UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

PLANIFICACIÓN ÓPTIMA DE DATACENTERS DESAGREGADOS

Diseño, implementación y evaluación de técnicas para la planificación óptima de Datacenters virtuales (VDC) en entornos de Datacenters desagregados interconectados con redes ópticas.

Proyecto de final de grado

Presentado a la

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

por

Rubén Serrano Moreno

En cumplimiento parcial

de los requerimientos del

GRADO EN SISTEMAS DE TELECOMUNICACIONES

Tutores: Jordi Perelló Muntan, Albert Pagès Cruz

Barcelona, Junio 2016

Abstract

Traditional DCs (Data Center) can suffer from computing resource underutilization due to the constrained capacity of server units. To overcome these limitations, the disaggregated DC paradigm has been recently introduced. Thanks to resource disaggregation, it is possible to allocate the exact amount of resources needed to provision a VDC (Virtual Data Center) instance. In this project, we focus on the static planning of an all-optical disaggregated DC infrastructure in support of a known set of VDC instances to be deployed on top of a shared physical infrastructure. To this end, we provide optimal and sub-optimal techniques to determine the necessary capacity (both in terms of computing and network resources) required to support the demand set. In this study we show how disaggregated DC architectures can help on reducing the amount of computing resources needed when allocating VDC instances on top of a shared physical infrastructure. Around 46% average reductions can be experimented for fairly balanced Virtual Machine (VM) profiles while the reductions can increase up to 60% in the case of highly specialized VMs. Such results indicate that disaggregated DCs can overcome the limitations of current architectures in regards of efficient computing resource utilization, pleading for their adoption in future DC architectures.

Resum

Els DCs (Data Center) tradicionals no aprofiten al màxim els seus recursos a causa de la restricció de capacitat dels seus servidors. Per millorar aquesta limitació, el paradigma de DC desagregat ha estat introduït recentment. Gràcies a la desagregació dels recursos, és possible assignar exactament els recursos necessaris per aprovisionar una instància VDC (Virtual Data Center). En aquest projecte ens centrem a la planificació estàtica d'una infraestructura de DC desagregat totalment òptica sent conegut el conjunt de VDCs que han de ser desplegades. En aquest context, proporcionem tècniques òptimes i sub-òptimes per determinar les necessitats de capacitat (tant en termes de recursos computacionals com de xarxa) que es requereixen per satisfer aquestes demandes. En aquest estudi hem mostrat com les arquitectures DC desagregades poden ajudar a la reducció de recursos computacionals necessaris per albergar instàncies VDC en la infraestructura física. Al voltant d'un 46% de reduccions de mitjana poden ser experimentades en perfils de VM (Màquines Virtuals) balancejades, mentre que aquestes reduccions poden arribar fins al 60% en el cas de VMs especialitzades. Aquests resultats ens indiquen que els DCs desagregats poden sobreposar-se a les limitacions de les arquitectures actuals en termes d'utilitzar eficientment els recursos computacionals, esperant la seva adopció en futures arquitectures DC.

Resumen

Los DCs (Data Center) tradicionales no aprovechan al máximo sus recursos debido a la restricción de capacidad de sus servidores. Para mejorar esta limitación, el paradigma de DC desagregado ha sido introducido recientemente. Gracias a la desagregación de los recursos, es posible asignar exactamente los recursos necesarios para aprovisionar a una instancia VDC (Virtual Data Center). En este proyecto nos centramos en la planificación estática de una infraestructura de DC desagregado puramente óptica siendo conocido el conjunto de VDCs que tienen que ser desplegadas. En este contexto, proporcionamos técnicas óptimas y sub-óptimas para determinar las necesidades de capacidad (tanto en término de recursos computacionales como de red) se requieren para satisfacer estas demandas. En este estudio hemos mostrado como las arquitecturas de DC desagregado pueden ayudar a la reducción de recursos computacionales necesarios para albergar instancias VDC en la infraestructura física. Alrededor de un 46% de reducciones promedio pueden ser experimentadas en perfiles de VM (Máquinas Virtuales) balanceadas, mientras que estas reducciones pueden llegar hasta al 60% en el caso de VMs especializadas. Estos resultados nos indican que los DCs desagregados pueden sobreponerse a las limitaciones de las arquitecturas actuales en términos de utilizar eficientemente los recursos computacionales, esperando su adopción en futuras arquitecturas DC.

Agradecimientos

En estos últimos meses muchas personas han contribuido a que este estudio se haga realidad. A todos ellas, me gustaría expresarles mi agradecimiento.

En primer lugar, a Albert Pagès y Jordi Perelló, codirectores, mentores y amigos. Sin su dirección exigente, pero, también, honesta, comprometida y brillante, este estudio no habría sido posible. De ellos he aprendido mucho, y sigo aprendiendo, y no sólo de investigación.

Quiero dedicar también unas líneas a Luís Rueda y Marc Vidal, viejos amigos, con los que llevo compartiendo estos estudios y enfrentándonos a numerosos retos desde tiempo atrás. Su apoyo leal e incondicional tiene parte de culpa de que ahora esté escribiéndolas.

También me gustaría expresar mi agradecimiento sincero a la gente del Departamento, tanto técnicos como investigadores, que siempre han estado dispuestos a echarme una mano cuando lo he necesitado.

Finalmente, quiero darle las gracias a mi familia. Quiero dedicarles a ellos muy especialmente este estudio, que es algo tan suyo como mío. Ellos han soportado mis días duros de trabajo y siempre me han apoyado incondicionalmente.

HISTORIAL DE REVISIÓN

Revisión	Fecha	Propósito
0	27/05/2016	Creación del documento
1	02/06/2016	Revisión del documento
2	07/06/2016	Ampliación del documento
3	21/06/2016	Redactado final del documento

MIEMBROS DEL PROYECTO

Nombre	E-mail
Rubén Serrano Moreno	ruben.serrano.moreno@gmail.com

ESCRITO POR:		REVISADO Y APROVADO POR:	
Fecha		Fecha	
Nombre	Rubén Serrano Moreno	Nombre	Jordi Perelló Muntan, Albert Pagès Cruz
Rango	Autor	Rango	Supervisores

Tabla de contenido

Listado de figuras

Listado de tablas

1. Introducción

Las infraestructuras Data Center (DC) son unas pieza fundamental en las telecomunicaciones y servicios cloud actuales, permitiendo el acceso a enormes cantidades de datos sin importar hora ni lugar. Gracias a los esfuerzos conjuntos de miles de servidores que se encuentran dentro de las instalaciones, complejos servicios de Internet y de Cloud (p.e buscadores, almacenamiento cloud) pueden llevarse a cabo. En los DCs tradicionales, los servidores están organizados en racks, cada uno de ellos equipado con un Top of the Rack (ToR) *switch*, que interconectan todos los servidores de un mismo rack y permiten el intercambio de información entre los diferentes racks mediante una red entrelazada intra-DC (DCN). Las arquitecturas DCN actuales están construidas principalmente por conmutadores eléctricos (p.e Ethernet), organizados en una arquitectura multi-capa, la cual proporciona varios puntos de agregación que implican la redundancia necesaria para mejorar la utilización de los recursos de red y garantizan la calidad del servicio (QoS) [1].

No obstante, el constante crecimiento del tráfico de Internet y de los servicios Cloud, promovido por aplicaciones que requieren un gran ancho de banda como son Big Data, Internet of Things y Video en streaming, requieren mayores infraestructuras DC en términos de capacidades computacionales y de red con el objetivo de acomodar todas las aplicaciones y servicios. Por ejemplo, se prevé que el tráfico global IP soportado por los DCs será prácticamente el doble que el actual en 2019, pasando de 5.6 ZB hasta 10.4 ZB por año [2]. Está crecimiento de tráfico sin precedentes está llevando al límite las capacidades de las actuales estructuras DCN basadas en conmutación eléctrica. Por esta razón, existe una especial atención en mejorar la actuación de la intra-DCNs que serán desarrolladas en las arquitecturas DC futuras. Bajo esta premisa, la tecnología óptica está ganando un considerable interés debido a su escalabilidad, ancho de banda y latencia igual que la reducción del consumo eléctrico. Muchos esfuerzos se están llevando a cabo para aplicar esta tecnología en las comunicaciones intra-DC de los futuros DC [3], ya sean redes híbridas eléctrico/ópticas (p.e [4]) o totalmente ópticas (p.e [5]).

Sin embargo, a pesar de los esfuerzos en mejorar las arquitecturas DCN, los actuales DC basados en servidores aún tienen que hacer frente a diversos problemas relacionados con la eficiente utilización de sus recursos. En general, los servicios/tareas de los DC son ejecutados sobre máquinas virtuales (VMs) las cuales son desplegadas en los servidores. A cada VM se le asigna un conjunto de recursos computacionales (núcleos CPU, memoria y almacenamiento) en función de las necesidades computacionales de la aplicación. Estos recursos son asignados de manera permanente, siendo totalmente reservados únicamente para el uso requerido de una VM. Destacar que la coexistencia de más de una VMs en un mismo servidor solo es posible si la capacidad total del servidor no supera a la suma de las necesidades de las VMs. No obstante, a pesar de la posibilidad de esta coexistencia, el tamaño heterogéneo de las VM y su asociación correspondiente a los recursos computacionales puede provocar no aprovechar eficientemente el servidor. Por ejemplo, puede pasar que una aplicación/servicio se esté ejecutando sobre uno servidor ocupando prácticamente la totalidad de uno de sus recursos (p.e núcleos CPU) mientras que el resto de recursos (p.e memoria y almacenamiento) estén prácticamente desutilizados. En esta situación, sería imposible asignar otra aplicación a este servidor, debido a que sobrepasaría la capacidad de uno de los recursos del servidor. Como ejemplo de esta implicación,

Google recientemente ha publicado sus datos sobre la utilización de sus infraestructuras DC, los cuales muestran la disparidad del uso de memoria/almacenamiento respecto el uso de CPU para realizar sus tareas [6]. Adicionalmente, es especialmente difícil configurar dinámicamente los recursos del DC en función de un perfil impredecible de tráfico.

Aparte de la potencial limitación en el uso de los recursos, las arquitecturas basadas en servidores también sufren limitaciones de modularidad que pueden provocar impacto en el comportamiento del sistema. Los servidores tradicionales son usualmente construidos a partir de la integración de sus principales componentes (CPU, módulos de memoria, disco, tarjeta de interfaz de red, …) en una única placa madre. Esta ha sido la base de la manufacturación de computadoras durante los últimos años. No obstante, esta integración es responsable de limitaciones en aplicar posibles mejoras al sistema. Esto principalmente sucede porque los ratios de los diferentes componentes (tamaño, velocidad, etc.) son substancialmente distintos para cada uno de ellos. Por ejemplo, las necesidades de CPU han aumentado alrededor del 60% cada año mientras las necesidades de memoria DRAM alrededor del 7%, por lo tanto hay una diferencia entre los dos componentes de 50% por año [7]. Esta disparidad en la evolución de los diferentes componentes integrados en un servidor no permite el uso de las tecnologías más avanzadas para cada uno de ellos en particular ya que se tiene que buscar un equilibrio que implique el buen rendimiento de ellos con el compromiso de que funcionan de manera conjunta.

Por lo tanto, para hacer frente a todos estos retos, nuevas arquitecturas de DC están siendo desarrolladas. Una solución interesante es el concepto de recursos desagregados [8], que consiste en, en vez de aglutinar todos los componentes en un sistema integrado, desagregar todos los componentes desacoplándolos físicamente y organizándolos en separadores (blades). Con este desacoplo de los componentes, es posible investigar nuevas tecnologías para mejorar cada uno de ellos en particular permitiendo así una optimización y costumización del sistema. Este concepto se ha desarrollado en el paradigma de DC desagregados [9]-[12], donde los recursos computacionales no están organizados en unidades de servidores, sino que están desplegados en separadores que actúan como hardwares autónomos. Los blades de los recursos pueden estar agrupados en racks que agrupen todo tipo de recursos o en mono-hardware racks que únicamente contengan blades de un recurso en concreto. Entonces, los blades deben estar interconectados mediante una estructura intra-DCN. La cual debe cumplir los estrictos requerimientos de latencia y de ancho de banda en la comunicación de los diferentes módulos de hardware que exigen las tecnologías ópticas que deben ser desplegadas [9]-[11].

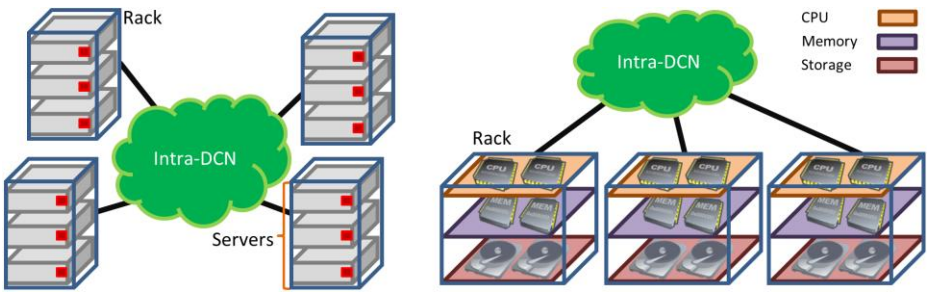


Figura 1. Esquema de una DC tradicional (izquierda) y de uno desagreagdo (derecha)

De este modo, los recursos computacionales pueden ser asignados ajustados a las necesidades de las VMs, esto implica menos recursos para satisfacer una demanda y una reducción de los costes asociados (CAPEX). Además, la desagregación proporciona modularidad al sistemas, permitiendo la actualización del hardware cuando se necesario. Por esta razón, los DCs desagregados interconectado ópticamente se prevén como una solución para el futuro. Todas estas características son especialmente beneficiosas cuando se trata de provisionar servicios complejos, como son Virtual Data Centers (VDCs), donde la óptima utilización de los recursos pasa a ser primordial. Con el objetivo de cuantificar la reducción de recursos computacionales que implicaría la colocación de VDC sobre una infraestructura desagregada, en este proyecto nos centramos en la planificación de colocación de instancias VDC estáticas sobre un DC desagregado interconectado ópticamente y la compararemos con arquitecturas de servidores. A continuación se introducirá el concepto de servicio VDC y la problemática que implica.

