Capítulo V - Diseño de una arquitectura heterogénea para la gestión eficiente de recursos FPGA en un cloud privado

Julio Proaño Orellana
María Blanca Caminero
Carmen Carrión

SciELO Books / SciELO Livros / SciELO Libros

PROAÑO ORELLANA, J., BLANCA CAMINERO, M., and CARRIÓN, C. Diseño de una arquitectura heterogénea para la gestión eficiente de recursos FPGA en un cloud privado. In: INGA ORTEGA, E., ed. *Aplicaciones e innovación de la ingeniería en ciencia y tecnología* [online]. Quito: Editorial Abya-Yala, 2019, pp. 165-199. ISBN: 978-9978-10-491-0.
<https://doi.org/10.7476/9789978104910.0007>.

All the contents of this work, except where otherwise noted, is licensed under a [Creative Commons Attribution 4.0 International license](#).

Todo o conteúdo deste trabalho, exceto quando houver ressalva, é publicado sob a licença [Creative Commons Atribuição 4.0](#).

Todo el contenido de esta obra, excepto donde se indique lo contrario, está bajo licencia de la licencia [Creative Commons Reconocimiento 4.0](#).

CAPÍTULO V

Diseño de una arquitectura heterogénea para la gestión eficiente de recursos FPGA en un cloud privado

Desing of a heterogeneous architecture for the efficient management of FPGAs in a Cloud

Julio Proaño Orellana*, María Blanca Caminero†, Carmen Carrión§

*Universidad Politécnica Salesiana, †Universidad de Castilla La Mancha

*email: jproanoo@ups.edu.ec

†email: mariablanca.caminero@uclm.es

§email: carmen.carrion@uclm.es

Resumen

El uso eficiente de recursos en entornos de computación en la Nube es uno entre los muchos temas que han atraído a muchos científicos en los últimos años. Hoy en día existe una amplia variedad de recursos informáticos disponibles en las infraestructuras de Cloud Computing, como CPUs multi-core, GPUs y FPGAs. En particular, arquitecturas basadas en aceleradores que usan FPGAs pueden proporcionar un importante valor agregado de negocio en entornos Cloud. Los dispositivos FPGA ofrecen un mejor desempeño debido a su gran capacidad de procesamiento con latencia predecible y bajo el consumo de energía. Sin embargo, la gestión de estos recursos en un entorno Cloud no es trivial, y es el foco de este trabajo. En este trabajo, se presenta una arquitectura destinada a integrar FPGAs como aceleradores en un Cloud privado. La propuesta toma en cuenta requisitos de calidad de servicio (QoS). Así, los indicadores de QoS están relacionados con los requerimientos de los usuarios. La arquitectura propuesta busca la combinación adecuada entre CPUs y FPGAs en un entorno Cloud.

Palabras clave

Cloud Computing, energía, FPGA

Abstract

The efficient use of resources in cloud computing environments is one of the many topics that have attracted many scientists in recent years. Today there is a wide variety of computing resources available in Cloud Computing infrastructures, such as multi-core CPUs, GPUs and FPGAs. In particular, accelerator-based architectures using FPGAs can provide

significant business value in Cloud environments. FPGA devices offer better performance due to their high processing capacity with predictable latency and low power consumption. However, the management of these resources in a Cloud environment is not trivial. In this work, an architecture designed to integrate FPGAs as accelerators in a private Cloud is presented. The proposal takes into account quality of service (QoS) requirements. Thus, the QoS indicators are related to the requirements of the users. The proposed architecture seeks the right combination between CPUs and FPGAs in a Cloud environment.

Keywords

Cloud Computing, energy, FPGAs

1. Introducción

En la actualidad el paradigma de Cloud Computing es un tema de gran interés tanto para la academia como para la industria. La industria de Cloud representada por los proveedores de servicios invierte grandes cantidades de dinero en nuevas infraestructuras con el objetivo de satisfacer la creciente demanda de recursos computacionales. Sin embargo, problemas como por ejemplo: la gestión eficiente de recursos, la seguridad, la confidencialidad de la información y el excesivo consumo de energía son cuestiones que aún no se han resuelto del todo. Así, la academia es la que se enfoca en resolver estos problemas a través de la investigación científica. [1]

Los entornos Cloud están conformados por dos elementos fundamentales, proveedores y clientes. Los proveedores, siendo los propietarios de los recursos, buscan maximizar sus ganancias ofreciendo recursos computacionales como un servicio a bajo coste. Por otro lado, los clientes buscan ejecutar sus aplicaciones sin tener que incurrir en gastos de manteniendo de infraestructura; dicho en otras palabras ellos únicamente buscan pagar por el uso de los recursos.

La interacción entre estos dos actores sigue un protocolo simple. Los clientes envían peticiones a los proveedores para utilizar recursos computacionales, los proveedores por su parte luego de procesar dichas peticiones seleccionan y encapsulan los recursos solicitados en una máquina virtual. Finalmente, el cliente recibe el acceso a esta máquina como respuesta a su requerimiento. [2]

Los proveedores de servicios Cloud son los responsables de asegurar una calidad de servicio (QoS) en la cual se consideran paráme-

tros como: la disponibilidad del servicio, el rendimiento, la latencia, etc. [3] Con el objetivo de asegurar a los clientes que estos requerimientos de calidad se cumplan, clientes y proveedores negocian un acuerdo denominado en inglés Service Level Agreement (SLA). [4]

Clientes y proveedores persiguen diferentes objetivos, por un lado, los clientes buscan utilizar recursos computacionales para sus aplicaciones con el mínimo costo, mientras los proveedores buscan maximizar sus ingresos manteniendo una calidad de servicio adecuada y disminuyendo sus gastos. Dicho de otra manera, los proveedores buscan ser eficientes en la administración de recursos y el uso de la energía. Sin embargo, según estudios publicados por Internet World Stats las regiones como África, Latinoamérica y el Caribe han mostrado un crecimiento del 7415% y 2029% en los años comprendidos entre 2000 y 2016. Como consecuencia, también el coste de consumo de energía y disipación de calor se han incrementado de manera exponencial. Un ejemplo claro se muestra en [5] donde los autores sostienen que el 42% del presupuesto total de los servidores de Amazon EC2 se corresponde con la energía tanto en generación como en disipación. De hecho, de acuerdo a un estudio realizado por McKinsey & Company [6] en promedio el 80% de la energía utilizada por los grandes centros de datos es únicamente para mantener a estos en estado de reposo y el 20% restante es usado para operaciones de cómputo.

Según un informe realizado por el departamento de Energía Lawrence Berkley National Laboratory [7] en el 2016, una estrategia como el uso de hardware más eficiente podría disminuir el consumo de energía en un 45% para el 2020. Este reporte menciona que el uso de infraestructura más eficiente en almacenamiento, red y procesamiento es fundamental para disminuir el excesivo consumo de energía de los grandes centros de datos.

Desde la industria, los principales proveedores de Cloud como Google, Amazon han optado como estrategia la utilización de infraestructuras más eficientes como el uso de dispositivos FPGA [8]. Como resultado han logrado reducir el consumo de energía. Además, han trasladado sus infraestructuras a lugares con bajas temperaturas, para disminuir el consumo de energía que se utiliza para enfriar sus granjas de servidores.

