta prueba hay tres placas de Raspberry Pi trabajando de forma simultánea, cada una de ellas mantiene sus cuatro cores trabajando a máximo rendimiento. El tiempo total de las pruebas es nuevamente de tres horas, igual que en la prueba anterior. El periodo de muestreo es de 10 segundos. La temperatura ambiente, en este caso, oscila entre los 15°C y 18°C. Para esta prueba una Raspberry Pi dispone de un disipador de calor, la cual se identifica con la letra D junto a su nombre. Las siguientes gráficas muestran los resultados de este experimento.

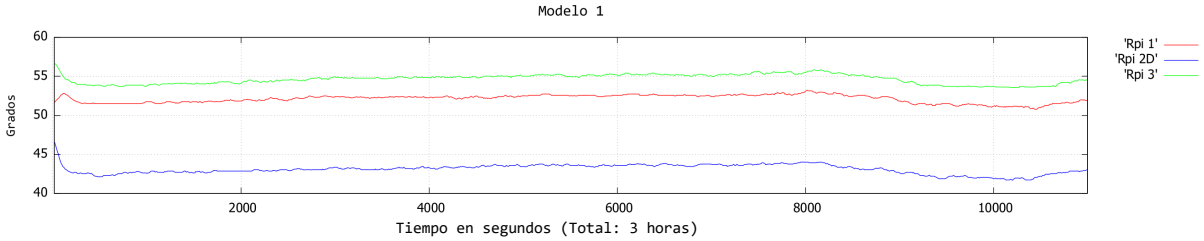

Figura 5.5: Temperatura del Experimento 2 sobre el Modelo 1

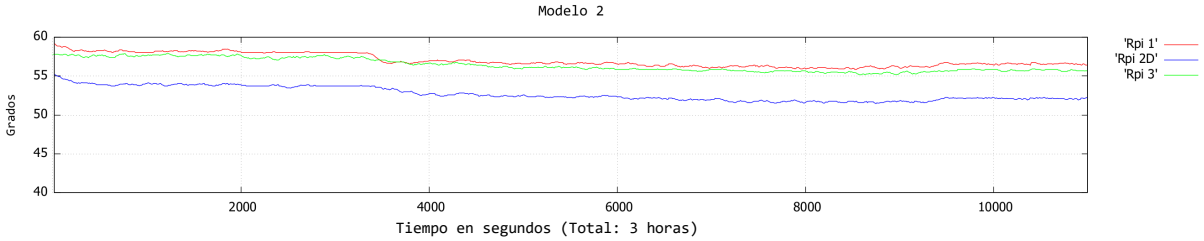

Figura 5.6: Temperatura del Experimento 2 sobre el Modelo 2

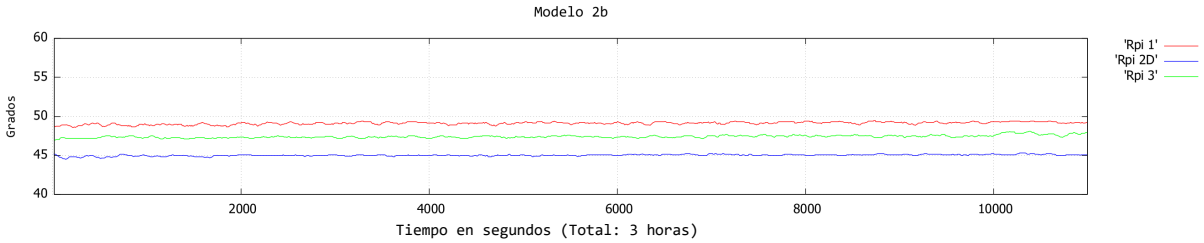

Figura 5.7: Temperatura del Experimento 2 sobre el Modelo 2b

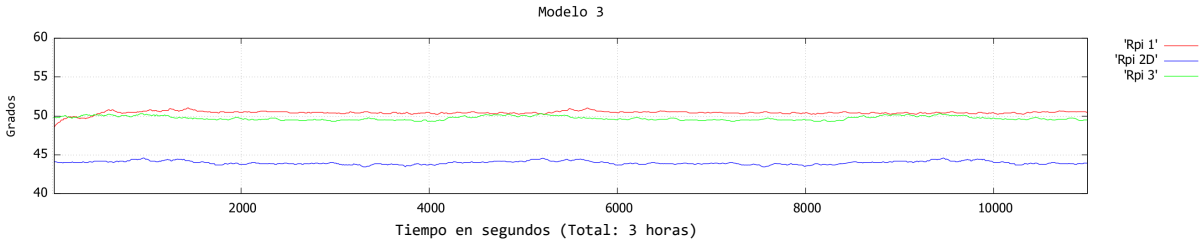

Figura 5.8: Temperatura del Experimento 2 sobre el Modelo 3

Hemos comprobado que una buena corriente de aire hace que los nodos que disponen de un disipador de calor instalado presenten unas temperaturas notablemente inferiores al resto.

Sin embargo, como pasa en el Modelo 2 (ver figura 5.6), si la corriente no es eficiente, dicha mejora se ve severamente afectada.

De forma similar a como ocurre en el Experimento 1, los modelos 2b y 3 ofrecen unos resultados mejores. Además, se mantienen más estables durante toda la prueba. El Modelo 1, (ver figura 5.5) es en el que más variación de temperatura se observa entre las Raspberry Pi, con y sin disipadores. Nuevamente, el Modelo 2 tiene unas altas temperaturas, cercanas a los 70°C.

Observamos además que al incluir más nodos de cómputo, no afecta a las temperaturas medias de cada uno, ofreciendo resultados similares a los obtenidos en el Experimento 1.

5.4. Experimento 3

Finalmente en esta última prueba hay 6 placas Raspberry Pi trabajando de forma simultánea. Cada una de ellas mantiene sus cuatro cores trabajando en paralelo. El tiempo total de las pruebas es nuevamente de tres horas y el muestreo se realiza cada diez segundos. La temperatura ambiente oscila entre los 16°C y 17°C. Para esta prueba existen dos Raspberry Pi que disponen de disipadores de calor, identificadas nuevamente con la letra D en la gráfica. Las siguientes gráficas muestran los resultados del Experimento 3.

