UNIVERSIDAD DE COSTA RICA

SISTEMAS DE ESTUDIOS DE POSGRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE APRENDIZAJE INSTITUCIONAL PARA LA NUBE: UNA PLATAFORMA CAPAZ DE ADAPTARSE DINÁMICAMENTE SEGÚN LA DEMANDA

Trabajo final de investigación aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Computación e Informática para optar al grado y título de Maestría Profesional en Computación e Informática

JOB NATÁN CÉSPEDES ORTIZ

Ciudad Universitaria Rodrigo Facio, Costa Rica

2016

Dedicatoria

A todas las personas extraordinarias que componen mi vida y de quiénes sobresalen las mujeres, me han llenado de bendiciones siempre, Grace, Antonio, Susana, Tony, Flor, Fanny, Keren, Raquel, Esther, Santiago y bebé. Te amo Karla.

Agradecimientos

Como estela de brillante tonalidad Divinidad son las innumerables personas, circunstancias y condiciones que me han ayudado de diversas formas, una y otra vez, no solo en esta fase de formación y superación sino durante toda la trayectoria. Muchas gracias a mi mamá; a mi esposa, al resto de la familia; a la tutoría de Ricardo Villalón; a la Unidad Metics, Silvia, Susan, Grace; a la Universidad de Costa Rica.

Un agradecimiento especial a usted estimada persona lectora; espero que encuentra aplicaciones en esta propuesta.

"Este trabajo final de investigación aplicada fue aceptado por la Comisión del Programa de Estudios de Posgrado en Computación e Informática de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Profesional en Computación e Informática"

M.Sc. Francisco Arroyo Mora

Representante de la Decana

Sistema de Estudios de Posgrado

Dr. Ricardo Villalón Fonseca

Profesor Guía

Dr. Vladimir Lara Villagrán

Director

Programa de Posgrado en Computación e Informática

Job Natán Céspedes Ortiz

Sustentante

Tabla de contenidos

RESUMEN

Los sistemas de gestión de aprendizaje convencionales (LMS por sus siglas en inglés) han sido desarrollados siguiendo los modelos de arquitectura tradicional en infraestructura tecnológica, donde la asignación de recursos se limita y se fija según las prestaciones del *hardware* utilizado. Por otro lado, han surgido alternativas disruptivas que utilizan el paradigma de la computación en la Nube para sobrepasar ese tipo de limitaciones. El modelo de la Nube ofrece un entorno de características que facilitan el desarrollo de soluciones elásticas. Bajo esas condiciones una plataforma tradicional estática se puede convertir en una dinámica con la capacidad de responder a variaciones en la demanda y auto servirse sólo de los recursos necesarios.

La plataforma de aulas virtuales institucional de la Universidad de Costa Rica utiliza Moodle como LMS; la propuesta actual para su arquitectura permite robustez para soportar el servicio, pero sigue el modelo de asignación de recursos tradicional. La Universidad tiene también un proyecto de Nube Académica que posibilita el desarrollo de soluciones elásticas en un entorno basado en el gestor de Nube OpenStack. El objetivo ha sido integrar este concepto de elasticidad en el diseño e implementación de una nueva plataforma que utilice Moodle como LMS, aprovechando las características de la Nube para obtener un grado de adaptabilidad dinámica.

Para lograrlo se identificaron, detallaron y diseñaron los componentes base de una arquitectura auto escalable para un LMS, según las características del problema, los requerimientos y los recursos disponibles. El diseño propuesto sirvió como insumo para implementar un prototipo en el entorno de la Nube disponible. Posteriormente se diseñaron, aplicaron y analizaron dos experimentos para evaluar la capacidad de adaptación dinámica del prototipo respecto a la demanda y para valorar su desempeño.

El resultado fue un prototipo elástico para el LMS Moodle, que es además tolerante a fallas y altamente disponible. Fue evidente a partir de la evaluación que el prototipo tiene un rango de adaptabilidad. Esta característica le permite ajustar su capacidad de forma dinámica ante cambios en la demanda, utilizando diferentes estrategias para agregar o liberar recursos. Aunque la variabilidad en los tiempos de respuesta y el consumo de recursos en el hipervisor durante la evaluación se incrementaron con la característica elástica; el prototipo tuvo un buen rendimiento respecto a las otras arquitecturas comparadas. La solución sirve como base, no solo para Moodle, sino también para otros sistemas tradicionales similares, que tienen en la Nube una forma para optimizar el uso de los recursos computacionales, disminuir los tiempos de implementación de las soluciones y aumentar las facilidades en las tareas de gestión de sus servicios.