Asignación del Virtual Data Center

Las infraestructuras DC contemporáneas proporcionan servicio a gran cantidad de consumidores, tanto usuarios individuales, negocios/compañías o instituciones públicas, todos ellos tienen necesidades heterogéneas en términos de utilización de los recursos, requiriendo la Calidad del Servicio (QoS) y grados de control sobre los recursos utilizados y sobre otros parámetros. En esta situación, la diversidad de consumidores (*tenants*, en inglés) se convierte en un pilar fundamental al cual deben dirigirse las infraestructuras DC modernas. No obstante, las arquitecturas tradicionales de telecomunicaciones y servicios cloud presentan ciertos inconvenientes en el reto que supone atender a gran cantidad de tenants. A más, la estructura de servicio es muy rígida, y los propietarios de las infraestructuras necesitan ofrecer servicios en capa superiores para gestionarlos con cierta adaptabilidad en función del servicio desarrollado. Esto implica que deban desarrollarse arquitecturas capaces de adaptarse a tráficos dinámicos, con un alto grado de heterogenia en las características de los servicio/aplicaciones. Por otro lado, el papel del usuario es de mero consumidor de servicio, con posibilidades prácticamente nulas en términos de gestión y operación de la infraestructura.

Para superar estas limitaciones, el concepto de Infraestructura de Servicio (IaaS) ha sido introducido [13]. La aparición de las IaaS surge de la necesidad de dar servicios a usuarios de telecomunicaciones e infraestructuras cloud que implique mejorar la explotación y gestión de las infraestructuras en un contexto dominado por el constante cambio de los requerimientos de los servicios y necesidades de los usuarios. En esencia, IaaS consiste en ofrecer una parte de la infraestructura física a entidades externas, dándoles la capacidad de gestionarlas como si fueran las suyas propias. La tecnología clave detrás de la IaaS es la virtualización, la cual permite abstraer los componentes físicos y vincularlos a múltiples elementos virtuales. Estos elementos virtuales pueden organizarse en infraestructuras virtuales a medida de los requerimientos exactos de recursos, gestión y controlo exigidos por las entidades externas.

Bajo esta premisa, el término servicio VDC (Virtual Data Center) ha sido introducido [14]. En esencia, un servicio VDC consiste en alquilar parte de la infraestructura DC física a una entidad externa (anteriormente llamadas también como tenants) como soporte para desarrollar su propio modelo de negocio. Cada VDC consiste en una infraestructura virtual que integra capacidades computacionales en forma de VMs, las cuales están

interconectadas a través de una red virtual formada por conexiones virtuales (virtual links, en inglés) que proporciona el necesario ancho de banda entre los nodos virtuales (VMs). Estas infraestructuras virtuales son usadas por los tenants para desarrollar sus aplicaciones con el objetivo de ofrecer servicios finales a los usuarios. Gracias a las VDCs, es posible la coexistencia de múltipes tenants sobre la misma infraestructura física, garantizado que cada tenant está aislado de sus vecinos y teniendo el control operacional y de gestión de su infraestructura VDC. Al mismo tiempo, los tenant son beneficiados del bajo coste de este servicio en comparación del coste que implicaría tener su propio DC.

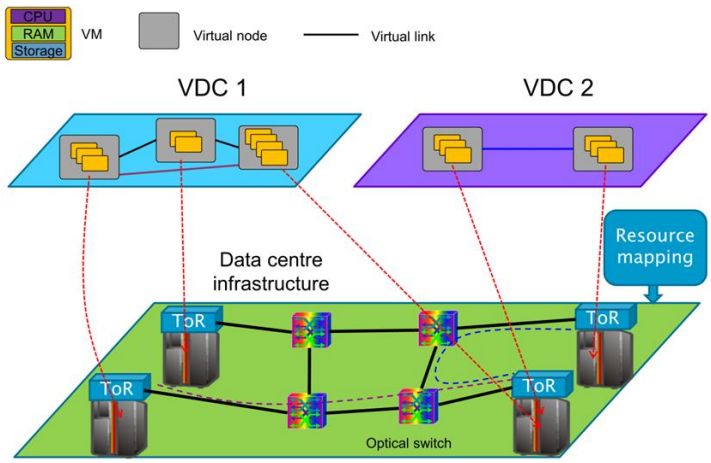


Figura 2. Esquema del mapeo de un VDC.

Uno de los aspecto claves de los servicios VDC reside en lo llamado mapeo (*embedding*) del VDC, que se refiere a como se vincula los recursos virtuales sobre los físicos. Esta operación implica principalmente dos fases: el mapeo de las VM sobre los servidores físicos y el mapeo de los link virtuales sobre el camino físico, el cual debe tener como origen y destino los servidores físicos donde han sido mapeados los nodos virtuales origen/destino del link virtual. Este camino que une físicamente dos servidores y que engloba varios enlaces (links físicos), en posteriores comentarios nos referiremos a él como *path*. La Figura 2 muestra un ejemplo de estos procesos. Idealmente, estas dos fases se deben llevar a cabo de manera conjunta o en un orden coordinado a fin de mejorar el uso de los recursos físicos e incrementar el número de instancias VDC que puedan ser soportador por la infraestructura física compartida [15],[16]. No obstante, como se ha dicho anteriormente, los DC actuales basados en servidores tiene diversas limitaciones en la utilización eficiente de sus recursos al mapear VDC. Por esta razón, los operadores de DC están forzados a sobreequipar de servidores sus infraestructuras para satisfacer los requisitos de las VDCs, con el incremento de costes (CAPEX) que eso implica. A luz de esto, los DC desagregados pueden ayudar en la reducción del número de recursos necesarios para provisionar VDCs, ya que las VMs pueden ser mapeadas exactamente con los recursos que necesitan a partir de los hardwares disponibles agrupados en blades. De esta manera, el operador DC necesitará un menor CAPEX para provisionar una demanda VDC en comparación de si fuera una arquitectura basada en servidores.

En nuestro estudio vamos a considerar el escenario en que el operador DC tiene una infraestructura física con la capacidad necesaria para servir el conjunto de instancias de

VDC que sabemos de antemano. En particular, asumiremos un escenario con un DC desagregado interconectado ópticamente, además su topología de red que interconecta los diferentes hardwares blades y racks ya vienen dadas. Un ejemplo de la arquitectura DC descrita está representada en la Figura 3. Como ya hemos comentado en la introducción, en nuestro estudio analizaremos una estructura DC desagregado donde los racks contienen recursos de los diferentes tipos (p.e núcleos de CPU, memoria y almacenamiento). En ese caso, los diferentes blades están interconectados entre ellos mediante una red óptica basada en el multiplexado compacto por división en longitudes de onda (en inglés Dense Wavelength Division Multiplexing DWDM). DWDM es un método de multiplexación en el que varias señales portadoras (ópticas) se transmiten por una única fibra óptica utilizando distintas longitudes de onda de un haz láser en cada una de ellas. Cada portadora óptica forma un canal óptico que podrá ser tratado independientemente del resto de canales que comparten el medio (fibra óptica) y contener diferente tipo de tráfico. De esta manera se puede multiplicar el ancho de banda efectivo de la fibra óptica. Cada hardware blade dispone de un *Switch and Interface Card* (SIC) que realiza una conversión eléctrica a óptica o viceversa, en función de si es el origen o el destino del canal respectivamente.

Tal y como se ha descrito, se utiliza un conjunto de fibras ópticas para las conexiones entre los diferentes blades como para la conexiones entre los diferentes racks. Bajo esta premisa, asumimos que las fibras asignadas para establecer conexiones entre los elementos internos del rack tienen las propiedades suficientes para mitigar limitaciones de latencia y de ancho de bandas necesarias para estas conexiones internas. Asimismo, cada rack está equipado con un conmutador óptico (Top of the rack “ToR”) el cual interconecta el hardware de un rack con el de los distintos racks gracias a un red DWDM de conmutadores. Esta comunicación entre rack es puramente óptica a través de conmutadores ópticos (e.g. *Optical Cross Connects* (OXCs)), incoloros, unidireccionales y sin contención (*Colorless, Directionless y Contentionless* (CDC)). De ese modo los ToR no necesitan realizar ningún tipo de procesamiento eléctrico, ya que ese proceso y las conversiones ópticas se dan en los SICs de cada blade. Gracias a este modelo, se establece un gran ancho de banda y una latencia ínfima en las comunicaciones entre los diferentes blades de la infraestrctura [17].

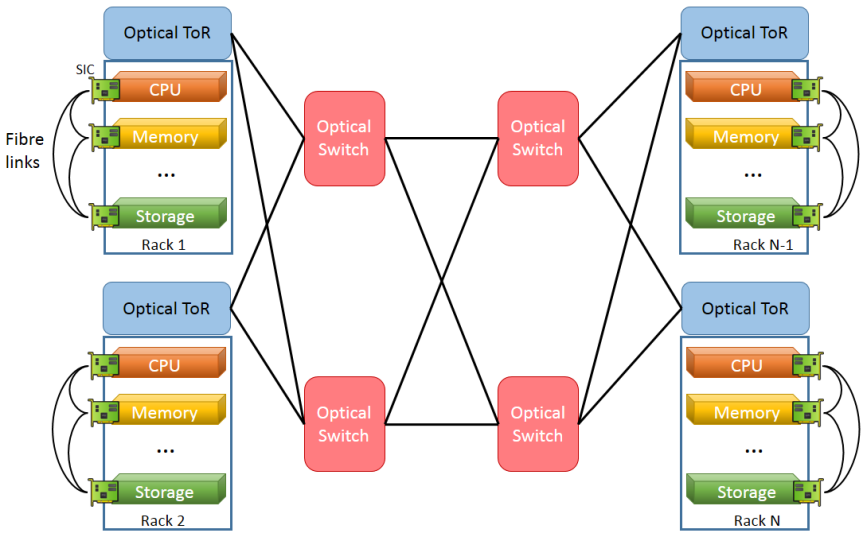


Figura 3. Arquitectura de DC desagregado que asumiremos

En referencia a las VDC, consideraremos que las VMs tienen que ser mapeadas sobre los recursos de un mismo rack, ya que un mapeo sobre distintos racks podría provocar problemas de ancho de banda y latencia. Además, consideraremos que cada una de las VMs pertenecientes a un mismo tenant (VDC) deben ser mapeadas en racks distintos, ganando así resiliencia ante posibles fallos de los racks. Sin embargo cabe recordar que no hay ninguna limitación en que VMs pertenecientes a diferentes tenant puedan ser mapeados en los recursos de un mismo rack. Respecto a los virtual link, consideraremos que cada uno de ellos es capaz de ofrecer el ancho de banda necesario en términos de longitudes de onda para ser mapeado. En redes transparentes como la que vamos a considerar, el tráfico es cursado por medio de conexiones totalmente ópticas, llamadas *lightpaths*. Un lightpath se origina en un transmisor óptico y termina en un receptor óptico, ocupando un único canal de longitud de onda en cada enlace atravesado. Al no ser procesado electrónicamente el tráfico soportado por el lighpath en los nodos intermedios, se pueden conseguir ahorros con respecto al uso de equipamiento de conmutación. Por lo tanto, el mapeo de los links virtuales consiste en determinar los ligthpaths necesarios entre racks para satisfacer los requerimientos de los links virtuales.

Con todo esto, nuestro problema de optimización consistirá en mapear las VDC requeridas de tal manera que se minimicen el total de recursos computaciones de los racks que albergan VMs y el número de diferentes longitudes de onda que será necesario equipar en la red óptica intra-rack para poder ser asignados todos los links virtuales. Con este fin, definimos como $G_n = (N_f, E_f)$ el grafo de red óptica intra-rack, con N_f como el conjunto de nodos físicos (ToRs y conmutadores ópticos) y E_f como el conjunto de fibra ópticas. Defineremos P como el conjunto de caminos físicos entre los ToRs del G_n ; mientras que $P_{e_f} \subseteq P$ se refiere al conjunto de caminos físicos que atraviesan el link e_f. A continuación, definiremos $N_{OS} \subseteq N_f$ como los conmutadores ópticos los cuales tienen un número de puertos igual a N_{port}. Además, nombramos R como el conjunto de racks del DC. Cada rack $r \in R$ tiene asignada una cantidad de núcleos de CPU, memoria y alamcenamiento igual a CPU_r, RAM_r y , HDD_r mientras que cada link de la red física esta equipada con W longitudes de ondas distintas. En referencia a las instancias de VDC, nombraremos D como el conjunto de demandas que tienen que ser servidas. Cada VDC está caracterizada por un grafo virtual $G_d = (N_v^d, E_v^d)$, con N_v^d como el número de VM y E_v^d como el conjunto de links virtuales. Cada VM $n_v \in N_v^d$ requiere un conjunto de capacidades computacionales CPU_{n_v}, $HDD_{n_v}$and RAM_{n_v} mientras que los link virtuales tienen asociados un ancho de banda B_{e_v}. Finalmente, nombraremos $a(\cdot)$ y $b(\cdot)$ al origen y al destino de cada link virtual o camino físico, mientras que $\delta^+(n_f)$ y $\delta^-(n_f)$ seran respectivamente los links de entrada/salida de un nodo físico, $Q_{\delta^+(n_f)} \subseteq P$ y $Q_{\delta^-(n_f)} \subseteq P$ son el conjunto de caminos físicos que contienen un link de entrada/salida del nodo físico n_f. Todas estas definiciones serán necesarias en futuros apartados para poder entender la formulación matemática de las soluciones óptimas que serán propuestas en los dos escenarios.