Como se explicó anteriormente es muy importante establecer una arquitectura que permita por un lado satisfacer la creciente demanda de recursos pero al mismo tiempo contralar el consumo de energía.

Este trabajo propone una arquitectura que aborda la integración de hardware eficiente como FPGAs dentro de un paradigma de Cloud Computing. La propuesta se enfoca en la gestión de recursos heterogéneos que incluyen el uso de FPGAs como aceleradores de código en la ejecución de tareas en un entorno de Cloud privado.

Las principales contribuciones de este artículo son: una revisión de la literatura enfocada en la integración y gestión de FPGAs dentro de entornos Cloud y el desarrollo teórico de una arquitectura que tenga como objetivo la integración de FPGAs en un Cloud privado. Finalmente, una evaluación como prueba de concepto de la propuesta.

El artículo se encuentra organizado de la siguiente manera, en la Sección 2 se muestra una revisión de los conceptos teóricos que soportan la propuesta. En la Sección 3, se exponen los principales trabajos relacionados con nuestra propuesta. En la sección 4, se dan detalles sobre la implementación y funcionamiento de la propuesta. En la sección 5 se muestra una evaluación como prueba de concepto de la arquitectura. Finalmente, en el capítulo 6 se discuten algunas conclusiones de este trabajo.

2. Marco teórico

A. *Arquitectura Cloud*

Desde el punto de vista de la computación distribuida, el paradigma de Cloud sigue una arquitectura por capas. Las tres principales capas son: la capa de plataforma de gestión, la capa de virtualización y la capa de hardware como se puede observar en la Figura 1. [9]

La capa de hardware está compuesta por un grupo de recursos tales como: servidores, dispositivos de red, fuentes de energía, sistemas de ventilación y disipación de calor. Además, cabe destacar que cada servidor se encuentra compuesto, por lo menos, por procesadores, memoria, discos de almacenamiento y tarjetas de red. También se pueden encontrar dispositivos especiales como por ejemplo co-procesadores, procesadores gráficos (GPUs), FPGAs, etc.

La capa de virtualización usa máquinas virtuales para partir los recursos computacionales disponibles y así facilitar su compartición y gestión. La virtualización permite crear un entorno virtual de manera segura y aislada con el objetivo de ejecutar una aplicación o un grupo de aplicaciones determinadas.

Desde el punto de vista de Cloud, los proveedores de servicio necesitan encontrar mecanismos para realizar una gestión eficiente y segura de los recursos computacionales. Y además, asegurar que dicho mecanismo sea capaz de escalar.

Los proveedores utilizan la virtualización para entregar servicios y/o desplegar servicios a los clientes de manera rápida y segura. A pesar de que el término de virtualización sea comúnmente relacionado con la emulación de un hardware, esta tecnología es muy útil para la compartición y gestión de recursos en un centro de datos. El mecanismo de gestión, a nivel macro se lo realiza a través de un software conocido como Gestor de Virtualización en inglés Virtual Machine Manager VMM. Existen diferentes tipos de VMMs, los más usados en entornos Unix son: KVM (Kernel-based Virtual Machine) y XEN, mientras que para los sistemas que se ejecutan en Windows se tiene Hyper-V y VMware vSphere. [10]

A pesar de las ventajas para la gestión, aislamiento y seguridad que ofrece la virtualización, esta tecnología presenta un sobre coste en el rendimiento del sistema debido a que el VMM debe multiplexar los procesos del sistema operativo local y los procesos del sistema operativo simulado. Sin embargo, existen técnicas como por ejemplo la paravirtualización que ayuda a reducir este sobre coste de rendimiento.

Finalmente, grandes empresas tecnológicas como Intel y AMD han desarrollado extensiones para mejorar el rendimiento de entornos virtuales. En particular, Intel VT-x [11] y AMD SVM [12]. Estos mecanismos permiten acelerar los entornos virtuales a través de funciones especiales y la gestión de memoria.

La última capa denominada plataforma opera a nivel de nodo de un servidor y se encarga de gestionar las distintas máquinas virtuales desplegadas en diferentes nodos de un centro de datos.

Esta capa está compuesta por un software que tiene la capacidad de conocer todos los recursos disponibles de varios nodos en un entorno Cloud. Además, esta capa es capaz de desplegar, moni-

torizar y gestionar una gran cantidad de máquinas virtuales desplegadas en la capa de virtualización. Dicho de otro modo, esta capa es la responsable de dirigir de manera efectiva la creación, monitorización, gestión, despliegue de toda la infraestructura virtual en un entorno de Cloud.

En la actualidad tanto en la academia como en la industria los VMMs más usados en entornos de Cloud son: Open Stack [13], Eucalyptus, VMware y Open Nébula [14]. Cada uno de estos contiene una serie de características que los investigadores aprovechan en base a sus necesidades.

En el siguiente apartado se detallan, aspectos sobre el paradigma de Cloud Computing basado en tecnologías de virtualización.

Figura 1. Arquitectura de Cloud Computing

B. Definición y modelos de despliegue en Cloud

De acuerdo con el Instituto Nacional de Estándares y Tecnología en inglés Nation Institute of Standars and Technology (NIST) [15], Cloud Computing es “un modelo ubicuo bajo demanda que comparte recursos computacionales (red, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y liberados con una gestión mínima por parte de los proveedores.”

Según la definición anterior Cloud Computing presenta una serie de caracterizas tales como:

- **Bajo-demanda.**- los proveedores de servicios Cloud permiten auto-aprovisionamiento de recursos de manera autónoma.

- **Acceso ubicuo.**- el acceso a los servicios de Cloud es a través de Internet.
- **Multi-usuario.**- los recursos en Cloud son compartidos por un sin número de usuarios de manera transparente y eficiente.
- **Elástico y escalable.**- Cloud tiene la capacidad de adicionar y liberar recursos de manera dinámica de acuerdo a los requerimientos de los clientes.
- **Monitorización de uso de recursos.**- para controlar, optimizar y con fines de facturación, las arquitecturas Cloud ofrecen herramientas para la monitorización del uso de recursos.

Por otro lado, el paradigma de Cloud ofrece tres tipos modelo de entrega de servicios que son: Software-como-un-Servicio (SaaS), Plataforma-como-un-Servicio (PaaS) e Infraestructura-como-un-Servicio (IaaS).

Figura 2. Infraestructura como servicio (IaaS) [3]

Figura 3 Software como servicio (SaaS) [3]

Figura 4. Plataforma como servicio (PaaS) [3]

En el modelo de *IaaS* como se muestra en la Figura 2, los proveedores ofrecen a sus clientes recursos computacionales (procesamiento, servicios de red, almacenamiento) encapsulados en una máquina virtual, por ejemplo: Amazon Web Service, Microsoft Azure, Google Cloud, Rackspace, etc.

En el caso de *SaaS* (ver Figura 3), los proveedores ofrecen a sus clientes una lista de aplicaciones que están en el Cloud, algunos ejemplos son: Twitter, Facebook, Flickr, Google Docs.

Finalmente en la Figura 4, se muestra el modelo *PaaS*. En este modelo, se ofrecen plataformas para el desarrollo de aplicaciones, como por ejemplo: Amazon Web Service (AWS), Windows Azure, Google App Engine.