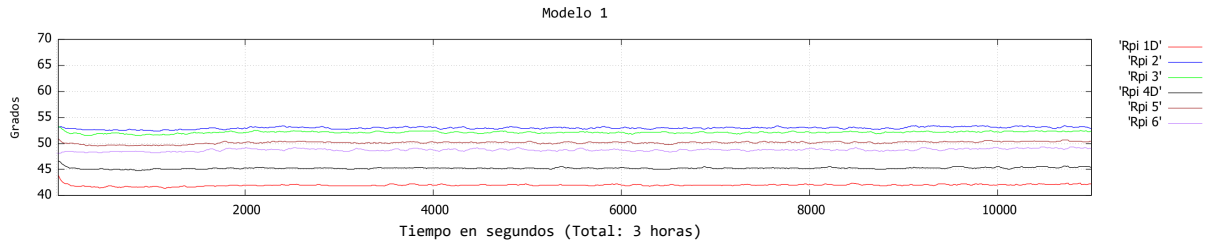

Figura 5.9: Temperatura del Experimento 3 sobre el Modelo 1

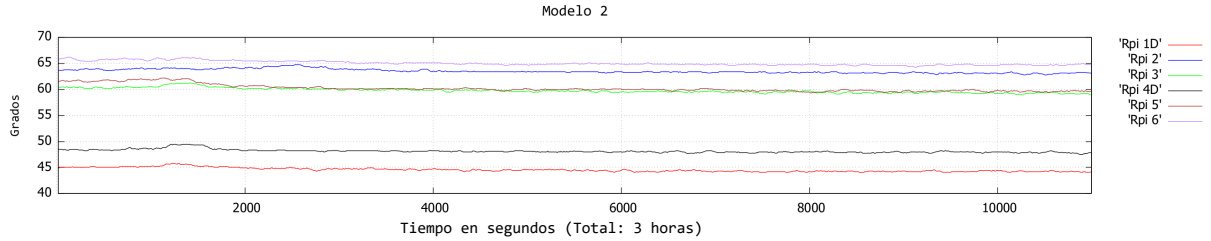

Figura 5.10: Temperatura del Experimento 3 sobre el Modelo 2

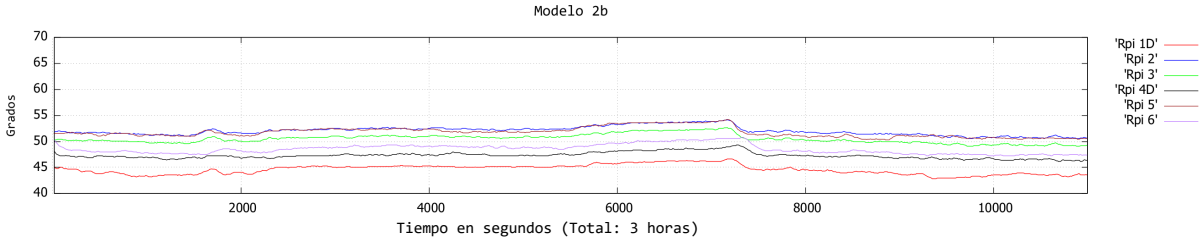

Figura 5.11: Temperatura del Experimento 3 sobre el Modelo 2b

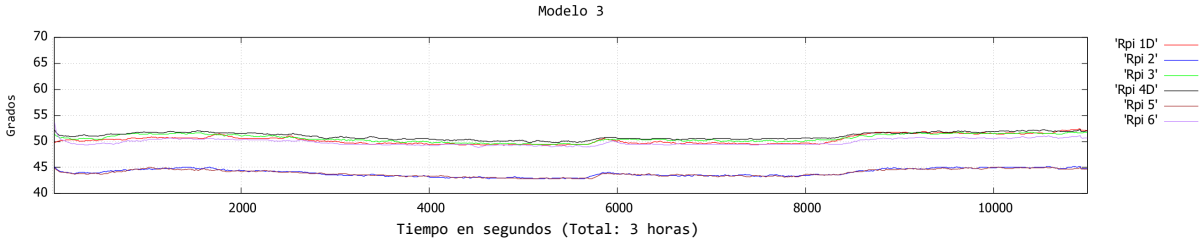

Figura 5.12: Temperatura del Experimento 3 sobre el Modelo 3

En esta prueba hemos detectado que definitivamente el Modelo 2 no cumple con las expectativas necesarias para el correcto funcionamiento del clúster debido a que alcanza temperaturas cercanas a la crítica. En el Módulo 2 las diferencias entre nodos con y sin disipador son muy notables y aquellos que no disponen de esta mejora alcanzan unas temperaturas muy próximas a los 80°C, temperatura crítica de parada, por lo que podemos descartar este modelo para el desarrollo final.

Los otros tres modelos ofrecen unos resultados semejantes a los anteriores experimentos. En particular, el Modelo 1 sigue mostrando una gran diferencia entre los nodos con disipador. El Modelo 2b, aunque ofrece algunas variaciones más notables durante la prueba, sigue manteniendo una relación parecida entre nodos, independientemente de que incluya o no disipadores.

El Modelo 3 presenta los mejores resultados, manteniendo temperaturas cercanas a los 50°C y una notable diferencia entre las placas de Raspberry Pi que disponen de disipador, las cuales se mantienen en 45°C.

5.5. Experimento 4

El objetivo de esta última prueba es comprobar el comportamiento del sistema durante un periodo de tiempo largo. Este experimento se realizó sobre el Modelo 1 durante 24 horas de forma ininterrumpida. El escenario es similar al del Experimento 3, con seis Raspberry Pi trabajando en paralelo. Sin embargo, este muestreo se realiza cada 60 segundos. No podemos certificar la temperatura ambiente durante toda la prueba, pero calculamos que osciló entre los 15°C y 18°C.

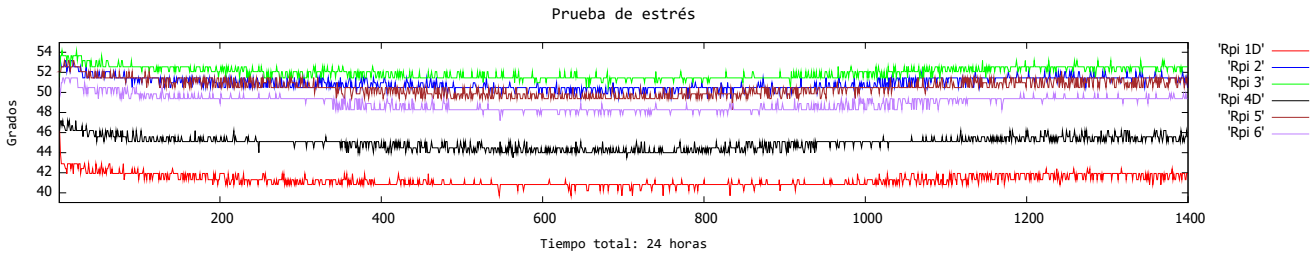

Figura 5.13: Temperatura del Experimento 4 sobre el Modelo 1

En este último experimento sacamos como conclusión que el sistema es estable durante una prueba de larga duración, no existen picos o caídas repentinas a lo largo de las 24 horas en las que se lleva a cabo. Las Raspberry Pi que tienen disipador muestran temperaturas bajas mientras que las que no lo tienen no superan en ningún momento los 55°C, en base a estos resultados, podemos afirmar que es un sistema fiable y regular.

Tras esta serie de experimentos, llegamos a la conclusión de que el sistema que proponemos puede responder sin problemas a las exigencias del entorno sobre el que será puesto en marcha de manera definitiva.

5.6. Conclusiones

Durante el desarrollo de los experimentos hemos podido apreciar algunos de los factores más influyentes sobre el sistema. En este apartado destacamos los que han sido más relevantes:

El flujo de las corrientes de aire afecta de manera muy significativa en el diseño. Merece la pena dedicar parte del trabajo a buscar una mayor eficiencia en la conducción de flujos de aire dentro del contenedor, ya que una buena distribución de la entrada y salida de aire permite reducir el número de ventiladores y establecer un punto de partida para la disposición del resto de elementos.

Al contar con un gran número de cables, también es necesario combinar el punto anterior con la accesibilidad general a los dispositivos. De esta manera, exponiendo las tarjetas SD, se evita tener que abrir el contenedor para acceder a las placas de Raspberry Pi.

La temperatura ambiente es otro de los factores influyentes en el comportamiento general. Aunque menos notorio, se pueden apreciar alteraciones en todos los dispositivos sujetos a este efecto.

En cuanto a la estabilidad, los resultados ofrecidos por la prueba de estrés (ver figura 5.13), muestran que el comportamiento ha sido similar al del resto de pruebas. Independientemente del tiempo empleado, podemos concluir que el sistema es estable y poco propenso a los cambios bruscos de temperatura en periodos largos de cómputo intensivo.

El uso de disipadores es muy conveniente. Son elementos de precio reducido y con una distribución de aire adecuada, aumentan tanto la vida útil del clúster como el comportamiento de los nodos ante los cambios bruscos de temperatura ambiente. En general, todos los nodos que disponían de esta mejora ofrecieron unas temperaturas inferiores a las del resto.

Capítulo 6

Modelo final

Me encanta que los planes salgan bien.

John 'Hannibal' Smith

En este capítulo se muestra el diseño del Modelo final del clúster, el cual incluye tanto el modelado en 3D como el resultado del mismo. Además, se presentan las pruebas de temperatura sobre este modelo.

6.1. Diseño del Modelo final

Basándonos en los resultados obtenidos en los capítulos 4 y 5, hemos realizado el Modelo final del clúster. Para su elaboración se han utilizado planchas de metacrilato de 4mm de ancho unidas con tornillería y ángulos metálicos. Cada pieza ha sido elaborada a mano como se muestra en la figura 6.1.

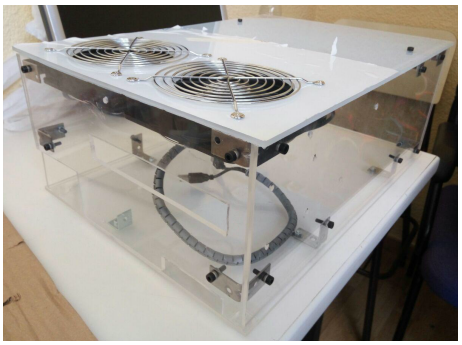

Figura 6.1: Elaboración Modelo final

El Modelo final incluye los aspectos más relevantes, en cuanto a disposición de los elementos dentro del módulo, observados a lo largo del capítulo 4.