1.1. Objetivos

El objetivo de este estudio reside en cuantifcar, en el contexto anteriormente expuesto, las mejoras en referencia al número de recursos, tanto ópticos como computacionales, que tienen las infraestrucuras de DC desagreado en comparación a los DC actuales, basados en arquitecturas con uso de servidores de computación tradicionales.

Para llevar a cabo el objetivo propuesto se deben cumplir los requerimientos y especificación presentada en la sección siguente.

1.2. Requerimientos y especificaciones

Requerimientos del proyecto:

- Comprensión del uso de las redes ópticas en un DC tradicional.
- Comprensión del paradigma de DC desagregado.
- Comprensión del concepto de VDC en infraestructuras de DC con DCN ópticas.
- Diseño y formulación de estrategias de optimización para la planificación de VDCs tanto en DCs desagregados como tradicionales con DCN ópticas.
- Implementación y programación de las estrategias.
- Validación de las estrategias y comparativa de los resultados obtenidos en escenarios de DCs desagregados o DC tradicionales.
- Redacción de la documentación

Especificaciones del proyecto:

- Programar en lenguaje JAVA un simulador que cree un DC tradicional y desagregado a partir de un documento .TXT que incluya los siguientes datos:
    - Número de longitudes de onda.
    - Bit-rate soportado por cada longitud de onda.
    - Número de servidores por rack.
    - Número de núcleos de CPU (cores) por servidor.
    - Capacidad de memória (RAM) por servidor.
    - Capacidad de disco duro (HDD) por servidor.
    - Número de puertos de los conmutadores de la red.
    - Matriz de conectividad de la red física del DC, esto es, la DCN.
- Programar en lenguaje JAVA un simulador capaz de crear VDC aleatorias de diferentes características. Dicho simulador deberá retornar un fichero .TXT que contenga la siguiente información:
    - Número de nodos y sus características.
    - Conexiones entre nodos y su bit-rate.
- Diseñar de forma teórica y programar mediante la API para JAVA que proporciona el software de optimización CPLEX [18] una resolución óptima al mapeo de VDC sobre las dos tipos de DC considerados, básicamente, DC desagregado y DC basado en servidores, desplegando ambos DC una DCN basada en tecnología de conmutación de longitud de onda. El algoritmo de mapeo deberá de tener en cuenta las siguientes restricciones:
    - Un nodo virtual no puede estar mapeado en más de un servidor.
    - Cada demanda ha de tener cada uno de sus nodos virtuales en racks diferentes.

- Todo enlace virtual debe ser mapeado.
- No superaremos la capacidad de los servidores (en caso desagregado, la capacidad de los racks).
- Por un mismo enlace físico no pueden pasar dos conexiones ópticas que utilicen la misma longitud de onda.
- Se debe mantener el uso de la misma longitud de onda en todos los enlaces físicos que forman un camino entre dos nodos.
- No se sobrepasara el número de puertos de los conmutadores ópticos.

La finalidad del algoritmo que se diseñará es encontrar la manera óptima de satisfacer las demandas, i.e. los VDCs, en la infraestructura física del DC utilizando tanto el menor número de servidores como el menor número de longitudes de onda. En el caso desagregado, dónde el concepto de servidor no existe, se minimizará igualmente el número de longitudes de onda y se intentará balancear la ocupación de los racks, de esta manera no habrá ningún rack sobrecargado.

- Diseñar y programar sobre el simulador de DC comentado anteriormente estrategias heurísticas que requieran un menor cálculo computacional que el optimizador. La estrategia se considerará válida siempre que no supera en más de 20% el resultado óptimo y tiempo computacional sea del orden igual o inferior de los “segundos”.

1.3. Plan de trabajo

El detalle de las tareas y los hitos expuestos en la planificación temporal se encuentran en el Anexo 1.

Planificación temporal

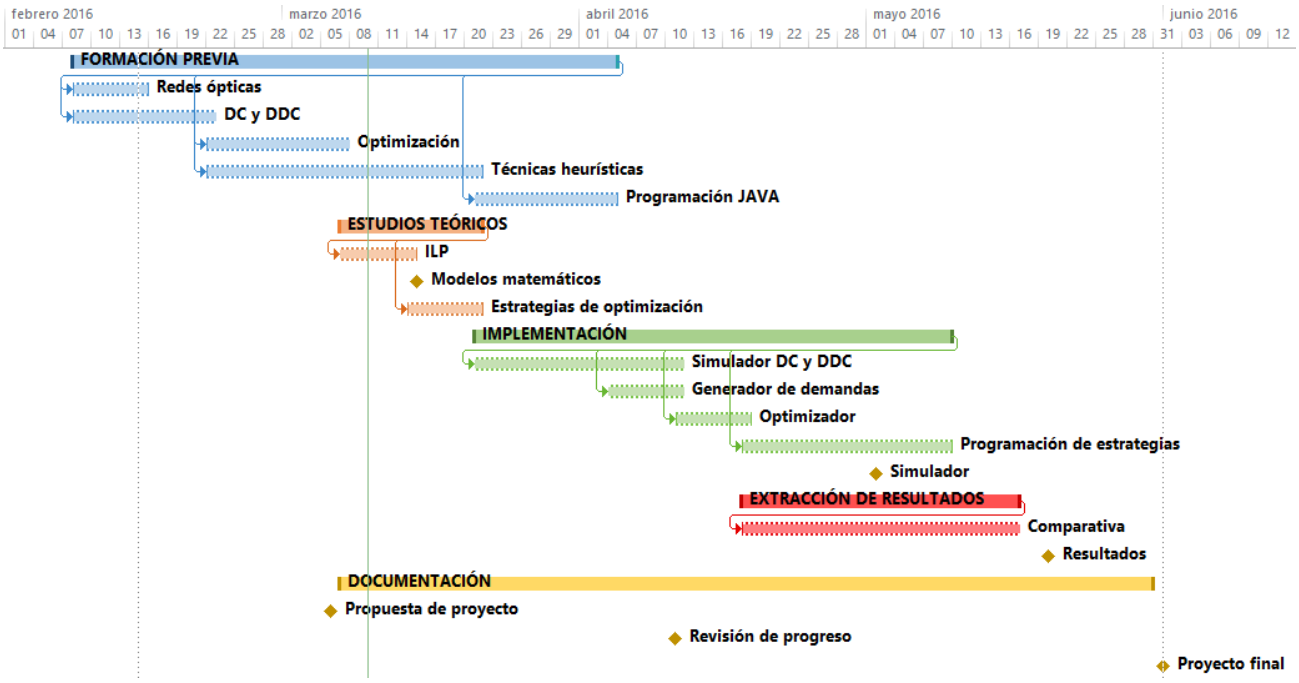


Figura 4. Planificación temporal del Proyecto.

1.4. Modificaciones e incidencias

Incidencias

Antes de todo, cabe destacar que en líneas generales se han cumplido con exactitud los plazos marcados en la propuesta del proyecto. Esto se debe a que a la hora de asignar fechas ya se intentó ser lo máximo preciso posible y teniendo en cuenta un pequeño margen temporal para posibles complicaciones de las cuales éramos conscientes que iban a aparecer durante el desarrollo del proyecto. Asimismo el progreso y la implicación en el proyecto están siendo totalmente acorde a los objetivos marcados, que aunque eran ambiciosos, se han cumpliendo en los plazos establecidos.

A nivel de incidencias, después de la amplia formación adquirida sobre técnicas metaheurísticas, a la hora de ponerla en práctica se valoró la posibilidad que para el escenario en concreto que afrontaríamos resolver, era más conveniente usar estrategias particulares más simples, como son el *first fit* o el *best fit*, en vez de seguir las estrategias genéricas marcadas por las metaheurísticas para la resolución óptima de los problemas. El hecho de que al final se decidiera por particularizar las estrategias para así limitar el abanico de opciones a contemplar para llegar a obtener la óptima, no aminora la importancia de la formación adquirida sobre las técnicas genéricas, ya que los conocimientos asumidos sobre técnicas como pueden ser los algoritmos genéticos o el recocido simulado (*Simulated Annealing* en inglés)[20] son una base ideal para llegar a conceptualizar el problema de la búsqueda de óptimos y los fundamentos a partir de las cuales llegar a formular estrategias más específicas. De esta manera llegamos a la conclusión de que procuraríamos encontrar la manera óptima de resolver cada sub-problema en particular, introduciendo cierta aleatoriedad dentro de nuestro escenario general, por ejemplo la ordenación en la que se atiende las demandas, el orden de mapeo de los nodos de cada demanda, la asignación de longitud de onda… para así explorar mejor el espacio de soluciones.

Por otra parte, durante la implementación de nuestro simulador de datacenter nos encontramos más dificultades de las previstas. Aunque se había procurado tener a nivel teórico todos los conceptos claros, en este punto del desarrollo había que saberlos combinar con un nivel de programación en JAVA exigente que en algunos momentos fue un pequeño obstáculo al que se le tuvo que dedicar una atención adicional para llegar a solventarlo. En particular, la mayor dificultad la encontramos en la implementación de los algoritmos de enrutamiento, ya que intentar programar el algortimo de Dijkstra[21], en que se busca el camino más corto entre dos vértices de un grafo (el término “corto” en este caso no se refiere siempre a distancia, ya que a cada conexión entre dos puntos se le puede asignar cualquier valor especifico) no fue una tarea sencilla y se tuvo refrescar conceptos de programación en JAVA, como el uso de los contenedores, para llegar a resolverla.

Al empezar a obtener los resultados de la solución óptima a través de la API de JAVA que proporciona el software CPLEX, nos sorprendió observar que en algunos casos particulares en los que había que conectar dos nodos próximos entre sí, la solución proponía una camino, que aunque siendo unos de los que no incrementaba el número de longitudes de onda utilizados, implicaba un gran número de saltos (entre nodos) en los que se dejaba en evidencia que teníamos que tener en cuenta ese nueva factor, ya que esa solución no era la óptima que pretendíamos hallar, ya que no se sacaba partido a la

propia estructura de la red para ahorrar recursos. Para solventar este aspecto, se incorporó a la fórmula a minimizar un factor que tuviera en cuenta el número de saltos (hops, en inglés) totales realizados por todos los caminos de la red.

DC:

$$\boldsymbol{MIN}[\boldsymbol{\alpha Nwavelengths} + (\mathbf{1}-\boldsymbol{\alpha})\boldsymbol{Nservers} + \mu\boldsymbol{Nhops}]$$

DDC:

$$\boldsymbol{MIN}[\boldsymbol{\alpha Nwavelengths} + (\mathbf{1}-\boldsymbol{\alpha})[\boldsymbol{\beta Nracks} + (\mathbf{1}-\boldsymbol{\beta})\boldsymbol{Ocupaci\acute{o}nRacks})] + \mu\boldsymbol{Nhops}]$$

Finalmente también tuvimos que resolver el problema que teníamos a nivel de tiempo computacional para llegar a obtener una solución óptima. Estaba claro que ese era un problema a tratar desde buen principio, de ahí la necesidad de encontrar estrategias heurísticas las cuales pretenden obtener una solución similar con la ventaja de que el tiempo computacional era reducido drásticamente. No obstante necesitábamos diversas soluciones óptimas para poderlas usarlas como referente para posteriormente compararlas con las soluciones obtenidas mediante las heurísticas desarrolladas, y dichos cálculos requerían demasiado tiempo. Así que para solucionarlo, decidimos poner una limitación temporal de tal manera que nos aseguraríamos de haber realizado una búsqueda prácticamente completa de las soluciones (alrededor del 95%) invirtiendo un tiempo que no nos dificultaría el desarrollo del proyecto.

Modificaciones

Tal y como ya se ha explicado anteriormente en el apartado de incidencias, la variación adoptada respecto al uso de las técnicas metaheurísitcas y la complejidad requerida en la programación de simulador ha implicado que tanto el tiempo como el contenido que se estableció para la fase de formación tuvieran que ser modificados, ya que debían incluir conceptos de soluciones específicas para problemas de optimización y formación en programación en JAVA. Esto ha implicado que la fase de formación se solapara en algunos casos con la de los estudios teóricos y la de implementación ya que en muchos casos se realizaba una formación adicional ad-hoc para solventar los obstáculos en particular que nos íbamos encontrando.