Finalmente, todos estos servicios de Cloud mencionados anteriormente pueden ser desplegados en entornos *Privados*, *Públicos*, *Comunitarios* e *Híbridos*. A continuación se da un breve resumen de los distintos modelos de despliegue de Cloud.

La Figura 5 muestra el despliegue en un entorno privado, cuya principal característica es que los recursos solamente pueden ser accedidos por los miembros de la misma organización.

Figura 5. Ejemplo de un Cloud Privado [3]

Por otro lado, la Figura 6 muestra la arquitectura para un modelo de Cloud *Pública* en el cual los recursos computacionales se ofrecen al público en general a través de Internet.

Figura 6. Ejemplo de Cloud Pública [3]

Finalmente, la Figura 7 muestra un modelo que combina dos o más de los modelos mencionados anteriormente.

Figura 7. Ejemplo de un Cloud Híbrido [3]

C. Calidad de Servicio (QoS) y asignación de recursos en Cloud

En la sección anterior se mencionaron la definición de un Cloud y los modelos servicio y despliegue. En esta sección nos referiremos a conceptos fundamentales de calidad de servicio y las diferentes técnicas usadas para la gestión de recursos en Cloud.

Los entornos Cloud ofrecen a sus clientes una gran cantidad de recursos disponibles bajo demanda siguiendo un modelo pay-as-you-go, pago por lo que uso [16]. Desde el lado del proveedor, los

servicios tienen cumplir con una cierta calidad. Estos parámetros de calidad tales como: rendimiento, latencia, disponibilidad son incluidos dentro del Acuerdo de Nivel de Servicio (SLA) [17].

La calidad del servicio en Cloud está estrechamente relacionada a la gestión de recursos del sistema. Una gestión eficiente permite a los proveedores aprovechar al máximo sus infraestructuras. Sin embargo, debido a la complejidad que presenta un Cloud, esta tarea no es trivial. Uno de los problemas más abordados es la asignación eficiente de recursos.

La asignación de recursos es una tarea que debe ser realizada de manera autónoma y bajo demanda. Por lo general se definen distintas políticas para llevar esta tarea. Las políticas más usadas para la asignación de recursos en Cloud incluyen, el control de admisión, la eficiencia energética, los modelos estadísticos, el balanceo de carga, criterios de SLA y el uso de dispositivos dedicados como FPGAs. [18]

La política de control de admisión es preventiva y su objetivo es evitar la sobrecarga de recursos y violaciones del SLA. Esta política se basa en la relación entre el conocimiento previo acerca del comportamiento del sistema y la carga [19]. Esta política puede resultar muy compleja para entornos muy dinámicos y puede causar una sub-utilización del sistema. Sin embargo, trabajos [20], demuestran que la combinación de estas políticas con criterios sobre aprovisionamiento (Over-booking) suelen dar un mejor rendimiento.

Por otro lado, las políticas basadas en eficiencia energética y modelos estadísticos buscan un balance entre la asignación de recursos, mantener una tasa aceptable de SLAs cumplidos y disminuir el consumo de energía. Proyectos como TANGO [21], se enfocan en la implementación de algoritmos de eficiencia energética orientado a computación de alto rendimiento. Trabajos como [5], [22], [23], usan algoritmos que consideran el consumo de energía.

Las políticas que consideran el balanceo de carga tienen como objetivo distribuir la carga a través de los recursos disponibles del sistema. Estas políticas integran criterios de migración de tareas a diferentes recursos dependiendo de la carga. Por ejemplo, si el sistema detecta que un nodo se encuentra subutilizado y otro sobreutilizado, el sistema se encarga de migrar las tareas desde el nodo sobreutilizado al nodo subutilizado, de tal manera que se consiga un

equilibrio. Sin embargo, la migración de tareas podría provocar una saturación de la red. [18]

Las políticas que toman en cuenta los SLAs [19] se enfocan en mapear el requerimiento de cada uno de los clientes y los recursos disponibles que pueden satisfacer el SLA. Por lo general se basan en algoritmos que realizan predicciones y toma de decisiones relacionados con la Inteligencia Artificial.

Finalmente, las políticas relacionadas con la utilización de dispositivos más eficientes son nuevas en el ámbito de Cloud. En este tipo de mecanismo las peticiones de los clientes son clasificadas en diferentes categorías dependiendo de los requerimientos para después ser asignadas a los recursos que mejor se adapten. Además, estos recursos poseen características muy favorables con respecto al uso de la energía y rendimiento. Sin embargo, es una tarea muy compleja encontrar los mejores recursos para todas las tareas por lo cual se puede tener algunos recursos subutilizados.

D. Técnicas de Planificación de recursos en Cloud

Desde el punto de vista de la industria las técnicas de planificación de recursos son utilizadas para maximizar los ingresos de los proveedores y reducir el tiempo de retorno de la inversión (ROI). El mecanismo más utilizado para conseguir este objetivo es compartir de manera eficiente los recursos computacionales disponibles. Sin embargo, esta tarea no es trivial debido a la gran variabilidad que presentan los entornos de Cloud. Según [24], la clave para conseguir el objetivo antes mencionado está en diseñar un sistema de planificación que sea eficiente, lo suficientemente justo e inteligente con la mínima interacción humana.

De manera general algunos mecanismos para la gestión se basan en sistemas de colas con prioridades para determinar ¿Qué recurso? va a ¿Dónde? y ¿Cuándo? Esta estrategia básicamente elabora listas dinámicas y ordena sus elementos según un criterio de prioridad.

Las técnicas de planificación de recursos se pueden implementar siguiendo una arquitectura por capas. Así, cada capa es responsable únicamente de una función específica. Por ejemplo: en un esquema de planificación con tres capas: Aplicación, Virtualización y

Despliegue. La primera capa es la responsable de clasificar las peticiones de los clientes de acuerdo a una determinada política como QoS. La segunda capa es responsable, por su parte, por su parte de gestionar los recursos virtualizados haciendo un mapeo entre los recursos físicos y los virtuales. Finalmente, la última capa es responsable de repartir las tareas a diferentes nodos del Cloud.

La planificación de recursos puede ser vista como un problema de optimización multiobjetivo. Por un lado minimizar el coste y maximizar la utilización de los recursos en el tiempo.

En [25], los autores clasifican las estrategias de planificación de recursos en aquellas que consideran el rendimiento (performance) y en las que consideran el coste. Las estrategias que consideran el rendimiento se dividen en First-come First-serve (primero en llegar es el primero en ser atendido), balanceadores de carga. Por otro las estrategias que toman en cuenta el coste se dividen en máximo rendimiento, mínimo coste de operación y la combinación de las dos.

A continuación se hace una revisión de trabajos relacionados con la integración, gestión y planificación de recursos como FPGAs dentro de entornos Cloud.

