Universidad Zaragoza
1542

Proyectos Docente e Investigador

Concurso de Acceso de los Cuerpos Docentes Universitarios

Plaza-Procedimiento Nº 2020–02

Actividades Docentes: Administración de Sistemas I, Proyecto Hardware y Garantía y Seguridad

Darío Suárez Gracia

Índice general

Índice de figuras

Índice de cuadros

Resumen

El presente documento incluye los Proyectos Docente e Investigador realizados por el candidato para el concurso de acceso a plazas de cuerpos docentes universitarios. (BOE Nº 21 de 24 de enero de 2020) TU. Plaza número 2020-02. La plaza se enmarca dentro del área de Arquitectura y Tecnología de Computadores perteneciente al Departamento de Informática e Ingeniería de Sistemas en la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza. La convocatoria de esta plaza junto a sus bases fueron anunciadas públicamente el día 20 de enero de 2020 una vez autorizada la contratación de las plazas por el Rector mediante resolución de 6 de junio de 2019 [1].

La primera parte se corresponde con una de las 3 asignaturas de la actividad docente de la plaza, en concreto, asignatura común Administración de Sistemas del grado en Ingeniería Informática impartido en la Escuela de Ingeniería Arquitectura de la Universidad de Zaragoza. El proyecto incluye el contexto y el entorno donde se imparte la asignatura y un posible temario para su impartición desglosado por módulos y temas.

La segunda parte del documento detalla el proyecto investigador del aspirante y aborda la mejora de las prestaciones y la eficiencia energética en el procesado de grafos mediante sistemas heterogéneos a múltiples niveles, desde los modelos de programación al hardware. El proyecto asume que el gran tamaño que han alcanzado hoy día los grafos junto con su utilización en un gran número de problemas demanda mejoras a todos los niveles y que el empleo de distintos dispositivos de computo, en especial las FPGA, puede ayudar en esta tarea.

[1] https://www.boe.es/boe/dias/2020/01/24/pdfs/BOE-A-2020-1090.pdf

Parte I

Proyecto Docente

Resumen

La primera parte de este documento presenta un posible proyecto docente para la asignatura Administración de Sistemas. Está dividido en 3 capítulos, el primero estadedicado al marco académico e institucional, el segundo realiza una propuesta de diseño curricular para la asignatura, y el último termina describiendo en un nivel mayor de detalle el desarrollo por módulos planteados, incluyendo el material de las clases de teoría, problemas y prácticas.

Capítulo 1

Marco Académico e Institucional

Este capítulo describe el entorno de la asignatura Administración de Sistemas impartida en la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza. Primeramente se hará un resumen breve de los aspectos relativos a las instituciones y a la legislación aplicada, para continuar con el contexto curricular de la asignatura.

1.1 Contexto Institucional y Legislativo

Las universidades en España se rigen fundamentalmente por la Ley Orgánica de Universidades 6/2001 del 21 de diciembre de 2001 [1]. Esta ley fue modificada por la Ley Orgánica 4/2007 del 12 de abril de 2007, que a su vez fue modificada por el Real Decreto-Ley 14/2012 del 20 de abril de 2012. La última modificación se realizó mediante la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y Garantía de los derechos digitales, para incluir una nueva letra l) en el apartado 2 del artículo 46, Derechos y deberes de los estudiantes, con el contenido siguiente:

l) La formación en el uso y seguridad de los medios digitales y en la garantía de los derechos fundamentales en Internet.

Tarea que curiosamente puede y debe realizarse en la asignatura de Administración de Sistemas. Finalmente, la Ley Orgánica de Universidades fue actualizada por última vez de acuerdo a la Ley 06/2018, de 3 de julio, de Presupuestos Generales del Estado para el año 2018.

A nivel continental, la Unión Europea considera que la educación y la cultura son esenciales para desarrollar una Europa más inclusiva, más cohesionada y más competitiva. Para afianzar estas líneas ha dejado plasmada su visión en un documento para que en el año 2025 exista una *European Education Area* donde estudiar y aprender en otro país de la unión sea la norma y donde toda la población pueda acceder a una educación de alta calidad con independencia de su origen social [Com17]. Aunque parece que esta *European Education Area* continua la labor desarrollada vía el Espacio Europeo de Educación Superior y no es rupturista con el modelo actual.

[1] `https://www.boe.es/eli/es/lo/2001/12/21/6/con`. Para una consulta general sobre legislación universitaria, la Biblioteca Jurídica Digital mantiene actualizado el Código de Universidades en la siguiente url: `https://boe.es/legislacion/codigos/codigo.php?id=133&modo=1&nota=0&tab=2`

1.1.1 Espacio Europeo de Educación Superior

En 1999, la declaración de Bolonia, estableció la construcción del Espacio Europeo de Educación Superior, EEES [2], y su puesta en marcha para el año 2010, con la visión de armonizar los sistemas educativos de los distintos países de la Unión Europea. En la actualidad los límites del EEES se extienden más allá de las fronteras europeas y comprenden 48 países, incluyendo la práctica totalidad del continente Europeo junto con Rusia y alguno de sus países limítrofes.

El EEES establece varios mecanismos para alcanzar sus objetivos. Por un lado, homogeneiza las métricas sobre el trabajo de los alumnos mediante el *European Credit Transfer System*, ECTS [Com16a]. Un curso académico equivale a 60 créditos ECTS y cada uno de estos créditos se corresponde con entre 25 y 30 horas de trabajo. En el caso concreto de España, un crédito ECTS equivale a 25 horas de trabajo concretamente. Los créditos están enfocados al aprendizaje e incluyen todas sus actividades, ya sean clases magistrales, seminarios, estudio personal, realización de trabajos/proyectos y/o exámenes. Además de permitir la movilidad entre instituciones, los créditos ECTS también permiten su transferencia entre programas siempre que los responsables de las instituciones y/o los programas así lo acuerden.

El segundo mecanismo principal del EEES es la estructura grado—postgrado. EL ciclo de grado ofrece una formación general mientras que la orientación del postgrado es hacia la especialización. Si bien es verdad que ambos están enfocados a cubrir las necesidades laborales de la sociedad.

El último mecanismo es un sistema de evaluación basado en acreditaciones que tanto de manera interna como externa debe garantizar la calidad de la formación y el seguimiento por parte de los centros de las directivas del EEES.

La universidad de Zaragoza implantó el EEES durante la primera década de este siglo y en la actualidad, la práctica totalidad de los planes antiguos ya han sido extinguidos.

1.1.2 La Universidad de Zaragoza

El germen de la Universidad de Zaragoza fue un estudio eclesiástico de artes del siglo XII y fue el Papa Sixto IV el día 13 de diciembre de 1474 quien la designo como *Universitas magistrorum*. Pero su despegue como gran institución no se produjo hasta el año 1542 cuando el emperador Carlos V elevó su rango hasta "Universidad general de todas las ciencias" a instancias de una solicitud de los síndicos de Zaragoza. Poco después, en 1554, el papa Julio III ratificó la condición de *Studium Generale* y bajo el auspicio de Pedro Cerbuna, quien en 1582 sufragó los gastos para abrir la nueva universidad, la institución fue inaugurada el 24 de mayo de 1583.

Las primeras facultades fueron Teología, Cánones, Leyes, Medicina y Artes y establecieron el prestigio de la institución que fue languideciendo a lo largo del siglo XVIII al no añadir los estudios en boga por aquel tiempo tales como Botánica o Matemáticas. Cuando el ministro Caballero redujo el número de universidades en España, la de Zaragoza no se vio afectada, aunque como el resto de universidades tuvo que adoptar los planes de estudio de la Universidad de Salamanca.

A lo largo del Siglo XIX varió el número de facultades, perdiendo para luego recuperar por ejemplo la facultad de Medicina por ejemplo, y no fue hasta bien entrado el siglo XX, cuando en 1921 se aprobaron unos estatutos autónomos que permitieron la impartición del doctorado ya creación de los primeros cursos de verano en España en la localidad oscense de Jaca.

La transformación del estado de las autonomías en España, durante los años 80 del siglo pasado,

[2] `http://www.ehea.info/`

volvió a reorganizar la universidad y los colegios universitarios que se habían formado en años anteriores pasaron a formar parte de las universidades de sus respectivas comunidades autónomas. Además comenzó la impartición de diplomaturas e ingenierías técnicas.

En la actualidad, la Universidad de Zaragoza cuenta con casi 40000 miembros repartidos en 23 centros, 18 propios y 5 adscritos, que jalonan la geografía aragonesa, maneja un presupuesto de más de 284 millones de € e imparte más de 50 grados adaptados al EEES. Dentro del *Academic Ranking of World Universities*, o índice de Shanghái, la Universidad de Zaragoza se encuentra entre la 501 y la 600 mejor del mundo. Es reseñable que en la titulación de Ciencias de la Computación, o Ingeniería Informática, impartida en la Escuela de Ingeniería y Arquitectura el puesto sube entre la 301 y la 400 mejor del mundo y el puesto decimotercero a nivel nacional [3]

1.1.3 La Escuela de Ingeniería y Arquitectura

En el año 1974 fue fundada la Escuela Técnica Superior de Ingenieros Industriales de Zaragoza. A finales de los años 80 del siglo pasado, se trasladó al barrio del Actur y cambio su nombre por Centro Politécnico Superior para iniciar los estudios de ingeniería de telecomunicaciones, ingeniería informática, ingeniería química y, finalmente en el curso 2008-2009, arquitectura. Poco tiempo después surgió la Escuela de Ingeniería y Arquitectura, EINA, fruto de la unión del Centro Politécnico Superior con la Escuela Universitaria de Ingeniería Técnica Industrial. La EINA aglutina más del 80% de los estudios de Ingeniería y Arquitectura de Aragón con sus casi 4000 alumnos de grado y representa un porcentaje similar de la actividad de I+D+i en el campo tecnológico dentro de la comunidad aragonesa [4].

En las aulas de la EINA se pueden estudiar más de 36 titulaciones diferentes entre estudios de grado, másteres oficiales, estudios propios y titulaciones en vías de extinción. Es destacable el alto número de institutos universitarios involucrados con la EINA como son el I3A, CIRCE, INA, ICMA, IUCA, IUMA,... que redundan en la calidad de la investigación y enseñanza del centro.

1.1.4 El Departamento de Informática e Ingeniería de Sistemas (DIIS)

El Departamento de Informática e Ingeniería de Sistemas, DIIS, es uno de los departamentos más grandes que forman la Universidad de Zaragoza. Su fundación se remonta a la primavera del año 1995 y desde el comienzo ha estado compuesto por 4 áreas de conocimiento:

- Arquitectura y Tecnología de Computadores
- Ciencias de Computación e Inteligencia Artificial
- Ingeniería de Sistemas y Automática
- Lenguajes y Sistemas Informáticos

La plantilla del DIIS está estabilizada alrededor de 160 personas: 106 profesores, 39 becarios y 13 miembros del personal de administración y servicios, a fecha de noviembre de 2019. Dicho profesorado imparte docencia en los siguientes centros:

- Escuela de Ingeniería y Arquitectura, Zaragoza (EINA)
- Escuela Politécnica Superior, Huesca (EPSH)

[3] Datos provenientes de `https://www.unizar.es/institucion/conoce-la-universidad/datos-basicos` y `https://www.unizar.es/rankings` actualizados por última vez el 25 de junio de 2019.

[4] Datos provenientes de `https://academico.unizar.es/grado-y-master/estadisticas`

- Escuela Universitaria de Estudios Empresariales, Huesca (EUEEH)
- Escuela Universitaria de Estudios Empresariales, Zaragoza (EUEEZ)
- Facultad de Ciencias Sociales y del Trabajo, Zaragoza (FCSTZ)
- Escuela Universitaria Politécnica de Teruel, Teruel (EUPT)
- Facultad de Ciencias, Zaragoza (FC)
- Facultad de Ciencias de la Salud y el Deporte, Huesca (FCSDH)
- Facultad de Educación, Zaragoza (FEZ)
- Facultad de Ciencias Sociales y Humanas, Teruel (FHCS)

Dentro del DIIS se realiza una gran labor investigadora a través del programa de doctorado en ”Ingeniería de Sistemas e Informática” que obtuvo la Mención de Calidad por el Ministerio de Educación y Ciencia en la resolución de fecha 28 de mayo de 2003 (referencia MCD2003-00466). Este programa aporta una visión pluridisciplinar gracias a las diferentes temáticas que estudian los distintos grupos de investigación del departamento:

- Affectivelab (Affectivelab)
- Computer Science for Complex System Modelling (COS2MOS)
- Grupo de I+D en Computación Distribuida (DisCO)
- Grupo de Arquitectura de Computadores de la Unizar (gaZ)
- Group of Discrete Event Systems Engineering (GISED)
- Graphics and Imaging Lab (GraphImag.Lab)
- Grupo de Sistemas de Información Avanzados (IAAA)
- Intelligent Networks and Information Technologies (iNiT)
- Interactive Systems, Adaptivity, Autonomy and Cognition (ISAAC Lab)
- Noesis (Noesis)
- Grupo de Robótica, Percepción y Tiempo Real (ROPERT) y
- Grupo de Sistemas de Información Distribuidos (SID)

1.1.5 El Área de Arquitectura y Tecnología de Computadores

Como parte de su labor docente, dentro del DIIS, el área de Arquitectura y Tecnología de Computadores es responsable de la impartición de la asignatura de Administración de Sistema así como del resto de asignaturas que tienen que ven con el diseño de computadores y sistemas, incluyendo el cómputo y las comunicaciones, abarcando desde supercomputadores y centros de datos a sistemas embebidos. Sus materias abarca entre otros el diseño lógico, microprocesadores, sistemas operativos, redes de computadores, seguridad, modelos de programación, paralelismo, servicios de alto nivel,... Un porcentaje muy grande de los profesores de esta área desarrollan su labor investigadora dentro del Grupo de Arquitectura de Computadores de Zaragoza, gaZ, que cuenta con más de 20 miembros entre personal funcionario, laboral y becarios.

1.2 Contexto Curricular de Administración de Sistemas

Administración de Sistemas pertenece al bloque común del segundo curso de los estudios de Grado en Ingeniería Informática impartidos en la Escuela de Ingeniería y Arquitectura en Zaragoza y en la Escuela Universitaria Politécnica de Teruel. Como ya se ha mencionado, el área de Arquitectura y Tecnología de Computadores del Departamento de Informática e Ingeniería de Sistemas se encarga de su docencia en ambos centros.

1.2.1 La Titulación de Grado en Ingeniería Informática

La sociedad del conocimiento actual se basa en generar, compartir y hacer disponible a todos sus miembros toda la información y conocimiento que permita una mejora de la condición humana. Esta ingente tarea requiere de la automatización del procesado de la información ya que sería inviable hacerlo de manera manual. El ingeniero alemán Karl Steinbuch en su ensayo "Informatik: Automatische Informationsverarbeitung" ya planteó este tratamiento automatizado de la información en el año 1957 y acuño el término informatik del que deriva el español informática a través del francés informatique. Por lo tanto la sociedad del conocimiento demanda y se basa en profesionales que puedan ayudarle a alcanzar su loable objetivo y dentro de estos profesionales, los graduados en Ingeniería Informática son un pilar fundamental ya que pueden atender las demandas de la sociedad asociadas a las tecnologías de la información y las comunicaciones. Los datos del Libro Blanco para el título de grado en Ingeniería Informática corroboran la afirmación anterior ya que indican que en Europa eran necesarios casi 4 millones de egresados en Ingeniería Informática [Eva05].

Desde un punto de vista histórico, fue la Conferencia de Decanos y Directores de Centros Universitarios de Informática, CODDI, quien acordó acuñar el título de grado en Ingeniería Informática en octubre de 2008 como sucesor de la fusión de las antiguas titulaciones de Ingeniero en Informática, Ingeniero Técnico en Informática de Gestión e Ingeniero Técnico en Informática de Sistemas.

En la Universidad de Zaragoza se imparte dicho grado de Ingeniería Informática en 2 campus:

- Escuela de Ingeniería y Arquitectura, Zaragoza
- Escuela Universitaria Politécnica, Teruel

En ambos centros el perfil recomendado de acceso es la modalidad de Ciencia y Tecnología en bachillerato junto con una buena formación matemática y conocimiento de ingles. Además se recomienda una notable capacidad y buena disposición tanto para el trabajo individual como en grupo. La organización del grado es tal que no requiere conocimientos informáticos previos y solo recomienda una atracción por las tecnologías de la información y las comunicaciones.

Organización del Grado

La consecución del grado en Ingeniería Informática requiere completar 240 créditos ECTS distribuidos en 4 cursos académicos. Durante todos los años, cada estudiante debería cursar 10 asignaturas entre los 2 semestres que sumaran 60 créditos por año a repartir entre un poco más de 30 semanas durante el curso. En concreto, los estudios se distribuyen de acuerdo al siguiente modo:

- 10 asignaturas de 6 créditos de formación básica de las que nueve se cusan en el primer año y la décima en el primer cuatrimestre del segundo año

- 17 asignaturas de 6 créditos de formación obligatoria que constituyen el núcleo de formación específica de los estudios. Catorce de ellas se cursan en los cuatrimestres tercero, cuarto y quinto de los estudios
- 8 asignaturas de 6 créditos de formación en una especialidad que se cursan en los tres últimos cuatrimestres de los estudios
- 16 créditos de formación optativa y 2 créditos de inglés
- 12 créditos del trabajo fin de grado

Este plan de estudios otorga un peso fundamental a la formación obligatoria, incluyendo las asignaturas de especialidad, como puede verse en el cuadro 1.1. Al terminar los estudios, los estudiantes validan sus conocimientos realizando un Trabajo Fin de Grado que consta de 12 créditos.

Cuadro 1.1: Distribución de asignaturas por tipo de asignatura

Tipo de asignatura	Créditos
Formación Básica	60
Obligatorias	152
Optativas	16
Prácticas externas (si se incluyen)	—
Trabajo fin de grado	12
Total	240

El plan de estudios de grado en Ingeniería Informática por la Universidad de Zaragoza permite a sus egresados ejercer la profesión de Ingeniero Técnico en Informática tal y como detalla el Anexo II del Acuerdo del Consejo de Universidades por el que se establecen recomendaciones para la propuesta por las universidades de memorias de solicitud de títulos oficiales en los ámbitos de la Ingeniería Informática, Ingeniería Técnica Informática e Ingeniería Química (BOE 12977/2009, Número 187 de 4 de agosto, en adelante “fichas del Consejo de Universidades”).

Las fichas del Consejo de Universidades definen 5 tecnologías específicas que han dado lugar a las 5 especialidades del grado:

- Ingeniería de Software (CEIS)
- Ingeniería de Computadores (CEIC)
- Computación (CEC)
- Sistemas de Información (CESI)
- Tecnologías de la Información (CETI)

En el Campus de Teruel se oferta las especialidades en Sistemas de Información y Tecnologías de la Información, mientras que en el Campus Río Ebro de Zaragoza se ofertan todas las especialidades: Computación, Ingeniería de Computadores, Ingeniería del Software, Sistemas de Información y Tecnologías de la Información. En la comunidad autónoma de Aragón, existe el Colegio Profesional de Ingenieros Técnicos en Informática de Aragón al que pueden colegiarse todos los egresados en cualquiera de las 5 especialidades.

Planificación y Especialidades del Grado Las 5 especialidades mencionadas anteriormente comparten las asignaturas básicas y comunes detalladas en el cuadro 1.2. Como puede apreciarse, en los 2 primeros años y el primer cuatrimestre de 3º todas las asignaturas son comunes o básicas. Una vez completado este periodo, los alumnos disponen de unos conocimientos y experiencias suficientes para elegir de una manera cuidadosa su especialidad. Como dato curioso, la asignatura Administración de Sistemas que presentada en este proyecto docente es cursada por los alumnos en el momento de realizar la elección de especialidad, en el semestre de primavera de segundo curso.

Cuadro 1.2: Relación de Asignaturas Básicas y Comunes del grado en Ingeniería Informática

Curso	Semestre Otoño	Semestre Primavera
1º	Fundamentos de administración de empresas	Arquitectura y Organización de Computadores 1
	Introducción a los computadores	Física y Electrónica
	Matemáticas I	Estadística
	Matemáticas II	Matemática discreta
	Programación I	*Programación II*
2º	*Programación de sistemas concurrentes y distribuidos*	*Arquitectura y Organización de Computadores 2*
	Sistemas Operativos	*Administración de Sistemas*
	Redes de Computadores	*Interacción Persona Ordenador*
	Teoría de la computación	*Tecnología de programación*
	Estructura de datos y algoritmos	*Bases de datos*
3º	*Proyecto Hardware*	*Proyecto Software*
	Sistemas Distribuidos	
	Ingeniería del Software	
	Inteligencia Artificial	
	Sistemas de Información	

[a] Las asignaturas en itálica son comunes y el resto básicas

Objetivos

El objetivo fundamental del grado en Ingeniería Informática es formar profesionales capaces de de analizar, concebir, redactar, organizar, planificar y ejecutar proyectos en el ámbito de la informática que tengan por objeto la concepción, el desarrollo o la explotación de equipos, sistemas, servicios y aplicaciones informáticas y de dirigir las actividades objeto de los proyectos citados.

La consecución del objetivo fundamental supone el cumplimiento de los objetivos general especificados en la memoria de verificación del grado [Zar10b]:

- Formar profesionales capaces de concebir, redactar, organizar, planificar, desarrollar y firmar proyectos en el ámbito de la ingeniería en informática que tengan por objeto, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 del Anexo II de las fichas del Consejo de Universidades, la concepción, el desarrollo o la explotación de sistemas, servicios y aplicaciones informáticas.
- Formar profesionales capaces de dirigir las actividades objeto de los proyectos del ámbito de la informática de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 del Anexo II de las fichas del Consejo de Universidades.

- Capacidad para concebir, desarrollar y mantener sistemas, servicios y aplicaciones informáticas empleando los métodos de la ingeniería del software como instrumento para asegurar su calidad, de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 del mencionado anexo.
- Formar profesionales capaces de concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes de acuerdo con los conocimientos adquiridos según lo establecido en el apartado 5 del Anexo II de las fichas del Consejo de Universidades.
- Formar profesionales con el conocimiento de las materias básicas y tecnológicas, que capaciten para el aprendizaje y desarrollo de nuevos métodos y tecnologías, así como de una gran versatilidad para adaptarse a nuevas situaciones.

1.2.2 La Materia de Administración de Sistemas

La materia de Administración de Sistemas pertenece al cuarto semestre del grado en Ingeniería Informática de la Universidad de Zaragoza como se aprecia en el cuadro 1.2. Administración de Sistemas es una asignatura básica que completa los contenidos de las materias Sistemas Operativos (semestre 3º) y Redes de computadores (semestre 3º). Además requiere de los conocimientos adquiridos en Arquitectura y Organización de Computadores I (semestre 2º) y de Programación 1 (semestre 1º). El carácter práctico de la asignatura ayuda a los alumnos a alcanzar un conocimiento razonable del funcionamiento de varios servicios fundamentales de un sistema operativo y de una red de computadores de tamaño pequeño o intermedio.

A su vez, Administración de Sistemas es prerrequesito en asignaturas de todas las especialidades del grado. En concreto pueden citarse: Administración de Sistemas 2 (semestre 6º), Seguridad Informática (semestre 7º), Centros de datos (semestre 7º), Diseño y Administración de redes (semestre 7º), Garantía y Seguridad (semestre 8º) o Sistemas de información (semestre 5º) como materias que aprovechan los conocimientos adquiridos en Administración de Sistemas.