También se decidió avanzar la fase de extracción de resultados ya que preveíamos que debido a la gran cantidad de tiempo que era necesario para poder obtener soluciones óptimas, podía llegar a ser un punto crítico que nos retrasara el progreso de proyecto. Así que decidimos que una vez tuviéramos programado nuestro modelo óptimo iniciaríamos el proceso de generación de resultados a la par que programaríamos nuestras estrategias particulares.

Finalmente en referencia al bloque de documentación, analizando el avance del proyecto y los resultados que esperamos obtener, contemplamos de manera bastante factible la posibilidad de realizar un artículo ya sea para alguna revista o conferencia científica, este aspecto no variaría nuestro plan de trabajo inicial pero sí que implicaría una serie de documentación adicional (o la misma redactada en una estructura diferente) con tal de cumplir los requisitos que nos demanden para la publicación.

2. Estado previo de la tecnolgia usada en el proyecto:

Desde finales de 2012, el Centro de Comunicaciones de Banda Ancha (CCABA) de la UPC ha participado como partner en los proyectos europeos LIGHTNESS (Low latency and high throughput dynamic network infrastructures for high performance datacentre interconnects, FP7-318606) y COSIGN (Integrating advanced optical hardware and SDN for future all optical DCNs, FP7-619572). Ambos proyectos se han centrado en la introducción de nuevas tecnologías de DCN completamente óptica para entornos de DC, capaces de proporcionar un ancho de banda muy superior a las DCNs eléctricas existentes hoy en día (p.e., basadas en Ethernet), además de requerir de un consumo energético muy inferior, un requisito crítico en los DCs de los próximos años.

En particular, dentro del proyecto COSIGN se han estudiado algunos de los servicios que los DC del futuro deberían ser capaces de proporcionar de forma eficiente a sus usuarios. Entre estos, se ha identificado especialmente interesante el servicio de DC virtual (VDC). Por otro lado, la comunidad científica (incluyendo también los miembros del proyecto COSIGN) prevé una evolución del modelo actual de DC centrado en los servidores a un modelo de DC desagregado, el cuál puede ofrecer una utilización mucho más eficiente de los recursos computacionales (CPU, memoria y disco duro) disponibles.

En este marco de trabajo, el presente TFG intenta atacar un problema de optimización completamente nuevo que los miembros del CCABA se han propuesto investigar, partiendo del conocimiento adquirido en los proyectos europeos LIGHTNESS y COSIGN.

Por lo tanto, no forma parte de alguna continuación de ningún trabajo (p.ej., un TFG) iniciado anteriormente. Cabe mencionar pero, que los resultados obtenidos se intentaran promover dentro del consorcio del proyecto europeo COSIGN, para que puedan dar lugar a colaboraciones a nivel internacional en un futuro próximo.

3. Metodologia/desarrollo del proyecto:

3.1. Metodologia

Tal y como se ha comentado anteriormente en el apartado de Requerimientos y Especificaciones, para poder realizar nuestro estudio es necesario crear un simulador mediante programación Java que tenga en cuenta todos los escenarios posibles y nos permita poder desarrollar nuestras técnicas heurísticas. Además, el simulador se complementa con el api de Java proporcionada por el software de optimización CPLEX para poder programar la formulación ILP. A continuación (Figura 5) se añade un esquema resumen de los ficheros que intervienen en nuestro simulador:

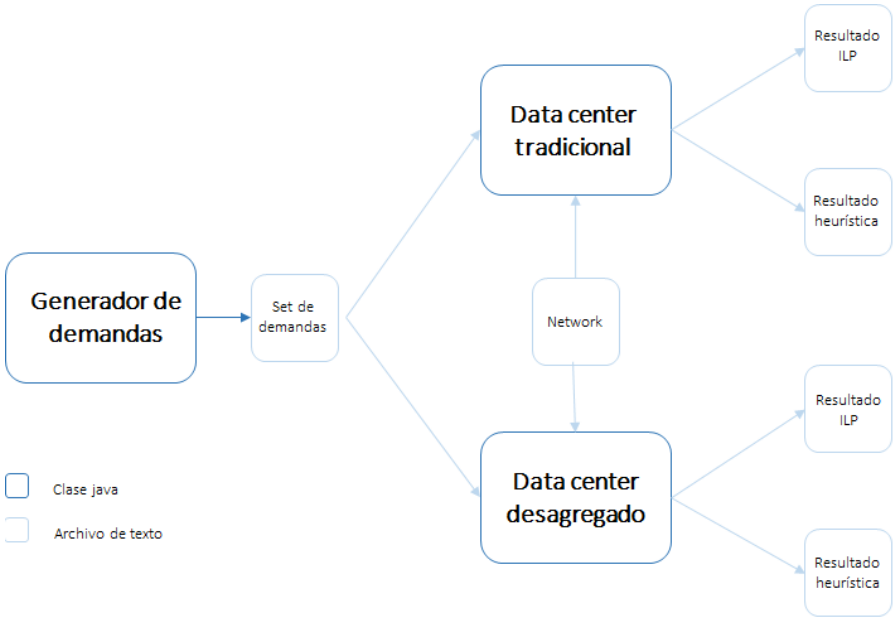


Figura 5. Esquema de los ficheros que intervienen en el simulador

Generador de demandas

En el generador de demandas se generan conjuntos de demandas aleatorias en función de los parámetros determinados por el usuario. Estas variables son las siguientes:

- Número de demandas
- Número de nodos virtuales de cada demanda, también se puede introducir un rango y asignarle una probabilidad a cada elemento. (por defecto entre 3 y 4 nodos virtuales equiprobables)
- Probabilidad con la que se crea un link virtual entre dos nodos de una misma demanda, teniendo en cuenta que los enlaces son bidireccionales (por defecto 50%)
- Ancho de banda de los links virtuales, también se puede introducir un rango y asignarle una probabilidad a cada elemento. (por defecto 100 Gbit/s)
- Perfiles de las demandas.

Finalmente al ejecutar el programa se genera un archivo .txt que contiene demandas con la estructura de la Figura 6:

8; 48; 800

8; 48; 800

22; 16; 200

22; 16; 200

[0:100:100:100]

[100:0:0:0]

[100:0:0:100]

[100:0:100:0]

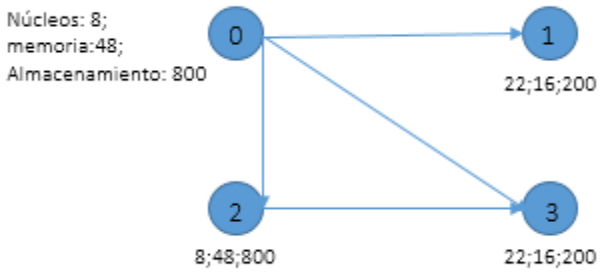


Figura 6. Ejemplo visual de una demanda VDC

Data center tradicional y desagregado

Las clases data center tradicional y la de data center desagregado aunque interiormente sus heurísticas y formulación sean diferentes, ya que en la segunda se deja de tener en cuenta el concepto de servidor, ambas tienen un funcionamiento similar.

Están programadas para recibir demandas que tengan la estructura anteriormente especificada en el generador de demandas. Además es necesario introducirles un documento .txt que especifique la estrucutura física del datacenter:

#número de longitudes de onda (w)
#BitRate per W
#número de servidores por rack
#número de núcleos por servidor
#cantidad de memória por servidor
#cantidad de alamcenamiento por servidor
#número de puertos
[0:0:0:0:0:0:1]*
[0:0:0:0:0:0:1]*
[0:0:0:0:0:0:1]*
[0:0:0:0:0:0:1]*
[0:0:0:0:0:0:1]*
[0:0:0:0:0:0:1]*
[1:1:1:1:1:1:0]

En el caso del DDC calcula las capacidades globales de los racks multiplicando el número de servidores por la cantidad de recursos por servidor. De esta manera se puede aprovechar la misma estructura de fichero .txt para las dos clases.

Tanto a la clase del simulador DC como DDC se le pasa por parámetro cuantas demandas queremos que procese y si queremos que realize el mapeo de dichas demandas mediante nuestra heurística o la formulación matemática ILP. En el caso de la ILP, también le debemos pasar los parámetro que ajustarán la función objetivo.

Al ejecutarse las clases generan un archivo .txt que muestra los resultados de la siguiente manera:

DC:

Resumen recursos utilitzados:

Núcleos: 12528.0 Memoria: 66816.0 Almacenamiento: 1044000.0

Porcentaje medio de los recursos utilitzados:

Núcleos%: 41.26% Memoria%: 44.04% Almacenamiento%: 44.89% Servidor: 43.39%

Tiempo computacional: 1081ms

Servidores utilitzados: 522

Lambdas utilitzadas: 40

DDC:

Utilització máxima del rack: 0.113020

Racks utilitzados: 6

Lambdas utilitzadas: 9

Resumen recursos utilitzados:

Núcleos: 319.0 Memoeria: 1420.0 Amacenamiento: 26200.0

Porcentaje medio de los recursos utilitzados:

% Núcleos%: 11.08% Memoria%: 9.24% Almacenamiento%: 10.92% Rack: 10.41%

Tiempo computacional: 43201992ms

Utilització del rack 0 :11.28%

Utilització del rack 1 :11.29%

Utilització del rack 2 :9.54%

Utilització del rack 3 :11.3%

Utilització del rack 4 :9.52%

Utilització del rack 5 :9.54%

3.2. Data center tradicional

3.2.1. Formulación matemática

A continuación se expone la formulación matemática óptima de un data center tradicional. El objetivo reside en mapear una serie de demandas sobre un datacenter en el cual sus racks están interconectados entre sí mediante un red óptica, con restricciones adicionales como seria que los nodos de una misma demanda tienen que estar mapeado en racks diferente (para ganar resiliencia ante posibles fallos), minimizando el número de longitudes de onda y de servidores utilizados. Parte de la nomenclatura utilizada en la formulación, ha sido expuesta en la descripicón del escenario de la introducción.

Las variables de decisión de la formulación son las siguientes:

$x_{d,n_v,r,s}$: Binaria; 1 si el nodo virtual n_v de la demanda d está mapeado sobre el servidor s del rack r; 0 en caso contrario.

$y_{d,e_v,p,w}$: Binaria; 1 si el link virtual e_v de la demanda d está mapeado usando la longitud de onda w sobre la ruta física p; 0 en caso contrario.

z_w: Binaria; 1 si la longitud de onda w está siendo utilizada en algún link de la red óptica DCN; 0 en caso contrario.

$s_{r,s}$: Binaria; 1 si el servidor s del rack r está siendo utilizado; 0 en caso contrario.

h_p: Real; número de hops (salto de link a link) del path p.

La formulación ILP en detalle está expuesta a continuación:

minimizar $\alpha\left(\sum_{w\in W} z_w + \varepsilon \sum_{d\in D}\sum_{e_v\in E_v^d}\sum_{p\in P}\sum_{w\in W} h_p y_{d,e_v,p,w}\right) + (1-\alpha)\sum_{r\in R}\sum_{s\in S_r} s_{r,s}$ (1)

sujeto a:

$$\sum_{r\in R}\sum_{s\in S_r} x_{d,n_v,r,s} = 1 \qquad \forall\, d\in D, n_v \in N_v^d \quad (2)$$

$$\sum_{n_v\in N_v^d}\sum_{s\in S_r} x_{d,n_v,r,s} \le 1 \qquad \forall\, d\in D, r\in R \quad (3)$$

$$\sum_{p\in P}\sum_{w\in W} y_{d,e_v,p,w} = B_{e_v} \qquad \forall\, d\in D, e_v\in E_v^d \quad (4)$$

$$y_{d,e_v,p,w} \le \sum_{s\in S_{a(p)}} x_{d,a(e_v),a(p),s} \qquad \forall\, d\in D, e_v\in E_v^d, p\in P, w\in W \quad (5.a)$$

$$y_{d,e_v,p,w} \le \sum_{s\in S_{b(p)}} x_{d,b(e_v),b(p),s} \qquad \forall\, d\in D, e_v\in E_v^d, p\in P, w\in W \quad (5.b)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} CPU_{n_v} x_{d,n_v,r,s} \le CPU_s \qquad \forall\, r\in R, s\in S_r \quad (6.a)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} RAM_{n_v} x_{d,n_v,r,s} \le RAM_s \qquad \forall\, r\in R, s\in S_r \quad (6.b)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} HDD_{n_v} x_{d,n_v,r,s} \le HDD_s \qquad \forall\, r\in R, s\in S_r \quad (6.c)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in P_{e_f}} y_{d,e_v,p,w} \le 1 \qquad \forall\, e_f\in E_f, w\in W \quad (7)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in Q_{\delta^+(n_f)}}\sum_{w\in W} y_{d,e_v,p,w} \le N_{port} \qquad \forall\, n_f\in N_{OS} \quad (8.a)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in Q_{\delta^-(n_f)}}\sum_{w\in W} y_{d,e_v,p,w} \le N_{port} \qquad \forall\, n_f \in N_{OS} \tag{8.b}$$

$$y_{d,e_v,p,w} \le z_w \qquad \forall\, d \in D, e_v \in E_v^d, p \in P, w \in W \tag{9}$$

$$x_{d,n_v,r,s} \le s_{r,s} \qquad \forall\, d \in D, n_v \in N_v^d, r \in R, s \in S_r \tag{10}$$

El objetivo a optimizar expuesto en la Ec. (1) es doble: por un lado, minimiza el número de diferentes longitudes de onda a equipar en los links de la red del DCN (primer término) mientras por otro lado se minimiza el número total de servidores para soportar las VMs de las VDC (tercer término). El parámetro α es un número real positivo en el rango de [0, 1] que se usa para ajustar el énfasis que se quiera otorgar a los dos sub-objetivos. Adicionalmente, un término secundario es añadido en el primer objetivo (el segundo término del sumatorio), con el parámetro ε siendo un número positivo de valor ínfimo (p.e., $\varepsilon \ll 1$). El objetivo de este término es priorizar la soluciones que requieran un menor numeró de láseres usados (wavelength channels) en la red, en el caso de que dos o más soluciones impliquen el mismo número de diferentes longitudes de ondas a equipar en los enlaces físicos.