Resumen en Inglés

Conventional learning management systems (LMS) have been developed following the traditional architecture models in technological infrastructure, where the allocation of resources is limited and set according to the features of the hardware used. On the other hand, disruptive alternatives that use the cloud computing paradigm have emerged to overcome these kind of limitations. The Cloud model offers an environment with features that make the development of elastic solutions easier. Under these conditions, a traditional static platform can become a dynamic one with the ability to respond to demand variations and serve itself only with the necessary resources.

The University of Costa Rica's institutional virtual classrooms platform uses Moodle as LMS; the current model for its architecture allows robustness to withstand the service, but it follows the traditional allocation of resources. The University also has an Academic Cloud project that enables an environment based on OpenStack Cloud manager to develop elastic solutions. The aim has been to integrate the elasticity concept in the design and implementation of a new platform, in a way that it continues using Moodle as LMS, takes advantage of the Cloud features and obtains a degree of dynamic adaptability.

To achieve this aim, the basic components for an auto scalable architecture were identified, detailed and designed, according to the characteristics of the problem and the existing requirements and resources. The proposed design provided an input to implement a prototype in the Cloud environment available. After that, two experiments were designed, performed and analyzed to evaluate the capacity of the prototype for dynamic adaptation in relation to the demand and also to evaluate its performance.

The result was an elastic prototype for a LMS platform based on Moodle that is also fault-tolerant and highly-available. From the evaluation, it was evident that the prototype has an adaptability range. This characteristic allows to adjust its capacity in a dynamic way, following different strategies to add or release resources when faced with changes in the demand. Even though the variability in response times and in the hypervisor consumption of resources increased with the elastic feature; the prototype performed well in relation to other architectures compared. The solution serve as a basis not only for Moodle but also for other similar traditional systems that have in the Cloud a way to optimize the use of computing resources reduce implementation times and increase facilities in managing tasks of their services.

Lista de cuadros

Lista de figuras

Lista de abreviaturas

API: Interfaz de programación de aplicaciones (en inglés).

CMP: Plataformas de gestión de la Nube (en inglés).

CMS: Sistemas de gestión de contenido (en inglés).

DBMS: Sistema de gestión de bases de datos (en inglés).

LCMS: Sistema de gestión de contenidos para el aprendizaje (en inglés).

LMS: Sistema de Gestión de Aprendizaje (en inglés).

METICS: Unidad de Apoyo a la Docencia Mediada con Tecnologías de la Información y Comunicación.

MOOC: Curso en línea, masivo y abierto (en inglés).

NIST: Instituto Nacional de Normas y Tecnología de los Estados Unidos (en inglés).

QoS: Calidad del servicio (en inglés).

SAN: Red de área de almacenamiento (en inglés).

SDN: Redes definidas por software (en inglés).

SLA: Acuerdo a nivel de servicio (en inglés).

SPOF: Puntos únicos de fallo (en inglés).

VLE: Sistema virtual de aprendizaje (en inglés).

1 Introducción

Las arquitecturas computacionales convencionales de servidores físicos suelen ser estáticas en la asignación de recursos y no son capaces de adaptarse dinámicamente a diferentes contextos de uso. Inician su ciclo de vida con un aprovisionamiento de recursos tecnológicos determinado y lo mantienen en el tiempo. La demanda del servicio que soportan, por otro lado, es relativa, depende de una serie de factores y puede presentar tanto incrementos como decrementos.

Este es el caso de la plataforma para el Sistema de Gestión de Aprendizaje institucional, LMS por sus siglas en inglés, de la Universidad de Costa Rica. Cuenta con una arquitectura tradicional donde la asignación de los recursos computacionales disponibles es igual al total de las prestaciones del hardware destinado para soportar el servicio. Cuando esa asignación ya no es suficiente es necesario realizar nuevas adquisiciones para reemplazar el hardware por otro de mayor capacidad. Mientras tanto el volumen de tráfico hacia la plataforma no solo aumenta con el tiempo, también varía según los ciclos lectivos, la oferta académica y las diferentes características de los entornos virtuales que hospeda, llegando incluso a disminuir de forma considerable durante los periodos de verano, donde hay un número inferior de estudiantes y una menor oferta de cursos.