La temporización de la asignatura dentro del grado adecuada ya que no se solapa con sus requisitos ni con sus dependencias, lo que simplifica el aprendizaje por parte del alumnado. A modo de curiosidad en la Escuela Universitaria Politécnica de Teruel, la materia se denomina de manera ligeramente distinta pasando a llamarse Administración de Sistemas 1.

Contexto Internacional

El *Computing Curricula 2005* incluye la administración de sistemas dentro de los 5 grados que plantea: ingeniería de computadores, computación, sistemas de información, tecnologías de la información e ingeniería del software [Com05] y el Computing Curricula 2020[5] refiere a CC2005 para la organización de los grados. Dentro del *Computer Engineering Curricula 2016* [Com16b], la asignatura propuesta se englobaría dentro de las siguientes áreas que plantea el CE2016: CE-NWK (Computer Networks), CE-SEC(Information Security), CE-SPE (Systems and Project Engineering), CE-SRM(System Resource Management). Por lo que se puede concluir que la Administración de Sistemases una asignatura relevante dentro de los estudios de informática en general.

[5] https://www.cc2020.net/

1.3 Mecanismos de Evaluación en la Universidad de Zaragoza

La Universidad de Zaragoza dispone de mecanismos de evaluación tanto de alumnado como del profesorado.

Para garantizar el cumplimiento y asegurar una evaluación correcta de los reglamentos de evaluación de la Universidad de Zaragoza, la Escuela de Ingeniería y Arquitectura dispone de varias comisiones: Comisiones de Garantía de la Calidad de la Docencia, Comisiones de Evaluación de la Calidad de las Titulaciones, Comisiones Académicas de las Titulaciones y Comisión de Control y Evaluación de la Docencia, todas ellas están formadas por profesores de múltiples departamentos y alumnos de distintos grados. Junto con los coordinadores de titulación velan por el aseguramiento de la calidad de la docencia y el cumplimiento de la "Normativa del Sistema Interno de Gestión de la Calidad de la Docencia de la Escuela de Ingeniería y Arquitectura. Comisiones Delegadas EINA" acordado el 28 de junio de 2012.

1.4 Evaluación del Alumnado

El "Reglamento de Normas de Evaluación del Aprendizaje de la Universidad de Zaragoza", acuerdo de 22 de diciembre de 2010 define los contenidos mínimos de las normas de evaluación a aplicar al alumnado que son: régimen de convocatorias, programación de las pruebas de evaluación, nombramiento de los tribunales de evaluación y procedimiento de revisión de las calificaciones.

A modo de resumen, las normas definen que todas las asignaturas de los planes de estudio de los títulos oficiales de grado y máster dispondrán cada curso de 2 convocatorias de evaluación siendo los centros responsables de aprobar el calendario de las pruebas de evaluación de cada curso académico. Los resultados de la evaluación de cada materia se calificarán de 0 a 10, con expresión de un decimal, añadiendo una calificación cualitativa según la siguiente escala: 0-4,9: Suspenso (SS); 5,0-6,9: Aprobado (AP); 7,0-8,9: Notable (NT); 9,0-10: Sobresaliente (SB). Además la normativa contempla la evaluación por compensación curricular de las asignaturas de carácter obligatorio o troncal. Dicha evaluación tiene en cuenta tanto la calificación de la asignatura pendiente, CAP, como la nota media ponderada, NM, a créditos de todas las asignaturas o materias obligatorias aprobadas de la titulación, redondeada a 2 decimales. Dependiendo de los créditos de la asignatura, 6 créditos en el caso de Administración de Sistemas por ejemplo, para obtener un aprobado por compensación será necesario que $0.65 \times NM + 0.35 \times CAP \geq 5.0$.

Siendo las directrices del EEES, la normativa de evaluación admite un mecanismo de evaluación continua que permita al alumno obtener la máxima calificación. Aunque es derecho del alumno poder hacer una prueba global de evaluación y que la calificación final sea la máxima calificación entre la continua y la global.

1.5 Evaluación del Profesorado

Al igual que para el alumnado, los profesores disponen de varias herramientas para obtener su evaluación. El objetivo fundamental fijado por la normativa anteriormente citada es la implantación de planes que permitan una mejora continua de la docencia y de los rendimientos académicos. En concreto, el artículo 109.4 de los Estatutos de la Universidad de Zaragoza detalla que "la evaluación de la actividad docente se configura como un derecho del personal docente, una garantía de los estudiantes y un deber de la Universidad a los efectos legales que procedan".

La herramienta principal para medir la calidad de la docencia son las encuestas realizadas por los alumnos al final de cada cuatrimestre. Con los resultados de las mismas, la Comisión de Control y Evaluación de la Docencia evalúa a cada docente otorgándole una calificación positiva-destacada, positiva o negativa de acuerdo a los criterios establecidos en la normativa [Zar16]. Para comprender en más profundidad los resultados y ver aquellos aspectos que los alumnos consideran mejores y peores, existe una plataforma de nombre ATENEA donde el profesorado puede consultar los resultados desglosados por múltiples criterios tales como asignatura o preguntas específicas.

Lamentablemente son pocos los alumnos que rellenan las encuestas con lo que para muchas asignaturas es difícil obtener unos resultados representativos. Independientemente de los mismos, el profesorado debe comprometerse con la calidad de su docencia y mejorarla continuamente por ejemplo mediante la realización de cursos de formación en el Instituto de Ciencias de la Educación de la Universidad de Zaragoza, la realización de proyectos de innovación docente con otros compañeros, e incluso, aprendiendo de las técnicas que utilicen bien sea en el aula, en libros o en plataformas de enseñanza “online”.

Capítulo 2

Administración de Sistemas: Diseño Curricular

En la subsección 1.2.2 se han visto los aspectos más formales de la asignatura de Administración de Sistemas y como esta resulta un complemento fundamental de las asignaturas de Sistemas Operativos y Redes de Computadores ya que desarrolla desde un punto de vista más práctico sus contenidos.

Este capítulo extiende la información de la subsección anterior respecto al contenido de la materia Administración de Sistemas y presenta unos breves apuntes históricos junto con las competencias y los resultados de aprendizaje que se pretende que alcance el alumnado una vez completada la asignatura.

2.1 Presentación de la materia

El glosario del Computing Curricula 2005 define la Administración de Sistemas como:

> **Systems Administration** –– The field of study which deals with the management of computing and communications resources, including networks, databases, operating systems, applications, and Web delivery. The management issues include installation, configuration, operation, and maintenance.

En el caso concreto de este proyecto docente se plantea una asignatura muy práctica con una fuerte carga en el aprendizaje de herramientas UNIX, diseño y programación de scripts y gestión de computadores y de redes que cubra la gestión tanto de los elementos de cómputo como de la infrastructura de red siempre de una manera segura y automatizada cuando sea posible. Debido a las limitaciones temporales, se cubrirán en mucha menor medida los servicios de alto nivel y las bases de datos ya que son estudiadas en asignaturas posteriores. Además, al tratarse de un grado en Ingeniería Informática, y aprovechando los conocimientos adquiridos en Sistemas Operativos y Redes de Computadores, esta propuesta intenta que los alumnos no solo sean capaces de manejar con soltura herramientas sino que además *visualizen* las tareas que realiza el sistema operativo cuando las invocan para alcanzar una comprensión mayor de los sistemas informáticos. Como ejemplo, la primera linea de los scripts, *shebang*, tiene la forma `#!<interprete>`

`<argumento-opcional>` para que la llamada al sistema `execve()`[1] reemplace el programa que este ejecutando por la ejecución del primer argumento de la llamada. En el caso de los scripts, mediante el *shebang* identificara su interprete y lo invocará añadiendo el propio script como primer argumento.

2.2 Revisión Histórica

Esta revisión histórica esta basada en el anexo: "A Brief History of System Administration" del libro *UNIX and Linux System Administration Handbook* [Nem+18].

Los comienzos de la Administración de Sistemas se remontan a los años 50 del siglo pasado con la aparición de los primeros computadores comerciales y está muy ligada a la historia de los sistemas operativos. En ese momento, los operadores de los computadores eran sus primeros administradores y sufrían problemas similares a los administradores actuales como la incompatibilidad entre sistemas. A modo de anécdota, cuando IBM anunció la aparición del modelo 704 que reemplazaba al modelo 701 con el que era incompatible, invitó a los operadores de los 701 existentes a reuniones para compartir sus problemas y soluciones y así facilitar su labor. El grupo entonces fundado, conocido como SHARE, sigue en funcionamiento ayudando a todos los clientes de IBM a compartir información de como gestionar sus sistemas.

La aparición de las computadores de tiempo compartido y el nacimiento del sistema operativo UNIX entre los años 1969 y 1973 marcan un punto de inflexión en la Administración de Sistemas. La invención de Ken Thompson, Rudd Canaday y Dennis Ritchie definía una filosofía basada en 3 principios:

- Escribir programas que hagan una cosa y la hagan bien.
- Escribir programas que trabajen juntos.
- Escribir programas que gestionen cadenas de texto como si fueran un interfaz universal.

Estos principios todavía vigentes ilustran perfectamente la filosofía UNIX y deben inculcarse desde el principio a los alumnos para ayudarles a comprender las partes más arduas.

La sentencia por monopolio a AT&T a principio de los años 80 no permitía a la empresa vender UNIX sino que debía licenciar la tecnología a otras entidades. La universidad de California en Berkeley recibió una de las cintas magnéticas con UNIX y el profesor Robert Fabry comenzó el desarrolló de *x*BSD, Berkeley Software Distribution, que luego fue el germen de SunOS de Sun Microsystems.

Unos pocos años después de la sentencia, multitud de universidades americanas disponían de UNIX y empezaron a emplear administradores de sistemas. El hecho de que muchos estudiantes quisieran aprender como funcionaban permitió la creación de comunidades organizadas de donde surgieron multitud de herramientas como `sudo` o manuales de buenas prácticas.

El número de administradores seguía en aumento y USENIX organizó la primera conferencia sobre administración de sistemas en el año 1987 y en 1989 apareció la primera edición de la "biblia" de la Administración de Sistemas [Nem+18]. En los años 90 del siglo pasado, surgió la eclosión tanto de Linux como de Windows lo que diversificó las necesidades de los administradores de sistemas. Además el uso de redes de computadores fue ubicuo aumentando el espectro de conocimientos requeridos por su parte.

[1] `http://man7.org/linux/man-pages/man2/execve.2.html`

Figura 2.1: Ken Thompson y Dennis Ritchie administrando un PDP-11

La aparición de los servicios instalados en grandes centros de datos y la adopción de tecnologías como la virtualización y la movilidad aseguran un prometedor futuro a los administradores de sistemas, la oficina de estadísticas laborales de los Estados Unidos estima que de 2018 a 2028 el número de administradores de sistema aumentará un 5% [Lab18], a la vez que exige a sus formadores una revisión continua del material para asegurar su puesta al día.

2.3 Competencias

A continuación de detallan las competencias de la asignatura divididas en 3 bloques: Básicas y Generales, Transversales, y Específicas[2].

- Competencias Básicas y Generales:
  - CB1 - Que los estudiantes hayan demostrado poseer y comprender conocimientos en un área de estudio que parte de la base de la educación secundaria general, y se suele encontrar a un nivel que, si bien se apoya en libros de texto avanzados, incluye también algunos aspectos que implican conocimientos procedentes de la vanguardia de su campo de estudio.
  - CB2 - Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas dentro de su área de estudio.
- Competencias Transversales:
  - CT04 - Capacidad para resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico.
  - CT10 - Capacidad para aprender de forma continuada y desarrollar estrategias de aprendizaje autónomo.
  - CT11 - Capacidad para aplicar las tecnologías de la información y las comunicaciones en la Ingeniería.

[2]Se indica al comienzo de cada competencia su identificador.

- Competencias Específicas:
  - CE01 - Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
  - CE02 - Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
  - CE04 - Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
  - CE05 - Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
  - CE10 - Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e implementar aplicaciones basadas en sus servicios.
  - CE11 - Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet y diseñar e implementar aplicaciones basadas en ellas.
  - CE14 - Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.
  - CE18 - Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Como se aprecia en la lista anterior, el número de competencias de la materia Administración de Sistemas es relativamente grande debido a la multitud de aspectos distintos que desarrolla.

2.4 Resultados de aprendizaje

Al terminar esta asignatura se espera del alumno que sea diestro en la gestión de múltiples computadores que trabajen de manera autónoma o conectados mediante una red. Su manejo del sistema será avanzado y será capaz de ocuparse de todo su ciclo de vida, desde la compra y posterior puesta en marcha hasta su reemplazo siendo capaz de migrar todas las aplicaciones y servicios que fuera necesario. En concreto, la memoria de verificación de la titulación define los siguientes resultados de aprendizaje [Zar10a]:

- Tiene un conocimiento de la función de un sistema operativo, de sus niveles de uso y gestión, y de los objetos comunes que proporciona.
- Entiende y sabe utilizar los servicios más importantes de un sistema operativo como usuario y mediante el interfaz de programación con llamadas al sistema.
- Conoce y aplica las características, funcionalidades y estructura de las redes de computadores e Internet.
- Sabe diseñar e implementar aplicaciones que utilicen comunicaciones en red de forma básica.
- Tiene conocimientos básicos para administrar y mantener sistemas, redes y aplicaciones informáticas.
- Conoce fundamentos básicos de la seguridad en los sistemas operativos y redes de computadores.

2.5 Metodología

La práctica totalidad de las interacciones entre profesores y alumnos se pueden clasificar en 4 tipologías: la clase de teoría, la clase de problemas, las prácticas de laboratorio y las tutorías. A continuación se detalla brevemente que metodologías se pueden emplear en los distintos tipos para intentar mejorar en lo posible el proceso de aprendizaje y la adquisición de competencias.

2.5.1 La Clase de Teoría

En la clase de teoría el profesor presenta la materia a tratar al alumnado siguiendo un discurso. Resulta muy útil para presentar ideas a un conjunto grande de alumnos y puede permitir un avance muy rápido si el alumno realiza algún tipo de trabajo previo para ayudar con el aprendizaje. Su mayor limitación radica en tratarse de una actividad prácticamente unidireccional que requiere una continua atención del profesor para intentar que los alumnos sigan el discurso. Para ello suele resultar útil romper la evolución lineal de la clase con interjecciones que estimulen la reflexión del alumno y permitan evaluar su asimilación del contenido. Dicha evaluación puede llevarse a cabo mediante la realización de preguntas y el posterior comentario de todo el aula de las respuestas. Para estimular la participación suele resultar útil hacer que la respuesta sea grupal, normalmente por parejas, ya que así los alumnos tienen también que explicarse entre si la materia y pueden auto-evaluar su grado de compresión. Cuando los alumnos son reacios a participar de la discusión, al autor de esta memoria le ha resultado muy útil establecer un orden en los alumnos. Al primer alumno se le pide por favor que conteste a la pregunta y que además indique quien será la siguiente persona en contestar entre aquellas que no hayan contestado todavía durante la clase. Este proceso se repite hasta que ya hayan participado todos los alumnos. Con esta sencilla técnica se ha observado que a las pocas semanas de su puesta en práctica, los alumnos se muestran muy participativos y al preguntarles si desean eliminarla del aula se muestran reacios a hacerlo.

2.5.2 La Clase de Problemas

El aprendizaje basado en problemas ha demostrado su gran utilidad para la enseñanza en áreas técnicas como es el caso de Administración de Sistemas. Es especialmente útil cuando los alumnos se involucran activamente en su resolución. En la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza las clases de problemas se imparten con la mitad de los alumnos del grupo facilitando un seguimiento más individualizado por parte del profesor hacia el alumnado.

En general las clases del problemas mantienen una estructura muy similar a la narrativa clásica y sus 3 partes: introducción, nudo y desenlace. En la introducción se plantea el problema, esto se puede hacer en el mismo aula o con anterioridad para que se disponga de más tiempo estudiando el problema. El nudo se podría emparentar con la fase en la que los alumnos comentan la solución ayudados por el profesor. Este momento puede ser muy útil para que los alumnos sean completamente protagonistas en el aula y compartan entre si sus soluciones y el profesor sirva de conductor de la discusión y de apoyo a cuestiones concretas. Finalmente en el desenlace se alcanza la solución correcta y el profesor se intenta asegurar de que todos los alumnos la entiendan lo mejor posible. Según la dificultad de los comandos requeridos para resolver los problemas, en algunas sesiones se adopta el modelo de *flipped classroom* para que los alumnos estudien en casa los comandos a emplear junto con sus opciones y en la clase los apliquen y resuelvan sus dudas con el profesor.

Debido al carácter práctico de la materia de Administración de Sistemas y a que buena parte de los problemas de la asignatura deben resolverse con scripts, resulta útil que el profesor se acom-

pañe de un computador y adopte un rol más pasivo para intentar incentivar que sean los alumnos los que tomen la iniciativa proponiendo su solución y apliquen la técnica de la evaluación cruzada en la que los alumnos se corrigen entre si. Una de las ventajas de esta aproximación es que los alumnos están más cerca de la realidad a la que se enfrentan en las clases del laboratorio y pueden ver sus errores en tiempo real algo que no tiene porque suceder al utilizar la pizarra. De hecho, cuando la conexión a internet en el Ed. Ada Byron sea fiable se emplearán soluciones como tmate, `https://tmate.io/` o flootty, `https://github.com/Floobits/flootty`, para compartir la terminal entre el profesor y los alumnos. Otra propuesta interesante que se esta estudiando para ayudar a los alumnos en su aprendizaje son los juegos online de plataformas como OverTheWire, `https://overthewire.org/wargames/`, para aprender técnicas de seguridad y mejorar en los conocimientos del shell. Para esta asignatura, se recomienda Bandit, `https://overthewire.org/wargames/bandit/`, ya que no requiere de conocimientos previos al comenzar y cubre muchos aspectos vistos en esta asignatura [3].

2.5.3 Las Prácticas de Laboratorio

Parte del trabajo real de un Ingeniero en Informática se desarrolla empleando un computador y por tanto es el laboratorio la parte más difícil a la que se enfrenta el alumno dentro de la asignatura.

La clase de laboratorio suele consistir en la resolución de algún caso práctico descrito en un guión con herramientas reales. Este guión puede incluir apartados a realizar antes de asistir al laboratorio y/o al terminar el mismo. El autor de esta memoria tuvo una experiencia muy positiva preparando unas preguntas de auto-evaluación que los alumnos debían realizar al terminar cada sesión de laboratorio. Cuestiones ligeramente más difícil que la propia sesión de laboratorio ayudaban a los alumnos a ver su nivel y parecía que estimulaban el estudio.

2.5.4 Las Tutorías

Las tutorías permiten responder de manera individualizada a los problemas de aprendizaje que le surgen a los alumnos. Una tutoría puede realizarse de manera física en un despacho o sala de reuniones o de manera telemática, la mayor parte de las veces por correo electrónico.

Al interactuar con pocos alumnos, normalmente 1 o 2, resulta fácil para el profesor comprender los problemas y las dificultades que encuentran en la materia. Además haciendo un seguimiento de sus preguntas es fácil ir adecuando las clases a sus necesidad, por ejemplo seleccionado un tipo de problemas en los que se haya visto más incidencia de dificultades lo que mejora ostensible la actividad facilitadora del profesor. Un aspecto trasversal de las tutorías es la ganancia de confianza por parte del alumno a la hora de plantear sus respuestas y/o preguntas en público en clase, lo que hace aumentar su aprendizaje.

2.6 Sistema de Evaluación

Para cumplir con la normativa de evaluación de la Escuela de Ingeniería y Arquitectura, la evaluación de Administración de Sistemas seguirá el procedimiento de evaluación global y constará

[3] Agradecimientos para el antiguo alumno de esta asignatura Javier Corbalán Colino por enseñar estos juegos al autor de este proyecto docente.

de 3 partes[4]:

- **Examen escrito** en el que se deberán resolver problemas, responder preguntas conceptuales o resolver algún ejercicio. Es necesario una calificación mínima de 5.0 puntos en el examen escrito para aprobar la asignatura. La calificación obtenida en este examen pondera un 65% de la nota de la asignatura.
- **Trabajo práctico en el laboratorio**. Se valorará que las soluciones aportadas se comporten según las especificaciones, la calidad de su diseño y el tiempo empleado. Es necesario una calificación mínima de 5.0 puntos en el trabajo práctico de laboratorio para aprobar la asignatura. La calificación obtenida pondera un 25% de la nota de la asignatura.
- **Realización y presentación de un trabajo práctico** de análisis y configuración de algún aspecto de sistema. La calificación obtenida en este trabajo pondera un 10% de la nota global de la asignatura.

Es obligatorio realizar las prácticas para poder presentarse al examen escrito. Se consideran las prácticas realizadas cuando el alumno entrega el material que se pide en el enunciado de las mismas, por el procedimiento que se establezca.

La nota en el convocatoria será la que corresponda a la suma ponderada de las tres pruebas, estando limitada a 4,5 puntos sobre 10 en el caso de no alcanzar un 5 sobre 10 en el examen escrito o en el trabajo práctico de laboratorio.

En el caso de que el alumno no logre superar la asignatura en la primera convocatoria, pero logre superar alguna(s) de las 3 partes de la prueba global, la calificación obtenida en dicha(s) prueba(s) se mantendrá para la convocatoria siguiente del mismo curso académico.

2.7 Bibliografía

La rápida evolución tanto del hardware como del software que forman los sistemas informáticos conlleva un rápido desfase del material bibliográfico tradicional en formato libro. No obstante existen libros de gran calidad que sin duda ayudan a los alumnos en el proceso de aprendizaje de la asignatura. En especial el primero de los citados a continuación resulta una guía indispensable para todo buen administrador de sistemas.

- UNIX and Linux System Administration Handbook, by Evi Nemeth, Garth Snyder, Trent R. Hein, and Ben Whaley. 5th Edition. July 2010, 1232 pages.
- Essential System Administration, Tools and Techniques for Linux and Unix Administration, by Æleen Frisch, 3rd Edition. August 2002, 1178 pages.
- Unix Shell Programming, by Stephen G. Kochan and Patrick Wood, 3rd Edition, March 2003, 456 pages.
- The Practice of System And Network Administration, by Thomas A. Limoncelli, Christina J. Hogan, and Strata R. Chalup. 2nd Edition. July 2007, 1056 pages.

Además de la documentación del sistema disponible vía las páginas man, abreviación de *manual page*, y las de info, Internet es una fuente de conocimiento infinita sobre la administración de sistemas. A continuación, el cuadro 2.1 muestra una colección no exhaustiva de enlaces a páginas

[4]El contenido de está sección reproduce buena parte de la información de la asignatura en su página web oficial: `http://titulaciones.unizar.es/asignaturas/30216/evaluacion15.html`.

con información muy útil. Durante el curso, se recomienda al alumno, buscar información en Internet sobre la asignatura pero siempre intentando verificar la calidad de las fuentes, ya que no siempre el material es bueno.

Cuadro 2.1: Breve colección de páginas web útiles para Administración de Sistemas.