Por lo que se refiere a las restricciones: la restricción (2) asegura que todos los nodos virtuales de la demanda (una instancia VDC) tienen que ser mapeados en los servidores físicos, forzando a que un nodo particular solo puede estar mapeado en un único servidor. La restricción (3) garantiza que todos los servidores de un rack en concreto como máximo puedan ser usados para mapear un nodo de una instancia VDC, esto implica que todos los nodos virtuales de una VDC están mapeados sobre racks diferentes, otorgando de esta manera cierto grado de resiliencia ante posibles fallos de los servidores y los racks. Sin embargo, recalcar que los nodos virtuales pertenecientes a diferentes instancias de VDCs pueden estar mapeados sobre un mismo rack o servidor físico. La restricción (4) es la restricción del mapeo de links, asegurando que cada link virtual se mapea con el número de ligthpaths necesario, mientras que la restricción (5) obliga al mapeo de los links virtuales sobre rutas físicas que interconecten puntos de entrada y de salida de los racks en los cuales los nodos origen y destinos de los links han sido mapeados. A continuación la restricción (6) asegura que la capacidad de los servidores no es sobrepasada, es decir, el sumatorio de los recursos de todos los nodos virtuales (VMs) mapeados en un servidor no excede la disponibilidad de ninguno de los recursos del servidor (núcleos CPU, memoria y almacenamiento). La restricción (7) es la conocida en inglés como la “clashing constraint”, en la cual se impide que dos ligthpaths sean mapeados sobre un mismo link físico usando la misma longitud de onda. La restricción (8) tiene en cuenta las limitaciones de los switches ópticos, limitando que el número de canales de longitudes de onda de entrada/salida no supere el límite de puertos del switch. Por último, la restricción (9) y (10) sirven respectivamente para identificar si una longitud de onda o un servidor son utilizados en la infraestructura del DC.

3.2.2. Técnicas heurísticas

Durante el proceso del mapeo de las demandas en nuestro data center tradicional hay varios puntos donde entra en juego la aleatoriedad y donde vamos a aplicar técnicas heurísticas particulares con el objetivo de obtener un resultado lo máximo parecido

posible al obtenido en la ILP, pero reduciendo el tempo computacional drásticamente hasta el orden de los milisegundos.

Los puntos en los cuales hemos obtenido claros beneficios al aplicar técnicas heurísticas son los siguientes:

- Orden en que se mapean las demandas.
- Orden en que se mapean los nodos virtuales de cada demanda.
- Técnica de mapeo de los nodos sobre los servidores físicos.
- Técnica de asignación de rutas y longitudes de onda.

Inicialmente ordenamos las demandas del set de demandas de mayor a menor según su recurso más restrictivo, eso quiere decir que la demanda mayor será aquella que incluya el nodo virtual en que alguno de sus recursos ocupe porcentualmente más un servidor. Por ejemplo si un nodo virtual ocupa un servidor de la siguiente manera: Núcleos: 25% Memoria: 25% Almacenamiento 25%, y otra: Núcleos: 10% Memoria: 50% Almacenamiento 10%. El nodo virtual mayor es el segundo y a su vez será mayor una demanda que contenga éste antes que una en que su mayor nodo virtual sea el primero. Una vez ordenadas las demandas, las mapearemos recursivamente.

Al mapear una demanda volveremos a ordenar sus nodos virtuales de mayor a menor tal y como se ha descrito anteriormente. Seguidamente maparemos cada nodo virtual en los servidores físicos teniendo preferencia en ocupar los servidores que ya han sido utilizados, y si no es posible en ninguno de ellos, lo colocaremos en el primer servidor libre del rack que tenga menos servidores ocupados.

Finalmente mapearemos los links virtuales obteniendo los K caminos más cortos (K-Shortest Path)[19] entre los nodos físicos donde han sido mapeados su nodo virtual origen y final. Y le asignaremos la longitud de onda mediante First Fit (la primera libre disponible).

A continuación se exponge el pseudcódigo de la heurística que ha sido implementada:

```
//Ordenamos las demandas d del set de demandas D de mayor a menor
Collection.sort (D)
//Mapeamos las demandas
For d=1 hasta |D|
        //ordenamos nodos virtuales Nv de cada demanda d de mayor a menor
        Collection.sort(Nv)
        For nv=1 hasta |Nv|
                //mapeamos los nodos virtuales con técnica Best-Fit
                For r=1 hasta |R|
                        If ( r no usado)
                        For sr=1 hasta |Sr|
                                If ( sr soporta nv)
                                        If( sr utilizado)
                                                Mapeamos nv sobre sr
                                                 r  usado
                                                mapeado true
```

else

if(sr<minServer)

minServer =sr

If(mapeado false)

Mapeamos nv sobre minServer

r de minServer usado

// mapeamos los link virtuales con técnica K-SP-FF

For lv=1 hasta |Lv|

rackOrigen = rack del servidor donde se ha mapeado el nodo virtual origen de lv

rackDestino = rack del servidor donde se ha mapeado el nodo virtual desino de lv

Mapeamos rackOrigen y rackDestino con técnica K-SP-FF.

Figura 7. Heurística DC basado en servidores

3.3. Data center desagregado

3.3.1. Formulación matemàtica

A continuación se expone la formulación matemática óptima de un data center desagregado. El objetivo reside en mapear una serie de demandas sobre un datacenter en el cual sus racks están interconectados entre sí mediante un red óptica, con restricciones adicionales como seria que los nodos de una misma demanda tienen que estar mapeado en racks diferentes (para ganar resiliencia ante posibles fallos), minimizando el número de longitudes de onda, el número de racks utilizados e intentando que los racks estén lo máximo balanceados entre sí, es decir que tengan un número similar de recursos utilizados. Parte de la nomenclatura utilizada en la formulación, ha sido expuesta en la descripicón del escenario de la introducción.

Las variables de decisión de la formulación son las siguientes:

$x_{d,n_v,r,s}$: Binaria; 1 si el nodo virtual n_v de la demanda d está mapeado sobre el servidor s del rack r; 0 en caso contrario.

$y_{d,e_v,p,w}$: Binaria; 1 si el link virtual e_v de la demanda d está mapeado usando la longitud de onda w sobre la ruta física p; 0 en caso contrario.

z_w: Binaria; 1 si la longitud de onda w está siendo utilizada en algún link de la red óptica DCN; 0 en caso contrario.

u_r: Binaria; 1 si el rack r está siendo utilizado; 0 en caso contrario.

u_{max}: Real; el promedio de los recursos utilizados del rack del datacentre que tenga dicho promedio más elevado.

h_p: Real; número de hops (salto de link a link) del path p.

La formulación ILP en detalle está expuesta a continuación:

$$\textit{minimizar } \alpha\left(\sum_{w\in W} z_w + \varepsilon \sum_{d\in D}\sum_{e_v\in E_v^d}\sum_{p\in P}\sum_{w\in W} h_p y_{d,e_v,p,w}\right) + (1-\alpha)(\beta\sum_{r\in R} u_r + (1-\beta)u_{max}) \quad (11)$$

sujeto a:

$$\sum_{r\in R} x_{d,n_v,r} = 1 \qquad \forall\, d\in D, n\in N_v^d \quad (12)$$

$$\sum_{n_v\in N_v^d} x_{d,n_v,r} \le 1 \qquad \forall\, d\in D, r\in R \quad (13)$$

$$\sum_{p\in P}\sum_{w\in W} y_{d,e_v,p,w} = B_{e_v} \qquad \forall\, d\in D, e_v\in E_v^d \quad (14)$$

$$y_{d,e_v,p,w} \le x_{d,a(e_v),a(p)} \qquad \forall\, d\in D, e_v\in E_v^d, p\in P, w\in W \quad (15.a)$$

$$y_{d,e_v,p,w} \le x_{d,b(e_v),b(p)} \qquad \forall\, d\in D, e_v\in E_v^d, p\in P, w\in W \quad (15.b)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} CPU_{n_v} x_{d,n_v,r} \le CPU_r \qquad \forall\, r\in R \quad (16.a)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} RAM_{n_v} x_{d,n_v,r} \le RAM_r \qquad \forall\, r\in R \quad (16.b)$$

$$\sum_{d\in D}\sum_{n_v\in N_v^d} HDD_{n_v} x_{d,n_v,r} \le HDD_r \qquad \forall\, r\in R \quad (16.c)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in P_{e_f}} y_{d,e_v,p,w} \le 1 \qquad \forall\, e_f\in E_f, w\in W \quad (17)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in Q_{\delta^+(n_f)}}\sum_{w\in W} y_{d,e_v,p,w} \le N_{port} \qquad \forall\, n_f\in N_{OS} \quad (18.a)$$

$$\sum_{d\in D}\sum_{e_v E_v^d}\sum_{p\in Q_{\delta^-(n_f)}}\sum_{w\in W} y_{d,e_v,p,w} \le N_{port} \qquad \forall\, n_f\in N_{OS} \quad (18.b)$$

$$y_{d,e_v,p,w} \le z_w \qquad \forall\, d\in D, e_v\in E_v^d, p\in P, w\in W \quad (19)$$

$$x_{d,n_v,r} \le u_r \qquad \forall\, d\in D, n_v\in N_v^d, r\in R \quad (20)$$

$$\frac{1}{3}\left(\frac{\sum_{d\in D}\sum_{n_v\in N_v^d} CPU_{n_v} x_{d,n_v,r}}{CPU_r} + \frac{\sum_{d\in D}\sum_{n_v\in N_v^d} RAM_{n_v} x_{d,n_v,r}}{RAM_r} + \frac{\sum_{d\in D}\sum_{n_v\in N_v^d} HDD_{n_v} x_{d,n_v,r}}{HDD_r}\right) \le u_{max} \quad \forall\, r\in R \quad (21)$$

El objetivo a optimizar expuesto en la Ec. (11) es doble: por un lado, minimiza el número de diferentes longitudes de onda a equipar en los links de la red del DCN (primer término), mientras por otro lado se minimiza el número total de racks para soportar las VMs de las VDC (tercer término) y su correspondiente balanceo en referencia a los recursos utilizados (cuarto término). Tanto el parámetro α como el parámetro β son

números reales positivos en el rango de [0, 1] que se usan para ajustar el énfasis que se quiera a otorgar a los sub-objetivos. Adicionalmente, un término secundario es añadido en el primer objetivo (el segundo término del sumatorio), con el parámetro ε siendo un número positivo de valor ínfimo (p.e., $\varepsilon \ll 1$). El objetivo de este término es priorizar la soluciones que requieran un menor numeró de láseres usados (wavelength channels) en la red, en el caso de que dos o más soluciones impliquen el mismo número de diferentes longitudes de ondas a equipar en los enlaces físicos.

Por lo que se refiere a las restricciones, la restricción (12) asegura que todos los nodos virtuales de la demanda (p.e., una instancia VDC) tengan que ser mapeados en los racks físicos, forzando a que un nodo particular solo puede estar mapeado en un único rack. La restricción (13) garantiza que todos los nodos virtuales de una VDC están mapeados sobre racks diferentes. Otorgando de esta manera algunos grados de resiliencia ante posibles fallos de los racks. Sin embargo, recalcar que los nodos virtuales pertenecientes a diferentes instancias VDC pueden estar mapeados sobre un mismo rack. La restricción (14) es la restricción del mapeo de links, asegurando que cada link virtual se mapea con el número de ligthpaths necesario, mientras que la restricción (15) obliga al mapeo de los links virtuales sobre rutas físicas que interconecten puntos de entrada y de salida de los racks en los cuales los nodos origen y destino los links han sido mapeados. A continuación la restricción (16) asegura que la capacidad de los racks no es sobrepasada, es decir, el sumatorio de los recursos de todos los nodos virtuales (VMs) mapeados en un rack no excede la disponibilidad de ninguno de los recursos del rack (núcleos CPU, memoria y almacenamiento). La restricción (17) es la conocida en inglés como la “clashing constraint”, en la cual se impide que dos ligthpaths sean mapeados sobre un mismo link físico. La restricción (18) tiene en cuenta las limitaciones de los switches ópticos, limitando que el número de canales de longitudes de onda de entrada/salida no supere el límite de puertos del switch. Por último, la restricción (19), (20) sirven respectivamente para identificar si una longitud de onda o un servidor son utilizados en la infraestructura del DC. La restricción (21) identifica el promedio máximo de recursos utilizados de entre todos los racks del DC.

3.3.2. Técnicas heurísitcas

Durante el proceso del mapeo de las demandas en nuestro data center desagregado hay varios puntos donde entra en juego la aleatoriedad y donde vamos a aplicar técnicas heurísticas particulares con el objetivo de obtener un resultado lo máximo parecido posible al obtenido en la ILP, pero reduciendo el tempo computacional drásticamente hasta el orden de los segundos.