6.1.1. Prototipo del Modelo final

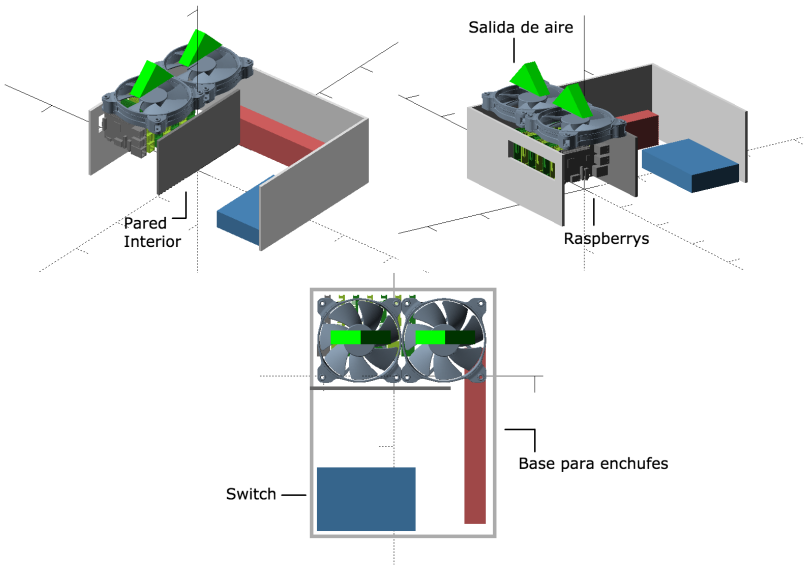

Figura 6.2: Distintas vistas del Modelo final en 3D

La ventilación se realiza desde la parte superior. Ambos ventiladores se colocan sobre las placas, creando una corriente de aire debida al emplazamiento de cada ventilador. En la vista frontal (ver figura 6.2), se puede observar, la apertura centrada sobre las placas, así como la pared interior incorporada en el Modelo 2b (ver figura 4.5). Esta apertura permite crear una corriente de aire que refrigera los procesadores de cada nodo. Las placas de Raspberry Pi han sido elevadas, dejando los cables de alimentación en la parte inferior del módulo, lo cual permite un mejor acceso a las placas y la posibilidad de eliminar el exceso de cableado.

Para mantener los cables en líneas ordenadas se han utilizado elementos de agrupación.

6.1.2. Modelo final

La figura 6.3 presenta una vista frontal del módulo. Se puede apreciar tanto la ranura de acceso a las placas, el conector RJ45 extra del front-end y el conector VGA. Éste último permite conectar un monitor al módulo para trabajar con el front-end desde un entorno gráfico.

Figura 6.3: Montaje del Modelo final en producción (vista frontal)

Como se pude apreciar en la figura 6.4 se han incluido cables RJ45 planos desde el switch a cada placa. Además se ha modificado el panel interior, realizado una serie de divisiones para facilitar el guiado del cableado dentro del módulo.

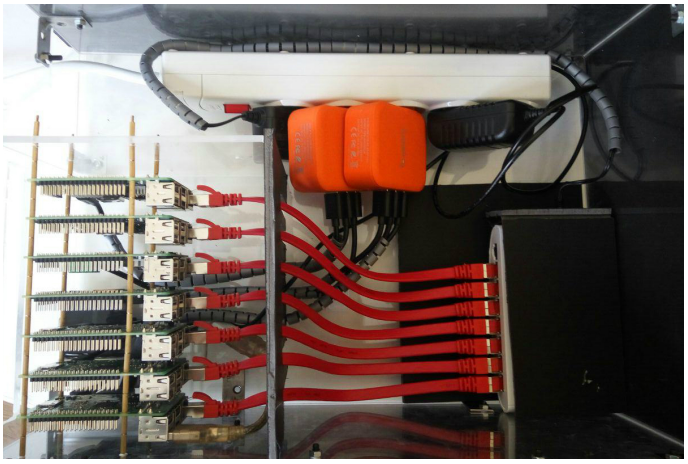

Figura 6.4: Montaje del Modelo final en producción (vista superior)

6.2. Experimentos

En esta sección se muestran los resultados obtenidos sobre el Modelo final en las pruebas de temperatura.

6.2.1. Experimento 1

El escenario de esta prueba consiste en una única placa Raspberry Pi trabajando con 4 cores en paralelo. El tiempo total de la prueba es de 3 horas, con un periodo de muestreo de 10

segundos. En la gráfica siguiente se muestra el tiempo total expresado en segundos.

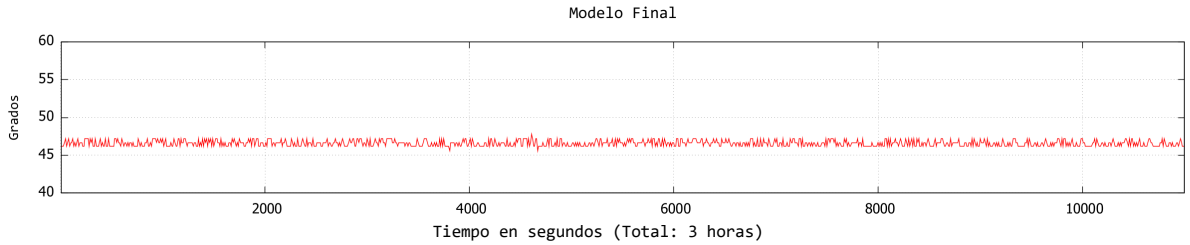

Figura 6.5: Resultados Prueba 1

La prueba realizada con una sola Raspberry Pi muestra una temperatura estable, lo cual es un buen indicativo de que el sistema podrá funcionar correctamente durante varias horas.

6.2.2. Experimento 2

El escenario de esta prueba consiste en 3 placas Raspberry Pi trabajando de forma simultánea. Cada una de ellas mantiene sus 4 cores trabajando a máximo rendimiento. El tiempo total de las pruebas es de 3 horas. Igual que en el experimento anterior, el periodo de muestreo es de 10 segundos y en cada gráfica muestra el tiempo total expresado en segundos.

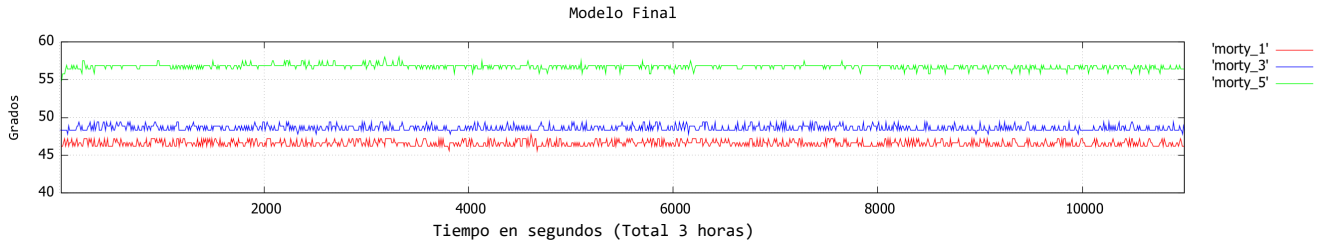

Figura 6.6: Resultados Prueba 2

Tras efectuar el segundo experimento podemos apreciar que el hecho de trabajar de manera conjunta no parece afectar al rendimiento o a la temperatura.

6.2.3. Experimento 3

Finalmente, el escenario de esta prueba consiste 6 placas Raspberry Pi trabajando de forma simultánea. Como en los casos anteriores, cada una de ellas mantiene sus 4 cores trabajando

en paralelo, el tiempo total de las pruebas es de 3 horas y el muestreo se realiza cada 10 segundos.