La demanda variable en contraste con la asignación y aprovisionamiento estático de recursos resultan en la subutilización de la plataforma de un servicio durante todo el ciclo de vida, hasta que esta alcance la capacidad máxima de utilización. Considerando el costo de la mayoría de los recursos y en especial el de los recursos críticos como por ejemplo el procesamiento, la memoria y el almacenamiento, lo óptimo es que sean aprovisionados dinámicamente, para que puedan ser reasignados y reutilizados por otros sistemas o aplicaciones según sea necesario. Sin embargo, una solución tradicional no ofrece esas características. Al alcanzar el máximo nivel de utilización para la cual fue diseñada debe ser reemplazada. Es necesario, entonces, ir más allá de las soluciones convencionales para implementar una alternativa que ofrezca la posibilidad de adaptarse

dinámicamente a la demanda, de forma automática para optimizar el uso de recursos, disminuir al mínimo la interacción humana, y con una asignación variable de recursos para extender el ciclo de vida de los sistemas, integrando otros conceptos importantes para un servicio institucional como lo son la tolerancia a fallas y la alta disponibilidad.

En la práctica, no es una tarea sencilla aplicar conceptos como escalabilidad, aprovisionamiento automático, tolerancia a fallos o alta disponibilidad sin afectar significativamente otros aspectos importantes, como por ejemplo el rendimiento. Existen diferentes correlaciones entre ellos, algunas de ellas negativas, que hacen que el problema, planteado en estos términos, gane en complejidad. Esto no quiere decir que es imposible lograr un diseño y una implementación robustos y confiables capaces de adaptarse según el contexto de uso. Haciendo uso de la infraestructura, la tecnología y los servicios apropiados se puede lograr una plataforma dinámica y auto escalable.

La *computación en la nube* es el entorno ideal para desarrollar este tipo de soluciones. Por medio de un modelo de servicios novedoso que hace uso de la virtualización y de la computación distribuida, mejora las condiciones para diseñar e implementarlas de manera que sean capaces de crecer o decrecer rápidamente según sea necesario con el objetivo de optimizar el uso de los recursos. A esa capacidad también se le conoce como *elasticidad* (Herbst, Kounev, & Reussner, 2012). El presente trabajo es una propuesta para integrar este concepto de elasticidad en el diseño e implementación de una plataforma LMS institucional en la Nube, de forma que se obtenga un grado de adaptabilidad dinámica adecuado a las necesidades de la Universidad.

Para instituciones como la Universidad de Costa Rica, donde conviven una variedad de servicios y donde se comparte una infraestructura tecnológica, esta posibilidad resulta muy beneficiosa. Además, es una propuesta novedosa, que aprovecha la tecnología de punta, optimiza el uso de los recursos computacionales disponibles, mejora los tiempos de despliegue e implementación de las aplicaciones y sistemas, y disminuye la intervención humana por medio de componentes automatizados. De este modo, es posible desplegar rápida y eficientemente un LMS elástico, que incorpora conceptos de tolerancia a fallas, alta disponibilidad y automatización necesarios para una

aplicación de tipo institucional, y que puede ofrecer un buen rendimiento en un entorno de uso dinámico. Además, facilita la gestión del ciclo de vida de la plataforma, porque permite realizar actualizaciones a los diferentes componentes de configuración y modificaciones a la asignación de los recursos (escalamiento vertical u horizontal) de manera rápida y eficaz. Por otro lado, su utilidad no es exclusiva para un LMS. La arquitectura se basa en el diseño de capas, por lo que también puede ser utilizada por otro sistema u aplicación que utilice este diseño. Como por ejemplo los sistemas de gestión de contenido, conocidos como CMS por sus siglas en inglés.