En los puntos en los cuales hemos obtenido claros beneficios al aplicar técnicas heurísticas son los siguientes:

- Orden en que se mapean las demandas.
- Orden en que se mapean los nodos virtuales de cada demanda.
- Técnica de mapeo de los nodos sobre los racks físicos.
- Técnica de asignación de rutas y longitudes de onda.

Inicialmente ordenamos las demandas del set de demandas de mayor a menor según su recurso más restrictivo, eso quiere decir que la demanda mayor será aquella que incluya el nodo virtual que alguno de sus recursos ocupe porcentualmente más un servidor. Por ejemplo si un nodo virtual ocupa un servidor de la siguiente manera: Núcleos: 25% Memoria: 25% Almacenamiento 25%, y otra: Núcleos: 10% Memoria: 50%

Almacenamiento 10%. El nodo virtual mayor es el segundo y a su vez será mayor una demanda que contenga éste antes que una en que su mayor nodo virtual sea el primero. Una vez ordenadas las demandas, las mapearemos recursivamente.

Al mapear una demanda volveremos a ordenar sus nodos virtuales de mayor a menor tal y como se ha descrito anteriormente. Seguidamente mapearemos cada nodo virtual en los racks físicos teniendo preferencia en colocar el nodo virtual mayor de la demanda en el rack que en promedio tenga sus recursos más desocupados, los siguientes nodos virtuales de la demanda irán colocados en los racks consecutivos a este último. De esta manera reducimos la probabilidad de que para conectar dos racks se tenga que pasar por los nodos superiores del datacenter que es donde se suelen congestionar los enlaces y requieren el uso de más longitudes de onda.

Finalmente mapearemos los links virtuales obteniendo los K caminos más cortos (K-Shortest Path) entre los nodos físicos donde han sido mapeados su nodo virtual origen y final. Y le asignaremos la longitud de onda mediante First Fit (la primera libre disponible).

A continuación se exponge el pseudcódigo de la heurística que ha sido implementada:

```
//Ordenamos las demandas d del set de demandas D de mayor a menor
        Collection.sort (D)
        //Mapeamos las demandas
        For d=1 hasta |D|
                //ordenamos nodos virtuales Nv de cada demanda d de mayor a menor
                Collection.sort(Nv)
                For nv=1 hasta |Nv|
                        //ordenamos los racks de menor a mayor ocupación
                        Collection.sort(R)
                                mapeamos nv sobre r más desocupados
                                mapeamos el resto de nv en los racks físicamente consecutivos a r
                // mapeamos los link virtuales con técnica K-SP-FF
                For lv=1 hasta |Lv|
                        rackOrigen = rack del servidor donde se ha mapeado el nodo virtual origen de lv
                        rackDestino = rack del servidor donde se ha mapeado el nodo virtual desino de lv
                        Mapeamos rackOrigen y rackDestino con técnica K-SP-FF.
```

Figura 8. Heurística DC desagregado

4. Resultados

En esta sección, vamos a evaluar la potencial reducción en términos de recursos computacionales en la asignación de instancias VDC sobre un DC desagregado en comparación a un DC con arquitectura basada en servidores. Con este objetivo, hemos asumido que el conjunto de servidores de un rack del DC tradicional contiene la misma cantidad de recursos del cúmulo de los hardwares del rack del DC desagregado. Esto nos permitirá realizar una comparación justa entre los dos escenarios. Recalcar que esta cantidad será lo suficientemente grande para permitirnos mapear todas las instancias de VDC, por lo tanto no tendremos en cuenta problemas de bloqueos de demandas. Por lo que se refiere al set de demandas, las VDCs son generadas aleatoriamente en dos pasos: primero, cada una se generará de manera equiprobable con 3 o 4 VMs. A continuación, las VMs son interconectadas entre ellas mediante virtual links con la misma probabilidad, es decir entre dos VMs de una VDC hay la misma probabilidad que se cree un virtual link como que no. No obstante, se impide a que se cree un grafo virtual no conectado (no haya ningún camino posible entre dos VMs). Por simplicidad, asumiremos que cada link virtual requiere una longitud de onda, de esta manera nos centramos a analizar las diferencias en la ocupación de los recursos computacionales. Con este fin, consideraremos cuatro tipo de perfiles de VM, cada uno con unas características determinadas de recursos computacionales (núcleos CPU, memoria y almacenamiento): GP = (8, 48, 800), CO = (22, 16, 200), MO = (3, 116, 200) and SO = (3, 16, 1800). Consideraremos que la capacidad por servidor es de (24, 128, 2000). La selección de estos valores está inspirada en el servicio Amazon Elastic Compute Cloud (EC2) [23]. Siguiendo este perfil de VM, consideraremos diferentes topologías de instancias VDC en función del tipo de sus VMs: T_1 = (25% GP, 25% CO, 25% MO, 25% SO), T_2 = (70% GP, 10% CO, 10% MO, 10% SO) and T_3 = (10% GP, 30% CO, 30% MO, 30% SO). Se ha tenido en cuenta esta consideración ya que es necesario analizar el comportamiento del DC desagregado y el basado en servidores ante VDC heterogéneas en términos de recursos computacionales.

Con esos parámetros, primero evaluaremos la actuación de las heurísticas propuestas en comparación de las formulaciones ILP, comparando el valor de la función objetivo y los tiempos de ejecución. El escenario considerado consiste en una estructura en forma de árbol de 6 racks interconectados en un conmutador óptico central, con un número de puertos lo suficientemente grande. En el caso de la arquitectura basada en servidores, cada uno de estos racks está equipado con 20 servidores. Para esta comparación nos centraremos en demandas del perfil T_1. Entonces, exploraremos conjuntos de demandas de tamaños distintos. Los resultados obtenidos están representados en la Tabla 1 con cada uno de estos valores obtenido a partir de la media de 20 instancias. Cada ejecución se ha realizado en PCs i7-4770 con 3.4GHz CPUs y 16GB de memoria, utilizando el software de optimización CPLEX v12.5. En nuestro caso hemos asignado tanto a α como β un valor de 0.5, lo cual nos da el scenario lo más compensado posible. No obstante está selección de valores depende de los criterios del operador DC en función de los intereses de su escenario.

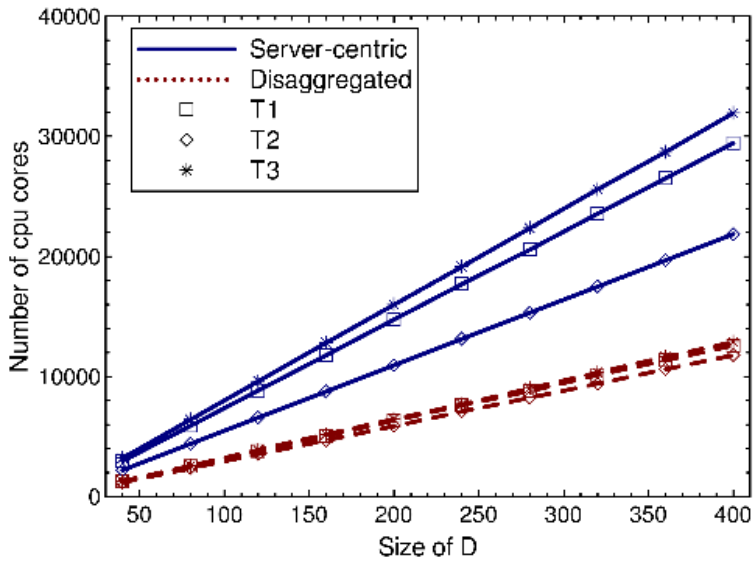
COMPARACIÓN DE HEURÍSTICA VS. FORMULACION ÓPTIMA MEDIANTE ILP

		ILP		Heuristica		
Escenario	$\|D\|$	Objectivo	Tiempo	Objectivo	Tiempo (ms)	Error (%)
Servidores	10	22.5	>12h	23.8	29.2	5.7
	15	30.5	>12h	31.5	33.4	3.3
	20	40.1	>12h	40.7	38.1	1.5
	25	49.6	>12h	50.5	40.2	1.8
	30	62.7	>12h	64.5	48.5	2.9
Desagregado	10	6.02	>12h	6.62	23.1	10.1
	15	8.54	>12h	9.04	26.7	5.8
	20	11.56	>12h	12.55	30.4	8.5
	25	12.57	>12h	13.57	32.3	7.9
	30	17.1	>12h	18.4	38.2	7.6

Tabla 2. Comparación de heurística contra formulaciones ILP.

Podemos apreciar que la heurística propuesta obtiene unos resultados favorables en comparación del caso óptimo obtenido a través de las formulaciones ILP, con menos de un 10% de error relativo. Adicionalmente, se reducen varios órdenes de magnitud el tiempo computacional necesario para obtener las soluciones. Por esta razón, concluimos que las heurísticas son exitosas aproximándose al resultado óptimo, por lo tanto pueden ser utilizadas para solucionar eficientemente problemas con gran cantidad de instancias.

Después de haber evaluado los mecanismos heurísticos, procederemos a analizar la cantidad necesaria de recursos computacionales con los que hay que equipar los racks del DC en función de los diferentes perfiles de VM de las instancias VDC y de diferentes tamaños de conjuntos de demandas. Con este fin, consideramos un escenario DC compuesto por 64 racks con 48 servidores cada uno conectados en una topología de árbol, como la mostrada en la primera imagen de la Figura10. La Figura 9 muestra la comparación de recursos computacionales necesarios (Núcleos de CPU, memoria y almacenamiento) para los tres perfiles de VM (T_1, T_2, T_3) y en las dos arquitecturas DC (basada en servidores y desagregado). Todos estos resultados son obtenidos mediante la heurística anteriormente presentada y promediando 100 instancias aleatorias para cada valor.



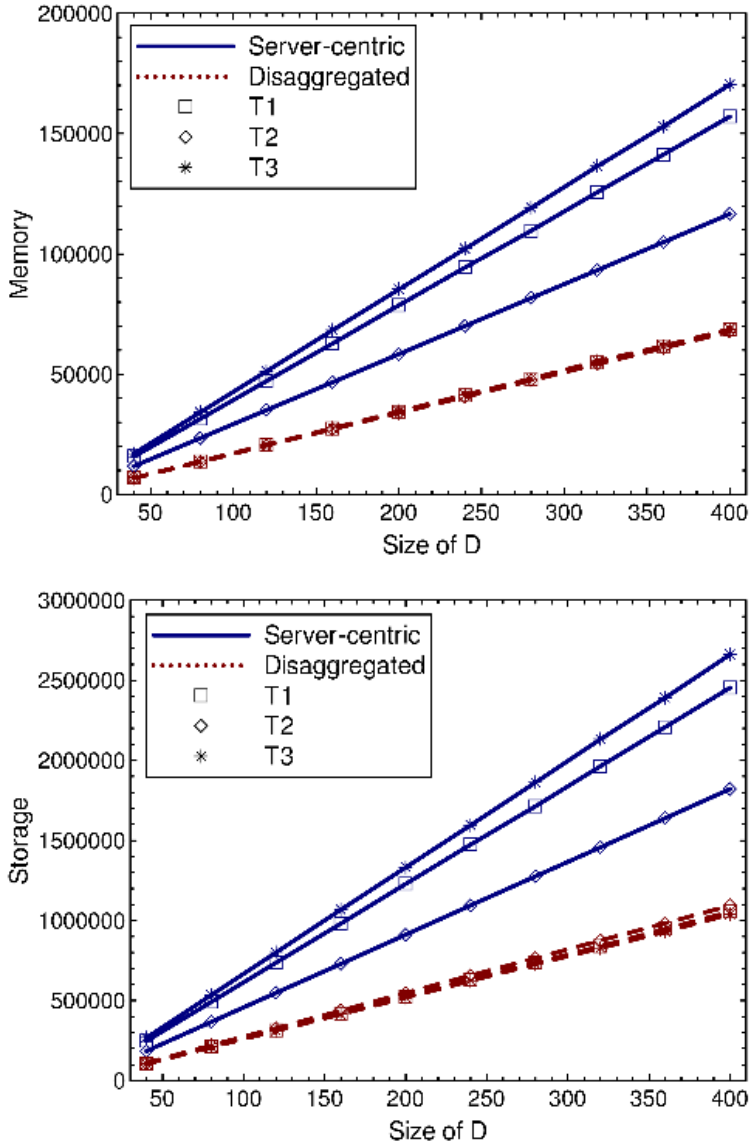


Figura 9. Comparación entre arquitectura de DC basada en servidores y desagregado en términos de uso de recursos computacionales (núcleos de CPU, memoria, almacenamiento)

Podemos observar como la arquitectura de DC desagregado permite una reducción substancial en términos de recursos computacionales. En concreto, reducciones de alrededor del 46% pueden ser apreciadas para todo tipo de recursos y tamaños de conjuntos de demandas en el caso de VMs que no son exigentes respecto a la cantidad de recursos que ocupan (T_2), mientras que la reducción de recursos puede llegar hasta alrededor del 60% en presencia de VMs exigentes computacionalmente, como en el caso de la T_3. Esto se debe, en que con este tipo de demandas es más complicado reutilizar un servidor que ya está alojando una VM. Por otro lado, en los DC desagregados, los recursos ocupados son exactamente los necesarios que las demandas requieren, de ahí que observemos estas reducciones.