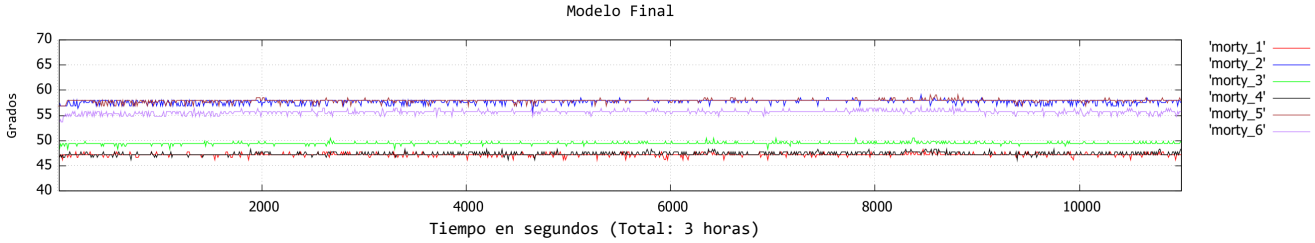

Figura 6.7: Resultados Prueba 3

6.3. Conclusiones

En general el Modelo final permanece estable en situaciones de cómputo masivo y presenta unos resultados adecuados en cuanto a temperaturas.

Como ocurre con los prototipos, las placas de Raspberry Pi que disponen de disipadores (ver capitulo 5) muestran diferencias de temperatura.

La apertura frontal permite un fácil acceso a las ranuras SD de las placas Raspberry Pi.

La disposición de los elementos facilita la escalabilidad de clúster.

Además, la agrupación del cableado y el uso de cables de red planos, simplifica el manejo de estos a la hora de manipular los componentes.

Capítulo 7

Software para la ejecución remota de prácticas

'long long long' is too long for GCC

Some GCC programer

En este capítulo presentamos el software responsable del envío de las prácticas desde la GUI del SIMCAN hasta el clúster, detallando la arquitectura empleada y el planificador de tareas.

7.1. Cliente-Servidor

Una vez desplegado el clúster, es necesario establecer una comunicación entre éste y el mundo exterior. Como ya se ha detallado anteriormente, la GUI del SIMCAN realizará la conexión con el front-end del clúster (ver figura 2.2) Para implementar esta comunicación, nos hemos decantado por una arquitectura Cliente-Servidor en Java[1] basada en sockets.

Los sockets son un mecanismo que nos permite establecer un enlace entre dos programas que se ejecutan generalmente, en ordenadores remotos. Habitualmente uno de estos programas adquiere el rol de servidor, el cual se mantiene escuchando y aceptando posibles peticiones de múltiples clientes. Java[2] a través de la librería java.net[19] nos proporciona dos clases: *Socket* para implementar la conexión desde el lado del cliente y *ServerSocket* que nos permitirá manipular la conexión desde el lado del servidor.

El servidor, el cual se ejecutará en el front-end del clúster, estará a la espera de recibir los archivos de configuración necesarios para lanzar una simulación. Todas las simulaciones

[1] https://www.java.com/es/

que lleguen al clúster serán gestionadas por un planificador, el cual gestionará los recursos disponibles del clúster para llevar a cabo las ejecuciones de las simulaciones. Por su parte, el programa cliente requerirá la IP del servidor para establecer conexión. Para facilitar el envío de estos archivos, estos, serán comprimidos en un *.tar*.

Para proporcionar una interfaz que no sobrecargue de información al usuario hemos diseñado una aplicación que permita a éste mandar estos archivos de forma cómoda. Esta interfaz se ha implementado utilizando la biblioteca gráfica Swing de Java.

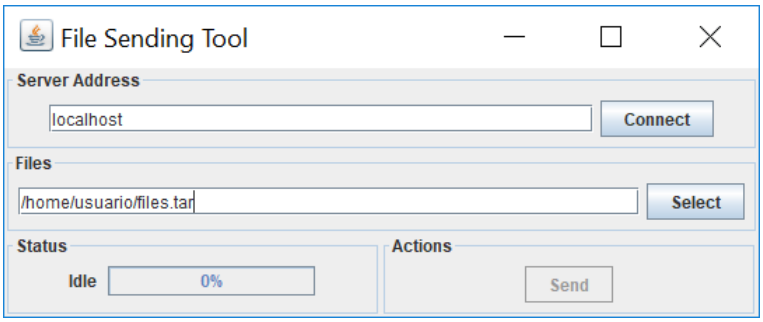

Figura 7.1: Interfaz Gráfica

En el envío de archivos se ha prestado especial atención a la seguridad, empleando un algoritmo de cifrado simétrico AES[2] con clave de 16 bits. Dicho algoritmo utiliza una estructura de bucle para realizar permutaciones de datos, reemplaza una unidad con otra para datos de entrada. Este cifrado utiliza la misma clave para cifrar y descifrar los datos, y aplica esa clave a los bloques de datos de longitud fija.

Servidor y Cliente han de tener acceso a la clave *'ProgramacionDeSistemasDistribuidos'*, la cual se puede modificar en el código en caso que el usuario desease hacerlo. El programa servidor utiliza ésta cuando cifra un mensaje y al enviarlo al destinatario, éste lo descifra utilizando esta misma clave de manera inversa. La figura 7.2 muestra el envío una vez completado.

[2]https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html

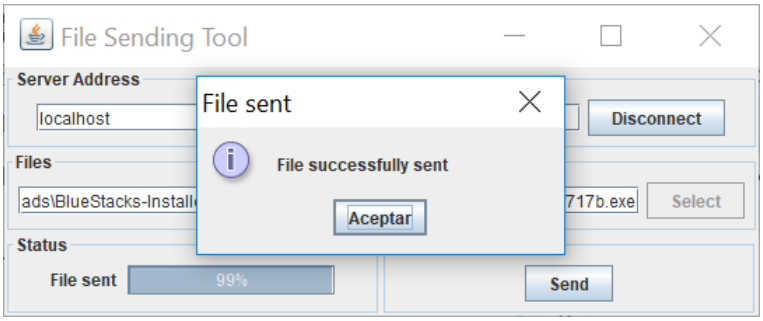

Figura 7.2: Interfaz Envío completado

En el esquema 7.3 se muestra el esquema de carpetas del front-end. El directorio *received* contiene los trabajos que le envía el cliente, los cuales descomprime y procesa en su directorio /home/pi/projects/simcan/simulations/scenario. El directorio *send* contiene los trabajos que están dispuestos para ser lanzados al back-end.

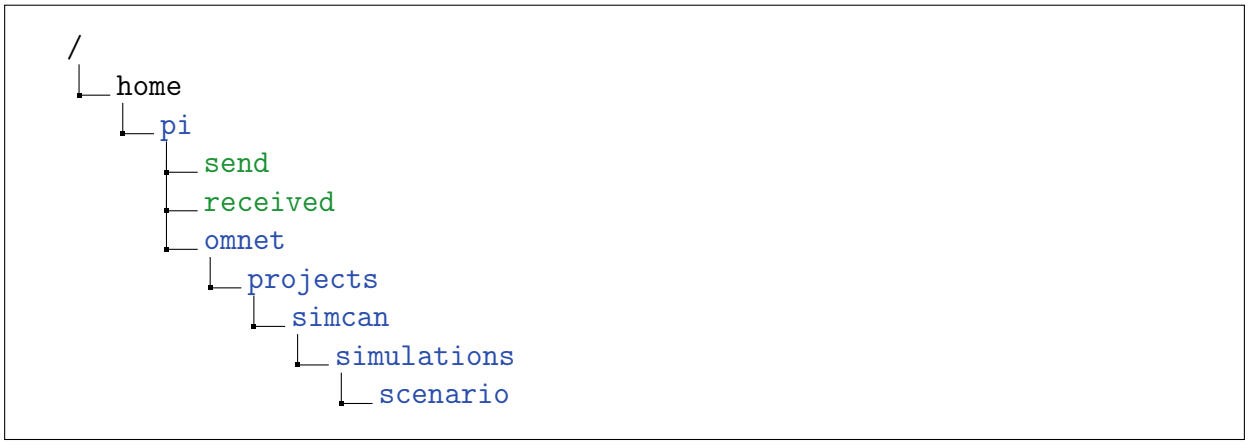

Figura 7.3: Esquema de directorios

7.2. Planificador

Para ejecutar las simulaciones en el back-end se ha desarrollado un planificador que distribuye éstos entre los nodos. Con el fin de conseguir un balance de carga equitativo, éste trabaja como un *buffer circular* implementando el algoritmo Round-Robin, en el cual, cada fichero se envía a un nodo distinto de la red para ser procesado.
De esta manera se explota la paralelización del sistema evitando que un nodo tenga una mayor carga de trabajo, mientras el resto permanecen inactivos.