Este trabajo ha sido estructurado de la siguiente manera. En el capítulo 2 se presenta una sección con antecedentes del LMS de la Universidad de Costa Rica, una breve caracterización de su uso y se detalla el contexto de actualización hacia una nueva plataforma. Además, se realiza una revisión literaria sobre trabajos similares que desarrollan también conceptos e ideas en el diseño e implementación de soluciones para LMS en la Nube. En el capítulo 3, se presenta el marco teórico alrededor de los LMS, la Nube y los sistemas para gestión de la Nube. En el capítulo 4 se describe el diseño del prototipo propuesto. El capítulo 5 detalla la implementación del prototipo, así como la configuración y los componentes de hardware y software requeridos. Los resultados con la evaluación de la propuesta se incluyen en el capítulo 6. El último capítulo recopila las conclusiones y el posible trabajo futuro.

1.1 Objetivo general

Integrar el concepto de elasticidad en el diseño e implementación de una plataforma LMS institucional utilizando el Sistema de Gestión de la Nube OpenStack de forma que se obtenga un grado de adaptabilidad dinámica en el entorno de la Nube.

1.2 Objetivos específicos

- Diseñar una arquitectura para un LMS institucional con características de elasticidad apoyándose en los servicios de OpenStack.
- Implementar el modelo propuesto utilizando OpenStack y sus servicios para desplegar la arquitectura en la Nube.

- Evaluar el prototipo en cuanto a elasticidad para determinar si es capaz de adaptarse dinámicamente según la demanda.

3 Antecedentes y revisión de la literatura

En esta sección se incluyen los antecedentes de la plataforma para el LMS institucional de la Universidad de Costa Rica y una revisión de investigaciones relacionadas con los objetivos de este trabajo. Se inicia con un breve repaso de la historia y la evolución de la plataforma junto con algunos de sus detalles técnicos y ciertas estadísticas de uso por parte de la comunidad universitaria para establecer el contexto y comprender mejor la función y el rol de esta herramienta tecnológica.

La Universidad de Costa Rica cuenta con una plataforma de aulas virtuales desde el 2004 a cargo de la Unidad de Apoyo a la Docencia Mediada con Tecnologías de la Información y la Comunicación (METICS). La plataforma creció impulsada por el uso y la promoción de las tecnologías de la información y la comunicación (TIC) así como por la capacitación en ellas y los avances tecnológicos (Chacón, 2009). Mediación Virtual fue en su momento uno de los pasos evolutivos, vino a reemplazar la primera versión conocida como "UCR Interactiva". La versión a la que se pasó incluía en un único servidor físico, el LMS de código abierto Moodle junto con los otros requerimientos como el sistema operativo Linux, el interpretador y las dependencias PHP y una base de datos SQL. Con estas características la plataforma llegó a soportar un volumen de tráfico importante. Por ejemplo, el número de usuarios inscritos pasó de ser 6.205 en el 2008 a 15.428 en el 2010 (Salazar, 2011). Ya para el segundo semestre del 2014 se ofrecieron 703 entornos virtuales con 9.652 usuarios activos, 1.288 de los cuales eran profesores, según las estadísticas de la aplicación.

Esas y otras estadísticas son importantes porque permiten identificar ciertos patrones de uso característicos. En la Figura 1 se comparan las estadísticas de acceso de usuarios a la plataforma en el año 2013 con las del año 2014. Se observa que, para cada año, hay dos periodos donde la cantidad de usuarios es consistente y relativamente alta y otros periodos donde el acceso y uso de la plataforma baja considerablemente. Este patrón coincide con el calendario universitario y los periodos lectivos de la Universidad, el I ciclo y el II ciclo son los dos principales periodos recién mencionados. La disminución en las visitas es más evidente en el III ciclo o periodo de verano donde hay

menos cursos impartidos y menos estudiantes matriculados. De forma similar el número de usuarios es menor los fines de semana y feriados, como por ejemplo durante Semana Santa. También es posible observar que se dan variaciones características dentro de un día y según la hora, como se observa en la Figura 2.