A continuación, para concluir nuestro estudio, analizaremos la influencia de la topología DC en términos de recursos para albergar las instancias VDC. Con este fin, además de tratar la topología Tree anteriormente descrita, usaremos topologías Fat-Tree y Spine-Leaf tal y como están representadas en la figura 10. El resto de parámetros del DC y de las VDC se mantendrán igual que en el caso anterior. Los resultados obtenidos revelan

que hay poco efecto en variar las topologías en términos de recursos computacionales, es decir, en el mapeo de las VM. Además el número de longitudes de ondas por links físicos es generalmente independiente al perfil de las VM. No obstante, observamos una gran dependencia entre el número de longitudes de onda con el que hay que equipar los links físicos y la topología DCN. Por esto, vamos a centrar nuestra atención únicamente en analizar demandas de perfil T_2. La figura 11 representa los resultados obtenido para las tres topologías en los dos tipos de arquitecturas DC en función del tamaño del conjunto de las demandas.

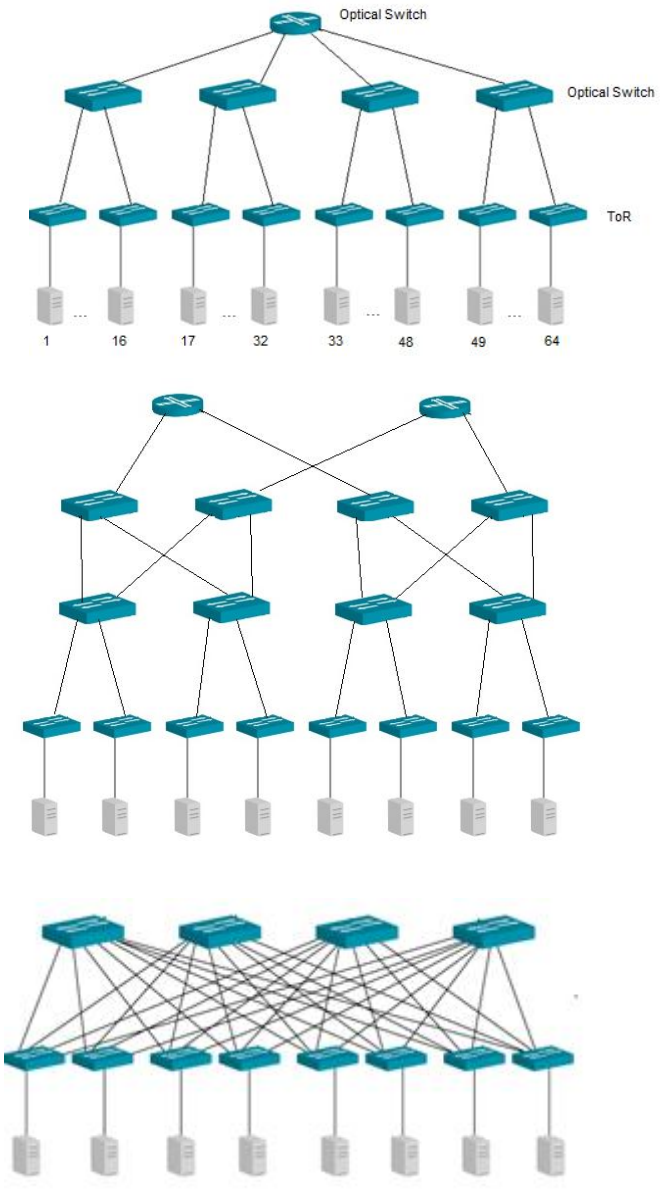


Figura 10. En orden descentende: Topologías Tree, Fat-Tree, Spine Leaf

Bajo esta premisa, podemos apreciar que la topología tree es la que requiere más longitudes de ondas por enlace físico, hasta tres veces más que la topología Fat-Tre y más de un orden de magnitud que la topología Spine-Leaf. Una mención especial requiere esta última topología (Spine-Leaf), la cual necesita una cantidad muy baja de

longitudes de onda. Esto es debido a la gran cantidad de caminos posibles que hay entre dos ToRs de la red, siendo así fácil reutilizar longitudes de ondas que ya han sido utilizadas en otros links de la red. En conclusión, podemos determinar que una DCN enmallada permite al operador gastar menos al equipar la red con diferentes canales de longitudes de onda, lo que implica menos transpondedores WDM en los orígenes (los blades en el caso desagregado y los ToR en la arquitectura basada en servidores). Sin embargo, este tipo de redes requieren un mayor número de nodos y links físicos, hecho que supone un incremento del coste de la red dependiendo del coste unitario de estos dos elementos. Por lo tanto, el operador DC debe decidir cuidadosamente que término de coste del DC para albergar VDC le interesa minimizar. La optimización del coste de la red, teniendo en cuenta de la topología óptima en número de nodos y links queda fuera del alcance de este estudio y se deja para futuros trabajos.

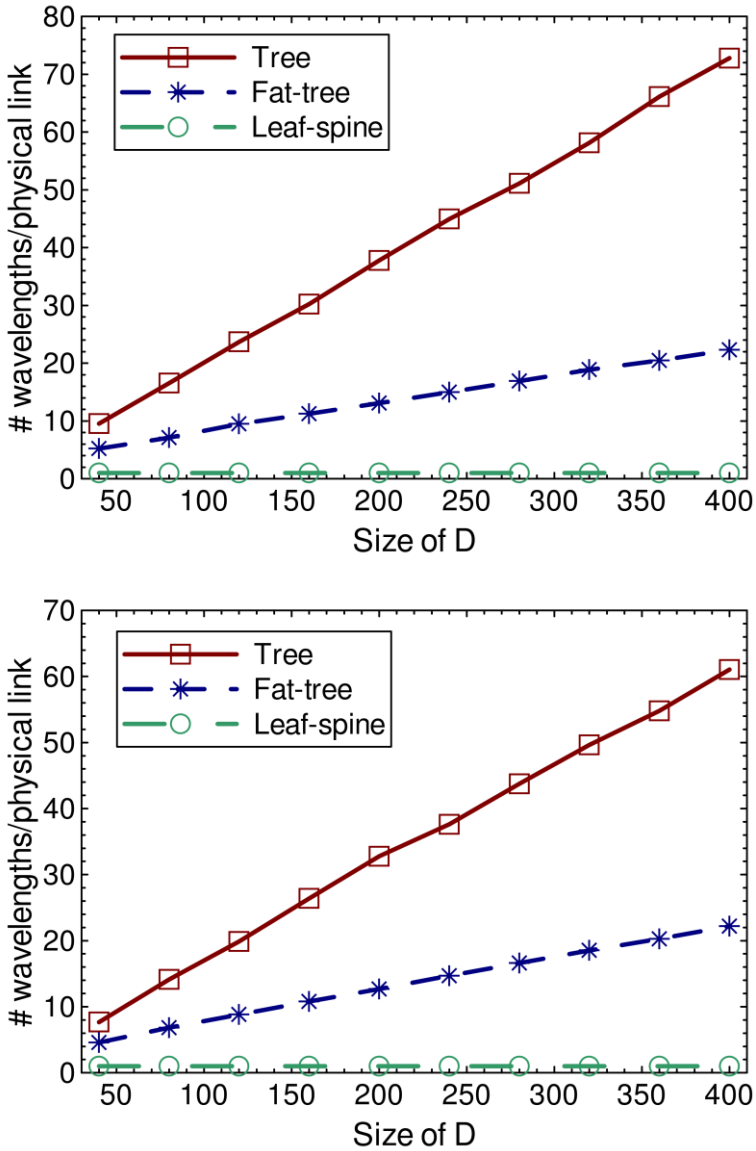


Figura 11.Comparación del número de longitudes de onda necesarios en función de la topologia del DC para las dos tipos de arqitecturas: basada en servidores y desagregada (orden descendente).

5. Costes

El cálculo estimado del presupuesto de este proyecto tiene dos partes distintas, ya que se trata de un estudio teórico y no se han realizado pruebas en un data center físico. De todos modos, hay otros aspectos a tener en cuenta para determinar el coste total del plan de trabajo, como serían: soporte utilizado y las horas dedicadas.

Respecto al soporte se ha utilizado un ordenador y como herramienta de diseño del simulador se ha usado Java con la API proporcionada por el software CPLEX, adicionalmente también se hizo uso de tres ordenador de mayor potencia para las ejecuciones de las ILP. Además, el proyecto tiene asignado 24 créditos ETCS y cada uno de ellos equivale en promedio a una dedicación de 27,5 horas.

La licencia de CPLEX no ha tenido ningún coste específico al ser estudiante de la UPC. Por lo que hace referencia a la amortización de un ordenador, en la cual su vida útil en una empresa es de 3 años, teniendo un coste de venta de 800€ el precio de la amortización es de 111€.

$$\frac{5\ meses}{36\ meses} * 100 = 14\% \longrightarrow 800\ € * 14\% = 111€$$

El coste de los ordenadores de cálculo se determina de la misma manera:

$$\frac{0.25\ meses}{36\ meses} * 100 = 0{,}69\% \longrightarrow\ 3 * 1200\ € * 0{,}69\% = 24{,}84\ €$$

Después, teniendo en cuenta las horas empleadas siendo un ingeniero junior con un salario de 12€/h:

$$24\ ETCS * \frac{27{,}5\ h}{1\ ETCS} * 12\frac{€}{h} = 7.920€$$

Con estos datos se puede aproximar un coste total del proyecto:

Item	Coste
Ordenadores de cálculo	24,84 €
Ordenador principal	111 €
Salario	7.920 €
TOTAL	8.055,84 €

Tabla 2. Resumen de costes

6. Impacto medioambiental

Los Data Centers son la base de la economía moderna: englobando tanto las pequeñas salas de servidores, las infraestrucuras DC de tamaño medio de las que disponen algunas coroporaciones como las megainfraestrucuras de servicios cloud pertenecientes Amazon, Facebook, Google y otros. No obstante, la explosión de contenido digital (Big Data), e-commerce y el tráfico de Internet han convertido los DC en el mayor consumidor creciente de electricidad en los países desarrollados, y esta es una de las claves en la construcción de los DC del futuro.

En 2013, los DC de U.S (Estados Unidos) consumieron alrededor de 91 billones de Kw/h de electricidad, equivalente a la electricidad que proporcionan las 34 plantas de electricidad mayores del país. Este consumo de electricidad está previsto que crezca aproximadamente hasta los 140 billones de Kw/h anuales en 2020, equivalente a la que proporcionan actualmente las 50 mayores plantas de electricidad, costando 13 billones de dólares anuales y emitiendo cerca de 100 millones de toneladas métricas de CO2 [22].

Year	End-use Energy (B kWh)	Elec. Bills (US, $B)	Power plants (500 MW)	CO2 (US) (million MT)
2013	91	$9.0	34	97
2020	139	$13.7	51	147
2013-2020 increase	47	$4.7	17	50

Tabla3. Previsión de crecimientos del consume électricos de DC en U.S (Fuente:NRDC)

En este contexto, diversas soluciones están siendo investigadas, como son el uso de energías renovables, nuevos sistemas de refigeración, etc. No obstante, una mayor eficiencias de los recursos de las infraestrucuras conllevan de manera directa grandes beneficios tanto en término económicos como en impacto mediambiental.

Por lo tanto, situándonos en el contexto anterior donde se realiza la previsión del consumo de DC en U.S, la implatanción de DC desagregados implicaría un ahorro de decenas de millones de toneladas métricas de CO2 en comparación de esas mismas infraestructuras pero con arquitecturas basadas en servidores.

7. Conclusiones e investigaciones futuras

Las instancias de Data Center Virtuales (VDC) requieren una asignación de recursos computacionales y de red. No obstante, las arquitecturas DC tradicionales basadas en servidores requieren el sobreaprovisionamiento de unidades de servidores para satisfacer el mapeo de las demandas VDC, lo que implica un aumento del CAPEX asociado. En este estudio hemos mostrado como las arquitecturas de DC desagregado pueden ayudar a la reducción de recursos computacionales necesarios para albergar instancias VDC en la infraestructura física. Alrededor de un 46% de reducciones promedio pueden ser experimentadas en perfiles de VM (Máquinas Virtuales) balanceadas, mientras que estás reducciones pueden llegar hasta al 60% en el caso de VMs especializadas. Estos resultados nos indican que los DCs desagregados pueden sobreponerse a las limitaciones de las arquitecturas actuales en términos de utilizar eficientemente los recursos computacionales, esperando su adopción en futuras arquitecturas DC.

Además hemos podido observer como una DCN enmallada permite al operador gastar menos al equipar la red con diferentes canales de longitudes de onda, lo que implica menos transpondedores WDM. Sin embargo, este tipo de redes requieren un mayor número de nodos y links físicos, La optimización del coste de la red, teniendo en cuenta de la topología óptima en número de nodos y links queda fuera del alcance de este estudio y se deja para futuros trabajos.

En todo nuestro estudio, al plantear la infraestructura de DC desagregado hemos asumido que los racks contienen blades hardware de todo tipo de recursos (Núcleos CPU, memoria y almacenamiento). Pero tal y como hemos planteado en la introducción también existiría la posibilidad de que los racks sean monotemáticos, es decir cada rack contuviera un único tipo de recursos computacional (p.e memoria), en ese caso el mapeo de una VM implicaría varios racks, lo que podría ser una limitante en términos de latencia y ancho de banda. No obstante, estudios futuros podrian analizar en cómo superar estas limitaciones y las ventajas que se obtendrían en comparación a la arquitectura desagregada en que los racks disponen de todo tipo de recursos computacionales.