El planificador está desarrollado mediante MPI (Message Passing Interface), cada nodo del back-end dispone de un *daemon* que se activa durante el arranque del sistema y queda a la espera peticiones desde el front-end. En el momento en que recibe una señal por parte del back-end, accede a la ruta en la cual se encuentra el fichero y comienza el cómputo de éste. Una vez finalizado, envía los resultados al usuario en un archivo comprimido que e informa al back-end de que ha terminado la tarea, y queda a la espera de nuevos trabajos.

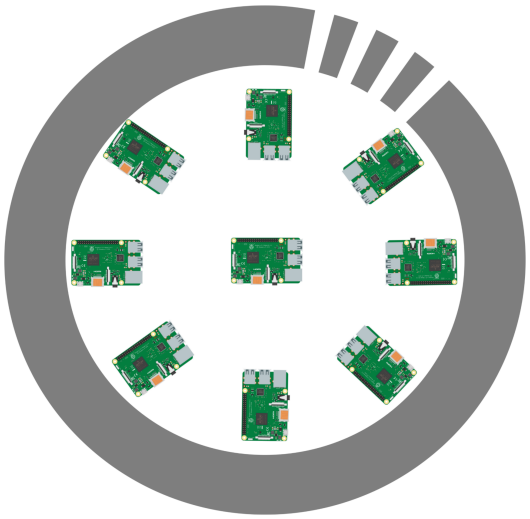

Figura 7.4: Planificador de tareas

El proceso desde que se modela un entorno hasta que se obtienen los resultados de la simulación se presenta en la figura 7.5.

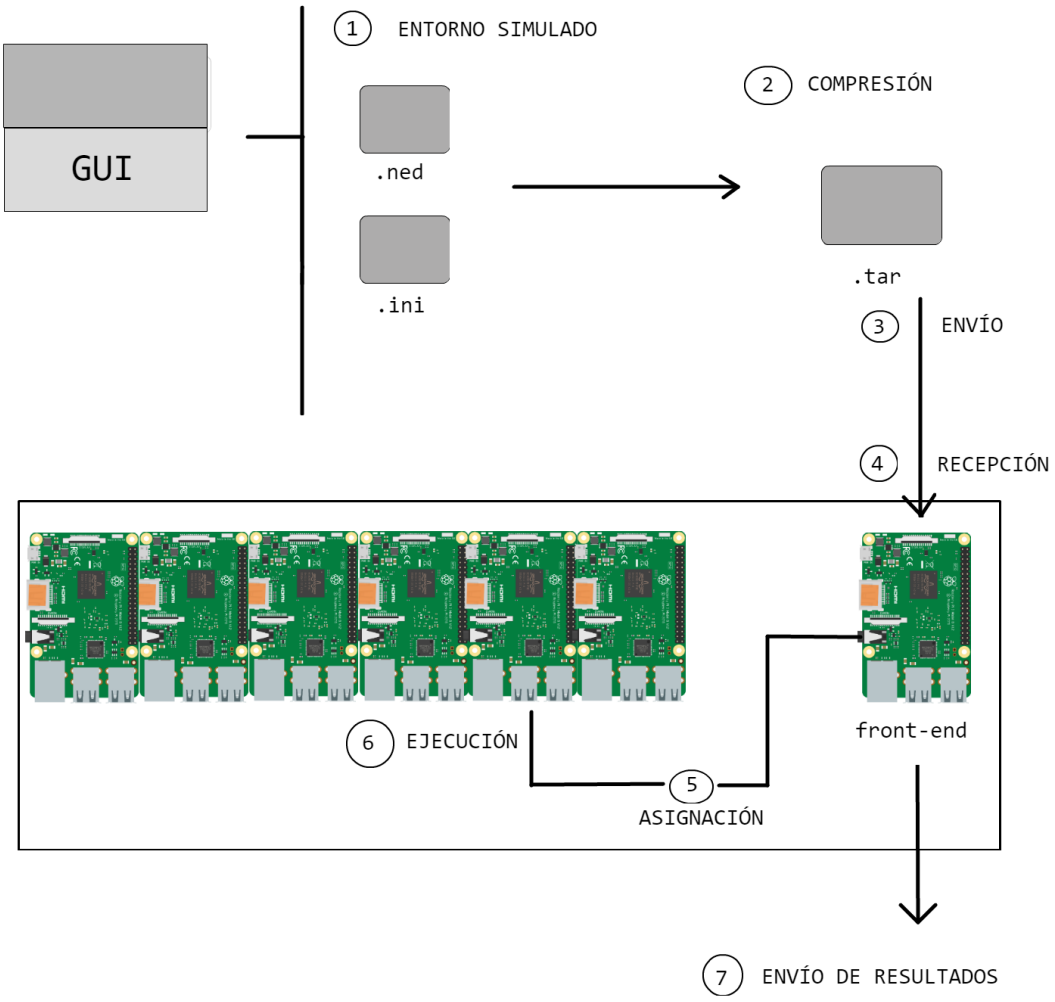

Figura 7.5: Esquema global del proceso

Inicialmente (paso 1) se dispone del desarrollo del entorno simulado, comprende los archivos .ned, que representan la topología de los sistemas distribuidos y los ficheros .ini, que incluyen la configuración de los parámetros. Seguidamente (paso 2), se comprimen estos ficheros, que representan el entorno, en un archivo .tar. Éste, se manda al clúster (paso 3) para que el front-end pueda recibirlo (paso 4). A continuación (paso 5), se asignan los recursos de cómputo a un nodo del back-end en función a los cores que estén libres y este procede a la ejecución de la simulación (paso 6). Por último, los resultados son enviados en un archivo comprimido al usuario (paso 7).

Capítulo 8

Experimentos de rendimiento

Nunca uses a un humano para hacer el trabajo de una máquina.

Morfeo

En este capítulo se presenta un estudio de rendimiento utilizando el clúster diseñado en este proyecto y un ordenador portátil de gama media.

8.1. Configuración de los Experimentos

El objetivo de los experimentos realizados en este capítulo es comparar el rendimiento del clúster frente a un ordenador convencional. Para realizar estos experimentos de rendimiento se ha generado una batería de pruebas utilizando las prácticas de la asignatura PSD.

8.1.1. Características del clúster

El clúster dispone de 6 placas de Raspberry Pi, como se especifica en el capítulo 2. Cada una de ellas dispone de un procesador ARM Cortex-A53 de 4 núcleos a 1,2 GHz y 1 GB de RAM LPDDR2. Además, dispone de una versión de OMNeT++ 4.6, INET y SIMCAN. El sistema operativo[12] instalado es un Debian *Jessie* de 64 bits.

8.1.2. Características del ordenador portátil

El modelo elegido para realizar las pruebas es un Xiaomi Mi Air, el cual dispone de un procesador Intel Core i5-6200u Quad Core de 2.3GHz por núcleo y 8GB de RAM. Este equipo contiene las mismas versiones de OMNeT++, INET y SIMCAN que las instaladas en el clúster. El sistema operativo[14] utilizado es un Ubuntu[4] versión 14.1 de 64 bits[7].

8.1.3. Carga de trabajo

Los experimentos llevados a cabo en esta sección están enfocados a estudiar el rendimiento en entornos distribuidos[17]. Para ello se hará uso de la aplicación modelada en SIMCAN *FileTransfer*.

La aplicación *FileTransfer* realiza una copia remota de un fichero. Esta aplicación permite dos operaciones, leer el fichero del servidor remoto para copiarlo en el nodo donde se ejecuta la aplicación, o copiar el fichero almacenado en el disco local del nodo donde se ejecuta la aplicación en un servidor remoto.

La carga de trabajo utilizada en los experimentos consiste en ejecutar la simulación de la aplicación *FileTransfer*, en diferentes entornos distribuidos, utilizando SIMCAN.

8.2. Experimento 1

Este experimento consiste en ejecutar la simulación de un entorno distribuido sobre el ordenador portátil y sobre una Raspberry Pi. El objetivo es comparar el tiempo requerido para su ejecución en ambos sistemas.