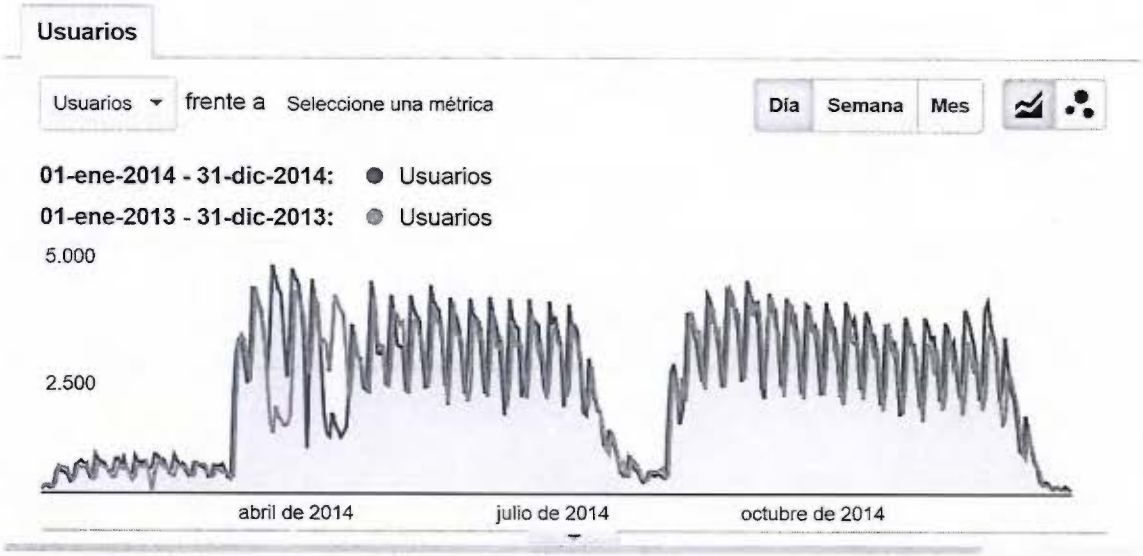


Figura 1. Comparativa anual del tráfico de usuarios en Mediación Virtual. Reporte de Google Analytics

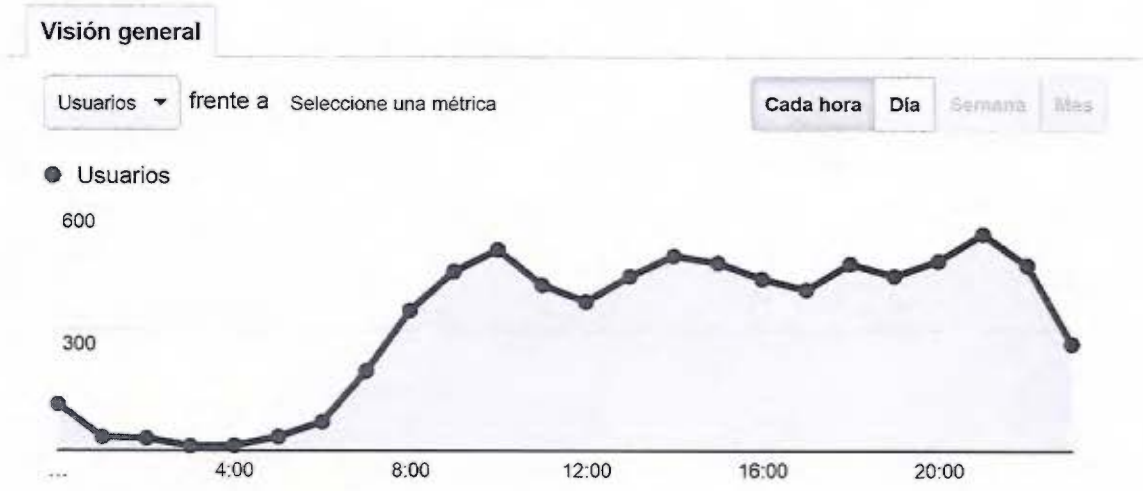


Figura 2. Tráfico de usuarios diario en Mediación Virtual. Reporte de Google Analytics