3. Trabajos relacionados

A. Recursos heterogéneos en Cloud

Existen diferentes tipos de clientes que usan los recursos de Cloud como por ejemplo, científicos, desarrolladores, investigadores etc. Todos ellos, necesitan un conjunto de recursos específicos creando una demanda muy variada. Esta diversidad de clientes demanda también una amplia gama de servicios también muy diversa. Así, los proveedores deben adaptarse a través de la adquisición de más y diversa infraestructura. Sin embargo, el hardware está limitado a factores como la densidad de potencia [W/mm^2] y la complejidad. La diversidad obliga a los proveedores en integrar arquitecturas heterogéneas. La integración de recursos heterogéneos puede ayudar a mejorar la eficiencia en el uso de los recursos a través de un mapeo de las necesidades de los clientes con los recursos más adecuados que contienen características específicas muy favorables para satisfacer un

SLA. Estas características pueden ser, por ejemplo, un mejor desempeño en el procesamiento gráfico mediante el uso de GPUs, procesadores usados como aceleradores de código incluso hardware dedicado para ciertas tareas específicas mediante el uso de FPGAs.

B. FPGAs en Cloud

Los dispositivos de hardware FPGA usados como coprocesadores ofrecen una mejora significativa para muchas aplicaciones [26]. Así, existen algunos esfuerzos para integrar FPGAs dentro de ámbito tradicional de las aplicaciones informáticas. Algunas soluciones han sido proporcionadas por la industria, tal como PicoComputing, Convey y Xillybus [27]. Estos productos conectan aplicaciones informáticas con la FPGA a través de una interfaz propietaria desarrollada bajo sus protocolos propios. Mitrion-C, Open Bio Project [28] intentan acelerar aplicaciones bioinformáticas portando secciones críticas de sus aplicaciones hacia FPGAs.

Por otro lado, arquitecturas de código abierto como RIFFA [29] proporcionan una plataforma muy práctica para usar dispositivos FPGA de Xilinx como aceleradores de código. Sin embargo, estos ejemplos no disponen de una arquitectura para desarrollar aplicaciones paralelas y distribuidas que usen múltiples nodos interconectados entre sí.

En otros trabajos los dispositivos FPGAs son utilizados solamente como elementos computacionales formando un clúster[30]. Sin embargo, no todas las aplicaciones pueden ser aceleradas de forma efectiva usando FPGAs. Por ejemplo, para aplicaciones basadas en operaciones con números de coma flotante o aplicaciones que requieren una frecuencia de reloj elevada, los procesadores gráficos GPUs son más eficaces como aceleradores de código.

El uso de GPUs como recursos computacionales dentro de infraestructuras de Cloud ha surgido en los últimos años. Por lo tanto, hoy en día algunas compañías ofrecen los servicios de GPU a sus clientes. Por ejemplo, Amazon EC2 [8] soporta el uso de GPUs para flujos de trabajo diseñados bajo CUDA y OpenCL [31].

El uso de FPGAs en sistemas Cloud cada vez va teniendo mucho éxito. Así, empresas como Nimbix [32] ofrecen un sistema de pago

en la nube con una variedad de plataformas para aceleración de código. Recientemente, esta compañía ha lanzado JARVICE, que es una plataforma que incluye la disponibilidad GPUs, DPS y FPGAs para un Catálogo de aplicaciones y además una API o un acceso para enviar tareas a través de una línea de comandos.

Por otra parte, hay algunos trabajos que combinan aceleradores (FPGAs, GPUs) y CPUs dentro de nodos en un clúster como AXEL, AXEL combina nodos heterogéneos con el modelo de programación MapReduce obteniendo como resultado una plataforma de computación de alto rendimiento y de bajo coste. En este trabajo el autor destaca que el mayor inconveniente es la dificultad de implementación de los diseños dentro de las FPGAs en comparación con la programación que ofrece por ejemplo CUDA para GPUs. Además, investigaciones recientes destacan el uso de la reconfiguración dinámica de los nuevos modelos de FPGAs. Por ejemplo en [33] se presenta un servicio de reconfiguración. Este servicio está basado en la administración eficiente de la funcionalidad de la reconfiguración parcial de dispositivos FPGAs, obteniendo así una gran disminución en el tiempo de configuración del dispositivo y optimizando al máximo el uso de recursos.

La investigación que más se acerca a nuestra visión es el proyecto Europeo FP7 HARNESS (Hardware and Network Enhanced Software Systems for Cloud Computing) Proyecto Europeo FP7. El objetivo de este proyecto es desarrollar una plataforma mejorada con acceso a una variedad de recursos computacionales, de comunicación y de almacenamiento. La plataforma consiste en una serie de componentes que pueden tener múltiples implementaciones. A continuación, una aplicación puede ser desplegada de diferentes maneras sobre los recursos obteniendo costos diferentes, mayor rendimiento y características de uso. Sin embargo, en este trabajo se enfoca como una extensión de los trabajos antes mencionados en el cual se toma en cuenta la eficiencia energética.

C. FPGAs en IaaS

Existen algunos trabajos en los cuales se integran FPGAs dentro de entornos Cloud para ofrecer estos como un servicio. En [34]

los autores proponen una arquitectura para proveer FPGAs como servicio en el Cloud de IBM. Esta propuesta está implementada sobre el gestor de Cloud OpenStack [13] y su característica principal es el uso de la reconfiguración parcial. La reconfiguración parcial es una característica de las FPGAs modernas y permite compartir dividir el hardware para ser utilizado por varios usuarios de manera simultánea. Existen trabajos similares a este en los cuales se ofrece FPGAs como parte de una arquitectura genérica a través de una extensión del gestor OpenStack [13].

En [35] se propone un hypervisor para la integración de FPGAs virtuales. En este trabajo el hypervisor tiene acceso a una base de datos que contiene la información de los dispositivos FPGA y este se encarga de su gestión e base a la demanda de los clientes.

Todos estos trabajos [36], [37], [38] se enfocan en ofrecer recursos FPGA como un servicio.

Desde el punto de vista de ofrecer software como un servicio (SaaS), el uso de FPGAs como aceleradores de código a tomando importancia recientemente. En 2014, Microsoft lanzó su proyecto Catapult [39], el cual se enfocó en acelerar con FPGAs el algoritmo Page-rank usado para búsquedas en Bing. Los resultados obtenidos fueron de alrededor de 95% en el rendimiento usando únicamente 22.7 [W] adicionales.

En otro trabajo los autores modificaron la arquitectura de Apache Hadoop [40] para incluir el uso de FPGAs como aceleradores de código. En este trabajo se utilizaron las FPGAs en conjunto con nodos workers. En un trabajo similar [41], se utilizaron FPGAs para acelerar los algoritmos tanto de los nodos mappers como reducers. En este trabajo la estrategia de gestión de recursos se realizó mediante el uso de un sistema multicolos para la asignación de tareas.

En [41], se utilizan FPGAs para acelerar algoritmos de Machine Learning como: K-means, Naive Bayes, k-Nearest Neighbours, Support Vector Machine (SVM). Los resultados indican que en promedio la arquitectura con FPGAs puede alcanzar una aceleración de 2.27x y con un consumo menor de 3.7x.

Existen también productos comerciales enfocados a la aceleración de aplicaciones en Cloud como por ejemplo Ryft3.

Este producto puede alcanzar una aceleración de hasta 100x y una reducción de energía total de 70%.

A continuación se expone la arquitectura propuesta para una integración eficiente de FPGAs dentro de un entorno de Cloud Computing privado.

4. Arquitectura propuesta

En esta sección se describe una arquitectura heterogénea para la construcción de servicios de infraestructura bajo demanda en los sistemas de Cloud Computing basada en [37], [36]. La arquitectura intenta abordar los desafíos de despliegue de una infraestructura como servicio (IaaS) en un centro de datos privado, en el que los dispositivos FPGA son considerados como recursos computacionales virtuales.