Como se puede comprobar en la figura 8.1, la capacidad de cómputo del ordenador portátil es superior a la de una Raspberry Pi. El tiempo empleado por este último es cercano a los 10 minutos, mientras que la Raspberry Pi necesita aproximadamente 40 minutos para completar la misma tarea. En base a estos resultados, se observa que una Raspberry Pi es 4 veces más lenta que el ordenador portátil.

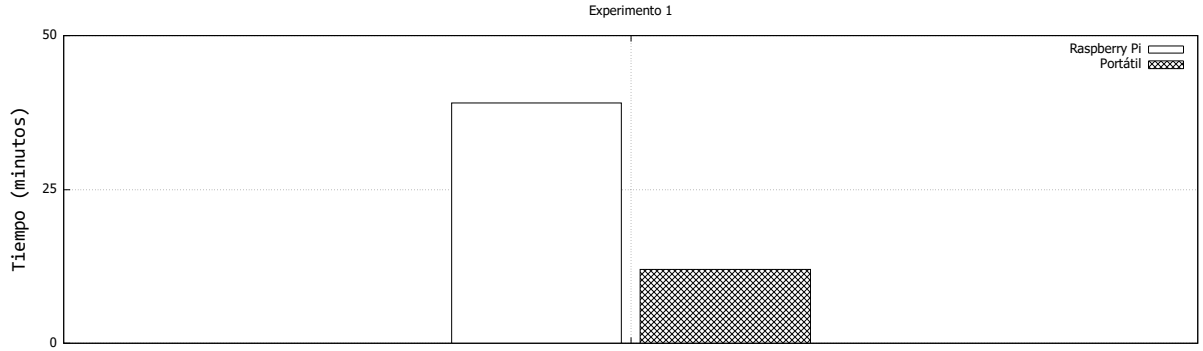

Figura 8.1: Ejecución de la simulación (Portátil Vs 1 Raspberry Pi)

De este experimento se puede concluir que, para una simulación, una Placa Raspberry Pi no puede competir con la capacidad de cómputo de un ordenador portátil.

8.3. Experimento 2

Este experimento consiste en medir la capacidad de cómputo de una Raspberry Pi y un ordenador portátil, trabajando con todos sus procesadores de forma concurrente.

Para ello, se han lanzado 4 simulaciones en paralelo sobre el ordenador portátil y sobre una Raspberry Pi. Cada simulación está formada por 4 entornos. La figura 8.2 muestra los tiempos requeridos por cada equipo para ejecutar la simulación.

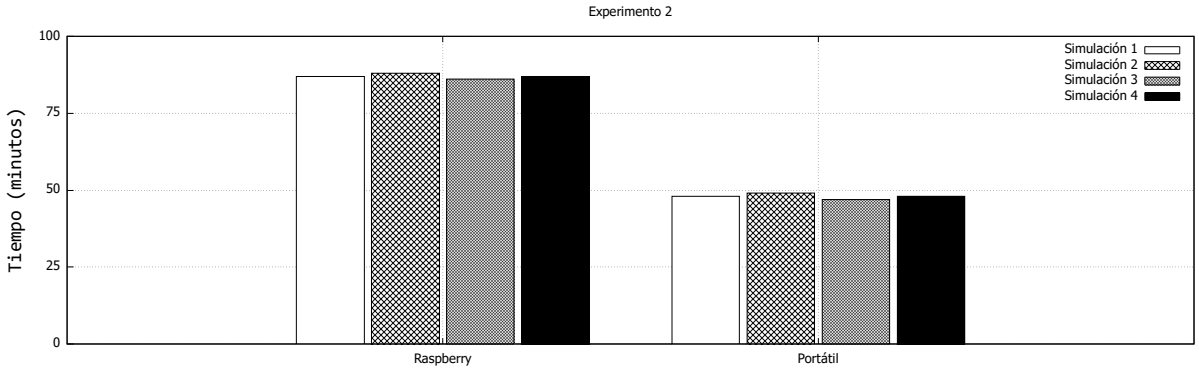

Figura 8.2: Ejecución de 4 simulaciones (Portátil vs 1 Raspberry Pi)

Se puede comprobar que el ordenador portátil ofrece unos resultados superiores a los de una Raspberry Pi. El tiempo requerido por ésta para ejecutar las 4 simulaciones es, aproximadamente, 90 minutos, mientras que el ordenador portátil es capaz realizar el mismo trabajo en menos de 50 minutos. Esto indica que puede completar el trabajo, aproximadamente, en la mitad de tiempo que una Placa Raspberry Pi.

Ante los resultados de los experimentos 1 y 2, podemos concluir que una Raspberry Pi es inferior, computacionalmente, a un ordenador portátil.

8.4. Experimento 3

Este experimento consiste en explotar el nivel de paralelismo que ofrece el clúster distribuyendo las simulaciones entre varios nodos. Para ello, se ejecutan 4 simulaciones, tanto en 4 placas Raspberry Pi, como en el ordenador portátil. Cada simulación está formada por 4 entornos.

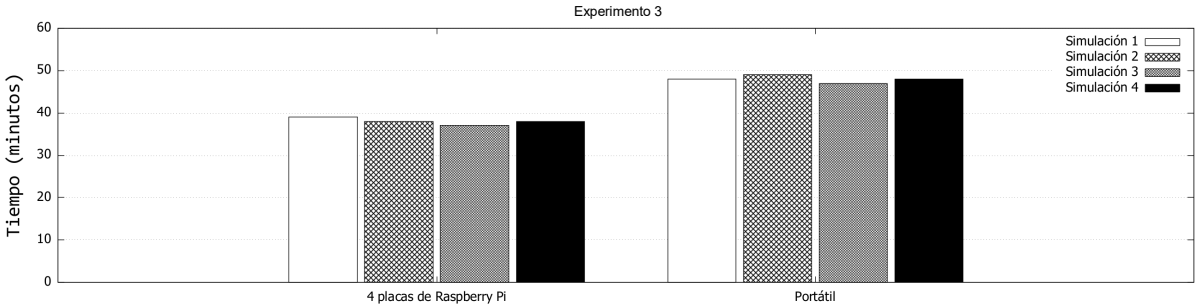

Figura 8.3: Ejecución de 4 simulaciones (Portátil Vs 4 Raspberry Pi)

Como se observa en la figura 8.3, la capacidad de paralelizar trabajos entre distintos nodos supone una mejora significativa frente a una única Raspberry Pi. Además, el tiempo empleado para ejecutar completamente la carga de trabajo en las placas Raspberry Pi es inferior al requerido por el ordenador portátil. En esta prueba, los nodos de Raspberry Pi son un 20 % más rápidos que el ordenador portátil. Esto es debido a que se explota el paralelismo del clúster para ejecutar las simulaciones.

8.5. Experimento 4

En este experimento se utilizan más recursos del clúster, tanto en la cantidad de nodos, como en el número de procesos por cada uno de ellos. Para ello, se han ejecutado 12 simulaciones, donde cada una contiene 12 entornos.
En total, se han utilizado 4 nodos de Raspberry Pi, cada uno ejecutando 3 procesos de forma concurrente. El ordenador portátil realiza todas las simulaciones utilizando sus 4 núcleos. La figura 8.4 muestra el tiempo que ha requerido cada equipo en finalizar todas las simulaciones.

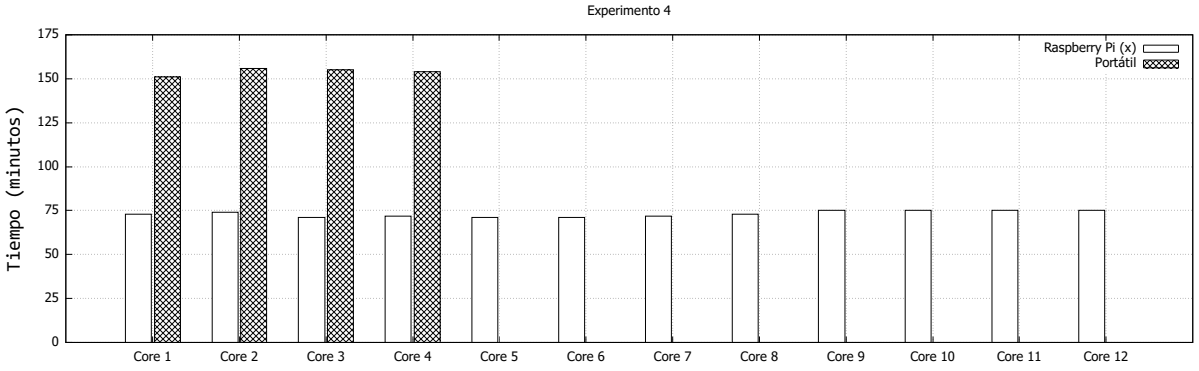

Figura 8.4: 12 Simulaciones

En este experimento se observa que cuanto mayor es el número de trabajos, más eficiente es el clúster frente al ordenador portátil. El clúster completa todas las simulaciones en un tiempo cercano a los 75 minutos, mientras que el ordenador portátil tarda un tiempo aproximado de 160 minutos para completar el mismo número de simulaciones. Esto es debido a su limitada capacidad de paralelización frente al clúster.