Naturalmente con el paso del tiempo se hizo necesario una modernización. A continuación se rapaza el último proceso de actualización para aumentar la capacidad y

robustez de la plataforma. La intención es que la información sirva más adelante para comparar las posibles gestiones e implicaciones de una solución tradicional frente a una solución en la Nube. Primeramente se selecciona un hecho del 2010 como el inicio del proceso. Ese año se presupuestó dentro del proyecto "Fortalecimiento del Acceso a Internet a la Población Estudiantil" la compra de nuevos servidores para la plataforma (Universidad de Costa Rica, 2010). El siguiente año los servidores ya habían sido adquiridos y se instalaron en el centro de datos institucional (Universidad de Costa Rica, 2011). Unos meses después, en el 2012, estaba listo la instalación de los sistemas operativos, la configuración de otros servicios y un espacio en la red de área de almacenamiento, SAN por sus siglas en Ingles (Universidad de Costa Rica, 2012). Para el año 2013 se contrató por medio de servicios profesionales la implementación y personalización de la plataforma según una arquitectura para los servidores propuesta por el Centro de Informática. En esa arquitectura se dividían los cuatro servidores físicos en dos *clusters*: uno web y otros de base de datos, como se puede observar en la Figura 3. Durante ese año el profesional contratado trabajó y entregó la versión conocida como "Se'kane", candidata para ser la nueva plataforma. No obstante, en marzo del 2014, en una nueva contratación se planteó una re implementación para resolver algunas dificultades técnicas existentes y para aprovechar el proyecto de la Nube Académica. De esa forma la nueva plataforma salió a producción en julio del 2015 utilizando el mismo nombre de Mediación Virtual.

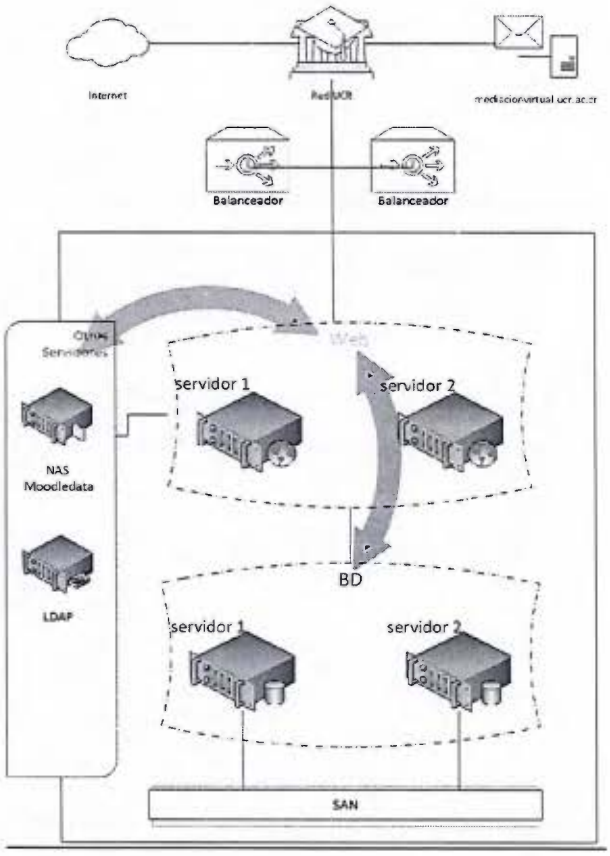


Figura 3. Arquitectura de servidores físicos inicial para la nueva Medición Virtual. Adaptación con permiso (Universidad de Costa Rica, 2012)

Ese proceso se desarrolló dentro del proyecto Multiversa. También hubo una vinculación estratégica con el proyecto de Nube Académica Computacional (NAC). En Multiversa participan múltiples instancias de la Universidad de Costa Rica para dar "un enfoque de flexibilidad curricular para transformar el quehacer docente a la multiversidad de entornos en los que se realizan los procesos de aprendizaje: físicos, virtuales y bimodales" (Universidad de Costa Rica, 2015). Tiene como meta a 10 años que al menos un 50% de las carreras tengan un grado virtual. Eso se podría traducir en 1500 cursos al año. Pero tiene también planes de acción a corto y mediano plazo. Además proyecta la impartición de al menos un 20% de cursos en un estilo similar al Curso en Línea Masivo y Abierto o MOOC, por sus siglas en Inglés (Universidad de Costa Rica, 2015).