Tema	Enlace	Contenido
Shell	`http://mywiki.wooledge.org/BashGuide`	Guía muy completa del lenguaje Bash
	`http://mywiki.wooledge.org/BashPitfalls`	Errores comunes al programar en lenguaje Bash
	`http://wiki.bash-hackers.org/scripting/tutoriallist`	Lista de tutoriales del lenguaje Bash. Los 2 primeros son muy recomendables.
Redes	`https://wiki.debian.org/DebianFirewall`	Guía de instalación de iptables, cortafuegos, en Linux Debian.
	`http://linux-ip.net/pages/diagrams.html`	Diagramas del funcionamiento en Linux de varios aspectos de red.
	`http://www.tldp.org/LDP/nag2/`	Linux Network Administrators Guide[a]
Almacenamiento	`https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html-single/Logical_Volume_Manager_Administration/`	Completo manual de administración de LVM.
	`https://ext4.wiki.kernel.org/index.php/Ext4_Howto`	Manual del sistema de ficheros Ext4
	`https://wiki.debian.org/SSDOptimization`	Información importante sobre el uso de SSD en Linux Debian.

[a] Existe una versión más moderna de este documento pero es de pago.

Capítulo 3

Desarrollo por Módulos

3.1 Organización de los temas en módulos

Para los estudiantes, dominar las competencias de la asignatura supone 150 horas de trabajo cuya organización puede ser analizada desde múltiples puntos de vista. Esta sección comienza realizando un análisis de acuerdo a la a la distribución temporal entre las distintas actividades de la asignatura como se muestra a continuación:

- 56 horas, aproximadamente, de actividades presenciales (teoría, problemas y prácticas de laboratorio).
- 81 horas de estudio personal efectivo
- 10 horas de realización/defensa de trabajos/proyectos prácticos (horas TP6)
- 3 horas de evaluación

Más de la mitad del tiempo, los estudiantes trabajan de manera independiente para adquirir los conocimientos de la asignatura y el resto se empleará sobre todo en la asistencia a clases y laboratorios.

El segundo análisis deja la distribución temporal y se basa en la distribución temática de la asignatura. Para Administración de Sistemas se propone cubrir su ficha docente en 30 temas muy concisos, que asumiendo un modelo presencial de la docencia, se cubrirán a ritmo medio de 1.25 temas por cada hora de teoría y a los que acompañaran sesiones de prácticas, problemas y trabajos prácticos multi-tema. Para modelos docentes *online*, esta granularidad tan fina, puede ayudar al alumnado a no perder el hilo de la asignatura y a realizar pequeños vídeos al profesorado para explicar los temas.

Debido a la dificultad de memorizar los 30 nombres y para facilitar la planificación tanto de los alumnos como del profesor, los temas pueden agruparse en módulos de acuerdo a su afinidad. De acuerdo a esta agrupación, la asignatura podría constar de 10 módulos tal y como muestra el cuadro 3.1.

De cara a flexibilizar el orden en su impartición, la mayor parte de los módulos son independientes entre sí y únicamente el módulo 3, Interacción con el sistema, en el que se presenta el interprete

Cuadro 3.1: Relación de temas de Administración de Sistemas agrupados en módulos

Módulo	Tema	Créditos ECTS
1. Introducción	1. Presentación de la Asignatura	0,05
	2. Introducción a la Administración de Sistemas	0,15
2. Normativa, aspectos legales y responsabilidad social	3. Normativa, aspectos legales, responsabilidad social	0,1
3. Interacción con el sistema	4. Introducción al shell Bash y comandos UNIX	0,3
	5. Bash: aspectos del lenguaje	0,3
	6. Shell: herramientas avanzadas	0,3
4. Seguridad	7. Criptografía básica	0,2
	8. Usuarios y control de acceso	0,2
	9. SSH: Secure Shell	0,2
	10. Seguridad en la infraestructura	0,2
	11. Otros aspectos sobre seguridad	0,1
5. Gestión básica del sistema	12. Arranque y parada del Sistema Operativo	0,1
	13. Configuración básica de red	0,5
	14. Instalación y gestión del software	0,1
	15. Configuración y gestión del núcleo del SO y sus drivers	0,2
	16. Interfaces de ventanas	0,1
6. Almacenamiento	17. El árbol de directorios	0,3
	18. Tipos, Atributos y Comandos con Ficheros	0,3
	19. Tipos de particiones y de sistemas de ficheros	0,2
	20. Tecnología de Almacenamiento	0,3
	21. Gestión de Memoria	0,1
	22. Sistemas de cuotas	0,1
	23. Gestión de volúmenes lógicos	0,3
	24. Copias de seguridad	0,2
7. Gestión de procesos	25. Procesos	0,3
	26. Automatización de tareas	0,3
8. Monitorización y Prestaciones	27. Monitorización del sistema	0,15
	28. Análisis del rendimiento del sistema	0,15
9. Servicios de sistema	29. Servicios de sistema: impresión, correo, NTP[a], …	0,2
10. Virtualización	30. Introducción a la virtualización de sistemas	0,1

[a] *Network Time Protocol*

y lenguaje de comandos Bash junto con multitud de utilidades UNIX es necesario para realizar problemas y resolver buena parte de las cuestiones que se plantean en el resto de temas.

Los 30 temas se imparten entre clases de teoría, de problemas y de laboratorio. Cada semana habrá 2 lecciones de teoría y una de problemas de 50 minutos de duración. Además, cada 2 semanas, los alumnos tendrán una sesión de laboratorio de 2 horas. Debido a que los temas anteriores tienen una duración variable al contrario que las clases que son siempre de 50 minutos, este capítulo detalla el emparejamiento entre temas y lecciones. En todas las clases de teoría y en algunas de problemas se presentará la materia mediante presentaciones de diapositivas. Esta organización sigue parcialmente el modelo que se emplea en la Escuela de Ingeniería y Arquitectura de la universidad de Zaragoza [1]. Durante parte del último cuatrimestre, la docencia ha sido impartida de manera *online* manteniendo el horario de clases. Estas clases podían ser vistas posteriormente en una lista de reproducción de videos, `https://www.youtube.com/playlist?list=PLD5WsNQ8L2_reWlQfkw97BAynwkkLheR8`. Observando los números de visualizaciones, se puede apreciar como dos tercios de los alumnos matriculados han visto algún video y que el más visualizado es el tema 13. Configuración básica de red, posiblemente porque su contenido es muy útil en varias prácticas y el trabajo final de la asignatura.

La siguiente lista resume la dedicación horaria del alumnado en la asignatura:

- Clases teóricas y de problemas (3 horas semanales).
- Clases prácticas de laboratorio (2 horas bisemanalmente). Son sesiones de trabajo de programación en laboratorio, tuteladas por un profesor, en las que participan los alumnos en grupos reducidos.

Cómo sistema operativo de referencia se propone el uso de Linux debido a su uso mayoritario en servidores y supercomputadores. De acuerdo a W3Cook en mayo de 2015 más del 98% de los servidores que alojan el millón de sitios más visitado según Alexa utilizan un sistema operativo derivado de UNIX y más del 96% Linux [w3c15].

A continuación, se describen los módulos y sus temas correspondientes.

3.2 Módulo 1: Introducción

Objetivos:

- Explicar los criterios de evaluación
- Describir otros aspectos relacionados con los problemas y las prácticas
- Conocer brevemente la historia de la Administración de Sistemas.

Temas:

1. Presentación de la Asignatura
2. Introducción a la Administración de Sistemas

Contenido: Este módulo empleará 2 lecciones de teoría en las que se describirán los aspectos formales de la asignatura: evaluación, entorno de prácticas, clases de problemas y luego se introducirá la asignatura.

La historia del sistema operativo UNIX servirá como hilo conductor de la Administración de Sistemas y se establecerán los principios básicos de UNIX que permiten al alumno comprender el

[1] `https://estudios.unizar.es/estudio/asignatura?anyo_academico=2019&asignatura_id=30216&estudio_id=20190148&centro_id=110&plan_id_nk=439`

funcionamiento tanto del sistema como de las herramientas que lo acompañan y la filosofía del curso. De acuerdo a Peter H. Salus está podría resumirse en [MPT78]:

- Escribir programas que hagan una cosa y la hagan bien
- Escribir programas que trabajen juntos
- Escribir programas que manejen cadenas de texto ya que son un interfaz universal

En este módulo no se realizarán ni clases de problemas ni prácticas.

3.3 Módulo 2: Normativa, Aspectos Legales y Responsabilidad Social

Objetivos:

- Gestionar políticas de administración y uso de sistemas informáticos
- Planificar y gestionar presupuestos
- Manejar la legislación vigente, en especial, la ley de protección de datos

Temas:

3. Normativa, aspectos legales y responsabilidad social

Contenido: Al igual que el anterior, este módulo solo consta de un tema dedicado a la organización de la infraestructura, su gestión, y la legislación que aplica sobre los administradores de sistema. En particular Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal [Esp11]. Este tema será impartido en una lección y pretende dar una visión general con muchos punteros que sirvan al alumno para ampliar, cuando sea necesario, la información sobre los mismos y para tomar consciencia de ellos. Como parte de la infraestructura, se abordará la problemática del impacto ambiental de las tecnologías de la información [GC15] y como la correcta aplicación de la Administración de Sistemas puede ayudar con los objetivos de desarrollo sostenible, sobre todo en los más próximos como son el 9, Industria, Innnovación e Infraestructuras, o el 7, Energía Asequible y no contaminante [Aus17; BM19].

Debido al carácter teórico de este tema no se realizarán ni problemas ni prácticas de laboratorio, pero su contenido se aprovechará en el resto de temas. Por ejemplo, en el módulo de almacenamiento se comentará la sostenibilidad del almacenamiento indefinido de datos y se nombrará el impacto ambiental de las infraestructuras requeridas. Es decir, al igual que el módulo de seguridad, este módulo enseña unas bases que son empleadas en multitud del resto de módulos.

3.4 Módulo 3: Programación para Administración de Sistemas

Objetivos

- Comprender la utilidad y las funcionalidades del Shell
- Conocer y utilizar los comandos básicos de UNIX
- Escribir scripts en bash utilizando *globbing*, tuberías, estructuras de control e iteración, …

Temas

4. Conceptos básicos de programación en Bash: variables, estructuras de control, tuberías, expansión, …
5. Filtros y conceptos avanzados de programación en Bash

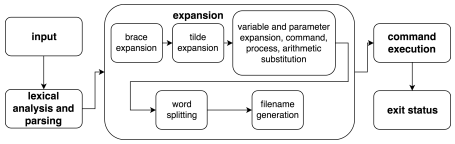

Figura 3.1: Organización interna del shell Bash

6. Comandos avanzados: `sed`, `awk`,...y su uso en *scripting*

Contenido: El módulo 3 consta de tres temas para aprender comandos y filtros UNIX-GNU y a manejar de manera fluida el intérprete de comandos bash. En el primer tema se introduce bash como intérprete y algunos comandos. Luego en el segundo se profundiza en los comandos como `grep`, `find`, `cut`, `cat` y otros. Además se introducen las variables y su expansión, estructuras de control e iteración, el entorno y las tuberías. En el último tema se ven comandos más avanzados como `sed` y `awk` junto con aspectos más avanzados del lenguaje como el alcance de las variables.

Para facilitar el aprendizaje de los alumnos este módulo se ve completamente en clase de problemas intercalando las explicaciones de teoría con ejemplos de comandos y scripts para que los alumnos asimilen rápidamente las abstracciones presentadas. De hecho, es recomendable pedir a los alumnos que realicen el trabajo previo de leer los temas antes de la clase de problemas para así centrar más tiempo en los ejemplos prácticos mediante el empleo de metodologías docentes activas como *Aprendizaje Basado en Problemas*, ABP, y *Flipped-Classroom*. Aunque en este módulo, se emplean ambas mucho, sobre todo la segunda, el resto de módulos también utilizan ABP porque permite introducir muy bien la teoría y aumentar el interés del alumno por la asignatura.

Una técnica útil al impartir la clases de este módulo consiste en partir en 2 la pantalla del computador para dejar en una mitad las transparencias del tema y en la otra una terminal para realizar demostraciones en directo de los comandos. Este módulo requiere 3 lecciones de problemas y de una práctica, como la práctica 1 de la sección 3.13.

3.5 Módulo 4: Seguridad

Objetivos:

- Establecer políticas de seguridad para usuarios: ¿quién puede acceder?, ¿a qué puede acceder? ¿Y de qué forma puede acceder?
- Conocer los principios de la criptografía de clave pública y privada
- Manejar con soltura SSH, *secure shell*
- Seguridad básica en redes. Topologías con cortafuegos y zonas desmilitarizadas

Temas:

7. Criptografía básica
8. Usuarios y control de acceso
9. SSH: *secure shell*
10. Seguridad en la infraestructura
11. Otros aspectos sobre seguridad

Contenido: El módulo 4 emplea 3 lecciones y media de clase de teoría y una lección de clase de problemas. En la primera lección se trabaja en los usuarios y el control de acceso, desde el fichero `/etc/passwd` hasta mecanismos sofisticados como Kerberos. En la segunda lección se termina

con la criptografía y se introduce ssh como reemplazo seguro a las antiguas alternativas como telnet. Ya en la última lección se explica como fortificar la infraestructura de sistemas y otros aspectos relevantes de la seguridad como la ingeniería social o el importante compromiso entre usabilidad/seguridad al que se enfrenta el administrador. Se explican jerarquías organizaciones, como tener un responsable de seguridad distinto de administración, para evitar disfunciones en la organización. En las clases de problema se verá como realizar script para añadirborrar usuarios al sistema y luego se hará una práctica de similar temática pero siempre con el acento puesto en la seguridad como podría ser la práctica 2 propuesta en la sección 3.13 de este documento [2]. Otras clases de problemas incluirán conocimientos de este módulo.

3.6 Módulo 5: Configuración Básica de Sistema

Objetivos:

- Entender con cierto nivel de detalle el proceso de apagado y encendido de un computador y sus implicaciones
- Comprender y gestionar interfaces de red, mecanismos de encaminamiento, protocolos de red (DHCP, encaminamiento …) mediante el uso de comandos y ficheros de configuración
- Conocer distintos sistemas de instalación de software (ficheros binarios y fuente), el formato de los paquetes, su instalación y borrado
- Entender la arquitectura del núcleo de un sistema operativo, saber que es un driver y como instalarlo
- Aprender la organización de los sistemas de ventanas como Xorg y saber realizar su configuración

Temas:

12. Arranque y parada del Sistema Operativo
13. Configuración básica de red
14. Instalación y gestión de sofware
15. Configuración y gestión del núcleo del SO y sus drivers
16. Interfaces de ventanas

Contenido: Este módulo requiere 3,5 lecciones de clase de teoría, 3 clases de problemas. La primera lección explica el proceso de *bootstraping* o arranque del sistema operativo y los mecanismos de los computadores para pasar el control desde que la máquina se enciende hasta que el sistema operativo toma el control. Debido a los riesgos de seguridad que suponen ataques durante el arranque, se presentaran los mecanismos para verificar un arranque seguro como UEFI SecureBoot y como emplear firmware auditable y seguro como coreboot, `https://www.coreboot.org`, o LinuxBoot, `https://www.linuxboot.org`. También se explicarán los servicios fundamentales del sistema operativo. En la parte de redes, se recordarán los niveles del modelo TCP-IP, aplicación, transporte, red, enlace y físico. La división entre redes y subredes, encaminamiento, redes virtuales (VLAN), … Una vez completado el repaso, se continua con los servicios, puertos y demonios del sistema operativo y con la configuración de nuevos nodos dentro de la red, incluyendo la asignación de direcciones IP, estática y dinámica, routing, Domain Name Server. Al completar el módulo, el alumno será capaz de gestionar y entender redes como la mostrada en la Figura 3.2.

El siguiente tema describe como instalar software y los formatos de los paquetes ya que una labor de los administraciones es gestionar las herramientas que necesitan los usuarios.

[2] El conocimiento necesario en redes para hacer la práctica se les suministrará al alumno en el propio guión.

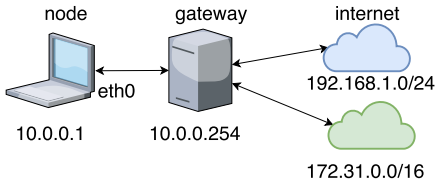

Figura 3.2: Ejemplo de red sencilla a configurar por parte de los estudiantes

El siguiente tema, *Configuración y gestión del núcleo del SO y sus drivers*, desarrolla los fundamentos del núcleo del sistema operativo y realiza una clasificación en tipos según su estructura: monolítica, modular o mixta. Después se centra en el núcleo Linux y su compilación desde código fuente. Al compilar el núcleo Linux también se compilan algunos módulos—componentes *software* que aislan funcionalidades del sistema operativo permitiendo que no consuman recursos cuando no estén en uso—y el tema detalla los comandos para operar con módulos como `lsmod` o `rmmod`. Una tarea de los módulos es dar soporte a nuevos dispositivos y el tema termina describiendo los dispositivos en UNIX y Linux.

El último tema comenta los sistemas de ventanas disponibles en Linux y su configuración incluyendo el servidor X y gestor de pantallas.

Debido al tamaño del módulo se dedicarán 3 lecciones de problemas y se intentará que la parte de redes se mezcle en problemas de otros temas para su mejor asimilación. Además las prácticas 2 y 3 utilizarán fundamentalmente estos temas. Mediante el uso de máquinas virtuales, los alumnos configuraran una red y utilizarán los conocimientos de todos los módulos vistos hasta ahora para instalar por ejemplo un servidor ssh al que solo puedan acceder un cierto número de nodos de una subred.

3.7 Módulo 6: Almacenamiento

Objetivos:

- Comprender los mecanismos de los ficheros en UNIX/Linux y su jerarquía
- Manejar con soltura las particiones y los sistemas de ficheros, incluyendo volúmenes dinámicos
- Discernir entre los tipos de tecnologías de almacenamiento y comprender los compromisos de rendimiento/coste/fiabilidad entre ellas
- Entender la gestión de memoria y su configuración
- Ser capaz de diseñar políticas de cuotas para usuarios
- Gestionar copias de seguridad

Temas:

17. El árbol de directorios: Representación y organización de la información
18. Tipos, Atributos y Comandos con ficheros
19. Tipos de particiones y de sistemas de ficheros
20. Tecnología de Almacenamiento
21. Gestión de memoria
22. Sistemas de cuotas
23. Volúmenes dinámicos y RAID

24. Copias de seguridad

Contenido: Al igual que el anterior, este módulo es bastante complejo y consta de 8 temas a impartir entre 8 y 9 lecciones[3]. Prácticamente se dedicará una lección por tema, aunque algunos de ellos como la gestión de volúmenes lógicos y los sistemas de ficheros requieran más. Ya que sobre ficheros y el sistema de ficheros han visto más materia en Sistemas Operativos, en esta asignatura se repasa su contenido pero no llega a ampliarse, excepto para las tecnologías de *journaling* y *copy-on-write*. Sobre los ficheros regulares, directorios y dispositivos de bloques únicamente se repasarán. Dentro de los ficheros se ve su tipo, el *magic number*, como se organiza un disco en particiones y se hacen dichas particiones. También se enseña a dar formato a los discos una vez particionados y las diferencias en configuración según su tecnología. Luego se describe el uso de particiones *swap* y como debe gestionarse la memoria de intercambio según sea la cantidad de memoria RAM del sistema, `vm.swappiness`, por ejemplo. Los grandes sistemas suelen utilizar volúmenes lógicos por su flexibilidad y el tema 23 describe desde la creación de volúmenes físicos, grupo volumen y volúmenes lógicos en el contexto de *Linux Volume Manager*. Además se realizará un breve estudio de los sistemas de ficheros encriptados, en especial para *ext4*. Finalmente, el último tema se dedica a las copias de seguridad y las herramientas que permiten hacerlas.

Junto con las clases de teoría se dedicarán 2 clases de problemas y una práctica, como la práctica 4 de la sección 3.13, para completar el aprendiza del módulo. La práctica podría incluir la instalación de un volumen lógico en un entorno con varios nodos para su uso como almacenamiento en un sistema de copias de seguridad.

3.8 Módulo 7: Gestión de Procesos

Objetivos:

- Controlar y observar procesos identificando sus distintos estados
- Manejar señales
- Alterar la prioridad de procesos
- Ser capaz de configurar y lanzar procesos periódicos

Temas:

25. Procesos
26. Automatización de tareas

Contenido: El módulo 6 contiene solamente 2 temas uno dedicado a la gestión de procesos y el segundo a la ejecución de programas, o tareas, en instantes de tiempo determinados o periódicos. Para su impartición se dedicarán hasta 4 lecciones, 2 lecciones por tema.

En el primero se describe el ciclo de vida de un proceso y los comandos fundamental para su visualización: `ps` y `top`. Además se recuerda el comando `kill` para enviar señales a procesos describiendo los tipos de señales que pueden ser enviadas. Todos los sistemas suelen correr más de un proceso por lo que es necesario saber como alterar las prioridades entre ellos, para ello se enseñan los comandos `nice` y `renice`.

El segundo tema del módulo, Automatización de tareas, se centra en la automatización de tareas, uno de los objetivos importantes de todo administrador de computadores. El tema comienza con la ejecución programada mediante el comando `at`, luego continua con la periódica, comando

[3] Aunque la planificación suponga 14 semanas de clase, en varios módulos se considera recomendable dejar holgura en la planificación porque hay semestres con únicamente 13 semanas

`cron`, y termina enseñando el comando `anacron` que permite la ejecución periódica aunque el sistema sea temporalmente apagado.

A este módulo le corresponden 2 lecciones de problemas en las que fundamentalmente se mezclaran scripts de este módulo mezclados con conocimientos de los módulos anteriores, especialmente el 5 de almacenamiento. Con las prácticas sucede algo similar y la sesión de prácticas correspondiente al módulo también requerirá conocimientos de los temas anteriores.

3.9 Módulo 8: Monitorización y Prestaciones

Objetivos:

- Ser capaz de definir políticas de monitorización
- Entender los compromisos de detalle/tamaño al monitorizar
- Conocer mecanismos de monitorización de procesos, usuarios y redes
- Conocer técnicas estáticas y dinámicas de análisis de prestaciones
- Manejar comandos de ajuste de prestaciones

Temas:

27. Monitorización del sistema
28. Análisis del rendimiento del sistema

Contenido: Diagnosticar problemas, resolver incidencias y detectar anomalías de funcionamiento es una actividad primordial del administrador de sistemas y objetivo de este módulo. El módulo consta de 2 temas repartidos en 2 lecciones de clase de teoría. En la primera lección se discutirán los requerimientos de la monitorización en las distintas categorías requeridas: procesos, red y sesiones. Después se verán políticas de almacenamiento, *logging*, y herramientas para su gestión como `syslog/syslogr` o `logrotate`.

El segundo tema pretende ofrecer una vista generalista de las herramientas disponibles en linux para analizar el rendimiento desde los contadores hardware, `perf`, logs del kernel, `ftrace`, redes, `ethtools` por citar algunas. Lo ideal sería mostrar todas las herramientas de la Figura 3.3 que describe Brendan Gregg en su página web [Gre16], aunque sea de manera superficial.

Respecto a la asignación de lecciones, a cada uno de estos temas les correspondería una lección de teoría y media de problemas. Además les corresponderá una posible sesión de práctica, práctica 6 de la sección 3.13, en la que los alumnos tendrían que diagnosticar el mal funcionamiento de un sistema, encontrarla y proponer una solución. Una posibilidad es utilizar este módulo con el siguiente para utilizar herramientas de diagnóstico y optimización en un servicio de alto nivel como podría ser un servidor web `apache`.