Las FPGAs virtuales son elementos computacionales administrados por el sistema junto con las máquinas virtuales que se ejecutan en los distintos procesadores (CPUs). Para ser más precisos, los dispositivos FPGA son usados como aceleradores en la ejecución de aplicaciones. En el resto de la sección se describe las principales características de la arquitectura que proponemos. Antes de explicar la propuesta, es muy importante mencionar que cada nodo se compone de al menos una CPU, y que pueden incluirse algunos elementos COTS extra. La CPU es la encargada de controlar el acceso y la asignación de tareas a los aceleradores. Por lo tanto, hace falta una conexión física entre la máquina virtual y los elementos de aceleración de hardware. Se propone usar el bus de datos PCI expres ya que este es el que proporciona el mejor rendimiento en cuanto a la cantidad de datos que se pueden enviar/recibir por unidad de tiempo. Además cabe destacar que para mejorar aún más el intercambio de datos entre máquinas virtuales y hardware de aceleración pretendemos usar la tecnología de virtualización que proporciona Intel VT-d [11]. Esta tecnología permite que los dispositivos de hardware accedan de manera directa a las máquinas virtuales, reduciendo el tiempo de inactividad y aumentando el rendimiento productivo por una mejor utilización de los recursos del centro de datos.

La mayoría de los entornos Cloud utilizan un software de administración de máquinas virtuales conocido como Virtual Machi-

ne Manager (VMM). Este elemento es el encargado de preparar, desplegar y monitorizar todos los recursos virtuales en un entorno Cloud. La arquitectura propuesta está desarrollada sobre el gestor VMM y permite utilizar las funcionalidades de este elemento y además extender dichas funcionalidades a la explotación de elementos de hardware adicionales como FPGAs o GPUs. Para ello se ha diseñado dos módulos principales, que son: Administrador de Hardware de Aceleración (HAM, por sus siglas en ingles) y una base de datos que hemos denominado Catálogo de Aplicaciones (Catalog) como se muestra en la Figura 8.

HAM es el cerebro de la arquitectura y se encarga de buscar recursos disponibles, además de preparar, desplegar y controlar la infraestructura virtual. Mientras que el segundo elemento es el encargado de guardar y actualizar la información sobre la disponibilidad y el estado de los recursos de todo el sistema. HAM usa las herramientas del administrador de máquinas virtuales para realizar acciones como: consulta de recursos, despliegue de nuevas máquinas virtuales, conectar una máquina virtual con un acelerador de hardware, etc.

Figura 8. Modelo de arquitectura propuesta [37]

El Catálogo, por otra parte, almacena la información sobre los recursos, como por ejemplo número de procesadores de cada nodo, identificación de dispositivos de aceleración, ficheros de aplicaciones (Bitstreams) disponibles para ser usados por estos dispositivos. Además, mantiene actualizada esta información para su uso futuro.

Como se puede observar en la Figura 9, HAM está compuesto por cuatro controladores: *Controlador de Catálogo (CC)*, *Controlador de Infraestructura Virtual (VIC)*, *Controlador Bitstreams (BC)* y finalmente un *Controlador de Tareas (JMC)*, cada uno de ellos tiene una funcionalidad específica que vamos a explicar a continuación.

El *Controlador de Infraestructura Virtual (VIC)* utiliza las funciones del administrador de máquinas virtuales para desplegar máquinas virtuales en los distintos nodos. Además, es el encargado de vincular cada máquina virtual con el hardware de aceleración respectivo.

El segundo elemento es el Controlador de Bitstreams (BC), este módulo ofrece dos funciones:

- a) se encarga de transferir el fichero bitstream desde el servidor hasta el cliente a través de una conexión SSH.
- b) es la encargada de configurar o programar el hardware FPGA usando las herramientas del fabricante.

Otro elemento fundamental es el *Controlador de Tareas (JMC)*, este módulo tiene la función de monitorizar el estado de una ejecución de una tarea, es decir, este módulo es una especie de demonio que se encarga de comunicar al sistema cuando la ejecución de una tarea ha cambiado de estado.

Finalmente, el último módulo es el encargado de interactuar con el *Catálogo (CC)*, este elemento se comunica con el Catálogo a través de operaciones de lectura y escritura. El Controlador del Catálogo mantiene actualizada la información de los recursos del sistema de tal manera que el administrador puede hacer uso de esta información cada vez que reciba una solicitud por parte del usuario o del propio sistema.

Figura 9. Gestor de aceleración de hardware [37]

A continuación se va a explicar los detalles de funcionamiento del sistema. Para ello, se ha dividido el proceso en tres fases. En la primera fase o puesta en marcha del sistema, el *Controlador de Catálogo* se pone en funcionamiento, consultando toda la información sobre los recursos disponibles (paso 1) en ese momento, como se puede ver en la Figura 10. Esta información es recolectada y enviada al Catálogo. Una vez con esta información (paso 2), el sistema se pone en modo de espera de peticiones. En este modo, el servidor se encuentra a la espera de llegada de una petición por parte del usuario o del propio sistema.

Figura 10. Controlador de bitstream [36]

La segunda fase inicia con una petición por parte del usuario. Esta petición se procesa por el Controlador de Infraestructura Virtual (VIC). Este controlador en primera instancia revisa si la solicitud esta correcta y completa (pasos 1 y 2). Los elementos que conforman esta solicitud son datos sobre la máquina virtual, como identificación, una imagen del sistema operativo que se pretende desplegar y la posibilidad de vincular esta máquina virtual con un hardware de aceleración (pasos 6, 7 y 8), como se puede observar en la Figura 11. Si toda esta información está completa y es correcta, se acepta la solicitud y se despliega el software virtual respectivo; en caso contrario, se muestra un mensaje de error.

Figura 11. Controlador de infraestructura Virtual [36]

La arquitectura también pretende proporcionar la opción de que el administrador pueda agregar más imágenes o elementos adi-

cionales al Catálogo como recursos al sistema. Finalmente, una vez que tenemos a nuestra disposición el software virtual desplegado (máquinas virtuales) (pasos 9, 10 y 11) el siguiente paso es preparar el hardware de aceleración. Para ello el sistema realiza los pasos detallados a continuación: El Controlador de Catálogo revisa si dispone de la aplicación (pasos 1 y 2), los ficheros bitstream en el caso de FPGA o de la aplicación en el caso de GPU como se muestra en la Figura 12. Si se dispone de la aplicación, el Catálogo pone a disposición del Controlador de Bitstreams el fichero (bitstream). Este elemento es transferido y/o configurado para ser utilizado como acelerador (pasos 3, 4 y 5). Una vez que todo este proceso se haya realizado correctamente, el sistema está listo para ser explotado por el usuario (paso 6).

Figura 12. Configuración de bitstreams [36]

En el caso de que no se disponga de la aplicación en el Catálogo, el usuario tiene la posibilidad de subir su propio fichero al Catálogo. Por último, hay que mencionar que cuando el usuario haya terminado de usar los recursos virtuales, estos serán liberados del sistema.