8.6. Conclusiones

El precio del ordenador portátil utilizado en los experimentos es de 574,99 €, mientras que el de cada placa Raspberry Pi 3 Model B es de 47,19 €. Así, el coste de cada placa Raspberry Pi es, aproximadamente, 12 veces menor que el coste del ordenador portátil. El precio total del clúster es ligeramente superior al coste del ordenador portátil

La capacidad de cómputo de una Raspberry Pi es inferior a la de un portátil de gama media. Esto queda reflejado en los dos primeros experimentos, donde se puede apreciar la diferencia de tiempos en la ejecución de simulaciones donde se utiliza una única placa Raspberry Pi 3.

La característica más destacada del clúster es la ejecución paralela de la carga de trabajo, tal y como queda reflejado en los experimentos 3 y 4. El ordenador portátil ofrece un rendimiento ligeramente inferior cuando compite contra 4 placas Raspberry Pi. Sin embargo, cuanto más se explota el paralelismo del cúster, mejor es el rendimiento obtenido, por lo que podemos concluir que es el aspecto más notable del clúster. Además, en los casos de cómputo masivo donde se utiliza un número elevado de núcleos, el clúster ofrece un mejor ratio coste/rendimiento que el ordenador portátil.

El ordenador portátil cumple de forma muy adecuada las labores de cómputo para las necesidades de simulación de un estudiante. Sin embargo, el clúster puede ofrecer mayor rendimiento en el caso de que se requiera una mayor cantidad de cómputo masivo.

Capítulo 9

Presupuesto

¿Por qué ganar trillones cuando podemos ganar... billones?

Dr. Maligno

A lo largo de este capítulo se desglosan los costes de este proyecto, entre los cuales se incluyen diseño, montaje, coste de los materiales y desarrollo del software.

9.1. Componentes y Materiales

En este apartado se presenta el desglose de gastos en material y componentes, así como la cantidad necesaria de cada uno, que se han utilizado tanto en el montaje del contenedor, como en el diseño del clúster (ver tablas 9.1 y 9.2).

El desglose de materiales de la tabla 9.1 incluye los utilizados durante la construcción[1] del contenedor. El cálculo de horas de trabajo necesarias para la elaboración de éste se incluye en la figura tabla 9.3, en el apartado de *Desarrollo Hardware*.

[1]No se incluyen los gastos en herramienta requeridas

Componentes	Importe	Unidades	Total
Base Múltiple de enchufes 5 Puertos	14,5	1	14,5
Plancha Metacrilato Transparente DINA3 4mm	7	3	21,99
Plancha Metacrilato Transparente DINA4 4mm	4	5	20,99
Tornillo Allen Cabeza Cilíndrica 10x30	0,15	20	3
Ángulo Corner Brace	0,5	14	7
Cable ZIP 1,5m	2,5	1	2,5
Total			***69,98***

Tabla 9.1: Coste de los materiales

La siguiente tabla muestra el desglose de materiales que componen el clúster. La lista de precios se ha elaborado utilizando el presupuesto que ofrecieron los distribuidores oficiales de la Universidad Complutense, así como otros gastos que hemos afrontado de proveedores externos.

Componentes	Importe	Unidades	Total
Raspberry Pi 3 Model B	47,19	7	330,33
Cable MicroUSB-USB M/M	3,63	7	25,41
Aukru Set 3 Disipadores de calor heatsinks	3,99	3	11,97
Aukru Cargador Micro USB 3000mA	7,99	1	7,99
Switch 8 Puertos D-Link DGS-1008D	29,04	1	29,04
Tarjeta MicroSD Kingston	15,6	7	109,2
Eluteng USB Cooling 120mm	10,89	2	21,78
Adaptador Nanocable USB 3.0 a Gb Ethernet	15,73	1	15,73
Cable de red LAN, 1 Gbit/s, CAT7	1,85	10	18,5
Lumsing Cargador USB 4 Puertos	6,99	2	13,98
Aukey Adaptador HDMI VGA	8,99	1	8,99
Total			***592,92 €***

Tabla 9.2: Coste de los componentes

9.2. Resumen de gastos

Para realizar el cálculo del coste por hora de trabajo supondremos que los autores de este trabajo pertenecen a la empresa Miguel&&Daniel Co. El salario bruto de cada uno es de 33.000€. Para realizar el cálculo del salario neto debemos aplicar la suma de un 6,35 % de retención de Seguridad Social más un 13,24 % de retención de IRPF, obteniendo un salario neto resultante de 26.535,3€.

Según el convenio, un empleado trabaja 1800 horas anuales. Por tanto, obtenemos un coste por hora de trabajo de 18,33€ por empleado. En este proyecto, el cual ha tenido una duración de 10 meses se han invertido una media de 4 horas al día de trabajo durante 5 días a la semana. Así, se han invertido un total de 800 horas (ver tabla 9.3)

Montaje	**Estimación Horas**	**Precio/hora**
Desarrollo Software	180	18,33€
Desarrollo Hardware	200	18,33€
Documentación	250	18,33€
Administración	20	18,33€
Pruebas	150	18,33€
Coste total	800	**14.664€**

Tabla 9.3: Coste Montaje

Por último, como se puede observar en la tabla 9.4, se presenta el gasto total del proyecto, el cual incluye el coste de los materiales, así como las horas que ha invertido cada trabajador.

Coste horas	**Coste materiales**	**Precio total**
14.664€	662,18€	**15.326,18€**

Tabla 9.4: Coste total

Capítulo 10

Conclusiones y trabajo futuro

No pienso nunca en el futuro porque llega muy pronto.

Albert Einstein

10.1. Conclusiones

Para nosotros ha sido todo un desafío llevar a cabo este proyecto, el cual ha requerido muchas reuniones, mucho trabajo y horas de esfuerzo, acompañadas de un gran proceso creativo para dar a luz este Trabajo de Fin de Grado. Como ya describimos en el capítulo 1, existen trabajos de mayor escala y presupuesto relacionados con el objetivo de este proyecto. Sin embargo, en nuestro caso, nos alegra saber que tendrá una aplicación real y podrá aplicarse a unas prácticas como las de la asignatura que nos atañe.

Dentro del apartado arquitectónico, queremos resaltar la importancia de realizar un estudio del diseño previo antes de entrar en la fase de construcción, ya que puede ahorrar tiempo y recursos humanos. La elección de los materiales es fundamental. En nuestro caso nos decantamos por el metacrilato, que pese a ser un material ligero y aislante, ha llegado a presentar problemas de manipulación y corte restándonos tiempo para otras tareas del proyecto.

Nuestro mayor reto han sido las altas temperaturas. El calor en los sistemas de alto rendimiento es un componente adverso muy a tener en cuenta. Durante el desarrollo de los modelos llegamos a una de las conclusiones más relevantes, pequeños recursos de bajo coste, como los disipadores, han demostrado que pueden aportar una gran mejora en el sistema.

Podemos concluir que hemos llevado a cabo con éxito el objetivo del proyecto, consiguiendo además resultados prometedores.

Como conclusión general, creemos que el proyecto se ha llevado de forma satisfactoria, pese al esfuerzo que nos ha conllevado el diseño, la implementación y las pruebas. Este tipo de arquitecturas de bajo coste podrían permitir múltiples aplicaciones a nivel académico si se partiese de una arquitectura ya existente como la que nosotros proporcionamos.