3.10 Módulo 9: Servicios de Sistema

Objetivos:

- Comprender la funcionalidad, puesta en marcha y configuración de servicios de sistema
- Conocer los servicios fundamentales como: impresión, NTP, correo electrónico, web, compartición de ficheros en red
- Instalar y configurar demonios para la gestión de servicios de red

Temas:

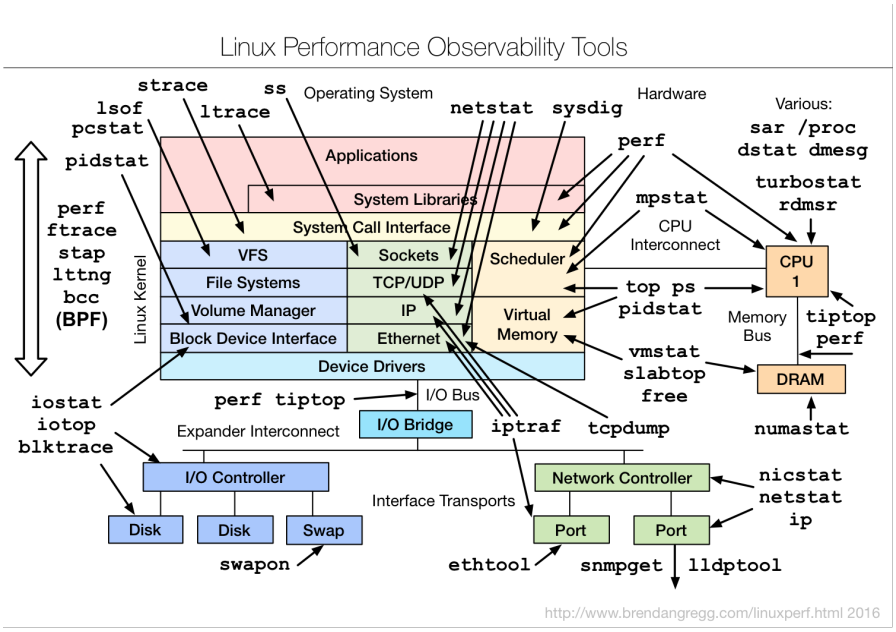

Figura 3.3: Herramientas para observar el rendimiento en Linux [Gre16].

29. Servicios de sistema: impresión, correo, NTP,…

Contenido: Este módulo únicamente contiene un tema que será explicado en 2 lecciones de teoría. No tendrá clases de problemas. En la primera lección se describirá el funcionamiento general de servicios de alto nivel y se tratará un servidor de impresión como `cups` y en la segunda sesión se abordarán otros servicios como la sincronización de relojes mediante NTP, *Network Time Protocol*, y el correo electrónico, protocolos IMAP y SMTP principalmente. Se abordará la temática de su configuración y aspectos relativos a su seguridad.

3.11 Módulo 10: Virtualización

Objetivos

- Comprender el concepto de virtualización
- Distinguir entre virtualización de sistema, de proceso, de red
- Conocer como gestionar a nivel básico herramientas de virtualización

temas:

30. Introducción a la virtualización

Contenido: Este breve módulo comprende únicamente una lección de teoría y no dispondrá de clases de problemas, aunque si habrá problemas relativos al mismo en la colección. La lección comenzará con la descripción del concepto de virtualización y su aplicación a los sistemas informáticos. Posteriormente, se analizarán los distintos tipos de virtualización para que los alumnos comprendan las diferencias entre una máquina virtual, un contenedor, un router virtual, …Y para finalizar, la lección se centrará en explicar los comandos básicos a emplear para realizar virtualización con *kvm*, *dockers*, *qemu*, …

3.12 Colección de Problemas de la Asignatura

Al tratarse de una asignatura eminentemente práctica, la realización de problemas tanto en papel como con el computador es de vital importancia para alcanzar los objetivos de aprendizaje. Por ello se ha diseñado una colección de problemas de distintos tipos, desde problemas breves donde se ha ofuscado el nombre de las variables y los alumnos deben descubrir la funcionalidad de los mismos a ejercicios completos de examen que requieren de entre 45 minutos y 1 hora para su resolución. En estos últimos se busca ayudar al alumno a resolver problemas complejos en los que intervengan varios módulos de la asignatura. Por ejemplo, el problema largo número 5 plantea la realización de un script para realizar el control térmico del procesador abarcando conocimientos de los módulos 3, 5 y 8 del temario.

La colección completa puede leerse en el apéndice 3.A

A modo de curiosidad, los nombres de las variables ofuscadas y algunas de las entradas de texto corresponden a personajes históricos de la Administración de Sistemas y de la ingeniería informática en general con especial énfasis en las mujeres para que los alumnos conozcan brevemente pinceladas sobre nuestra historia.

3.13 Sesiones de Laboratorio

Como se ha mencionado anteriormente, el programa de la asignatura se compone de 6 sesiones de laboratorio, prácticas, donde los estudiantes pueden experimentar con la materia. Para proponer estas sesiones se ha intentado que su realización vaya próxima a la impartición de los temas en clase para asentar el conocimiento de los alumnos. A continuación se detallan las 6 sesiones:

1. **Introducción a Bash**: En esta sesión los estudiantes instalarán una maquina virtual para un sistema operativo GNU/Linux y realizarán scripts sencillos para familiarizarse con el entorno. Ejemplos de estos scripts podrían ser buscar ficheros dentro del sistema y luego mostrar por pantalla aquellos que contengan la palabra *hola* o crear un directorio cada 10 segundos añadiendo la hora al nombre del directorio.
2. **Redes y Autentificación**: Los estudiantes tendrán que diseñar una topología de red en la que haya al menos 3 máquinas en una red de área local. Una de ellas estará dentro de una zona desmilitarizada y albergará un servidor SSH que será accesible tanto desde el interior de la red de área local como desde el host. Además se configurará el servidor para que no permita el acceso como administrador y que haya un usuario que solo se pueda conectar utilizado claves (`Match User …`).
3. **Usuarios y Kernel**: Los alumnos compilarán un módulo del kernel de linux. Por ejemplo un generador de números aleatorios, `https://github.com/davidscholberg/merandom`, y luego podrán usarlo para generar claves de usuarios en un script de creación de cuentas. Las cuentas se crearan desde un fichero de texto que tendrá que ser parseado. Además la creación de cuentas se podrá hacer en máquinas reales o remotas.
4. **Ficheros y Discos**: Para comprender la gestión de discos, volúmenes, LVM, los alumnos tendrán que crear un grupo volumen formado por varios volúmenes físicos que disponga de un número distinto de volúmenes lógicos que se montarán al arrancar el sistema. Además, en la segunda parte de la práctica utilizaran `debugfs` para analizar el *journal* de una partición `ext4`. El análisis podría consistir en borrar un fichero y recuperarlo con la información del *journal*.

5. **Automatización de Tareas**: En esta sesión de laboratorio los alumnos tendrán que instalar actividades en el sistema para que se realicen de manera periódica como puede ser la limpieza de antiguos ficheros temporales que no están en uso. La práctica requerirá que los alumnos realicen una búsqueda de tareas cruciales en un administrador de sistemas, que las implementen (incluyendo la instalación del software necesario) y hagan que se ejecuten de manera periódica en varias máquinas a la vez.
6. **Análisis de logs**: La última sesión de laboratorio se corresponde con los módulos 8 y 9 del temario. En ella los alumnos instalarán el demonio `rsyslog` y alguna herramienta de análisis de logs como `logchecker` o `logwatch`. Utilizando herramientas para conocer el estado del sistema, como el número de procesos activos, espacio libre en disco, espacio libre en memoria, alarmas térmicas, servicios, ... tendrán que enviarlas desde 3 nodos a una máquina que hará de receptor de todos los mensajes y donde se hará en análisis.

Para ayudar al alumnado en la realización de las prácticas y afianzar buenas técnicas de desarrollo, se dispone de un conjunto de test unitarios para que los scripts de las prácticas puedan ser evaluados antes de ser enviados a corregir. Estos tests realizan pruebas muy sencillas y concretas para ayudarles a encontrar los errores. De manera consciente, la duración de los tests en caso de fallo es relativamente alta, más de 10 segundos, para evitar tentaciones de resolver las prácticas "por fuerza bruta". Los tests se acceden vía un repositorio git, `https://gitlab.unizar.es/dario/as_tests_practicas`, y además se dispone de script para realizar la corrección de manera automática.

3.14 Trabajo de la Asignatura

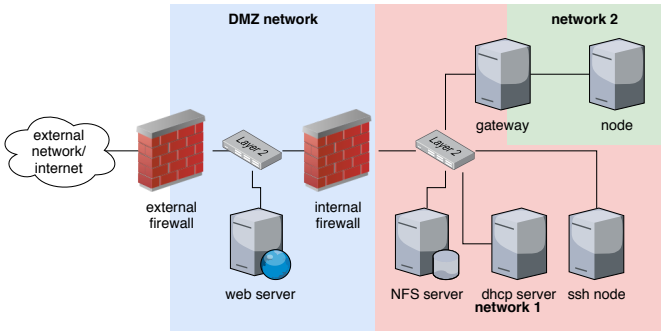

Figura 3.4: Topología de red del trabajo de asignatura, se distingue una zona desmilitarizada para albergar un servidor web y luego dos redes internas con un nodo realizando tareas de pasarela entre ellas.

Para que los alumnos puedan comprobar si han adquirido los conocimientos de la asignatura Administración de Sistemas, se plantea un trabajo final donde se apliquen la mayor parte de conocimientos aprendidos. Los alumnos tendrán que preparar una red de área local con 8 máquinas virtuales en la que 2 hagan tareas de cortafuegos, una haga de servidor `NFS`, otra tenga un servicio `dhcp` para asignar direcciones IP dinámicas al resto y haya un servidor web en una zona desmilitarizada que sea accesible desde el exterior de la red tal y como muestra la figura 3.4. El tráfico será filtrado para que desde el exterior no se pueda llegar a las máquinas de la red local.

El servidor `NFS` será accesible únicamente para las máquinas internas a la red y montará una

partición que se encuentre dentro de un grupo volumen. Se podría montar de este modo el directorio `/usr` por ejemplo. Los alumnos deberán entregar un diagrama de red, los scripts necesarios para levantar todos los servicios, así como los ficheros de configuración relevantes de cada una de las máquinas. Para asegurarse la consistencia en las operaciones se deberá también emplear el protocolo `ntp` asegurando que los relojes de todas las máquinas estén sincronizados. Además el puerto 22 del nodo ssh será accesible desde internet y los alumnos deberán decidir cuál es la política de acceso al servidor web tanto desde el exterior como desde el interior.

Cuando todo el sistema esté perfectamente montado, los alumnos utilizarán los *scripts* del proyecto Linux Test Project, `https://github.com/linux-test-project/`, para realizar pruebas de robustez y rendimiento en sus nodos. Por ejemplo, podrían realizar pruebas sobre `iptables` o sobre el servidor `NFS`.

3.A Colección de Problemas de Administración de Sistemas

Colección de problemas

Administración de Sistemas. Universidad de Zaragoza

17 de enero de 2020

Índice general

Notas preliminares

Esta breve colección de problemas se corresponde a la Asignatura Administración de Sistemas impartida en el cuarto semestre del grado en Ingeniería Informática de la Universidad de Zaragoza. La colección se divide en 2 bloques. El primero contiene problemas cortos y el segundo está dedicado a problemas más extensos provenientes de exámenes de la asignatura.

Si detecta cualquier errata en alguno de los problemas planteados por favor comuníqueselo a Darío Suárez Gracia, dario@unizar.es.

Problemas breves y cuestiones

1. ¿Describe cuál es el objetivo del siguiente script?

```
#! /bin/bash
ONE=1
chet=0
FOX=0

for brian in ./*
do
    echo "$brian" | grep -q " "
    if [ $? -eq $FOX ]
    then
        ramey=$brian
        n=`echo $ramey | sed -e "s/ /_/g"`
        mv "$ramey" "$n"
        chet=$((chet += 1))
    fi
done

if [ "$chet" -eq "$ONE" ]
then
    echo "$chet thing changed."
else
    echo "$chet things changed."
fi

exit 0
```


Nota: Brian Fox y Chet Ramey son 2 desarrolladores de Bash.

2. ¿Indica cuál es el objetivo del siguiente script?

```
#!/bin/bash
E_BADARGS=85

if [ ! -r "$1" ]
then
echo "Usage: $0 files-to-process"
exit $E_BADARGS
fi


cat "$@" |
tr A-Z a-z |
tr ' ' '\012' |
tr -c '\012a-z'  '\012' |
grep -v '^$' |
sort |
uniq |

exit $?
```

3. Se desea programar un script en bash que reciba un fichero con una lista de usuarios, un usuario por línea, como parámetro de entrada y determine si el usuario existe en el sistema. En caso afirmativo, el script revisará los permisos del directorio .ssh verificando si son correctos. Cuando los permisos sean incorrectos, se mostrara un mensaje de error por pantalla y si el usuario no pertenece al sistema se borrara el directorio `/home/usuario` si existe.

4. Un administrador ha encontrado un script y no identifica cuál es su función. ¿Puedes por favor ayudarle?

```
#!/bin/bash

FRANCES=10
ALLEN=9

# $RANDOM is an internal Bash function (not a constant) that returns a
# pseudorandom [1] integer in the range 0 - 32767. It should not be used
# to generate an encryption key.
i=$RANDOM
let "i %= $FRANCES"

if [ "$i" -lt "$ALLEN" ]
then
    echo "i = $i"
    ./$0
fi
exit 0
```

Nota: Frances E. Allen es una pionera en compilación y paralelismo ganadora del premio Turing en 2006

5. Realiza un script en bash que reciba como argumento uno o varios enlaces a páginas web y cuente el número de enlaces seguros, https, que contenga cada una de las páginas. Para la cuenta es suficiente contar el número de veces que aparezca la cadena https en el fichero. El script devolverá un error si es llamado sin ningún argumento. Cuando la ejecución sea correcta, la salida del script contendrá una línea por página web procesada y cada línea estará formada por el nombre de la web y el número de enlaces seguros.

6. Programa un script que dado un directorio, busque aquellos ficheros que no hayan sido modificados en el último mes y los guarde en un fichero tar. El nombre del fichero tar resultante contendrá un prefijo recibido como argumento seguido de guión bajo y de la fecha actual. Opcional: en vez de un directorio, utilizar como entrada múltiples directorios. El script tendrá al menos 2 argumentos, el primero será el prefijo y después se incluirán los nombres de los directorios de entrada.

7. Implementa un script para cambiar el password de un usuario verificando que su longitud es de al menos

8 caracteres, que tenga letras mayúsculas y minúsculas y dígitos. ¿Dispone linux de algún mecanismo mejor para hacer esta comprobación?

8. Haz un script que muestre todos los usuarios de un sistema, leyendo `/etc/passwd`, cuyo shell por defecto sea bash.

9. Siguiendo con el ejemplo anterior, escribe un script para mostrar los usuarios de un sistema agrupados por su shell por defecto. Para el siguiente fichero de entrada:

```
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
```

La salida debería ser:

```
shell /usr/sbin/nologin
nobody
shell /bin/false
syslog
messagebus
```

10. Realiza un script que muestre por pantalla todos los directorios que ha visitado un usuario y que todavía sean válidos. El nombre de usuario será pasado como parámetro del script. Para el cálculo de las rutas podrás asumir que el primer directorio visitado fue el $HOME del usuario y sólo se mostrarán los directorios que hayan sido accedidos con `cd` y que se encuentren en la historia del usuario. La salida sólo deberá mostrar las rutas canónicas de todos los directorios visitados cuyas rutas sean absolutas, empiecen por `/`, o empleen la expansión de tilde con `~` y `~+`.

11. Escribe un script para añadir un nodo al fichero `/etc/hosts` verificando que el nodo es accesible y que no exista ya en el fichero.

12. Programa un script que cuente el número de nodos que son visitados para alcanzar la dirección `www.google.com`. Puedes utilizar una herramienta como `traceroute`. Ademas de mostrar los nodos visitados, la salida de `traceroute` contiene los round-trip time, RTT, cada uno de los servidores visitados. Utiliza dichos valores de RTT para mostrar los 5 nodos con mayor RTT. Ya que `traceroute` devuelve el RTT para 3 paquetes, idealmente deberás hacer la media de los 3 paquetes para calcular el RTT medio de cada nodo. En caso de que se produzcan time-outs con `traceroute`, lo que dificulta procesar su salida, puedes hacer que el script tome como argumento la siguiente entrada:

```
traceroute to www.google.com (172.217.19.132), 30 hops max, 60 byte packets
 1  155.210.152.254 (155.210.152.254)  0.847 ms  1.244 ms  1.650 ms
 2  155.210.251.9 (155.210.251.9)  0.522 ms  0.519 ms  0.510 ms
 3  r-icuz-man.unizar.es (155.210.248.45)  0.753 ms  0.749 ms  0.737 ms
 4  193.144.0.81 (193.144.0.81)  2.303 ms  2.621 ms  2.606 ms
 5  XE5-0-1.unizar.rt1.ara.red.rediris.es (130.206.195.5)  2.258 ms  2.250 ms  2.240 ms
 6  UNIZAR.AE6.telmad.rt4.mad.red.rediris.es (130.206.245.94) 11.187 ms 11.019 ms 11.001
 ↪  ms
 7  google-router.red.rediris.es (130.206.255.2)  10.995 ms  11.176 ms  10.896 ms
 8  72.14.235.20 (72.14.235.20)  26.804 ms  26.784 ms  11.093 ms
 9  209.85.246.133 (209.85.246.133)  31.408 ms  26.956 ms  31.566 ms
10  209.85.247.194 (209.85.247.194)  31.549 ms  31.538 ms  31.519 ms
11  66.249.94.79 (66.249.94.79)  31.518 ms  31.509 ms  31.611 ms
12  par03s12-in-f4.1e100.net (172.217.19.132)  26.700 ms  26.684 ms  26.647 ms
```

Para el cálculo de la media, se puede utilizar el comando `bc` o `awk`.

13. Describe la razón del mal funcionamiento del siguiente script e indica como debería arreglarse. *Examen 12 de Junio de 2014*:

```
#!/bin/bash


while read linea ; do
    ssh usuario@maquina cmd $linea
done < fichero
```

14. Indica en qué casos este siguiente comando podría fallar y como solucionarlo.

```
mv $source_file $destination_file
```


15. ¿Cuál es la salida del siguiente script? Además de indicar la salida indica los motivos que la causan.

```
#!/bin/bash

f() {
  if [ -t 0 ]; then
    echo "$1"
  else
    read -r var
    echo "$var"
  fi
}

f 'hello' | f
echo "$var"
```

16. El siguiente script intenta buscar los ficheros que cumplen un patrón, tienen la extensión .txt, para escribirlos por pantalla pero presenta varios fallos. ¿Podrías indicar cómo solucionarlos?

```
#!/bin/bash

re='\.txt$'
for file in ./*
do
  if [ $file =~ "$re" ]
  then
    echo "Match found: $file"
  fi
done
```

17. ¿Qué tráfico permitirá la siguiente configuración de iptables?

```
root@borodino:/etc# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

18. El manual de iptables de debian propone el siguiente fichero de configuración para un *firewall* en una red doméstica.

```
#!/bin/sh

PATH='/sbin'

## INIT

# Flush previous rules, delete chains and reset counters
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

# Default policies
iptables -P INPUT   DROP
iptables -P OUTPUT  DROP
iptables -P FORWARD DROP

echo -n '1' > /proc/sys/net/ipv4/ip_forward
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_source_route
echo -n '0' > /proc/sys/net/ipv4/conf/all/accept_redirects
```

```
echo -n '1' > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo -n '1' > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# Enable loopback traffic
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Enable statefull rules (after that, only need to allow NEW conections)
iptables -A INPUT   -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT  -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Drop invalid state packets
iptables -A INPUT   -m conntrack --ctstate INVALID -j DROP
iptables -A OUTPUT  -m conntrack --ctstate INVALID -j DROP
iptables -A FORWARD -m conntrack --ctstate INVALID -j DROP


## INPUT

# Incoming ssh from the LAN
iptables -A INPUT -i eth1 -s 192.168.0.0/24 \
                  -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT


## OUTPUT

# Enable al outgoing traffic to internet
iptables -A OUTPUT -o eth0 -d 0.0.0.0/0      -j ACCEPT

# Enable access traffic, from the firewall to the LAN network
iptables -A OUTPUT -o eth1 -d 192.168.0.0/24 -j ACCEPT


## FORWARD

# We have dynamic IP (DHCP), so we've to masquerade
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables        -A FORWARD     -o eth0 -i eth1 -s 192.168.0.0/24 \
                               -m conntrack --ctstate NEW -j ACCEPT

# Redirect HTTP (tcp/80) to the web server (192.168.0.2)
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
                              -j DNAT --to-destination 192.168.0.2:80

iptables        -A FORWARD    -i eth0 -p tcp --dport 80 \
                              -o eth1 -d 192.168.0.2    \
                              -m conntrack --ctstate NEW -j ACCEPT


## LOGGING

iptables -A INPUT   -j LOG --log-level DEBUG --log-prefix '[FW INPUT]:    '
iptables -A OUTPUT  -j LOG --log-level DEBUG --log-prefix '[FW OUTPUT]:   '
iptables -A FORWARD -j LOG --log-level DEBUG --log-prefix '[FW FORWARD ]: '
```

Describe el número de intefaces de red de las que dispone el *firewall* así como la posible configuración de red con la funcionalidad de los distintos nodos y servicios dispuestos. ¿Para que se realiza *IP masquerading*?

19. Indica por favor en que se diferencian source NAT, `SNAT`, y masquerading, `MASQ`, dentro de iptables.
20. Un ataque por denegación de servicio, *denial-of-service (DoS)*, intenta hacer innacesible un servicio, red

o computador. Un DoS puede realizarse por ejemplo realizando muchas peticiones a un servidor. Una posible defensa sencilla ante este tipo de ataques puede consistir en descartar el tráfico proveniente de un nodo cuando este supere un umbral de tráfico hacia el servidor. Implementa un mecanismo que proteja un servidor web, puerto 80, descartando las conexiones de aquellos nodos que se intenten conectar a él más de 15 veces en un minuto.

21. Escribe por favor un script que muestre por pantalla la fecha actual y en la línea siguiente el PID, el nombre, y el porcentaje de memoria consumida, *resident set size*, respecto al total del sistema de todos los procesos activos. Una vez que tengas completado el script, indica como se podría añadir un trabajo `cron` que lo ejecutara los lunes y martes entre las 10 y las 11 de la mañana cada 5 minutos.
22. Imagina una red de 7 sensores conectados a una red IP mediante WiFi a los que hay que monitorizar. La monitorización consiste en hacer ping hacia ellos 3 veces, IPs en rango 172.16.1.[1-7], una vez al día. Si alguno de los nodos no responde a alguno de los ping, deberemos enviar un único email al administrador con las IPs de los nodos caídos. La máquina desde la que se lanzarán los pings no está siempre encendida pero si se enciende deberá realizar la monitorización. Escribe por favor el script necesario para monitorizar los nodos y el código necesario para garantizar su ejecución diaria.
23. ¿Describe cuál es la salida del siguiente script y explica su causa? Debes comentar como afectan las funciones y los sub-shells a la creación de procesos y como se ven afectadas las variables en ambos casos.