Bibliografia:

[1]. T. Wang, Z. Su, Y. Xia and M. Hamdi, “Rethinking the Data Center Networking: Architecture, Network Protocols, and Resource Sharing”, IEEE Access, vol. 2, pp. 1481-1496, December 2014.

[2]. Cisco, “Cisco Global Cloud Index: Forecast and Methodology, 2014–2019”, [Available online]: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html, accessed May 2016.

[3]. C. Kachris and I. Tomkos, “A Survey on Optical Interconnects for Data Centers”, IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1021-1036, Fourth Quarter 2012.

[4]. N. Farrington et al., “Helios: a hybrid electrical/optical switch architecture for modular data centers”, Proceedings of ACM SIGCOMM 2010, 2010, pp. 339–350.

[5]. J. Perelló et al., “All-Optical Packet/Circuit Switching-based Data Center Network for Enhanced Scalability, Latency and Throughput”, IEEE Network, vol. 27, no. 6, pp. 14-22, December 2013.

[6]. C. Reiss et al., “Google cluster-usage traces: format + schema”, Google Technical Report, 2012, [Available Online] http://code.google.com/p/googleclusterdata/wiki/TraceVersion2, accessed May 2016.

[7]. M. Byrne, “Memory Is Holding Up the Moore's Law Progression of Processing Power”, 2014, [Available Online]http://motherboard.vice.com/read/memory-is-holding-up-the-moores-law-progression-of-processing-power, Accessed May 2016.

[8]. Open Compute Project, [Available online] http://www.opencompute.org/, Accessed May 2016.

[9]. Y. Yan et al., “All-Optical Programmable Disaggregated Data Centre Network Realized by FPGA-Based Switch and Interface Card”, IEEE/OSA Journal of Lightwave Technology, vol. 34, no. 8, pp. 1925-1932, April 2016.

[10]. S. Han et al., “Network Support for Resource Disaggregation in Next-Generation Datacenters”, Proceedings of 12th ACM Workshop on Hot Topics in Networks (HotNets 2013), November 2013.

[11]. J. Weiss et al., “Optical interconnects for disaggregated resources in future datacenters”, Proceedings of 40th European Conference and Exhibition on Optical Communications (ECOC 2014), September 2014.

[12]. J. Kyathsandra et al., “Rack Scale Architecture: Designing the Data Center of the Future”, IBM Technical report.

[13]. K.-K. Nguyen, M. Cheriet and M. Lemay, “Enabling infrastructure as a service (IaaS) on IP networks: from distributed to virtualized control plane”, IEEE Communications Magazine, vol. 51, no. 1, pp. 136-144, January 2013.

[14]. Interoute VDC service, [Available Online] https://cloudstore.interoute.com/, Accessed may 2016.

[15]. A. Pagès et al., “Optimal Virtual Slice Composition Toward Multi-Tenancy Over Hybrid OCS/OPS Data Center Networks”, IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 10, pp. 974-986, October 2015.

[16]. M. G. Rabbani et al., “On tackling virtual data center embedding problem”, 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 177-184, May 2013.

[17]. Amazon Elastic Compute Cloud service, [Available Online] https://aws.amazon.com/ec2/,Accessed May 2016.

[18]. IBM CPLEX Optimizer, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[19]. J. Y.Yen.“Finding the k shortest loopless paths in a network”, Management Sciences, 17:712-716, 1971.

[20]. Pham D.T and Karaboga D. “Inteligent Optmisation Techniques”, pp51-218, Springer (London),2000.

[21]. Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs" . Numerische Mathematik 1: 269–271

[22]. Pierre Delforge, “Critical Action Needed to Save Money and Cut Pollution”, 2015, [Available Online] https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts-energy.

[23]. Amazon Elastic Compute Cloud service, [Available Online] https://aws.amazon.com/ec2/, Accessed May 2016.

Anexos:

Anexo 1.Bloques de trabajo, tareas e hitos

TAREA PRINCIPAL		**Ref.:** 1	
Nombre: FORMACIÓN			
Fecha de inicio:	Semana 0	**Fecha de finalización:**	Semana 8
Descripción de la Tarea: Durante esta tarea se obtendrán los conocimientos necesarios previos para poder desarrollar el proyecto con garantías. Se parte de los conocimientos base aprendidos de comunicaciones ópticas pero se hace hincapié en las partes que tendrán un peso específico en el proyecto. Para realizar esta tarea se hará uso de diferente tipo de material didáctico, como pueden ser documentos científicos o material audiovisual. En este apartado también se tiene en cuenta todos los conceptos que son necesarios sobre programación en java y como poderlos combinar en nuestro simulador de DC y DDC, como por ejemplo la programación de los algoritmos de enrutamiento.			
Participante: Rubén Serrano Moreno		**Responsables:** Jordi Perelló Muntan, Albert Pagès Cruz	
Tareas Internas:			
Nombre	REDES ÓPTICAS		
Descripción	Profundizar en el funcionamiento de redes ópticas, tipos de enrutamiento, restricciones, características, dispositivos implicados, concepto de redes virtuales, técnicas de mapeo…		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	1.1	Semana 0	Semana 1
Nombre	DC y DDC		
Descripción	Entender el funcionamiento de un DC tradicional, los elementos que lo componen, sus interconexiones y el servicio que ofrecen. Entender las implicaciones que supondría quitar el concepto de server y por lo tanto pasar a una DC desagregado. También se incluyen el estudio de los algoritmos de enrutamiento para establecer conexiones entre los diferentes nodos.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	1.2	Semana 0	Semana 2
Nombre	OPTIMIZACIÓN		

Descripción	Saber analizar un problema para entender sus restricciones y su función a optimizar y aprender a plasmar estos conceptos de manera matemática mediante diferentes ecuaciones con sumatorios.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	1.3	Semana 2	Semana 3
Nombre	TÉCNICAS HEURÍSTICAS		
Descripción	Estudiar algoritmos de metaheurísitca general y analizar técnicas que pretenden simular la solución óptima de problemas típicos de optimización como sería el *bin packing.* También se estudia modelos de ejemplo en que se resuelven problemas de optimización con estrategias particulares a fin de proporcionar ideas o conceptos que puedan ser aplicables en nuestro escenario		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	1.4	Semana 2	Semana 6
Nombre	PROGRAMACIÓN JAVA		
Descripción	Repaso tanto de conceptos básicos de programación en java, como de conceptos más avanzados para resolver los obstáculos que surgen durante la realización del simulador, como podría ser el uso del api CPLEX o la implementación del algoritmo Dijkstra.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	1.5	Semana 6	Semana 8

<table>
<tr><th colspan="2">TAREA PRINCIPAL</th><th colspan="2">Ref.: 2</th></tr>
<tr><td colspan="4">Nombre: ESTUDIOS TÉORICOS</td></tr>
<tr><td>Fecha de inicio:</td><td>Semana 4</td><td>Fecha de finalización:</td><td>Semana 6</td></tr>
<tr><td colspan="4">Descripción de la Tarea:
Esta tarea engloba el análisis y la resolución del problema, en concreto, del mapeo de VDC sobre los DC y el planteamiento del caso DC desagregado. Se tiene que proponer el modelo matemático que resuelve el problema de manera óptima y las estrategias que se podrían llevar a cabo para intentar simular ese resultado: el orden con que se reciben las demandas, la asignación de longitudes de ondas, el mapeo de los nodos…</td></tr>
<tr><td colspan="2">Participante: Rubén Serrano Moreno</td><td colspan="2">Responsables: Jordi Perelló Muntan, Albert Pagès Cruz</td></tr>
<tr><td colspan="4">Tareas Internas:</td></tr>
<tr><td>Nombre</td><td colspan="3">ILP</td></tr>
<tr><td>Descripción</td><td colspan="3">Formulación del modelo matemático que resuelve de manera óptima el mapeo de VDC sobre DC tradicionales y desagregados.</td></tr>
<tr><td>TI responsable</td><td>TI Ref:</td><td>Fecha de inicio:</td><td>Fecha de finalización:</td></tr>
<tr><td>Rubén Serrano</td><td>2.1</td><td>Semana 4</td><td>Semana 5</td></tr>
<tr><td>Nombre</td><td colspan="3">ESTRATEGIAS DE OPTIMIZACIÓN</td></tr>
<tr><td>Descripción</td><td colspan="3">Análisis de todas las estrategias posibles sobre los elementos del problema que permiten variaciones para poder intentar alcanzar el resultado óptimo.</td></tr>
<tr><td>TI responsable</td><td>TI Ref:</td><td>Fecha de inicio:</td><td>Fecha de finalización:</td></tr>
<tr><td>Rubén Serrano</td><td>2.2</td><td>Semana 4</td><td>Semana 6</td></tr>
</table>

TAREA PRINCIPAL		**Ref.:** 3	
Nombre: IMPLEMENTACIÓN			
Fecha de inicio:	Semana 6	**Fecha de finalización:**	Semana 13
Descripción de la Tarea: En esta tarea se programará en java la estructura del DC tradicional y desagregado. También será necesario implementar un generador aleatorio de demandas. A partir de estos elementos ya tendremos la base para obtener los resultados óptimos mediante la API JAVA de CPLEX una vez programado nuestro modelo matemático. El simulador también debe incluir las diferentes estrategias de mapeo que habremos propuesto para aproximarnos al resultado óptimo (las técnicas heurísticas).			
Participante: Rubén Serrano Moreno		**Responsables:** Jordi Perelló Muntan, Albert Pagès Cruz	
Tareas Internas:			
Nombre	SIMULADOR DC Y DDC		
Descripción	Creación de la estructura de un DC y un DDC a partir de la información contenida en un fichero .TXT.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	3.1	Semana 6	Semana 9
Nombre	GENERADOR DE DEMANDAS		
Descripción	Genera un fichero .TXT con un set de demandas (VDCs) aleatorio o determinista, en función del perfil de demandas que se quieran obtener.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	3.2	Semana 7	Semana 9
Nombre	OPTIMIZADOR		
Descripción	Entender las funcionalidades de CPLEX para poder programar el modelo matemático en java.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	3.3	Semana 9	Semana 10
Nombre	PROGRAMACIÓN DE ESTRATEGIAS		
Descripción	Programar las diferentes estrategias que pretendan aproximar-se al modelo matemático óptimo.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	3.4	Semana 10	Semana 13

TAREA PRINCIPAL		**Ref.:** 4	
Nombre: EXTRACCIÓN DE RESULTADOS			
Fecha de inicio:	Semana 9	**Fecha de finalización:**	Semana 16
Descripción de la Tarea: Analizar el comportamiento de las estrategias con diferente número de demandas aleatorias para llegar a la conclusión de cuál es la que se asemeja más a la óptima. Una vez comprobada cuál es la estrategia válida, se realizan diferentes simulaciones, con diferentes estilos de demandas y de topologías de DC, para poder constatar las ventajas que tendría un DC desagregado respecto a uno tradicional. Para poder realizar un número tan grande de simulaciones y obtener sus resultados es necesario automatizar el proceso mediante scripts.			
Participante: Rubén Serrano Moreno		**Responsables:** Jordi Perelló Muntan, Albert Pagès Cruz	
Tareas Internas:			
Nombre	COMPARATIVA		
Descripción	Obtención de los datos y comparativa tanto Heurística vs Óptima, como de DC vs DC desagregados.		
TI responsable	**TI Ref:**	**Fecha de inicio:**	**Fecha de finalización:**
Rubén Serrano	4.1	Semana 9	Semana 16

TAREA PRINCIPAL		**Ref.:** 5	
Nombre: DOCUMENTACIÓN			
Fecha de inicio:	Semana 3	**Fecha de finalización:**	Semana 16
Descripción de la Tarea: Realizar todos documentos necesarios para el correcto seguimiento del proyecto y para formalizar el trabajo realizado y los resultados obtenidos. La documentación del proyecto se basará en tres documentos: - La propuesta del trabajo y su planificación inicial. - La revisión crítica. - El documento final. De manera adicional, en este apartado también se incluye la adaptación de la documentación al formato que sea requerido para poder publicar un artículo en alguna revista o conferencia científica.			
Participante: Rubén Serrano Moreno		**Responsables:** Jordi Perelló Muntan, Albert Pagès Cruz	

Hitos

WP	Tarea	Título	Hito/ Entregable	Semana
5	-	Propuesta del proyecto	PropuestaDeProyecto.doc	3
2	2.1	Modelo matemático de optimización	Escrito manual de la resolución optimizada del problema.	5
3	-	Simulador de la estructura de los data centers y de las estrategias de optimización.	DC.jar y DDC.jar	12
5	-	Revisión del avance del proyecto	CriticalReview.doc	9
4	4.1	Obtención de los resultados	HeuVsOpt.xls Nodes.xls Demandes.xls	15
5	-	Proyecto final	TFG.doc	16

Glosario

CAPEX: CAPital EXpenditures
CDC: Colorless, Directionless y Contentionles.
DC: Data Center.
DCN: Data Center Network.
DWDM: Dense Wavelength Division Multiplexing.
IaaS: Infraestructure as a Service.
ILP: Integer Linear Programming
K-SP-FF: K-Shortest Path-First fit
OXC: Optical Cross Connects.
QoS: Quality of Service.
SIC: Switch and Interface Card.
ToR: Top of the rack.
VDC: Virtual Data Center.
VM: Virtual machine.