10.2. Trabajo Futuro

Este proyecto se centra en el ámbito educativo. Creemos que este trabajo puede brindar a los futuros alumnos, tanto de la asignatura de Programación de Sistemas Distribuidos, como de la Facultad de Informática en general, un proyecto abierto y muy interesante para seguir desarrollando y ampliando, en la medida que ellos deseen gracias a nuestras detalladas guías y especificaciones que esperamos realicen con la misma pasión y dedicación que nosotros hemos entregado.

Los trabajos que se realicen en el futuro pueden abarcar desde la mejora del propio diseño arquitectónico, la escalabilidad del sistema o la incorporación de diferentes elementos, ya que la Raspberry Pi dispone de una gama de periféricos que pueden ser conectados a su GPIO (del inglés, Entrada/Salida de Propósito General).

Una de las mejoras más interesantes que se podrían aplicar de manera inmediata, dado que la arquitectura actual ya lo soporta, sería la obtención de datos de carga en tiempo real, representada en una aplicación de monitorización de manera más visual. Además, incluir nuevos planificadores para el reparto de carga.

10.3. Conclusions

Implementing this project from scratch has been a challenge. There have been many meetings, a lot of work and hours of effort, accompanied by a great creative process to successfully carry out this TFG. As we already described in chapter 1, there are other projects of greater scale and budget related to our work. However, in our case, we are glad to know that it will be applied to practices like those of the PSD course that concerns us.

Regarding the architectural section, we want to highlight the importance of a through study of the previous design before entering the construction as it can save time and human resources. The choice of materials is fundamental. In our case, we opted for the methacrylate, which despite being a light and insulating material, has come to present problems of manipulation and cutting down time for other tasks of the project.

Our biggest challenge has been the high temperatures. The heat in the high performance systems is a very adverse component to consider. As our models and designs progressed,

we discover a relevant conclussion, small and low-cost resources, such as heat sinks, have a direct input on the overall system temperature.

We can conclude that we have accomplish the main objective of this work, achieving promising results.
As a general conclusion, we believe that this project has been succesfully implemented, despite the effort that has gone into design, implementation and testing, this type of low-cost architectures could allow multiple applications at the academic level.

10.4. Future work

This project is focused on the educational field. We believe that our work can provide to the future students of Distributed Systems Programming a very interesting project to be expand at using detailed guides and specifications, which we hope to carry out with the same passion and dedication that we have given.

The works that are carried out in the future may include, among other aspects, the improvement of the own architectonic design, the scalability of the system or the incorporation of different elements, since the Raspberry Pi has a range of peripherals that can be connected to its GPIO.

One of the most interesting improvements that could be applied immediately, given that the current architecture already supports it, would be to obtain load data in real time represented in a monitoring application in a more visual way. In addition, include new planners for load sharing.

Apéndice A

Aportación individual de cada integrante del proyecto

En este apartado detallamos la aportación individual de cada miembro del equipo. La división de tareas ha sido planificada en base a la experiencia y conocimiento de cada miembro.

A.1. Daniel Quiñones

Durante la etapas de virtualización y de instalación en el entorno real, se ha encargado de las tareas de configuración, así como la instalación del sistema operativo Raspbian Jessie, la plataforma de simulación SIMCAN, OMNeT++ e INET .

Llevó a cabo la instalación y configuración del servidor DHCP en el entorno virtual. Elaboró la guía de instalación de DHCP en Raspbian Jessie y, posteriormente, su configuración en el entorno real.

En cuanto a la fase de diseño de prototipos, se ha encargado de ejecutar pruebas de temperatura y recopilación de datos desde el modelo 1 hasta el modelo final.

En la fase de arquitectura hardware, diseñó el Modelo 3 y lo implementó en 3D con la herramienta *OpenScad*. Posteriormente, realizó en cartón dicho modelo, siendo el último prototipo en este material, dando paso al modelo final. Este último ha sido una tarea compartida por ambos.

Se ha encargado de la implementación del software cliente - servidor, así como la interfaz gráfica integrada en SIMCAN creada con la biblioteca gráfica *Swing*.

En cuanto a la memoria[8], ha sido una tarea compartida entre los dos, donde ambos han participado en las tareas de redacción, revisión y la creación del diagrama de Gantt en LA-

TEX para planificar y programar tareas a lo largo del proyecto.

En la elaboración de este documento ha participado en las tareas conjuntas de redacción y revisión.

A.2. Miguel Romero

Encargado de realizar la instalación inicial de SIMCAN en la arquitectura ARM de las placas Raspberry Pi 3. Esto resultó definitivo para abrir la posibilidad de elaborar el resto de este TFG. Posteriormente se ha encargado de generar las guías de instalación de OMNeT++, INET y SIMCAN en *Debian Jessie* que han servido como base para la configuración del entorno virtual y real del clúster.

En la fase de configuración ha realizado la instalación del servidor NFS y posterior documentación de ésta. Además programó de los *daemon* de sistema utilizados en *front-end* y *back-end*.

En cuanto a la elaboración del clúster, ha realizado el diseño de los modelos 1, 2 y 2b. Ha confeccionando el modelado en 3D de los mismos a través de la herramienta *OpenScad*. Además, construyó los prototipos en cartón de los modelos 1, 2 y 2b.

En los experimentos de diseño se ha encargado de programar el script de medición de temperaturas. Posteriormente ha realizado la ejecución el estudio térmico sobre los diferentes modelos, así como la posterior recopilación y procesado de datos. En esta fase también ha sido el encargado de realizar la elaboración de las gráficas de temperatura mediante la herramienta *gnuplot*.

Ha participado en el diseño del modelo final, modelado del mismo con *OpenScad*, y construcción del módulo en metacrilato.

En cuanto al desarrollo software, se ha encargado de elaborar el programa de comunicación, basado en sockets, entre nodos del *back-end*.

En la elaboración de este documento ha participado en las tareas conjuntas de redacción y revisión.

Apéndice B

Software desarrollado en el proyecto

El software implementado en este TFG se encuentra alojado en la siguiente dirección de Google Drive: https://drive.google.com/open?id=16EQndlwwS0Bt3fNVCtMeyvA_aeMEy5FB

Bibliografía

[1] *Raspberry Pi* Página principal del proyecto. https://www.raspberrypi.org/.

[2] Eckel, B. *Piensa en Java*. PRENTICE-HALL, 2007.

[3] Eckel, B. *Piensa en C++*. PEARSON, 1995.

[4] *NFS. Archlinux*, disponible en: https://wiki.archlinux.org/index.php/NFS.

[5] Lynx, J. *A Simple Guide to Help You Get the Most Out of Your Raspberry Pi 3*. CreateSpace Independent Publishing Platform, 2016.

[6] Samarth, S. *Learning Raspberry Pi*. Packt Publishing, 2015.

[7] Zadok, E. *LINUX: NFS and automounter administration*. SYBEX INC, 2001.

[8] Gómez Martín, MA. y Gómez Martín, PP. *TEXiS: Una plantilla de LATEX para Tesis y otros documentos*. Universidad Complutense de Madrid, 2009.

[9] *DHCP. Archlinux*, disponible en: https://wiki.archlinux.org/index.php/Dhcpcd.

[10] *Top500 List*, disponible en: https://www.top500.org.

[11] Nuñez Covarrubias, A. y Pickin S. *Programacion de Sistemas Distribuidos*. Universidad Complutense de Madrid. 2015.

[12] Moreno Vozmediano, R. y Fabero Jiménez, JC. *Sistemas Operativos y Redes*. Universidad Complutense de Madrid. 2012.

[13] Moreno Vozmediano, R. y Fabero Jiménez, JC. *Redes*. Universidad Complutense de Madrid. 2013.

[14] Igual Peña, F. *Ampliación de Sistemas Operativos*. Universidad Complutense de Madrid. 2013.

[15] Carretero, J. *SISTEMAS OPERATIVOS. Una vision aplicada*. McGRAW-HILL/INTERAMERICANA DE ESPAÑA, 2001.

[16] García Población, O. *Unix y Linux. Guía práctica, 3ª edición.* RA-MA S.A. Editorial y Publicaciones, 2004.

[17] R.Leon Casiano, C. *Programación Distribuida y Mejora del Rendimiento.* Universidad de La Laguna, 2012.

[18] *Raspbian Jessie*, disponible en: https://www.raspberrypi.org/blog/raspbian-jessie-is-here/.

[19] González de Miguel, A. *Tecnología de la Programación, componentes visuales.* Universidad Complutense de Madrid, 2018.