```
#!/bin/bash

var0=0
export var1=1

function f_var0 {
    echo -e "\tfunction f_var0, begin f_var0, var0: $var0"
    var0="zero"
    echo -e "\tfunction f_var0, end f_var0, var0: $var0"
}

function f_var1 {
    echo -e "\tfunction f_var1, begin f_var1, var1: $var1"
    var1="one"
    echo -e "\tfunction f_var1, end f_var1, var1: $var1"
}

echo "Program begins, var0: $var0, var1: $var1"

f_var0
f_var1

echo "Before updating, var0: $var0, var1: $var1"

var0=5
var1=1

( f_var0; f_var1 )

echo "After sub-shell, var0: $var0, var1: $var1"

exit 0
```

24. Explica por favor cuál es el resultado de la ejecución del siguiente comando: *Examen Septiembre 2016*

```
tar cj *.html | ssh user@unizar.es tar xj
```

25. Indica por favor cuál es el resultado de la ejecución del siguiente script: *Examen Septiembre 2016*

```
#!/bin/bash

var1="grace"
export var2="hopper"
```



```
/bin/bash -c "echo \$var1-\$var2"
/bin/bash -c "echo $var1-$var2"
```

26. Explica el funcionamiento de este script y que efectos puede tener en el sistema: *Examen Junio 2016*

```
#!/bin/bash
./$0|./$0&
```

27. Indica los problemas que podrían ocurrir al ejecutar esta secuencia de comandos:

```
cat $(find . -type f) > ../contenido_ficheros.txt
```

28. ¿Cuál es la salida del siguiente comando?:

```
echo /usr/bin/l*p3
```

29. Un sistema no dispone del comando filtro `uniq`. Podrías por favor escribir un script de `bash` para reemplazarlo. El script tendrá 2 parámetros, `input_file` y `output_file` y al terminar la ejecución del script `output_file` contendrá todas las líneas de `input_file` que no estén repetidas en líneas consecutivas.
30. Escribe un script `killall.sh` que mate todos los procesos cuyo nombre coincida con el primer argumento del script. Si el script recibe varios parámetros deberá devolver un mensaje de error. Además el script comprobará que dispone de privilegios de administración.

Problemas largos

1. Un proceso zombie, estado Z, es un proceso que ha completado su ejecución pero que aún mantiene la entrada en el tabla de procesos. Ocurren cuando un proceso hijo termina y el padre no ha ejecutado la correspondiente llamada a `wait()`. La unica manera de eliminar a un proceso zombie es matando al padre para que el proceso init los adopte y ejecute el `wait()` correspondiente.

   Realiza un script que busque procesos zombies que lleven más de 24 horas en ejecución, puedes utilizar `etime` de `ps`. Para todos los procesos encontrados deberás enviar la señal `KILL` a su proceso padre y un mail al dueño del proceso para notificarle la defunción del mismo. Además al terminar, el script deberá mandar un mail al administrador con la lista de todos los procesos matados con formato:

   ```
   pid, comando
   ```

   Además este proceso deberá ser ejecutado cada 24 horas. *Examen Junio 2016*

2. Escribe en bash una utilidad de copia de ficheros entre nodos utilizado scp/ssh. El script tendrá 3 o más argumentos. El primero será el usuario y la máquina destino, por ejemplo: `usuario@maquina.unizar.es`. El segundo argumento será el nombre del directorio destino donde se desea copiar los ficheros y directorios. El resto de argumentos serán el nombre de los ficheros y directorios a copiar. En caso de que un argumento sea un directorio hay que copiar todos los ficheros contenidos en él pero no sus subdirectorios. En el destino todos los ficheros serán copiados en la misma carpeta eliminando el nombre del directorio cuando sea necesario. Para reducir ancho de banda antes de copiar cada elemento, el script calculará la firma de los ficheros con md5, sintaxis `md5sum fichero`. Dicha firma será enviada al nodo destino en donde se comprobará si el fichero ya existe y tiene la misma firma. Cuando el fichero no exista en el destino o la firma sea distinta, se copiará.

   Nota: Se debe asumir que el usuario dispone de las claves ssh adecuadas para acceder a la máquina remota sin necesidad de introducir ningún password y que el directorio destino también existe. Además se supondrá que ninguno de los ficheros a copiar puede ser modficado desde el comienzo del cálculo de la firma md5 y el final de la copia al nodo destino. *Examen Septiembre 2016*

3. Se quiere monitorizar en tiempo real un servicio web con personas reales. Para facilitar su horario de trabajo, el servidor sólo aceptará peticiones en horario de 9 a 2 y de 4 a 7 de lunes a viernes. Indica el mecanismo para implementar este comportamiento y escribe los comandos necesarios para ponerlo en marcha, incluyendo algún script que fuera necesario. Se supondrá que el servidor web utilizado es apache y se podrá arrancar/apagar mediante los comandos `service apache2 start/stop`. Cuando el servidor esté apagado también será necesario configurar un firewall para bloquear el tráfico al puerto 80 fuera de ese horario.

   El servidor web genera un fichero de logs en el que aparece la IP, el puerto origen y la hora separados por espacios para cada una de las conexiones realizadas por los clientes. Realiza un script que busque en el fichero las direcciones IP que se han intentado conectar desde la misma subred a la que esta conectada el interfaz `eth0` del servidor para posteriormente bloquear todo el tráfico proveniente de dichos nodos y

enviar un email al usuario root de la máquina bloqueada para notificarle el bloqueo. El nombre del fichero será el primer argumento en la invocación del script y si esté es invocado con un número incorrecto de argumentos devolverá un error.

Nota: Se puede asumir que la máscara de red del servidor es `255.255.0.0`. *Examen Septiembre 2016*

4. Se disponen de 10 discos duros (`/dev/sd[a-j]`) con capacidad 1 terabyte cada uno y particionados con una única partición (`/dev/sd[a-j]1`). Fueron instalados para disponer de un volumen lógico en el directorio `/home` del sistema. Para alargar su vida útil se desea que sean empleados sólo cuando sea necesario hacerlo y que el resto del tiempo no sean visibles en el sistema. El administrador encargado del mismo necesita un script para agrandar el volumen lógico `/home` cada vez que esté lleno en más de un 90%. Esta comprobación se realizará cada 10 minutos y si es positiva habrá que añadir un disco al volumen lógico. Cuando ya estén en uso todos los discos duros, el script deberá mandar un email a los 10 usuarios que más espacio en disco estén consumiendo pidiéndoles que reduzcan su uso de espacio, el email se seguirá enviando indefinidamente mientras que la utilización de espacio siga por encima del 90%. Escribe por favor el script o los scripts necesarios para cumplir esta tarea.

   Notas:

   - Se deberá asumir que, al principio de la ejecución del programa, sólo se está usando `/dev/sda1`.
   - Los nombres del grupo volumen y del volumen lógico donde está montado la carpeta home son `vg_pool` y `lv_home` y el tipo de la partición del directorio `/home` es `ext4`.
   - Todos los usuarios tienen su home en el directorio `/home/<usuario>`
   - El comando `du` tiene la opción `–d` que limita el número de sub-directorios que se muestran por pantalla. Por ejemplo `–d0` sólo muestra por pantalla el uso de espacio del directorio/os pasado/os como argumentos. La salida incluye un argumento por línea. *Examen Junio 2017*

5. Un administrador desea monitorizar la temperatura de la CPU 0 de un multiprocesador, llamada `cpu0`, y limitar su frecuencia máxima de operación cuando su temperatura exceda un valor umbral durante un periodo dado de funcionamiento. Para ello necesita un script que tome 4 parámetros como argumentos: temperatura umbral, duración monitorización, intervalo muestreo y frecuencia máxima. Con estos argumentos, el script deberá comprobar durante un tiempo igual a duración monitorización si la temperatura media, medida cada intervalo, es mayor que la temperatura umbral y establecerá como frecuencia máxima el cuarto parámetro del script. El script verificará al comienzo de su ejecución que la nueva frecuencia máxima es un valor de frecuencia válido y menor que la máxima posible. Además, se asegurará que la duración total es múltiplo del intervalo de muestreo.

   Notas:

   - La unidad de todas las medidas de tiempo, duración monitorización e intervalo, es el segundo.
   - La temperatura de la `cpu0` se puede consultar leyendo el fichero `/sys/class/termal_zone0/temp`
   - La lista de frecuencias disponibles se puede consultar en `/sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies`. Este fichero muestra una lista de frecuencias disponibles en kilohercios separadas por espacios. Por ejemplo: 384000 486000 594000 702000 810000 918000 1026000
   - La frecuencia de la `cpu0` se puede cambiar escribiendo su valor en el fichero `/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq`.
   - Para hacer operaciones con números de coma flotante se puede utilizar `bc`. Ejemplo: `# echo "1 / 2" | bc –l` muestra por pantalla 0.5. *Examen Junio 2017*

6. El comando lastb muestra una salida tipo:

   ```
   root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
   root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
   root     ssh:notty    222.186.160.49   Fri May 29 11:33 - 11:33  (00:00)
   root     ssh:notty    lubov3110.static Fri May 29 11:33 - 11:33  (00:00)
   ```

   Con los datos de los intentos de login fallidos. Las columnas indican el usuario, TTY donde se intentó logear, IP o host desde el que se intentó la conexión, fecha del intento, y duración de la sesión (00:00). Escribir un script que revise la salida de este comando y muestre por pantalla el nombre del usuario y el día cuando el usuario tenga más de 10 intentos fallidos de login en el mismo día. Además deberá programarse la ejecución del script todos los días a las 23:59 horas y se enviará un email con la lista de dichos usuarios a root. *Examen Junio 2015*

7. Empleando el pseudo-sistema de ficheros `/proc` se desea obtener un script que muestre una lista de procesos que contenga la siguiente información:

   ```
   id proceso, nombre, memoria_residente_proceso_e_hijos
   ```

La lista contendrá únicamente procesos que sean hijos del proceso init, ID 1, y el cálculo de la memoria residente se hará sumando a la memoria residente del proceso la memoria residente consumida por sus hijos, nietos de init. La lista será visualizada al final de la ejecución y estará ordenada por el tamaño de memoria residente como se ve a continuación:

```
805, lightdm, 63052 KB
906, tmux: server, 21544 KB
846, systemd, 18292 KB
408, sshd, 12660 KB
853, at-spi-bus-laun, 9444 KB
882, systemd, 8200 KB
862, at-spi2-registr, 6892 KB
340, systemd-logind, 4828 KB
167, systemd-journal, 4660 KB
347, dbus-daemon, 4028 KB
188, systemd-udevd, 3952 KB
344, rsyslogd, 3304 KB
820, VBoxService, 3064 KB
```

`/proc` contiene un directorio por proceso con nombre `/proc/<pid>` siendo `<pid>` el identificador del proceso. Para la generación de la lista se puede consultar el fichero `/proc/<pid>/status` correspondiente al proceso con id `<pid>` y que tiene el siguiente formato (campos separados por uno o varios espacios):

```
Name:   exim4
Umask:  0000
State:  S (sleeping)
Pid:    745
        PPid:   1
        VmSize: 56152 kB
        VmLck:      0 kB
        VmPin:      0 kB
        VmHWM:   2588 kB
        VmRSS:   2588 kB
```

Una vez obtenida la lista, el script deberá cambiar la prioridad de ejecución a +15 para los 4 procesos que entre ellos y sus hijos consuman más memoria. Para evitar problemas con la creación y/o terminación de procesos, el script deberá guardar al principio de su ejecución el estado de todos los procesos.

Nota: Se puede asumir que el script se ejecuta con privilegios de administración y que al cambiar la prioridad el proceso sigue vivo. La opción `-s` del comando `tr` reemplaza múltiples ocurrencias de un carácter repetido por una única ocurrencia. *Examen Junio 2018*

8. Dada la subred 172.16.0.0/24 se desea asegurar que todos los nodos de dicha subred comparten el mismo fichero de nombres, `/etc/hosts`. Para ello, se plantea una solución con 2 scripts. El primero que se encargará de acceder a todas las máquinas, mezclará los ficheros `/etc/hosts` para que no haya ninguna IP repetida, copiará el fichero resultante en todos los nodos y relanzará el servicio de red, y el segundo que comprobará todos los días a las 3.14 horas que todos los nodos comparten el mismo fichero `/etc/hosts` que `172.16.0.1`. Al terminar la comprobación, se deberá enviar un mail al usuario root con la lista de los nodos, una IP por línea, para los que el fichero sea distinto, si hubiera alguno. El motivo del mensaje será lista nodos.

   Nota: Se puede asumir que sólo se emplearán direcciones IPv4, que si hay direcciones IP repetidas apuntarán al mismo nombre y que todos los nodos de la red están vivos. El usuario user tiene acceso vía ssh con clave sin password en todas las máquinas de la subred y privilegios de administración mediante `sudo`. *Examen Junio 2018*

9. Un administrador quiere minimizar el uso de recursos por parte del kernel de Linux y necesita un script que compruebe una vez cada hora todos los módulos del kernel que están cargados y descargue aquellos que no estén en uso, es decir aquellos para los que el valor de la columna `Used by` sea igual a 0. *Examen Septiembre 2018.*

10. Además desea saber cuales son los módulos más empleados. Para ello requiere de un único script que, durante 24 horas, empezando a las 6 de la mañana, compruebe cada 2 horas los módulos que estén usados 3 o más veces. Todos los módulos que cumplan la condición durante las 12 comprobaciones, serán añadidos a una lista que deberá enviarse por mail al usuario user con asunto "módulos más empleados". La lista estará ordenada alfabéticamente.

Notas:

- Se puede asumir que el script se ejecuta con privilegios de administración.
- El comando date permite conocer la fecha y la hora actual
- Si se desea se puede emplear el fichero /var/run/modules.txt como almacenamiento temporal
- La opción -s del comando tr reemplaza múltiples ocurrencias de un carácter repetido por una única ocurrencia.
- El comando rmmod descarga un módulo del sistema
- El comando lsmod muestra los módulos cargados del kernel y tiene el siguiente formato:

```
Module                  Size  Used by
fuse                   98304  3
iptable_filter         16384  0
```

Examen Septiembre 2018. Este problema debe realizarse junto con el anterior

11. La Universidad de Zaragoza, unizar, cuya red es `155.210.0.0/16`, desea conocer cuantos de sus nodos emplean Linux, su versión del kernel y el porcentaje de utilización de cada versión. La universidad sabe que todas sus máquinas Linux disponen de una cuenta con usuario user con privilegios de administración y que sólo las máquinas Linux de su red responden a ping. Escribe un script que escanee todos los nodos de la red de unizar, muestre por pantalla cuantos son Linux y la lista de versiones del kernel junto a su porcentaje de utilización entre los nodos Linux ordenada de mayor a menor. Por ejemplo:

```
68 nodos linux
4.9.0-5-amd64 55%
3.10.0-693.11.6.el7.x86_64 40%
3.10.0-514.el7.x86_64 5%
```

Notas:

- El comando bc permite realizar operaciones aritméticas con números reales. Ejemplo: `echo "5 % 2" | bc`
- El comando `uname -r` muestra la versión del kernel, 3.10.0-514.el7.x86_64, tal y como se muestra en el ejemplo anterior.

Examen Septiembre 2018

12. La computación *serverless* es un modelo de servicios en cloud en el que el proveedor es responsable del servidor y de la administración de los servicios y el cliente únicamente paga por los dichos servicios consumidos. Un administrador necesita un script en `bash` que siguiendo el modelo *serverless* permita ejecutar scripts de `bash` y ficheros binarios con ciertas restricciones en un nodo remoto.

    Podrías ayudar al administrador escribiendo dicho script cumpliendo la siguiente especificación:

    - El script serverless.sh tendrá un único parámetro que debe ser un programa ejecutable, es decir, otro script bash o un fichero binario. Para comprobar la validez del parámetro se deberá emplear el comando `file` y verificar que el tipo de fichero es `ELF 64-bit LSB shared object` o `Bourne-Again shell script`.
    - Para su ejecución, el fichero deberá ser copiado al nodo 192.168.56.1 donde en caso de que sea binario deberá comprobarse con el comando `ldd` que están instaladas todas las bibliotecas dinámicas necesarias. Para ello es necesario asegurar que no aparece '`not found` al ejecutar dicho comando. Si aparece, la ejecución deberá ser abortada.
    - Al copiar el fichero este deberá tener un nombre único para evitar sobreescrituras si varios clientes envían a ejecutar un script con el mismo nombre.
    - La salida del comando serverless.sh contendrá las salidas standard y de error del comando del cliente.
    - Para evitar sobrecargar el nodo, el script comprobará antes de lanzar el parámetro la carga del nodo 192.168.56.1 durante los últimos 5 minutos y si es mayor de .6, esperará 3 minutos antes de volver a lanzar el script o binario del parámetro.

    Notas adicionales:

    - El usuario `as` tiene acceso al nodo 192.168.56.1 mediante ssh y clave pública.
    - La salida del comando `file` es:


```
as@as: file /bin/ls
/bin/ls: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked
as@as: file serverless.sh
serverless: Bourne-Again shell script, ASCII text executable
```

    - Dado un fichero binario, la salida del comando `ldd` es:

```
as@as: ldd /bin/wrong_ls
ldd /bin/ls
linux-vdso.so.1 (0x00007ffe5479c000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007ffbf3106000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ffbf2d67000)
libwrong.so.6 => not found
```

    - El segundo campo de `/proc/loadavg` es la carga del sistema durante los últimos 5 minutos.

```
as@as: cat /proc/loadavg
0.01 0.02 0.00 1/75 645
```

Examen Junio 2019

13. Escribe un script en bash que se conecte a la máquina remota con IP 155.210.3.14 vía `ssh` y reduzca la prioridad a +15 a todos los procesos de los usuarios que tengan más de 10 procesos en estado running o slepping. Este script se ejecutará a las 6:15 los martes desde enero a junio y a las 15:16 los jueves de julio a diciembre y además enviará un email con asunto "lista de procesos" a una dirección que se recibirá como argumento del script (en caso invocar el script sin argumento, el email se enviará a `as@as.unizar.es`). Dicho email listará todos los procesos cuya prioridad haya sido modificada cuando la lista de procesos no esté vacía. La primera línea contendrá una lista ordenada alfabéticamente de los usuarios y debajo de cada usuario aparecerá el nombre de los procesos tal y como se muestra a continuación.

```
awinlock,eclarke,jbartik,mwescoff
sshd,sshd,vim,emacs
sshd,ps,bash,bash
bash,cron,dhclient
```

Notas adicionales:

- El usuario `as` dispone de acceso mediante clave pública al nodo 155.210.3.14 y de privilegios de administración sin password vía `sudo`.
- Cada columna de los procesos no es necesario que aparezca ordenada.
- La cabecera `comm` muestra el comando ejecutable de cada proceso en `ps`.
- Los estados running y sleeping se representan mediante una R y una S, respectivamente.
- El comando `test -s <fichero>` comprueba si un fichero está vacío.
- El comando `paste` permite concatenar columnas.

Examen Junio 2019

14. En la red 192.168.4.0/24 se sabe que todos sus nodos ejecutan linux y emplean el fichero `/etc/resolv.conf` para gestionar la resolución de nombres, DNS. Dicho fichero contiene líneas con el formato `nameserver <servidor>` que indican el servidor o los servidores a los que se lanzan las consultas DNS para que las resuelvan.

    Para reducir la latencia de dicho servicio se desea encontrar los servidores DNS más rápidos empleados por los nodos de la red. Para ello debes escribir un script que se conecte a todos los nodos de dicha red para leer sus servidores DNS. Después deberá emplear el comando `dig` para calcular su latencia y determinar el más rápido realizando una consulta a `www.unizar.es`. Una vez determinado el mejor servidor deberá enviar un mail al usuario root de todos los nodos que no lo estén empleando con asunto: "servidor dns" y cuerpo: "actualize su dns a <IP_servidor_más_rápido>". Además debes indicar como conseguir que el script se ejecute en días alternos a las 7 y 37 de la mañana.

    Notas adicionales:

    - El usuario `as` tiene acceso a todos los nodos de 192.168.4.0/24 mediante ssh y clave pública.
    - Ejemplo de fichero `/etc/resolv.conf`:

```
nameserver 8.8.8.8
nameserver 103.232.32.246
nameserver 46.18.43.8
```


    - La sintaxis del comando `dig` es `$ dig @<dns_server> <name_to_resolve>`. Por ejemplo `$ dig @8.8.8.8 www.unizar.es` resuelve la IP de `www.unizar.es` en el servidor DNS `8.8.8.8`.
    - Dig muestra la latencia de la consulta tal y como muestra el siguiente fragmento de su salida:

```
; <<>> DiG 9.10.3-P4-Debian <<>> @8.8.8.8 www.unizar.es

;; Query time: 14 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
```

Examen Septiembre 2019

15. Se desea obtener una herramienta que permita compartir ficheros encriptados mediante clave simétrica en un nodo local. La herramienta será un script de `bash` con la siguiente sintaxis `encript.sh <fichero_entrada> <usuario_destino>` y tendrá la siguiente especificación:

    - El script generará un fichero de salida llamado `<fichero_entrada>.enc` en el directorio actual.
    - Si el fichero a encriptar o el usuario_destino no existen se mostrará el siguiente mensaje de error: "Parámetros inválidos" y el código de salida del programa será 85.
    - La clave se generará leyendo 128 caracteres del dispositivo `/dev/urandom`.
    - Todas las claves generadas se almacenarán en el directorio `/var/keys`.
    - Cada par de usuarios empleará una clave única llamada `<usuario_origen>_<usuario_destino>.key`, dicha clave sólo sera generada cuando no exista.
    - Se creará un grupo llamado `<usuario_origen>_<usuario_destino>` para aumentar la seguridad de la herramienta. A dicho grupo pertenecerán la clave, cuando se genere, y el fichero de salida. El propietario de ambos ficheros podrá ser tanto el usuario origen como el destino.
    - Los permisos del fichero de salida y de la clave deberán ser únicamente lectura para el dueño y el grupo.
    - Para encriptar el fichero se empleara openssl cuya sintaxis es `openssl enc -aes-128-cbc -pass file:<cclave> -a -d -in <fichero_entrada> -out <fichero_salida>`.
    - Indique por favor tanto el comando a ejecutar para que encript.sh siempre corra con privilegios de administración como la razón si fuera necesario.

    Notas adicionales:

    - `head -c <n> <fichero>` lee `<n>` carácteres de `<fichero>`.

Examen Septiembre 2019

Parte II

Proyecto Investigador

Resumen

Esta parte del documento presenta un pequeño proyecto de investigación denominado *Procesamiento de Grafos en Sistemas Altamente Heterogéneos* y que aborda los problemas de la baja intensidad aritmética y de grandes requerimientos de sincronización del procesamiento por grafos desde la hipótesis que la combinación de distintos dispositivos de cómputo puede mejorar su procesamiento tanto en términos de velocidad, medida en vértices por segundo, como de eficiencia energética.

La premisa fundamental del proyecto es que compartiendo información entre la jerarquía de memoria, la red de interconexión y el *runtime*, se puede llegar a mejores diseños *hardware* y a obtener un mayor rendimiento y una mayor eficiencia energética. Por ejemplo, conocer la localización de unos datos en un chip puede elegir cual es el mejor dispositivo para su ejecución cuando el *runtime* puede elegir entre varios. Además incluye un resumen de los antecedentes investigadores del candidato y un posible esquema de trabajo para los próximos 3 años.

El capítulo 4 describe cual es el contexto actual dentro de arquitectura de computadores donde podría aplicarse este proyecto y motiva su realización mostrando los principales problemas que se pretender abordar. Además comenta los trabajos previos del candidato relevantes para este proyecto. Posteriormente, el capítulo 5 describe los objetivos del proyecto, y, finalmente, el capítulo 6 introduce la metodología y el plan de trabajo junto con las tareas y los resultados esperados del proyecto.

Capítulo 4

Contexto y Motivación

4.1 Introducción

De manera general, en matemáticas y, por ende, en informática, los grafos son estructuras que representan las conexiones, o relaciones, entre entidades. A las entidades se les conoce como vértices y a las conexiones como aristas. Debido a la abundancia de problemas actuales que pueden ser representados como grafos, como por ejemplo las relaciones entre páginas web [Pag+99], planificación de rutas [CGR94], descubrimiento de fármacos [Fle18], ...su rápido procesamiento es un reto de la computación actual, y en conferencias como IEEE High Performance Extreme Computing Conference se realizan concursos de procesado de grafos, `http://graphchallenge.mit.edu/`, con el objetivo de mejorar su procesamiento [Dre+16]. Desde un punto de vista de computación, los grafos presentan múltiples desafíos debido principalmente a las siguientes causas:

- **Poca intensidad computacional**: la mayoría de algoritmos sobre grafos realizan muy pocos cálculos sobre los mismos. Por ejemplo, *single-source shortest path*, SSSP, calcula el camino más corto desde un vértice al resto y puede implementarse realizando una suma, una comparación y a veces una actualización, escritura, por cada nodo visitado.
- **Gran volumen de sincronización**: Para ejecutar de manera concurrente algoritmos como SSSP, puede ser necesario serializar las actualizaciones, en cuyo caso se limita mucho el escalado de las implementaciones paralelas sobre todo sino cuentan con soporte hardware.
- **Poca localidad**: Según los algoritmos y la topología de los grafos, el orden en el que se atraviesan los vértices o aristas puede ser bastante aleatorio, lo que reduce tanto la localidad espacial como la temporal.

Esta 3 causas dificultan el procesamiento de grafos incluso para chips de última generación.

Durante los últimos años la industria de fabricación de chips ha entrado en un periodo conocido como la era posterior a la ley de Moore debido a que no se vislumbra un candidato claro a las tecnologías CMOS y se está llegando al final de su escalado [Bal+19]. De acuerdo al International Roadmap for Devices and Systems, en la actualidad nos encontramos en la edad del *escalado equivalente* que se caracteriza por reducir únicamente las dimensiones horizontales de los transistores y por la introducción de nuevos materiales y propiedades físicas [Bal+19]. Además, las

estructuras en 3 dimensiones reemplazan a los transistores planos como se está viendo con el caso de los FinFET. Dicho documento establece que en el año 2025 se producirá un nuevo cambio de edad que nos llevará a la edad del *escalado de potencia en 3D* en donde la mayor parte de las estructuras serán en 3 dimensiones y la reducción de consumo será el objetivo principal de diseño. Otro posible freno en la miniaturización de la escala de integración es el coste de fabricación de las salas blancas, ya que de acuerdo a la ley de Rock, o segunda ley de Moore, el coste de fabricación se duplica cada 4 años [Ros03; RS11].

Ante estos cambios tecnológicos la respuesta de la industria ha sido la especialización del hardware, ya que por un lado los límites del paralelismo a nivel de instrucción junto con el ya mencionado estancamiento tecnológico han hecho que los procesadores de propósito general estén compuestos por múltiples *cores* y que las plataformas de computo, bien sean integradas o discretas, dispongan de aceleradores en forma de GPUs, FPGAs, o ASICs. Esta especialización no esta exenta de dificultades ya que los aceleradores, en sentido amplio, suelen ser difíciles de programar y poco o nada flexibles para ejecutar de manera eficiente código que difiera ligeramente de aquel para el que han sido concebidos.

Volviendo a las aplicaciones objeto de estudio de este proyecto y de cara a su aceleración, una característica importante de los grafos es el grado, número de aristas incidentes, de sus vertices. Según la topología, la distribución del grado es muy cambiante complicando el diseño de aceleradores hardware para grafos debido a la complejidad por un lado repartir el grafo en sub-grafos para su procesado y porque dependiendo del algoritmo a procesar, la actualización de alguna propiedad de los vértices se puede ver serializada. Por ejemplo, al ejecutar el algoritmo de búsqueda del camino más corto, SSSP, en el grafo de la Figura 4.1, si se realizaran 2 particiones basadas únicamente en el identificador de vértice ([1-4] y [5-8]), el trabajo a realizar quedaría desbalanceado. Además, según el algoritmo podría darse el caso que partiendo del vértice 1, la actualización de la distancia al vértice 5, requiriera desde 2 a 4 actualizaciones.

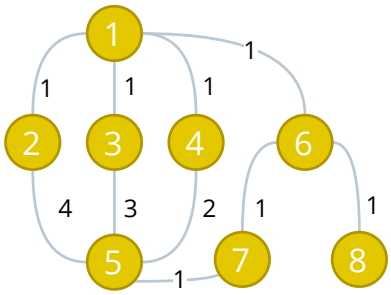

Figura 4.1: Ejemplo de grafo con 8 vértices y 10 aristas. El número en cada una de ellas representa su coste

Como se ha visto, la distribución del grado puede ser muy variable lo que dificulta el reparto a priori el trabajo entre varios dispositivos y este será uno de los problemas a tratar en este proyecto. Además, como ya se ha mencionado, los algoritmos de grafos tienen una intensidad aritmética pequeña y a veces puedes requerir de múltiples operaciones de sincronización en datos con poca localidad por lo que para su aceleración se requiere una orquestación completa para el *software* y el *hardware* trabajen en conjunto sobre los grafos.

En resumen, este proyecto aborda el desafío de acelerar aplicaciones con grafos, cuyas características principales son la baja localidad de memoria, posible gran cantidad de operaciones de sincronización, y el bajo ratio entre cómputo y comunicación, en plataformas heterogéneas. De

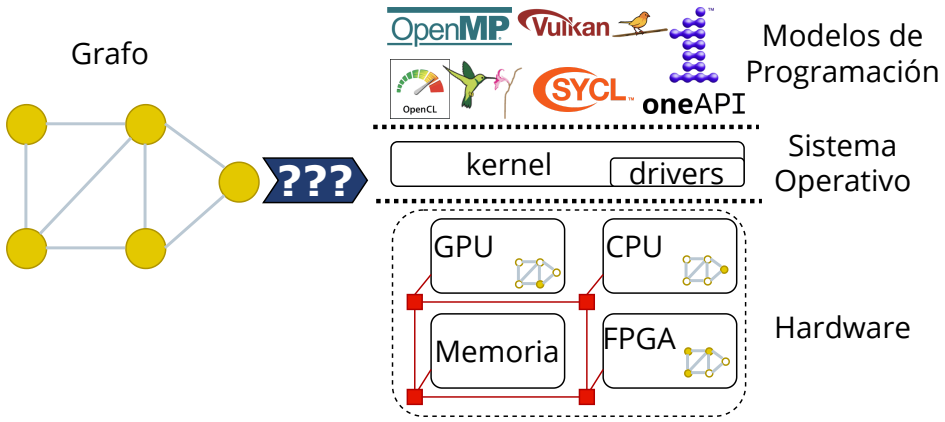

Figura 4.2: Esquema general del proyecto que buscar responder a la pregunta: ¿cómo podemos operar de manera efectiva grafos con sistemas heterogéneos?. Para cada dispositivo, se muestra un posible reparto de nodos en una hipotética ejecución heterogénea.

acuerdo a la Figura 4.2, para afrontar este reto se trabajará desde el modelo de programación hasta los distintos dispositivos para intentar buscar una solución completa que facilite sinergias entre los distintos niveles.

El tamaño de los grafos puede ser de tal magnitud que también se han propuesto multitud de estrategias para su procesamiento en sistemas distribuidos [MWM15]. Por ejemplo, la red neuronal ResNet-50 visualizada en la figura 4.3 tiene más de 3 millones de vértices. Debido al gran alcance que supone cubrir el procesamiento en múltiples nodos, este proyecto de investigación se centra en exclusiva en sistemas con un único nodo compuesto por varios dispositivos de cómputo heterogéneos.

4.2 Estado del Arte

Esta sección realiza una breve recopilación de trabajos sobre aceleración de grafos organizados en 2 grandes bloques de acuerdo a sus contribuciones, ya sean software y hardware. En el primer bloque también se incluyen los modelos de programación que podrían emplearse en este proyecto.

4.2.1 Software

El uso de abstracciones de programación cada vez más altas consigue que aumente la productividad de los programadores y permite una migración, en principio más fácil, entre plataformas y/o arquitecturas. Centrándonos en plataformas móviles, Google ofrece a sus desarrolladores Java como lenguaje principal pero además Renderscript para aplicaciones que requieran más potencia computacional y Vulkan para gráficos en 3 dimensiones [Goo16a; Goo16b]. Otro gran fabricante como Apple fue uno de los primeros impulsores de OpenCL para computación heterogénea [Gas+11] a la par que ofrecía Grand Central Dispatch (GCD) para mejorar la ejecución de código concurrente en iOS y OS X. Incluso uno de los últimos lenguajes de programación propuesto por Apple, Switf, para iOS, OS X, tvOS y watchOS, expresa la concurrencia con los mecanismos de GCD [Gro16].

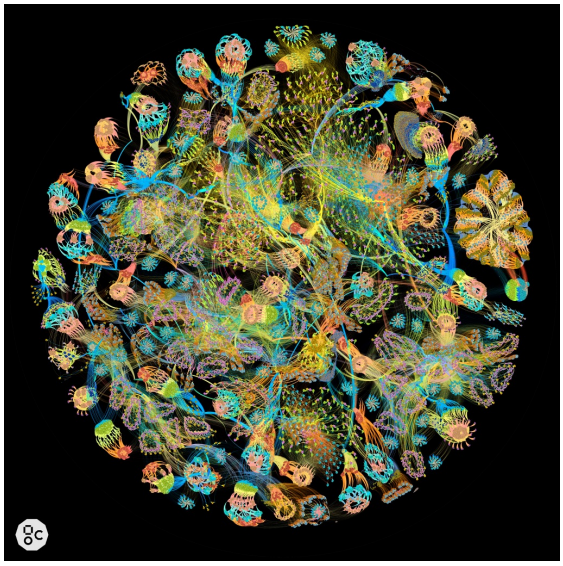

Figura 4.3: Disposición completa del grafo de entrenamiento de ResNet-50 compuesta por alrededor de 3 y 10 millones de vértices y de aristas, respectivamente. © Matt Fyles, Graphcore.

Estos modelos al igual que otros como Intel Thread Building Blocks, TBB, Intel Cilk Plus o Qualcomm Heterogeneous Compute SDK facilitan la tarea de los desarrolladores a la hora de aprovechar la potencia de los sistemas paralelos y heterogéneos [Cor16a; Cor16b; Inc16]. Los ejemplos anteriormente citados se circunscriben al entorno móvil pero esta tendencia cubre un amplio espectro pasando por los ordenadores personales con entornos como C++ Accelerated Massive Parallelism, C++ AMP, de Microsoft [Cor16c], e incluso llegando hasta los supercomputadores con propuestas como las directivas Open ACC [Con16] de Cray entre otros para acelerar principalmente utilizado procesadores gráficos.

Además, otros entornos clásicos de programación paralela como OpenMP están abrazando esta tendencia y además de soporte al paralelismo también dan soporte a entornos heterogéneos y aceleradores [Boa15]. No solo la industria sigue esta tendencia sino que también existen propuestas provenientes de ambientes académicos como OmpSs, StarPU, o EngineCL [ATN10; Cen13; NBB20]. De hecho las últimas tendencias de la industria intentan que se puedan emplear aceleradores sin tanto *boiler-code* como sucede con OpenCL y un único fichero fuente tal y como hace Intel oneAPI, `https://software.intel.com/en-us/oneapi`, o SYCL de Khronos [KRH15b].

Para trabajar con bases de datos de grafos se suele emplear *Bulk Synchronous Parallelism* (BSP). Este modelo se estructura en 3 elementos: nodos que realizan las operaciones, una red para encaminar mensajes entre pares de nodos, y finalmente soporte para la sincronización. La ejecución de un algoritmo basado en BSP es iterativa y consta generalmente de 3 pasos: computación concurrente, comunicación y sincronización mediante barreras donde todos los nodos se sincronizan hasta realizar la siguiente iteración [Val90]. Siguiendo este modelo hay múltiples bases de datos como Google Pregel, Apache TinkerPop o Neo4j [Mal+10; Rod15; Web12].

A medio camino entre los lenguajes de propósito general y los de propósito específico, DSL por sus siglas en inglés, se encuentran entornos como Galois [NLP13; Pin+11] o GraphMap [Sun+15] que permiten implementar de manera sencilla complejos algoritmos para grafos

a la vez que soportan sofisticados planificadores para reordenar tareas de grano muy fino y de este modo alteran el orden de procesamiento de vértices y/o aristas. Otros *frameworks* como GraphGen proponen un DSL para traducir los grafos a una representación intermedia, GEM, y desde allí generar código para los dispositivos como FPGAs [Nur+14].

Este proyecto plantea el uso de lenguajes y modelos de programación específicos para sistemas heterogéneos como oneAPI o SYCL ya que ofrecen un mejor soporte para FPGAs comparados con otros DSLs más orientados a procesadores de propósito general.

4.2.2 Hardware

Desde el hardware se han realizado también multitud de propuestas para acelerar el procesado de grafos. Estas pueden dividirse según el tipo de dispositivo empleado, ya sea una CPU, GPU, FPGA o ASIC. Para los 2 primeros dispositivos, Aasawat *et al.* y Shi *et al.* resumen muchos trabajos realizados hasta la fecha [ARR18; Shi+18]. Este proyecto esta más orientado hacia la aceleración en FPGA, por lo que está sección está centrada tanto en ASICs como en FPGAs ya que son las familias más próximas.

Dentro de los ASICs, Graphicionado es un acelerador de dominio específico que combina la especialización del camino de datos y del acceso a memory junto con pequeños bloques reconfigurables para permitir la ejecución de distintos algoritmos de grafos [Ham+16]. Ozdal *et al* proponen una arquitectura base para acelerar grafos que puede ser ajustada mediante la inserción de estructuras de datos y funciones propias de cada algoritmo dentro de las plantillas de SystemC propuestas. Su diseño está orientado a solventar las limitaciones del rendimiento en CPU y GPUs debidas fundamentalmente a la convergencia asimétrica, la ejecución asíncrona, los cuellos de botella en los accesos a memoria y el desbalanceo de carga ya que los grafos reales suelen seguir distribuciones basadas en ley potencial [Ozd+16].

Para grafos de grandes dimensiones también se han propuesto soluciones basadas en múltiples FPGAs como ForeGraph [Dai+17]. Siguiendo el modelo de Microsoft Catapult [Put+14], ForeGraph permite conectar multiple processing elements, PE, que trabajan en porciones de los grafos de manera paralela. Para la obtención de resultados, se emplea un modelo analítico ya que la propuesta únicamente evalúa la implementación de un PE en una FPGA con 2GB de memoria DRAM DDR4 y para el coste de los accesos a memoria se emplea DRAMSim2 [RCJ11]. Al contrario, HitGraph emplea una única FPGA y busca aumentar la reutilización de datos y un aumento del paralelismo mediante estructuras de datos que reducen los accesos a memoria externa no secuenciales, un esquema para evitar recorrer varias veces la misma arista en algoritmos no-estacionarios y mecanismos propios para agrupar y filtrar actualizaciones hacia memoria [Zho+19].

Dentro de las soluciones heterogéneas, GridGAS propone una plataforma de computación CPU+FPGA específicamente optimizada para grafos extremadamente grandes [ZL18a; ZL18b]. La plataforma emplea un modelo de computación centrado en las aristas y busca minimizar los accesos de entrada/salida. La heterogeneidad viene dada porque la CPU es responsable de leer los datos del grafo desde el disco duro en la fase de *scatter* y enviárselos a la FPGA para después, en la fase *gather* recibir los resultados de la FPGA, con lo que todo el cálculo intensivo se realiza en la FPGA. También hay propuestas para emplear nuevas tecnologías de memoria con gran ancho de banda como *Hybrid Memory Cube*, HMC, para acelerar grafos. Khoram *et al.* se centran en un algoritmo de búsqueda en anchura y mejorar el rendimiento empleando la memoria HMC desde el software al agrupar elementos para aumentar la localidad y desde el hardware con una unidad de acceso a memoria capaz de aprovechar la mayor localidad producida por el agrupamiento [Kho+18; ZKL17].

4.3 Antecedentes y Trabajo Previo del Candidato

Esta sección describe los resultados de investigación del candidato relacionados directamente con el proyecto. Estos cubren el periodo de la tesis doctoral y postdoctoral en posiciones académicas y en empresa. Como los objetivos del proyecto cubren aspectos de modelos de programación, *runtime* y también de microarquitectura y diseño, esta sección se dividirá en 2 partes. Siguiendo el order cronológico del grueso de los trabajo, se comenzará presentando los trabajos centrados en la microarquitectura en la subsección 4.3.1 y se continuará con la subsección 4.3.2 para los trabajos centrados en modelos de programación y paralelismo. Como la eficiencia energética se ha abordado en ambos bloques, los trabajos en consumo se presentarán según su próximidad a ambas áreas.

4.3.1 Jerarquía de Memoria y Redes

La tesis doctoral del candidato dirigida por los Profs. Víctor Viñals Yúfera y Teresa Monreal Arnal centró sus esfuerzos en los primeros niveles de la jerarquía de memoria proponiendo una cache llamada light *Non Uniform Cache Architectura*, NUCA, para aprovechar al máximo la localidad temporal [Suá+09].

El control fino de la localidad temporal es posible gracias al uso de teselas que combinan en un ciclo de procesador el acceso a un pequeño banco de cache, entre 8 y 32 KiBytes, y un salto de red. La latencia de la red es muy pequeña gracias a solapar parte de las actividades de la red con las de la memoria y gracias a la utilización de múltiples redes especializadas en cada una de las actividades de la cache. Este diseño fue implementado en un proceso de 90nm disponible para universidades europeas para probar su viabilidad y durante su diseño se añadieron mecanismos estáticos para reducir el consumo ya que cuando no hay localidad la continua migración de bloques entre teselas penaliza el consumo energético [Gra+12]. Este nuevo diseño fue denominado *Light Power NUCA*, *LP-NUCA*.

Posteriormente el candidato trabajo en mejorar el consumo dinámico de las *LP-NUCA* estudiando el tipo de acceso a las teselas, serie o paralelo, y proponiendo un controlador basado en la búsqueda del montículo para detectar fases de poca localidad y evitar que los bloques sin localidad contaminaran la cache [Suá+14].

Los conocimientos adquiridos durante la realización de la tesis sirvieron para explorar el impacto del consumo energético en procesadores en varios proyectos final de carrera que resultaron en publicaciones docentes [ASY07; Gut+09]. La primera de ellas obtuvo en el año 2007 el premio al mejor artículo en el Workshop on Computer Architecture Education.

Dentro del periodo post-doctoral se continuó trabajando en jerarquía de memoria y en redes *on chip*. En la primera temática, los trabajos propusieron caches de instrucciones teseladas [Fer+13], y luego técnicas para tolerar fallos en caches de último nivel alimentadas a voltajes cercanos a la tensión umbral [Fer+14; Fer+16; Fer+19]. En la segunda temática, redes, se abordó el estudio del impacto de las topologías de red en el rendimiento [Ort+16a] y como se pueden aprovechar los patrones de acceso a memoria definidos por su modelo *petición–respuesta* para generar caminos de vuelta más rápidos basados en circuitos virtuales [Ort+14; Ort+16b].

4.3.2 Modelos de programación, *Runtimes* y Aplicaciones

El candidato tuvo la suerte de contar con un permiso a efectos de investigación por parte de la Universidad de Zaragoza para realizar su labor investigadora en una empresa. Durante este

periodo de 3 años, su labor se centro en el modelo de programación y *runtime* Qualcomm Heterogeneous Compute SKD [Inc16]. Aquí fue responsable, entre otros, del Interfaz de programación de aplicaciones, API, de consumo de Heterogeneous Compute SDK y de la ejecución de tareas en el procesador digital de señal, DSP, Qualcomm Hexagon. Las APIs de consumo ya permiten al programador especificar sus requerimientos de calidad de servicio y pueden considerarse como parte de la semilla para este proyecto. Estas APIs permiten por un lado elegir el modo estático de funcionamiento de los procesadores y la unidad de procesamiento de gráficos, GPU. Es decir, cuantos procesadores y o GPUs queremos activas y a que frecuencia y por otro lado puede regular de manera dinámica ambos parámetros en los procesadores si el programador especifica un objetivo de rendimiento y la aplicación muestra un cierto patrón repetitivo en la ejecución.

El estudio de la ejecución de tareas en Heterogeneous Compute SDK intentando acelerarlas sirvió para la solicitud de varias patentes internacionales [KNS18; Suá+16; Suá+18; ZSK19] y para vislumbrar ciertos puntos en donde una comunicación más fluida entre el *hardware* y el *software* serviría para mejorar la eficiencia energética. La colaboración con usuarios de Heterogeneous Compute SDK sirvió para estudiar sus cargas de trabajo y ver otros cuellos de botella. Una de estas cargas de trabajo fue los navegadores de internet donde se estudio su concurrencia y eficiencia energética [Cas+15; Sam+12].