B. Estrategia de gestión de recursos

La estrategia de asignación se implementa dentro del Controlador de Infraestructura Virtual, y se describe en la Figura 11. Se usa un método de aprendizaje supervisado para estimar el uso del tiempo de una solicitud de servicio particular, en base a los datos anteriores, es decir, los datos de entrenamiento.

El servicio VIC lleva a cabo dos etapas:

1. Implementa un modelo de predicción que aprovechan la información almacenado en el Catálogo. Tenga en cuenta que en nuestro sistema los usuarios no necesitan tener conocimiento previo de la duración real del servicio, por lo que el tiempo de finalización debe estimarse en este punto.
Los datos de entrada se refieren a la complejidad del software (por ejemplo, la cantidad de los datos a procesar), y el resultado es el tiempo de procesamiento esperado en un recurso de procesamiento específico. Por lo tanto, este modelo se usa directamente por el sistema para hacer predicciones sobre el tiempo de ejecución esperado para las nuevas solicitudes enviadas.
2. El clasificador toma una decisión sobre qué tipo de se necesitan recursos para cumplir con la solicitud de servicio basado en la estimación de tiempo de ejecución total, incluyendo el momento de implementar la infraestructura virtual () y los parámetros de QoS establecidos en la solicitud de servicio. Por lo tanto, la solicitud de servicio puede se clasifica para ser ejecutado dentro de un elemento de procesamiento estándar (CPU) o con la cooperación de hardware de aceleración FPGA.

El objetivo principal de esta estrategia de asignación es evitar la violación de SLA utilizando elementos de proceso estándar y elementos de proceso muy rápidos como aceleradores trabajando juntos, al programar su uso en un “inteligente” camino.

Considere un ejemplo simple en donde una solicitud de servicio, como una Transformada de Fourier para 1024 Gb de datos, se envía al sistema. Supongamos también que no hay hardware de aceleración disponible. Entonces, el clasificador estima el tiempo necesario para completar esta solicitud usando un modelo de regresión lineal. Como posible escenario, el cliente define un tiempo para

usar el software como un servicio, pero el sistema puede estimar que el tiempo definido por el usuario no es suficiente para completar la solicitud si únicamente se ejecuta en CPUs convencionales. En ese caso, si la solicitud es aceptada, una violación de los niveles de servicio especificados se produciría. Sin embargo, cuando el sistema sabe que tiene hardware de aceleración, el servicio VIC clasificaría esta solicitud para programarse en el conjunto de recursos con hardware de aceleración CPU+FPGA. De esta manera, los niveles de servicio solicitados se cumplirían. Como efecto secundario, y en el caso particular de FPGA como aceleradores, existen ventajas adicionales, como el menor consumo de energía.

Una vez que la solicitud de servicio ha sido clasificada para su ejecución, el JMC debe buscar un espacio libre dentro del conjunto adecuado de recursos para ejecutar la solicitud de servicio, y dentro del tiempo intervalo establecido en la solicitud de servicio. Para tal fin, se pueden aplicar varios criterios de búsqueda, como First-Fit o incluso del que mejor ajuste (es decir, programar la solicitud en el primer espacio libre el tiempo suficiente para cumplir la solicitud). La solicitud dentro del intervalo de tiempo especificado por el usuario, o en la ranura cuya duración mejor se ajusta al tiempo estimado de ejecución del servicio. Criterios adicionales pueden incluir tratar de minimizar el número de ejecuciones VMs para minimizar el consumo de energía.

En esta propuesta se considera el algoritmo First-Fit como criterio de asignación de recursos. Esto implica el uso de la estimación calculada en el módulo JMC. Esto se hace para elegir el recurso más adecuado para asistir a la solicitud actual, evitando así el desperdicio de uso de recursos (es decir, cuando el usuario ha especificado un tiempo marco para la ejecución del servicio mucho más grande que el real tiempo de ejecución esperado). Además, el tiempo de implementación también son considerados en esta propuesta. Este tiempo considera las acciones de iniciar la máquina virtual con los parámetros tales como imágenes, memoria, software, y controladores previamente definidos (ver Figura 14).

5. Evaluación

El enfoque descrito en secciones anteriores ha sido implementado en un entorno privado real heterogéneo en la Universidad de

Castilla La Mancha (España) [37], en el cual se ha incluido una FPGA. El objetivo de esta evaluación es realizar una prueba de concepto de la arquitectura propuesta. Además, los experimentos también se han diseñado para garantizar la repetitividad.

Para comparar el impacto de agregar hardware como dispositivos de aceleración de aplicaciones hemos dividido estos experimentos en dos escenarios. Por un lado, un escenario estándar de Cloud privada con solo CPU como procesador, este constituirá nuestra infraestructura de control y se usa como referencia para la comparación.

El otro escenario consiste en incluir una FPGA como un elemento de procesamiento de aceleración en uno de los nodos (ver Figura 13).

En ambos escenarios, los clientes enviarán solicitudes para un servicio particular. Para este caso se ha seleccionado la Transformada Discreta de Fourier.

Para evaluar el desempeño se ha considerado la tasa de cumplimiento del acuerdo de nivel de servicio (SLA). Así, se contabiliza el número de cumplimientos de SLA. Por el contrario, una violación de SLA ocurre cuando el sistema no puede completar el servicio de software dentro de los requisitos de tiempo establecidos en la solicitud del usuario. Por lo tanto, SLA cumplidos son aquellos que se completan sin sufrir una violación.

La siguiente subsección entra en más detalles sobre banco de pruebas y la carga de trabajo.

Figura 13. Entorno de pruebas

A. Carga de trabajo

La infraestructura del banco de pruebas se divide en dos actores: el proveedor de servicios de Cloud y los clientes. Respecto al

proveedor, el módulo HAM ha sido desarrollado sobre el administrador de infraestructura virtual (VMM), OpenNebula [7]. Además, se ha configurado KVM [23] como hypervisor.

El hardware utilizado consiste en dos diferentes servidores. Uno de ellos es un Intel Core i5 con 6 Gb de la memoria RAM, con un Xilinx FPGA Virtex 6 entrelazado a su bus PCIe. La FPGA Virtex 6 es un ML605 Evaluation Kit, basado en Xilinx XC6VLX-240T-1FFG1156 [24].

El otro servidor consiste en un Intel Quad Core con 6Gb de RAM memoria sin FPGA. El servicio ofrecido es una emulación de la Transformada Discreta de Fourier. La transformada Discreta de Fourier (DFT) es una herramienta importante en muchas ramas de la ciencia y la ingeniería [25], como en el procesamiento de señales digitales o procesamiento de imágenes. El DFT es también usado para resolver eficientemente ecuaciones diferenciales parciales, y para realizar otras operaciones como convoluciones [26]. Para la evaluación de la propuesta se usa el algoritmo de Cooley-Tukey [27]. Se ha elegido este algoritmo para reducir la complejidad computacional de $O(N^2)$ a $O(N \log 2N)$ [28], y la regularidad del algoritmo lo hace adecuado para la implementación en hardware. Los detalles sobre el algoritmo completo se detallan en [28].

Para emular la parte de clientes solicitando el servicio con diferentes necesidades informáticas, el algoritmo se ejecuta un cierto número de veces sobre el la misma cantidad de datos. Entonces, el parámetro asociado con cada servicio solicitado indica la complejidad del trabajo.