La NAC es un proyecto para implementar el modelo de Nube privada en la Universidad de Costa Rica. Tuvo sus inicios en el 2012 y se planteó con el objetivo de "apoyar la docencia, investigación, acción social, vida estudiantil y el apoyo administrativo con tecnología de procesamiento y almacenamiento" (Agüero, 2014). Con este proyecto se logró "poner en operación una infraestructura de servicios para uso de la comunidad que sea propio de una nube computacional" (CITIC, s.f.). Con él se busca potenciar un catálogo de servicios capaces de crecer cualitativa y cuantitativamente en el tiempo.

3.1 Revisión de la literatura

En esta apartado se resumen y discuten varias investigaciones que en mayor o menor medida tienen relación con los objetivos de este trabajo. El criterio base de selección es que los trabajos aborden la temática de los LMS y la Computación en la Nube. Se espera que el análisis de los trabajos consultados ayude a formar un mejor panorama del estado de la cuestión. Se inicia con el resumen de cada uno de los trabajos y su principal relación con la presente investigación, luego se agrupan y discuten según ejes temáticos.

El trabajo de He, Qiu, & Zhai (2015) considera las posibles implicaciones de integrar el LMS Moodle a la Nube y su potencial para la informatización de la educación. Los temas que recopilan son de gran relevancia para esta investigación, sin embargo no ahondan en el cómo lograrlos. Los autores Coutinho, Moreira, Paillard, & Maia (2014) mencionan ciertos pasos a seguir para instalar Moodle en la Nube y también señalan que esto conlleva algún tipo de impacto en el rendimiento. Su propuesta es una arquitectura, una metodología de despliegue y una estrategia de elasticidad en la capa de datos en la Nube capaz de asegurar la calidad del servicio (QoS por sus siglas en ingles) y capaz de cumplir con un acuerdo a nivel de servicio (SLA por sus siglas en ingles). No obstante, el despliegue y la evaluación de la propuesta queda como trabajo futuro. Los investigadores Bione & de Souza (2014) integran Moodle con un *cluster* Postgres-XC de servidores físicos. Al final ellos llegan a la conclusión de que esta receta es viable para lograr una capa de datos robusta, escalable, económicamente accesible y

abierta tanto en servidores físicos como en la Nube. En el trabajo de Guo, Shi, & Zhang (2013) se diseña, implementa y evalúa un *cluster* virtualizado de Moodle. Ellos descubren que luego de sobrepasar un determinado volumen de tráfico este *cluster* tiene un mejor rendimiento que cuando se utiliza un único servidor. Además señalan que el uso de los recursos es optimizado por los métodos y funciones que ofrecen el hipervisor y la virtualización. En otro trabajo, Morgado & Schmidt (2012) presentan una herramienta para migrar una instalación de Moodle automáticamente a una plataforma de Nube. La novedad respecto a los trabajos anteriores es que esa plataforma de la Nube utiliza tecnología de Microsoft. Por último Piovesan, Hoff do Amaral, Arenhardt, & Medina (2012) proponen y desarrollan un sistema virtual de aprendizaje (VLE por sus siglas en inglés) que adecua dinámicamente el contenido según la velocidad de conexión de cada estudiante. La adaptación sucede a nivel de la aplicación y según el contexto computacional del usuario.Si bien es algo de gran interés en incorporar, queda fuera del alcance especifico de esta investigación donde se busca un grado de adaptación a nivel de la arquitectura según variaciones de la demanda del servicio que se presta.

Discusión de los trabajos seleccionados según ejes temáticos

A continuación se categorizan los trabajos mencionados anteriormente según cuatro ejes temáticos. Los cuatro ejes son: 1) los sistemas de gestión de Nube utilizados; 2) los sistemas de gestión de aprendizaje utilizados; 3) como se aborda el tema de la elasticidad y 4) como se evalúan las propuestas:

1) **Los sistemas de gestión de Nube utilizados:** Existe una variedad de gestores de Nube en el mercado, tanto propietarios como de código abierto. Se define este eje temático con el objetivo de conocer si existe algún tipo de criterio en común en los trabajos consultados para seleccionar este componente de software. En tres de ellos se utiliza algún tipo de software para la gestión de Nube. En cada uno de esos tres casos es un software diferente. Piovesan et al. (2012) utilizan Eucalyptus. Guo et .al (2013) hacen uso del software de VMWARE como base en su propuesta. Mientras que Morgado & Schmidt (2012) tienen una especificación propia del gestor de Nube Azure de Microsoft.