En fechas recientes, y en colaboración con los grupos de arquitectura de computadores de las universidades de Zaragoza, Cantabria, Bristol y Málaga se ha continuado trabajando en la planificación de cargas de trabajo dentro de *runtimes*, bien sea para soportar mayores niveles de heterogeneidad (empleando de manera concurrente CPUs, GPUs, y FPGAs) [Guz+18; Guz+19], y para realizar la planificación con CPUs y FPGAs en sistemas embebidos [Núñ+19; Rod+19a] y en chips integrados con CPUs y FPGAs, como Intel HARP, [Rod+19b].

Además, en los últimos años, se han abordado problemas que unen las 2 grandes líneas anteriores y que vislumbran el interés del candidato por el estudio de problemas "verticales" que cubran desde las aplicaciones hasta la implementación hardware. En concreto, se han propuesto mecanismos para tolerar el envejecimiento del banco de registros de GPUs mediante la compresión de datos de acuerdo a los patrones de acceso más empleados en modelos de programación para GPU como OpenCL [Can+17; Val+19].

También el candidato dispone de experiencia profesional trabajando en estos temas ya que durante el verano de 2018 estuvo trabajando como investigador visitante en la empresa Bigstream, `https://www.bigstream.com`, dedicada a la aceleración de plataformas de *big data* como Apache Spark.

Capítulo 5

Objetivos

5.1 Descripción General

El objetivo principal de este proyecto es analizar y proponer mejoras para el procesado de grafos en sistemas heterogéneos en términos de rendimiento, eficiencia energética y productividad, entendida esta como la complejidad requerida para escribir el código. Uno de los requisitos para asegurar el aprovechamiento de un sistema heterogéneo es garantizar la eficacia durante la computación de cada uno de los dispositivos que lo componen y de la comunicación entre ellos. Este objetivo requiere trabajar de manera simultanea y cooperativa en varios ámbitos desde la implementación hardware al software de planificación de alto nivel que ayude a la productividad de los programadores. En concreto, este proyecto plantea cuatro objetivos específicos para intentar mejorar el estado del arte en el procesamiento de grafos:

- Procesado eficiente de grafos en FPGA
- Procesado eficiente de grafos en CPU y GPU
- Planificación heterogénea para grafos
- Soporte hardware a la planificación heterogénea

5.2 Objetivos Específicos

O1: Aceleración de grafos en FPGA

Los excelentes resultados de la multitud de propuestas sobre aceleradores grafos bien sean ASICs o FPGAs permiten afirmar que estas plataformas son adecuadas para este tipo de problemas [Che+19; Ham+16; Kho+18; Muk+18; ZL18b].

Nuestro trabajo dentro de esta linea se centrará en estudiar las capacidades aritmético-lógicas de las FPGAs para ver como permiten codificar la información de los vértices y las aristas en modos que permitan realizar múltiples operaciones de actualización de las propiedades de los vértices de manera paralela. Además, se estudiarán mecanismos de reordenación para procesar los vértices o las aristas capaces de aumentar el paralelismo. Para ello se aprovechará la experiencia obtenida en años recientes optimizando aplicaciones en FPGA [Guz+19; Rod+19a; Rod+19b].

La segunda área de interés dentro de este objetivo específico es el aprovechamiento de nuevas memorias como *High Bandwidth Memory* (HBM) ya que ofrecen un ancho de banda mucho mayor que las memorias DDR tradicionales a través del acceso concurrente a múltiples bancos. Para ello se dispone de una tarjeta Intel Stratix X MX que cuenta con 32GB de memoria HBM y buscaremos codificar la información de los grafos de tal modo que sea útil para realizar las operaciones y también para su acceso y actualización aprovechando estas memorias. En conjunción con el objetivo siguiente, se analizará la manera de minimizar las comunicaciones entre aceleradores, ya que la mayoría de los algoritmos de grafos no son intensivos en cálculo y la comunicación puede volverse el primer factor a optimizar.

O2: Aceleración de grafos en CPU y GPU

La ejecución heterogénea de grafos en las que cada dispositivo procesa una parte de los vértices o de las aristas require que todos ellos obtengan un rendimiento razonable y que la cantidad de trabajo sea tal que un único dispositivo no sea más eficaz que varios de ellos distintos. Al contrario, en el caso de *pipelines*, donde el reparto se realiza por dispositivos y no por vértices/aristas, se busca que cada dispositivo tenga un rendimiento excelente en su tarea o tareas concretas a realizar.

Este proyecto pretende hacer más hincapié en las FPGAs como dispositivo acelerador, pero es necesario que la ejecución en CPU y GPU sea optimizada para poder alcanzar resultados realistas. Para poder balancear, es necesario que la velocidad de los distintos dispositivos no difiera en demasía. Esta tarea buscará soluciones existentes de optimización de grafos en CPU y GPU, ya que hay multitud, y las optimizará en el caso de que sea necesario. Dentro de este objetivo específico, y pensando en reducir el coste de la comunicación entre dispositivos, se estudiará el empleo de tecnologías como Nvidia GPUDirect RDMA [1] para que la FPGA pueda leer y escribir directamente en la memoria de la GPU. En este sentido, el uso de codificaciones de grafos que minimicen el espacio en memoria y por tanto el coste de la comunicación también será estudiado.

O3: Planificadores para grafos en sistemas CPU, GPU y FPGA

Este objetivo consiste en desarrollar nuevos algoritmos de planificación orientados a grados aprovechando las características de los modelos de programación recientes como SYCL y/o oneAPI para conseguir un código portable a la vez que altamente eficiente [KRH15a]. De este modo se buscará aumentar la productividad de los programadores.

Debido al gran coste de enviar trabajo a los aceleradores discretos, como las FPGAs, se buscaran estrategias de planificación que por un lado minimicen el número de lanzamiento de kernels sobre ellos y que a la vez no requieran de *work-stealing* ya que su coste en este tipo de plataformas suele ser grande. Debido a que las plataformas de experimentación constan de aceleradores discretos, los planificadores tendrán muy en cuenta el coste de la comunicación y la ubicación de los datos de cara a repartir el trabajo.

También queremos afrontar el estudio de como debe realizarse el reparto de trabajo entre los dispositivos, o bien, mediante un *pipeline* entre ellos para que cada dispositivo realice una parte del algoritmo para todos los vértices del grafo, por ejemplo, simulando un esquema bulk synchronous parallel model se podrían emplear FPGAs y GPUs para calcular propiedades de los vértices y luego emplear las CPUs para realizar la sincronización/actualización de las propiedades debido a que su soporte para operaciones atómicas es mejor. Por el contrario, se podría pensar en otra

[1] `https://docs.nvidia.com/cuda/gpudirect-rdma/index.html`

estrategia en la que se repartan los vértices entre dispositivos y cada dispositivo realize las tareas de computo y actualización de las propiedades de sus vértices asignados. Se buscaran planificadores, similares a Galois [Pin+11], en los que la prioridad de los vértices sea variable y pueda ser calculada por el propio planificador ya que el orden de procesamiento de los vértices puede tener un gran impacto en el rendimiento final. En ambos casos, se podrían emplear modelos de programación basados en tareas como oneAPI, OmpSs, … [Cen13; Inc16]

Gracias al soporte de medición de energía por parte de los dispositivos de manera continua en tiempo de ejecución es posible pensar en políticas que no solo optimicen tiempo de ejecución sino también energía y/o potencia media. En general, las mejores políticas para el tiempo de ejecución también lo son para la energía total, pero pueden aumentar la potencia media. O bien el dispositivo más rápido es también el que más consumo, o la energía total tan apenas cambia pero si lo hace el tiempo de ejecución. En entornos, como un centro de datos cuyo consumo pueda estar limitado, es útil fijar una potencia media como objetivo del planificador tal y como por ejemplo hace el modelo Intel *Running Average Power Limit*, `https://01.org/blogs/2014/running-average-power-limit-%E2%80%93-rapl`.

O4: Soporte *hardware* en planificadores para grafos

Con granularidades de trabajo pequeña, es decir, cuando las operaciones a realizar por cada vértice o arista visitada son pocas, la sobrecarga de la planificación puede suponer un obstáculo insalvable para la ejecución heterogénea. Una posible solución es ayudar al planificador mediante soporte *hardware* para acelerar las tareas más críticas. Ya existen propuestas industriales y académicas para intentar resolver este problema [KNS18; Mor+19; Suá+16; Suá+18; ZSK19]. Estas aproximaciones se pueden dividir en 2: aquellas que buscan acelerar la sincronización entre los dispositivos, por ejemplo los *mutex* y aquellas que intentan acelerar las operaciones sobre datos compartidos, por ejemplo, mediante colas FIFO punto a punto entre dispositivos.

La aproximación final elegida se deja abierta y se concretará cuando se hayan realizado los primeros experimentos sobre plataformas reales ya que los cuellos de botella donde el soporte *hardware* podría ser de más ayuda son desconocidos. En principio, mejorar la sincronización ayudaría a permitir planificadores operando con grano más fino y sería muy útil en los planificadores que repartan los vértices y las aristas entre dispositivos. Por otro lado, unas comunicaciones de baja latencia y muy alto ancho de banda ayudarían en caso de emplear un modelo *pipeline*. Además, en ambos casos, es muy posible que la topología exacta de los grafos afecte en ambos casos y que se pueda añadir soporte *hardware* por ejemplo para trabajar con vértices de grado pequeño.

Capítulo 6

Metodología y Plan de Trabajo

Este capítulo consta de dos secciones. La sección 6.1 describe la metodología a emplear y la sección 6.2 detalla las tareas y los objetivos esperables del proyecto.

6.1 Metodología

Como es habitual en este tipo de proyectos, se seguirá una metodología cuantitativa en la la observación detallada del comportamiento de las aplicaciones sobre grafos a estudiar permita luego determinar patrones que permitan por un lado acelerar su ejecución y/o reducir el consumo energético en los distintos dispositivos de computo y por otro proponer estrategias de planificación eficaces junto con el soporte hardware necesario para ayudar en dicha planificación. Tres de los cuatro objetivos específicos requieren de plataformas reales de experimentación y el cuarto si que posiblemente empleará herramientas de simulación, excepto si el soporte *hardware* puede ser implementado en la propia FPGA que hace de dispositivo de cómputo.

6.1.1 Cargas de Trabajo

Existen multitud de algoritmos y benchmarks para grafos. Para supercomputadores, el sempiterno Graph500 incluye dos de los problemas más empleados con grafos, la búsqueda en anchura (BFS) y el camino más corto desde una única raíz (SSSP) [Mur+10]. Tanto BFS como SSSP están incluidos en otros conjuntos de benchmarks como Chai [Góm+17]. Otros conjuntos como *Pannotia* incluyen algoritmos empleados habitualmente en grafos como su coloreado, Page Rank, Floyd-Warshall,... [Che+13], y realizan su análisis sobre plataformas GPU. Más orientada hacia CPUs se encuentra el conjunto de benchmarks CRONO que ofrece la mayoría de los algoritmos citados anteriormente además de la búsqueda en profundidad [Ahm+15]. También GAP de la Universidad de California en Berkeley ofrece benchmarks para grafos con sus datos de entrada de referencia [BAP15]. Otros autores han partido de GAP para realizar mejoras del procesado de grafos en CPU [Eye+18].

Gracias a la amplia disponibilidad de implementaciones de algoritmos de grafos en distintos lenguajes de programación, será más fácil obtener código optimizado para cada uno de los dispositivos y, a priori, los esfuerzos mayores deberán realizarse en conseguir que cada una de estas

implementaciones optimizadas emplee el mismo formato de datos de entrada y de salida para poder realizar la ejecución heterogénea.

6.1.2 Herramientas para la Caracterización y el Análisis de las Cargas de Trabajo

El grupo gaZ dispone de varias plataformas heterogéneas con CPUs, GPUs, y FPGAs para realizar los experimentos reales. En concreto 2 nodos singulares, uno con un Intel Core i7, una GPU Nvidia Titan X y una FPGA Intel Stratix V y otro con un procesador Intel®Xeon®Bronze 3204 y 2 FPGAs, una Intel Stratix X GX y otra Stratix X MX con memoria HBM. Esta plataforma dispone de canales suficientes de PCI express para que todos los dispositivos puedan comunicarse al menos a velocidad PCIe x16 y también podría acoger una GPU adicional. Para realizar mediciones de potencia, estás se realizarán mediante contadores hardware en todos los casos, excepto en la FPGA Intel Stratix V que require de un vatimetro externo [1].

6.1.3 Herramientas de Simulación

A continuación se describen los simuladores que se podrían emplear de cara a evaluar las propuestas sobre soporte hardware para la planificación ya que es posible que no pudieran evaluarse sobre hardware real.

- **gem5** Surge de la combinación de 2 simuladores más antiguos: M5 y GEM [Bin+06; Mar+05] gracias a la colaboración de las universidades americanas de Michigan, Texas y Wisconsin, Princeton, MIT y las empresas AMD, ARM, HP y MIPS [Bin+11]. Soporta una buena parte de los repertorios de instrucciones mas utilizados en la actualidad: ARM, Alpha, MIPS, Power, SPARC y x86 y permite simular un sistema completo, de hecho arranca Linux y Android. En ese último caso se puede utilizar NoMali para emular el trabajo de rendering que haría un procesador gráfico o gem5-gpu [Pow+14]. Su mayor limitación es que los modelos de ejecución de los cores no están muy desarrollados y es difícil por ejemplo simular una máquina con soporte de ejecución fuera de orden en algunas arquitecturas.
- **ESECS** es un simulador de multiprocesadores con modelos detallados de potencia, temperatura y consumo capaz de modelar modernos procesadores con ejecución fuera de orden [KR13]. ESECS toma como base el simulador SESC [Ren+05]. Soporta la arquitectura MIPS64r6 y permite trabajar con configuraciones heterogéneas. Además de modelar en detalle la microarquitectura del procesador: buffer de reordenación, ventana de lanzamiento, la jerarquía de memoria incluyendo los controladores de memoria principal [KR13]. Para el cálculo del consumo energético integra el simulador McPAC [Li+13].

Además de los simuladores citados, se podrían desarrollar simuladores funcionales del soporte hardware propuesto para posteriormente realizar la obtención del retardo, consumo y área con herramientas de síntesis como Synopsys Design Compiler, [2].

[1] Además del vatímetro se emplea una tarjeta elevadora del bus PCI express que fue amablemente donada por Luis Piñuel Moreno de la Universidad Complutense de Madrid.

[2] `https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-nxt.html`

6.2 Plan de Trabajo y Descripción de Tareas

En función de la metodología anteriormente descrita, el plan de trabajo propuesto está organizado en 3 grandes tareas que cubren los 4 objetivos planteados originalmente. La primera tarea engloba la aceleración de los algoritmos de grafos en dispositivos, la segunda se decida a la planificación, primero sin y luego con soporte hardware, y finalmente hay una tarea transversal dedicada a la difusión de los resultados.

6.2.1 Tarea 1: Aceleración de Grafos

Sub-tarea 1.1: Análisis de Cargas de Trabajo Para los principales algoritmos de grafos se analizarán en detalle sus propiedades, incluyendo sincronización, localidad, tipos de operaciones,… para su posterior clasificación según los patrones que muestren y así buscar las mejoras estrategias de implementación que se abordarán en las sub-tareas siguientes. Además, se definirán unas API, *Application Programming Interface*, a seguir por las implementaciones de los algoritmos tanto en FPGA como en CPU y GPU, de cara a facilitar posteriormente las implementaciones heterogéneas. Las interfaces constarán de elementos para invocar kernels en dispositivos y buffers para compartir datos y/o especificar dependencias.

Sub-tarea 1.2: FPGA

Conocidas las propiedades de los grafos se planteará cual es la mejor implementación, si emplear múltiples pipelines independientes que procesan los vértices y las aristas o si es mejor dedicar recursos hardware a realizar algún tipo de reordenamiento de los mismos para luego evitar detenciones en el pipeline de la FGPA y/o el uso de recursos extra. A continuación, se estudiará la viabilidad de aumentar el ancho de banda en la actualización de las propiedades de los vértices de manera paralela mediante el empleo de memorias HBM. En todo momento se evaluarán que algoritmos tienen mayores dependencias y serían los mejores candidatos para su ejecución heterogénea.

Durante toda la sub-tarea, se asegurará que las implementaciones siguen las APIs definidas en la sub-tarea anterior. De cara a la productividad, se revisarán distintos lenguajes de programación como oneAPI o SYCL para comprobar si hay algún compromiso entre rendimiento y productividad.

Sub-tarea 1.3: CPU y GPU

Esta sub-tarea explorará las distintas excelentes alternativas que hay sobre aceleración de grafos en CPU y GPU para buscar implementaciones que posteriormente permitan una ejecución heterogénea después de ser modificadas para ser adaptadas a las APIs propuestas en la primera sub-tarea. En dichas implementaciones, se buscará que el código tolere su descomposición en distintos elementos para posibilitar las implementaciones heterogéneas.

Resultados Esperados

- Definición de un pequeño conjunto de APIs independientes del dispositivo.
- Implementación eficiente de algoritmos de grafos en FPGA mediante propuestas en el recorrido de grafos que aprovechen el paralelismo masivo de las FPGAs.
- Implementación competitiva de algoritmos de grafos en CPU y FPGA.

- Análisis detallado de las posibilidades de planificación heterogénea para los algoritmos estudiados.

Es muy posible que se trabaje de manera iterativa entre las sub-tareas hasta alcanzar los resultados esperados ya que se aprovechará el conocimiento aprendido trabajando con cada una de las plataformas en el resto.

6.2.2 Tarea 2: Planificación

Sub-tarea 2.1: Planificadores

Debido a la gran tamaño de algunos grafos, por ejemplo, el grafo social de los usuarios activos por mes de Facebook tenía 2.5 miles de vértices en diciembre de 2019 [3], una estrategia de aceleración es el empleo simultaneo de varios dispositivos, ya sea mediante *pipelines* entre dispositivos o el reparto de porciones de un grafo entre dispositivos.

El gran reto para conseguir una ejecución heterogénea eficiente estriba en trabajar con tareas de grano muy fino y con una intensidad computacional muy pequeña. La actualización de la propiedad de un nodo, por ejemplo, su distancia al vértice raíz, conlleva como mucho una suma por vértice pero siempre requerirá un acceso sin patrón conocido a memoria. Por lo tanto se buscaran *runtimes* cuya sobrecarga an generar pipelines y/o lanzar pequeñas tareas sea lo menor posible. Entre las posibles candidatas estarían EngineCL, SYCL o oneAPI, por ejemplo, mediante `cl::sycl::pipe` en SYCL para conectar dispositivos.

El otro aspecto crítico será la capacidad del planificador de evitar conflictos en las actualizaciones de los nodos por lo que se buscarán planificadores capaces de reordenar los accesos a los vértices de manera similar a Galois [Pin+11] o se buscará algún mecanismo de reordenamiento muy ligero pero que ayude a la planificación sin necesidad de recorrer varias veces cada grafo. En este caso es posible que se vislumbre el potencial de soluciones *hardware* cuyo soporte aceleraría el pre-ordenamiento, la propia planificación o ambos. Dichas soluciones serán evaluadas en entornos sintéticos o en las propias FPGA si fuera posible.

Sub-tarea 2.2: Soporte *hardware*

Resultados Esperados

- Análisis detallado sobre estrategias de reparto de grafos. En concreto, entre modelos *pipeline* en los que cada dispositivo se especializa en una operación precisa o modelos con paralelismo a nivel de elemento (vértice o arista).
- Planificador multi-FPGA y/o CPU-GPU-FPGA para la aceleración de grafos.
- Propuesta de mecanismos *hardware* orientados a mejorar la planificación en términos de tiempo de ejecución y de consumo energético.

6.2.3 Tarea 3: Difusión y Explotación de Resultados

Sub-tarea 3.1: Difusión

El relativo buen conocimiento del estado del arte y las tendencias tecnológicas permite acometer con ciertas garantías las tareas de difusión y explotación de este proyecto. Respecto a la difusión,

[3] `https://s21.q4cdn.com/399680738/files/doc_financials/2019/q4/FB-12.31.2019-Exhibit-99.1-r61_final.pdf`

se espera enviar los frutos de este trabajo a un par de revistas de alto impacto y otros tres artículos de conferencias, workshops de prestigio. Además se realizarán tareas de diseminación sobre empresas locales que se pueden beneficiar del empleo de la computación heterogénea y se intentará seguir colaborando con empresas extranjeras en estas temáticas.

- Envió de publicaciones a revistas y conferencias de reconocido prestigio.
- Diseminación de las posibilidades de la computación heterogénea a empresas locales.

6.2.4 Cronograma Temporal

La figura 6.1 muestra una estimación de la temporización propuesta asumiendo una duración del proyecto de 3 años. Al plantearse un modelo de trabajo iterativo en el que conforme se va aprendiendo más de las cargas de trabajo se van refinando tanto el código a ejecutar en los dispositivos como los planificadores hay bastante solapamiento entre tareas.

Tarea	Comienzo	Duración	De julio 2020 a junio 2023
Aceleración de Grafos	julio 2020	2 años	
Análisis Cargas Trabajo		*18 meses*	
FPGA		*18 meses*	
CPU y GPU		*12 meses*	
Planificación	diciembre 2020	2,5 años	
Planificadores		*30 meses*	
Soporte hardware		*12 meses*	
Difusión	julio 2020	3 años	
Difusión y Explotación		*36 meses*	

Figura 6.1: Temporización de las tareas del proyecto, las sub-tareas se muestran en cursiva.

Durante los 2 primeros años del proyecto, se empleará la mayor parte del tiempo en la aceleración de los grafos, especialmente en las FPGAs, y a la vista de los análisis realizados, se irán diseñando las políticas de planificación. Una vez completada la aceleración en los dispositivos y sin terminar la planificación, se abordará el diseño de soporte *hardware* para mejorar la planificación. Ya que en este momento, se debería haber alcanzado el máximo rendimiento posible por parte de los dispositivos.