En la emulación, los clientes envían solicitudes de servicio DFT al proveedor. En esta solicitud se definen dos parámetros. El primero se refiere al tiempo de uso, y contiene el tiempo de inicio y finalización de la tarea. Por ejemplo, $t_{start} = 12:00:00$ y $t_{end} = 12:35:00$. Esto significa que el servicio debe ser provisto dentro de este periodo de tiempo. Sin embargo, estos parámetros de entrada son no obligatorios. Si no se proporciona t_{start} , el servicio será considerado como una solicitud inmediata y el tiempo del sistema se asignará a t_{start} (es decir, $t_{start} = \text{tiempo}$). Si no se proporciona t_{end} , el servicio será considerado como uno de mejor esfuerzo, sin ninguna restricción de tiempo de finalización. En este caso, cualquier recurso libre se puede asignar para la ejecución del servicio.

Figura 14. Definición de una tarea de usuario

En el segundo parámetro se define el nombre del servicio como tal y la complejidad computacional del servicio definida como el número de DFTs para calcular sobre un cierto conjunto de datos.

Se utiliza una distribución de Poisson como tasa de llegada de solicitudes, con una tasa $\lambda = 1/1$, es decir, se genera una solicitud de servicio cada minuto. Los parámetros de cada solicitud de servicio se generan de forma aleatoria. En particular, cada cliente generará una solicitud de servicio con N aleatoriamente entre 30000 y 300000. La hora de inicio (t_{start}) se ha establecido en la hora actual cuando se genera cada solicitud. Se debe tener en cuenta que la hora de inicio se puede configurar en cualquier momento después del tiempo de envío.

Para la hora de finalización, se han considerado dos tipos de solicitudes de servicio. Por un lado, los “clientes normales” que envían solicitudes cuyo tiempo de finalización (t_{end}) se calculará como una distribución normal con $\mu = 9$ minutos y $\sigma = 0.5 * \mu$. Por otro lado, los “clientes en tiempo real” exhiben tiempos de uso mucho más exigentes. Su tiempo de finalización se establecerá según una distribución normal con $\mu = 0.9$ minutos y $\sigma = 0.5 * \mu$. Recuerde que el sistema rechazará aquellas solicitudes de servicio que no se pueden atender dentro del marco de tiempo especificado con los recursos disponibles. Por lo tanto, se espera que los “clientes en tiempo real” necesiten hardware de aceleración para ser atendido adecuadamente.

En la siguiente sección, se analiza el impacto de agregar elementos de FPGA como hardware de aceleración y su administración con las técnicas propuestas.

B. Resultados

La arquitectura y los algoritmos descritos en las secciones anteriores se han probado generando una carga de trabajo compuesta por 60 clientes. La misma carga de trabajo tiene sido aplicado a los dos escenarios, servidores solo con CPUs y servidores con FPGA. Además, cada experimento ha sido repetido tres veces, y los resultados que se muestran aquí son los valores promediados.

Figura 15. Total de peticiones que cumplen SLA

Figura 16. Distribución de peticiones

En cada experimento, tanto “clientes normales” como Los “clientes en tiempo real” se envían solicitudes de servicio al sistema. Como los nodos tienen ocho núcleos y una FPGA la razón de solicitudes de tiempo real es de 1/7; es decir que por cada siete solicitudes normales se genera 1 de tiempo real. Las solicitudes han sido mezcladas al azar siguiendo esta proporción.

Figura 17. Total de peticiones que cumplen SLA

Figura 18. Total de peticiones que cumplen con SLA

Como se muestra en la Figura 18, el número de solicitudes aceptadas para el escenario en el que se utilizan solo CPUs es 51, y el número de solicitudes sin violación de SLA es 29. Por lo tanto, resulta

que el 57% de las solicitudes han sido atendidas con éxito. En el caso del escenario de CPU y FPGA, el número de solicitudes aceptadas es de 40, mientras que el 100% de las solicitudes cumplen el SLA. A pesar del hecho de que el número de solicitudes aceptadas es ligeramente menor que en el escenario en el cual únicamente se usa CPU, el porcentaje de solicitudes que se sirven con éxito es del 100%.

Finalmente, la proporción de solicitudes atendidas por cada clase de recursos se muestra en la Figura 16. Se puede ver que se ha asignado el 67% de las solicitudes aceptadas al FPGA. En resumen, la fiabilidad de la arquitectura y las técnicas propuestas al proporcionar garantías de QoS se refleja en los resultados. Estos son resultados prometedores, especialmente porque solo hay un FPGA disponible como acelerador.

Para el segundo escenario los resultados son similares como se puede observar en la Figura 17, en el caso que se usa la combinación de CPU y FPGA todas las solicitudes aceptadas por el sistema son cumplidas. Finalmente, en la Figura 19 se puede observar que la introducción de solicitud de mayor demanda hace que el sistema distribuya de mejor manera las peticiones a los recursos para este caso se concentran más peticiones para los recursos que tienen CPU y FPGA (67%). Así, el sistema puede lograr distribuir de manera más eficiente la carga, aprovechando mejor las características de los recursos.

Figura 19. Distribución de peticiones

6. Conclusiones

Una infraestructura heterogénea Cloud en el que los aceleradores tales como FPGA y GPU se utilicen para reducir el consumo de energía puede ser una excelente forma de utilizar los recursos de una manera más eficiente. Por lo tanto, en este trabajo nos hemos enfocado en una arquitectura que soporte agregar un elemento de computación de hardware como acelerador y que pueda ser ofrecida al usuario como servicio. El trabajo futuro consiste en la implementación de la arquitectura propuesta. Además en un futuro se propone enfocar todos los esfuerzos en el diseño de una herramienta para planificación del uso efectivo de los recursos en el tiempo. Adicionalmente, exploraremos la factibilidad de usar la tecnología que ofrece Intel (SR-IOV) [15], para que múltiples máquinas virtuales puedan compartir el hardware de aceleración al mismo tiempo, junto con las funcionalidades que ofrece la reconfiguración dinámica y reconfiguración parcial de FPGAs [5].

Agradecimientos

Este trabajo se ha desarrollado con la colaboración del Instituto de Investigación en Informática de Albacete (España) y la Universidad Politécnica Salesiana (Quito-Ecuador). Además, es muy importante agradecer a la Secretaría Nacional de Ciencia y Tecnología (Senescyt) por el apoyo a proyectos de investigación. Finalmente, un agradecimiento especial al grupo ARCO de la Universidad de Castilla La Mancha, por el uso de su infraestructura y el soporte dado durante todo el desarrollo de este proyecto de investigación.

Referencias

- [1] L. Wang *et al.*, “Cloud computing: a perspective study,” *New Gener. Comput.*, vol. 28, no. 2, pp. 137–146, 2010.
- [2] R. Buyya, C. S. Yeo, and S. Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,” in *High Performance Computing and Communications, 2008. HPCC’08. 10th IEEE International Conference on*, 2008, pp. 5–13.
- [3] T. Erl, R. Puttini, and Z. Mahmood, *Cloud computing: concepts, technology & architecture*. Pearson Education, 2013.