Bibliografía

[Ahm+15] M. Ahmad y col. «CRONO: A Benchmark Suite for Multithreaded Graph Algorithms Executing on Futuristic Multicores». En: *2015 IEEE International Symposium on Workload Characterization*. Oct. de 2015, págs. 44-55. DOI: 10.1109/IISWC.2015.11.

[ARR18] T. K. Aasawat, T. Reza y M. Ripeanu. «How Well do CPU, GPU and Hybrid Graph Processing Frameworks Perform?» En: *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Mayo de 2018, págs. 458-466. DOI: 10.1109/IPDPSW.2018.00082.

[ASY07] Alicia Aśın Pérez, Daŕıo Suárez Gracia y Víctor Viñals Yúfera. «A proposal to introduce power and energy notions in computer architecture laboratories». En: *WCAE '07: Proceedings of the 2007 workshop on Computer architecture education*. San Diego, California: ACM, 2007, págs. 52-57. ISBN: 978-1-59593-797-1. DOI: http://doi.acm.org/10.1145/1275633.1275644.

[ATN10] Cédric Augonnet, Samuel Thibault y Raymond Namyst. *StarPU: a Runtime System for Scheduling Tasks over Accelerator-Based Multicore Machines*. Research Report RR-7240. INRIA, mar. de 2010, pág. 33. URL: https://hal.inria.fr/inria-00467677.

[Aus17] Sustainable Development Solutions Network (SDSN) Australia. *Getting started with the SDGs in universities: A guide for universities, higher education institutions, and the academic sector*. Inf. téc. Australasian Campuses Towards Sustainability (ACTS), 2017.

[Bal+19] Francis Balestra y col. *International Roadmap for Devices and Systems ™. 2018 Edition. Executive Summary*. Inf. téc. IEEE, 2019.

[BAP15] Scott Beamer, Krste Asanović y David Patterson. *The GAP Benchmark Suite*. 2015. arXiv: 1508.03619 [cs.DC].

[Bin+06] Nathan L. Binkert y col. «The M5 Simulator: Modeling Networked Systems». En: *IEEE Micro* 26.4 (jul. de 2006), págs. 52-60. ISSN: 0272-1732. DOI: 10.1109/MM.2006.82. URL: http://dx.doi.org/10.1109/MM.2006.82.

[Bin+11] Nathan Binkert y col. «The Gem5 Simulator». En: *SIGARCH Comput. Archit. News* 39.2 (ago. de 2011), págs. 1-7. ISSN: 0163-5964. DOI: 10.1145/2024716.2024718. URL: http://doi.acm.org/10.1145/2024716.2024718.

[BM19] Javier Benayas y Carmelo Marcén. *Hacia una Educación para la Sostenibilidad, 20 años después del Libro Blanco de la Educación Ambiental en España*. Inf. téc. Centro Nacional de Educación Ambiental (CENEAM). Organismo Autónomo de Parques Nacionales, Ministerio para la Transición Ecológica, 2019.

[Boa15] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*. Inf. téc. Nov. de 2015.

[Can+17] Francisco Candel y col. «Exploiting Data Compression to Mitigate Aging in GPU Register Files». En: *29th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2017, Campinas, Brazil, October 17-20, 2017*. IEEE Computer Society, 2017, págs. 57-64. DOI: 10.1109/SBAC-PAD.2017.15. URL: https://doi.org/10.1109/SBAC-PAD.2017.15.

[Cas+15] C. Cascaval y col. «Concurrency in Mobile Browser Engines». En: *IEEE Pervasive Computing* 14.3 (jul. de 2015), págs. 14-19. ISSN: 1536-1268. DOI: 10.1109/MPRV.2015.58.

[Cen13] Barcelona Supercomputing Center. *OmpSs Specification*. 2013. URL: https://pm.bsc.es/ompss-docs/specs/.

[CGR94] Boris V Cherkassky, Andrew V Goldberg y Tomasz Radzik. «Shortest paths algorithms: Theory and experimental evaluation». En: *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*. 1994, págs. 516-525.

[Che+13] S. Che y col. «Pannotia: Understanding irregular GPGPU graph applications». En: *2013 IEEE International Symposium on Workload Characterization (IISWC)*. Sep. de 2013, págs. 185-195. DOI: 10.1109/IISWC.2013.6704684.

[Che+19] Xinyu Chen y col. «On-The-Fly Parallel Data Shuffling for Graph Processing on OpenCL-based FPGAs». En: *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2019, págs. 67-73.

[Com05] The Joint Task Force for Computing Curricula 2005. *Computing Curricula 2005*. Inf. téc. ACM y IEEE, 2005.

[Com16a] European Commission. *European Credit Transfer and Accumulation System (ECTS)*. Abr. de 2016. URL: http://ec.europa.eu/education/ects/ects_en.htm.

[Com16b] The Joint Task Force for Computer Engineering Curricula 2016. *Computer Engineering Curricula 2016*. Inf. téc. ACM y IEEE, 2016.

[Com17] Directorate-General for Communication (European Commission). *Strengthening European Identity through Education and Culture*. Inf. téc. European Union, 2017.

[Con16] Open ACC Consortium. *Directives Open ACC*. 2016. URL: http://www.openacc.org/.

[Cor16a] Intel Corporation. *Intel® Cilk Plus*. 2016. URL: https://software.intel.com/en-us/intel-cilk-plus.

[Cor16b] Intel Corporation. *Intel® Threading Building Blocks*. 2016. URL: https://www.threadingbuildingblocks.org/.

[Cor16c] Microsoft Corporation. *C++ AMP (C++ Accelerated Massive Parallelism)*. 2016. URL: https://msdn.microsoft.com/en-us/library/hh265137.aspx.

[Dai+17] Guohao Dai y col. «ForeGraph: Exploring large-scale graph processing on multi-FPGA architecture». En: *FPGA 2017 - Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (2017), págs. 217-226. DOI: 10.1145/3020078.3021739.

[Dre+16] P. Dreher y col. «PageRank Pipeline Benchmark: Proposal for a Holistic System Benchmark for Big-Data Platforms». En: *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Mayo de 2016, págs. 929-937. DOI: 10.1109/IPDPSW.2016.89.

[Esp11] Gobierno de España. *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. 5 de mar. de 2011. URL: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750.

[Eva05] Agencia Nacional de Evaluación de la Calidad y Acreditación. *Libro Blanco: Título de Grado en Ingeniería Informática*. Agencia Nacional de Evaluación de la Calidad y Acreditación, 2005.

[Eye+18] Stijn Eyerman y col. «Many-Core Graph Workload Analysis». En: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. SC '18. Dallas, Texas: IEEE Press, 2018.

[Fer+13] Alexandra Ferrerón-Labari y col. «Shrinking L1 Instruction Caches to Improve Energy-Delay in SMT Embedded Processors». En: *Architecture of Computing Systems - ARCS 2013 - 26th International Conference, Prague, Czech Republic, February 19-22, 2013. Proceedings*. Ed. por Hana Kubátová y col. Vol. 7767. Lecture Notes in Computer Science. Springer, 2013, págs. 256-267. DOI: 10.1007/978-3-642-36424-2_22. URL: https://doi.org/10.1007/978-3-642-36424-2%5C_22.

[Fer+14] Alexandra Ferrerón-Labari y col. «Block Disabling Characterization and Improvements in CMPs Operating at Ultra-low Voltages». En: *26th IEEE International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2014, Paris, France, October 22-24, 2014*. IEEE Computer Society, 2014, págs. 238-245. DOI: 10.1109/SBAC-PAD.2014.12. URL: https://doi.org/10.1109/SBAC-PAD.2014.12.

[Fer+16] Alexandra Ferrerón-Labari y col. «Concertina: Squeezing in Cache Content to Operate at Near-Threshold Voltage». En: *IEEE Trans. Computers* 65.3 (2016), págs. 755-769. DOI: 10.1109/TC.2015.2479585. URL: https://doi.org/10.1109/TC.2015.2479585.

[Fer+19] Alexandra Ferrerón-Labari y col. «A fault-tolerant last level cache for CMPs operating at ultra-low voltage». En: *J. Parallel Distrib. Comput.* 125 (2019), págs. 31-44. DOI: 10.1016/j.jpdc.2018.10.010. URL: https://doi.org/10.1016/j.jpdc.2018.10.010.

[Fle18] Nic Fleming. «How artificial intelligence is changing drug discovery». En: *Nature* 557.7706 (2018), S55-S55.

[Gas+11] Benedict Gaster y col. *Heterogeneous Computing with OpenCL*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[GC15] Erol Gelenbe e Yves Caseau. «The Impact of Information Technology on Energy Consumption and Carbon Emissions». En: *Ubiquity* 2015.June (jun. de 2015). DOI: 10.1145/2755977. URL: https://doi.org/10.1145/2755977.

[Góm+17] J. Gómez-Luna y col. «Chai: Collaborative heterogeneous applications for integrated-architectures». En: *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Abr. de 2017, págs. 43-54. DOI: 10.1109/ISPASS.2017.7975269.

[Goo16a] Google. *Renderscript Framework*. 2016. URL: https://developer.android.com/guide/topics/renderscript/compute.html.

[Goo16b] Google. *Vulkan Graphics API*. 2016. URL: https://developer.android.com/ndk/guides/graphics/index.html.

[Gra+12] D. S. Gracia y col. «LP-NUCA: Networks-in-Cache for High-Performance Low-Power Embedded Processors». En: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.8 (ago. de 2012), págs. 1510-1523. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2011.2158249.

[Gre16] Brendan Gregg. *Linux Performance*. 10 de mayo de 2016. URL: http://www.brendangregg.com/linuxperf.html.

[Gro16] Dave Grove. *Talking about Swift Concurrency on Linux*. 2016. URL: https://developer.ibm.com/swift/2016/02/22/talking-about-swift-concurrency-on-linux/.

[Gut+09] Sergio Gutiérrez Verde y col. «Processor Energy and Temperature in Computer Architecture Courses: a hands-on approach». En: *Proceedings of the 2009 Workshop on Computer Architecture Education*. 2009.

[Guz+18] Maria Angelica Davila Guzman y col. «Towards the Inclusion of FPGAs on Commodity Heterogeneous Systems». En: *2018 International Conference on High Performance Computing & Simulation, HPCS 2018, Orleans, France, July 16-20, 2018*. IEEE, 2018, págs. 554-556. DOI: 10.1109/HPCS.2018.00092. URL: https://doi.org/10.1109/HPCS.2018.00092.

[Guz+19] Maria Angelica Davila Guzman y col. «Cooperative CPU, GPU, and FPGA heterogeneous execution with EngineCL». En: *The Journal of Supercomputing* 75.3 (2019), págs. 1732-1746. DOI: 10.1007/s11227-019-02768-y. URL: https://doi.org/10.1007/s11227-019-02768-y.

[Ham+16] Tae Jun Ham y col. «Graphicionado: A high-performance and energy-efficient accelerator for graph analytics». En: *Proceedings of the Annual International Symposium on Microarchitecture, MICRO* 2016-Decem (2016), págs. 1-13. ISSN: 10724451. DOI: 10.1109/MICRO.2016.7783759.

[Inc16] Qualcomm Technologies Inc. *Symphony System Manager SDK*. 2016. URL: https://developer.qualcomm.com/forums/software/symphony-system-manager-sdk.

[Kho+18] Soroosh Khoram y col. «Accelerating graph analytics by co-optimizing storage and access on an FPGA-HMC platform». En: *FPGA 2018 - Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Vol. 2018-February. 2018. ISBN: 9781450356145. DOI: 10.1145/3174243.3174260.

[KNS18] Tushar Kumar, Aravind Natarajan y Darío Suárez Gracia. «Random-access disjoint concurrent sparse writes to heterogeneous buffers». Patent US10031697. 24 de jul. de 2018.

[KR13] E. K. Ardestani y J. Renau. «ESESC: A Fast Multicore Simulator Using Time-Based Sampling». En: *International Symposium on High Performance Computer Architecture*. HPCA'19. 2013.

[KRH15a] Ronan Keryell, Ruyman Reyes y Lee Howes. «Khronos SYCL for OpenCL: A Tutorial». En: *Proceedings of the 3rd International Workshop on OpenCL*. IWOCL '15. Palo Alto, California: Association for Computing Machinery, 2015. ISBN: 9781450334846. DOI: 10.1145/2791321.2791345. URL: https://doi.org/10.1145/2791321.2791345.

[KRH15b] Ronan Keryell, Ruyman Reyes y Lee Howes. «Khronos SYCL for OpenCL: a tutorial». En: *Proceedings of the 3rd International Workshop on OpenCL*. 2015, págs. 1-1.

[Lab18] Bureau of Labor Statistics. *Network and Computer Systems Administrators*. 2018. URL: http://www.bls.gov/ooh/computer-and-information-technology/network-and-computer-systems-administrators.htm.

[Li+13] Sheng Li y col. «The McPAT Framework for Multicore and Manycore Architectures: Simultaneously Modeling Power, Area, and Timing». En: *ACM Trans. Archit. Code Optim.* 10.1 (abr. de 2013), 5:1-5:29. ISSN: 1544-3566. DOI: 10.1145/2445572.2445577. URL: http://doi.acm.org/10.1145/2445572.2445577.

[Mal+10] Grzegorz Malewicz y col. «Pregel: A System for Large-Scale Graph Processing». En: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of*

Data. SIGMOD '10. Indianapolis, Indiana, USA: Association for Computing Machinery, 2010, págs. 135-146. ISBN: 9781450300322. DOI: 10.1145/1807167.1807184. URL: https://doi.org/10.1145/1807167.1807184.

[Mar+05] Milo M. K. Martin y col. «Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset». En: *SIGARCH Comput. Archit. News* 33.4 (nov. de 2005), págs. 92-99. ISSN: 0163-5964. DOI: 10.1145/1105734.1105747. URL: http://doi.acm.org/10.1145/1105734.1105747.

[Mor+19] Lucas Morais y col. «Adding Tightly-Integrated Task Scheduling Acceleration to a RISC-V Multi-Core Processor». En: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO '52. Columbus, OH, USA: Association for Computing Machinery, 2019, págs. 861-872. ISBN: 9781450369381. DOI: 10.1145/3352460.3358271. URL: https://doi.org/10.1145/3352460.3358271.

[MPT78] M. D. McIlroy, E. N. Pinson y B. A. Tague. «Unix Time-Sharing System Forward». En: *The Bell System Technical Journal* 57.6, part 2 (1978), p.1902.

[Muk+18] Anurag Mukkara y col. «Exploiting locality in graph analytics through hardware-Accelerated traversal scheduling». En: *Proceedings of the Annual International Symposium on Microarchitecture, MICRO* 2018-Octob (2018), págs. 1-14. ISSN: 10724451. DOI: 10.1109/MICRO.2018.00010.

[Mur+10] Richard C Murphy y col. «Introducing the graph 500». En: (2010).

[MWM15] Robert Ryan McCune, Tim Weninger y Greg Madey. «Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing». En: *ACM Comput. Surv.* 48.2 (oct. de 2015). ISSN: 0360-0300. DOI: 10.1145/2818185. URL: https://doi.org/10.1145/2818185.

[NBB20] Raúl Nozal, Jose Luis Bosque y Ramon Beivide. «EngineCL: Usability and Performance in Heterogeneous Computing». En: *Future Generation Computer Systems* 107 (2020), págs. 522-537. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.02.016. URL: http://www.sciencedirect.com/science/article/pii/S0167739X19314323.

[Nem+18] Evi Nemeth y col. *UNIX and Linux System Administration Handbook*. 5th Edition. Prentice Hall, 2018.

[NLP13] Donald Nguyen, Andrew Lenharth y Keshav Pingali. «A Lightweight Infrastructure for Graph Analytics». En: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. SOSP '13. Farminton, Pennsylvania: Association for Computing Machinery, 2013, págs. 456-471. ISBN: 9781450323888. DOI: 10.1145/2517349.2522739. URL: https://doi.org/10.1145/2517349.2522739.

[Núñ+19] José Núñez-Yáñez y col. «Simultaneous multiprocessing in a software-defined heterogeneous FPGA». En: *The Journal of Supercomputing* 75.8 (2019), págs. 4078-4095. DOI: 10.1007/s11227-018-2367-9. URL: https://doi.org/10.1007/s11227-018-2367-9.

[Nur+14] E. Nurvitadhi y col. «GraphGen: An FPGA Framework for Vertex-Centric Graph Computation». En: *2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines*. 2014, págs. 25-28.

[Ort+14] Marta Orſın y col. «Dynamic construction of circuits for reactive traffic in homogeneous CMPs». En: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*. Ed. por Gerhard P. Fettweis y Wolfgang Nebel. European Design y Automation Association, 2014, págs. 1-4. DOI: 10.7873/DATE.2014.254. URL: https://doi.org/10.7873/DATE.2014.254.

[Ort+16a] Marta Orſın-Obón y col. «Analysis of network-on-chip topologies for cost-efficient chip multiprocessors». En: *Microprocessors and Microsystems - Embedded Hardware Design* 42 (2016), págs. 24-36. DOI: 10.1016/j.micpro.2016.01.005. URL: https://doi.org/10.1016/j.micpro.2016.01.005.

[Ort+16b] Marta Orſın-Obón y col. «Reactive circuits: Dynamic construction of circuits for reactive traffic in homogeneous CMPs». En: *J. Parallel Distrib. Comput.* 95 (2016), págs. 57-68. DOI: 10.1016/j.jpdc.2016.04.002. URL: https://doi.org/10.1016/j.jpdc.2016.04.002.

[Ozd+16] Muhammet Mustafa Ozdal y col. «Energy Efficient Architecture for Graph Analytics Accelerators». En: *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016* (2016), págs. 166-177. DOI: 10.1109/ISCA.2016.24.

[Pag+99] Lawrence Page y col. *The pagerank citation ranking: Bringing order to the web*. Inf. téc. Stanford InfoLab, 1999.

[Pin+11] Keshav Pingali y col. «The Tao of Parallelism in Algorithms». En: *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '11. San Jose, California, USA: Association for Computing Machinery, 2011, págs. 12-25. ISBN: 9781450306638. DOI: 10.1145/1993498.1993501. URL: https://doi.org/10.1145/1993498.1993501.

[Pow+14] Jason Power y col. «gem5-gpu: A Heterogeneous CPU-GPU Simulator». En: *Computer Architecture Letters* 13.1 (ene. de 2014). ISSN: 1556-6056. DOI: 10.1109/LCA.2014.2299539. URL: http://gem5-gpu.cs.wisc.edu.

[Put+14] Andrew Putnam y col. «A reconfigurable fabric for accelerating large-scale datacenter services». En: *ACM/IEEE 41st International Symposium on Computer Architecture, ISCA 2014, Minneapolis, MN, USA, June 14-18, 2014*. IEEE Computer Society, 2014, págs. 13-24. DOI: 10.1109/ISCA.2014.6853195. URL: https://doi.org/10.1109/ISCA.2014.6853195.

[RCJ11] Paul Rosenfeld, Elliott Cooper-Balis y Bruce Jacob. «DRAMSim2: A Cycle Accurate Memory System Simulator». En: *IEEE Comput. Archit. Lett.* 10.1 (ene. de 2011), págs. 16-19. ISSN: 1556-6056. DOI: 10.1109/L-CA.2011.4. URL: https://doi.org/10.1109/L-CA.2011.4.

[Ren+05] Jose Renau y col. *SESC simulator*. http://sesc.sourceforge.net. Ene. de 2005.

[Rod+19a] Andrés Rodríguez y col. «Exploring heterogeneous scheduling for edge computing with CPU and FPGA MPSoCs». En: *Journal of Systems Architecture - Embedded Systems Design* 98 (2019), págs. 27-40. DOI: 10.1016/j.sysarc.2019.06.006. URL: https://doi.org/10.1016/j.sysarc.2019.06.006.

[Rod+19b] Andrés Rodríguez y col. «Parallel multiprocessing and scheduling on the heterogeneous Xeon+FPGA platform». En: *The Journal of Supercomputing* (2019). DOI: 10.1007/s11227-019-02935-1. URL: https://doi.org/10.1007/s11227-019-02935-1.

[Rod15] Marko A. Rodriguez. «The Gremlin Graph Traversal Machine and Language (Invited Talk)». En: *Proceedings of the 15th Symposium on Database Programming Languages*. DBPL 2015. Pittsburgh, PA, USA: Association for Computing Machinery, 2015, págs. 1-10. ISBN: 9781450339025. DOI: 10.1145/2815072.2815073. URL: https://doi.org/10.1145/2815072.2815073.

[Ros03] P. E. Ross. «5 Commandments [technology laws and rules of thumb]». En: *IEEE Spectrum* 40.12 (2003), págs. 30-35. ISSN: 1939-9340. DOI: 10.1109/MSPEC.2003.1249976.

[RS11] K. Rupp y S. Selberherr. «The Economic Limit to Moore's Law». En: *IEEE Transactions on Semiconductor Manufacturing* 24.1 (feb. de 2011), págs. 1-4. ISSN: 1558-2345. DOI: 10.1109/TSM.2010.2089811.

[Sam+12] A. Sampson y col. «Automatic discovery of performance and energy pitfalls in HTML and CSS». En: *Workload Characterization (IISWC), 2012 IEEE International Symposium on*. Nov. de 2012, págs. 82-83. DOI: 10.1109/IISWC.2012.6402904.

[Shi+18] Xuanhua Shi y col. «Graph Processing on GPUs: A Survey». En: *ACM Comput. Surv.* 50.6 (ene. de 2018). ISSN: 0360-0300. DOI: 10.1145/3128571. URL: https://doi.org/10.1145/3128571.

[Suá+09] Darío Suárez y col. «Light NUCA: a proposal for bridging the inter-cache latency gap». En: *Prooceedings of the 12th Design, Automation and Test in Europe Conference and Exhibition (DATE'09)*. Abr. de 2009.

[Suá+14] Daŕıo Suárez Gracia y col. «Revisiting LP-NUCA Energy Consumption: Cache Access Policies and Adaptive Block Dropping». En: *ACM Trans. Archit. Code Optim.* 11.2 (jun. de 2014), 19:1-19:26. ISSN: 1544-3566. DOI: 10.1145/2632217. URL: http://doi.acm.org/10.1145/2632217.

[Suá+16] Darío Suárez Gracia y col. «Directed Event Signaling For Multiprocessors Systems». Patent WO/2016/022308. 11 de feb. de 2016.

[Suá+18] Darío Suárez Gracia y col. «Identifying enhanced synchronization operation outcomes to improve runtime operations». Patent US10114681. 30 de oct. de 2018.

[Sun+15] Narayanan Sundaram y col. «GraphMat: High Performance Graph Analytics Made Productive». En: *Proc. VLDB Endow.* 8.11 (jul. de 2015), págs. 1214-1225. ISSN: 2150-8097. DOI: 10.14778/2809974.2809983. URL: https://doi.org/10.14778/2809974.2809983.

[Val+19] Alejandro Valero y col. «An Aging-Aware GPU Register File Design Based on Data Redundancy». En: *IEEE Trans. Computers* 68.1 (2019), págs. 4-20. DOI: 10.1109/TC.2018.2849376. URL: https://doi.org/10.1109/TC.2018.2849376.

[Val90] Leslie G. Valiant. «A Bridging Model for Parallel Computation». En: *Commun. ACM* 33.8 (ago. de 1990), págs. 103-111. ISSN: 0001-0782. DOI: 10.1145/79173.79181. URL: https://doi.org/10.1145/79173.79181.

[w3c15] w3cook.com. *OS Market Share and Usage Trends*. 2015. URL: http://www.w3cook.com/os/summary/.

[Web12] Jim Webber. «A Programmatic Introduction to Neo4j». En: *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*. SPLASH '12. Tucson, Arizona, USA: Association for Computing Machinery, 2012, págs. 217-218. ISBN: 9781450315630. DOI: 10.1145/2384716.2384777. URL: https://doi.org/10.1145/2384716.2384777.

[Zar10a] Universidad de Zaragoza. *Memoria de verificación de la titulación de grado en Ingeniería Informática en la Universidad de Zaragoza*. Universidad de Zaragoza, 2010.

[Zar10b] Universidad de Zaragoza. *Memoria de verificación del grado en Ingeniería Informática*. Universidad de Zaragoza, 2010.

[Zar16] Universidad de Zaragoza. *Acuerdo de 23 de febrero de 2016, del Consejo de Gobierno de la Universidad de Zaragoza, por el que se aprueba la normativa básica sobre el procedimiento y los criterios de valoración de la actividad docente del profesorado por parte de los estudiantes*. 23 de feb. de 2016. URL: https://zaguan.unizar.es/record/60999/files/Reglamento_2016.pdf.

[Zho+19] Shijie Zhou y col. «HitGraph: High-throughput Graph Processing Framework on FPGA». En: *IEEE Transactions on Parallel and Distributed Systems* 30.10 (2019), págs. 2249-2264. ISSN: 15582183. DOI: 10.1109/TPDS.2019.2910068.

[ZKL17] Jialiang Zhang, Soroosh Khoram y Jing Li. «Boosting the Performance of FPGA-Based Graph Processor Using Hybrid Memory Cube: A Case for Breadth First Search». En: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. FPGA '17. Monterey, California, USA: Association for Computing Machinery, 2017, págs. 207-216. ISBN: 9781450343541. DOI: 10.1145/3020078.3021737. URL: https://doi.org/10.1145/3020078.3021737.

[ZL18a] Y. Zou y M. Lin. «Very Large-Scale and Node-Heavy Graph Analytics with Heterogeneous FPGA+CPU Computing Platform». En: *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. Jul. de 2018, págs. 638-643. DOI: 10.1109/ISVLSI.2018.00121.

[ZL18b] Yu Zou y Mingjie Lin. «GridGAS: An I/O-Efficient heterogeneous FPGA+cpu computing platform for very large-scale graph analytics». En: *Proceedings - 2018 International Conference on Field-Programmable Technology, FPT 2018* (2018), págs. 249-252. DOI: 10.1109/FPT.2018.00045.

[ZSK19] Han Zhao, Darío Suárez Gracia y Tushar Kumar. «Proactive resource management for parallel work-stealing processing systems». Patent US10360063. 23 de jul. de 2019.