- [4] L. Wu, S. K. Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments," *J. Comput. Syst. Sci.*, vol. 78, no. 5, pp. 1280–1299, 2012.
- [5] J. Shuja *et al.*, "Survey of techniques and architectures for designing energy-efficient data centers," *IEEE Syst. J.*, vol. 10, no. 2, pp. 507–519, 2016.
- [6] J. Glanz, "Power, pollution and the internet," *New York Times*, vol. 22, 2012.
- [7] A. Shehabi *et al.*, "United states data center energy usage report," 2016.
- [8] K. Freund, "Amazon and xilinx deliver new fpga solutions." *Forbes*, 2017.
- [9] R. Uhlig *et al.*, "Intel virtualization technology," *Computer (Long Beach, Calif.)*, vol. 38, no. 5, pp. 48–56, 2005.
- [10] M. Fenn, M. Murphy, J. Martin, and S. Goasguen, "An evaluation of KVM for use in cloud computing," in *Proc. 2nd International Conference on the Virtual Computing Initiative, RTP, NC, USA*, 2008.
- [11] D. Abramson *et al.*, "Intel Virtualization Technology for Directed I/O.," *Intel Technol. J.*, vol. 10, no. 3, 2006.
- [12] I. AMD and O. Virtualization, "Technology (IOMMU) Specification," 2007.
- [13] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: toward an open-source solution for cloud computing," *Int. J. Comput. Appl.*, vol. 55, no. 3, 2012.
- [14] B. Sotomayor, R. S. Montero, I. M. Llorente, I. Foster, and others, "Capacity leasing in cloud systems using the opennebula engine," in *Workshop on Cloud Computing and its Applications*, 2008, vol. 3, p. 22.
- [15] P. Mell, T. Grance, and others, "The NIST definition of cloud computing," 2011.
- [16] M. Sachdeva, P. Rana, R. Kapoor, and M. Shahid, "Cloud computing-pay as you go technology," in *Proceedings of the 5th National Conference on Computing for National Development*, 2011, vol. 161.
- [17] R. Buyya, S. K. Garg, and R. N. Calheiros, "SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions," in *Cloud and Service Computing (CSC), 2011 International Conference on*, 2011, pp. 1–10.
- [18] J.-H. Morin, J. Aubert, and B. Gateau, "Towards cloud computing SLA risk management: issues and challenges," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 2012, pp. 5509–5514.
- [19] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, pp. 606–610.
- [20] L. Tomás and J. Tordsson, "Improving cloud infrastructure utilization through overbooking," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, 2013, p. 5.
- [21] M. Balakrishnan *et al.*, "Tango: Distributed data structures over a shared log," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2013, pp. 325–340.
- [22] A. Berl *et al.*, "Energy-efficient cloud computing," *Comput. J.*, vol. 53, no. 7, pp. 1045–1051, 2010.

- [23] T. V. T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, 2010, pp. 1–8.
- [24] M. Mezmaiz *et al.*, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, 2011.
- [25] K. Bloor, R. Chirkova, Y. Viniotis, and T. Salo, "Dynamic request allocation and scheduling for context aware applications subject to a percentile response time SLA in a distributed cloud," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, 2010, pp. 464–472.
- [26] G. A. Constantinides, "FPGAs in the cloud," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, p. 167.
- [27] E. Billauer, "Xillybus." Haifa, 2017.
- [28] S. Mohl, "The Mittrion-C programming language," *Mittrionics Inc*, 2006.
- [29] M. Jacobsen and R. Kastner, "RIFFA 2.0: A reusable integration framework for FPGA accelerators," in *2013 23rd International Conference on Field programmable Logic and Applications*, 2013, pp. 1–8.
- [30] M. Oikawa, A. Kawai, K. Nomura, K. Yasuoka, K. Yoshikawa, and T. Narumi, "DS-CUDA: A middleware to use many GPUs in the cloud environment," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, 2012, pp. 1207–1214.
- [31] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Comput. Sci. Eng.*, vol. 12, no. 3, pp. 66–73, 2010.
- [32] K. Paranjape, S. Hebert, and B. Masson, "Heterogeneous computing in the cloud: Crunching big data and democratizing HPC access for the life sciences," *Intel Corp.*, 2010.
- [33] P. Grigorasi, M. Tottenham, X. Niu, J. G. F. Coutinho, and W. Luk, "Elastic Management of Reconfigurable Accelerators," in *2014 IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2014, pp. 174–181.
- [34] I. Arpacı, "Understanding and predicting students' intention to use mobile cloud storage services," *Comput. Human Behav.*, vol. 58, pp. 150–157, 2016.
- [35] R. Moreno-Vozmediano, Montero, and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer (Long. Beach. Calif.)*, vol. 45, no. 12, pp. 65–72, Dec. 2012.
- [36] J. Proaño, C. Carrión, and M. B. Caminero, "An Open-Source Framework for Integrating Heterogeneous Resources in Private Clouds," in *CLOSER*, 2014, pp. 129–134.
- [37] J. P. Orellana, M. B. Caminero, and C. Carrión, "On the provision of SaaS-level quality of service within heterogeneous private clouds," in *Utility and*

- Cloud Computing (UCC)*, 2014 IEEE/ACM 7th International Conference on, 2014, pp. 146–155.
- [38] J. Proaño, C. Carrión, and M. B. Caminero, “Towards a green, QoS-enabled heterogeneous cloud infrastructure,” in *Parallel and Distributed Processing Symposium Workshops*, 2016 IEEE International, 2016, pp. 7–16.
- [39] A. Putnam *et al.*, “A reconfigurable fabric for accelerating large-scale datacenter services,” in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, pp. 13–24.
- [40] Y.-M. Choi and H. K.-H. So, “Map-reduce processing of k-means algorithm with FPGA-accelerated computer cluster,” in *Application-specific Systems, Architectures and Processors (ASAP)*, 2014 IEEE 25th International Conference on, 2014, pp. 9–16.
- [41] K. Neshatpour, M. Malik, and H. Homayoun, “Accelerating machine learning kernel in hadoop using fpgas,” in *Cluster, Cloud and Grid Computing (CCGrid)*, 2015 15th IEEE/ACM International Symposium on, 2015, pp. 1151–1154.

Sobre los autores

Julio Proaño es profesor titular la Universidad Politécnica Salesiana (Quito-Ecuador). Es PhD en Tecnologías Informáticas Avanzadas en la Universidad de Castilla (La Mancha, España). Recibió su título de grado en Ingeniería Electrónica en la Universidad Salesiana Politécnica (Cuenca-Ecuador) en 2008 y su Maestría en Informática en la Universidad de Castilla (La Mancha, España en 2012. Sus intereses actuales de investigación son los Gestión de recursos heterogéneos (FPGAS) y QoS con SLA en entornos de Cloud Computing. Él es un miembro del IEEE.

Maria Blanca Caminero es Profesora Asociada en Arquitectura de Computadores y Tecnología en Computación Departamento de Sistemas La Universidad de Castilla La Mancha. Ella tiene un PhD. en Informática de la Universidad de Castilla La Mancha, y sus intereses actuales de investigación son Soporte de QoS y meta-programación en Grids y Nubes. Ella es miembro del IEEE

Carmen Carrión es Profesora Asociada en Arquitectura de Computadora y Tecnología en el Departamento de Sistemas Computacionales de la Universidad de Castilla La Mancha. Tiene un doctorado en Física de la Universidad de Cantabria, y sus intereses incluyen meta-programación y QoS en Grids and Cloud.