2) **LMS:** El LMS es otro componente de software que al igual que los gestores de Nube tiene una variedad de alternativas tanto propietarias como de código abierto. A fin de conocer cuál(es) LMS seleccionaron los investigadores para sus propuestas se planteó este eje temático. A diferencia de los gestores de Nube todos los trabajos consultados coinciden en el mismo LMS: Moodle. No se utilizan otras alternativas para este componente de software en ninguna de estas investigaciones.
3) **Elasticidad:** El tema de la elasticidad, su relación con la computación en la Nube y la forma en cómo poder implementarlo es un eje temático de esta investigación, por eso se plantea de la misma forma para la revisión de literatura. Los trabajos se seleccionaron con base en su temática sobre LMS y la computación en la Nube pero ninguno a excepción de uno abordan el tema de la elasticidad. Coutinho et al. (2014) presentan una propuesta elástica en la Nube. Ellos describen los componentes, servicios y la metodología que integran su propuesta haciendo referencia a otros trabajos. Lo hacen de una manera genérica con la finalidad de que puedan ser desplegados en cualquier implementación de la Nube. Sin embargo el despliegue y las pruebas son parte del trabajo futuro sugerido.
4) **Evaluación:** Se define el eje temático de la evaluación para conocer los métodos utilizados en los trabajos consultados. Para evaluar sus propuestas y/o prototipos los investigadores recurren a diferentes herramientas para simular el tráfico web, estresar los prototipos y/o las propuestas y obtener métricas al respecto. Guo, Shi, & Zhang (2013) utilizaron el programa Siege mientras que Bione & de Souza (2014) y Coutinho et al. (2014) utilizaron Jmeter. Piovesan et al. (2012) utilizan también otra herramienta, el módulo de Firebug en el navegador junto con estadísticas internas de la aplicación LMS, para obtener valores sobre la velocidad de carga.

4 Marco teórico

En esta sección se ofrece una recopilación de los conceptos que se han identificado como fundamentales para la investigación, con el objetivo de establecer una base teórica sobre la cual apoyar el desarrollo de los objetivos planteados. La recopilación abarca temas sobre los sistemas informáticos para el aprendizaje, el paradigma de la computación en la Nube y otros aspectos relacionados como por ejemplo la elasticidad, la alta disponibilidad y la tolerancia a fallas.

Se inicia con la definición de un sistema de gestión de aprendizaje (LMS), su relación con el *e-learning* y Moodle como LMS. Un LMS es un paquete de software que se utiliza para administrar uno o más cursos de uno o más alumnos. Un sistema de este tipo esta típicamente basado en web lo que posibilita a los estudiantes el autentificarse, el inscribirse en cursos, el finalizar cursos y el realizar las evaluaciones. (Learning Systems Architecture Lab at Carnegie Mellon, 2003). Para Szabo & Flesher (2002) el LMS es la infraestructura que ofrece y administra el contenido educacional, identifica y evalúa el aprendizaje o los objetivos de formación individuales y organizacionales, registra el progreso hacia el cumplimiento de esos objetivos y recoge y presenta los datos para supervisar el proceso de aprendizaje de la organización como un todo.

Es común encontrar que estos sistemas son nombrados de otras formas, como por ejemplo con el término de entorno virtual de aprendizaje (VLE por sus siglas en inglés) o el de sistema de gestión de contenidos para el aprendizaje (LCMS por sus siglas en inglés). Lo que los caracteriza a todos es la posibilidad de proporcionar *e-learning*. El *E-learning* es el concepto donde se agrupan "todas aquellas metodologías, estrategias o sistemas de aprendizaje que emplean tecnología digital y/o comunicación mediada por ordenadores para producir, transmitir, distribuir y organizar conocimiento entre individuos, comunidades y organizaciones." (Bernardez, 2007)

Un ejemplo de un sistema que ofrece la posibilidad de realizar *e-learning* es Moodle. Este LMS es de código abierto y está diseñado para proporcionar a los educadores, los administradores y los estudiantes con un único sistema robusto, seguro e integrado para crear entornos de aprendizaje personalizados. (Moodle, 2014) Es

desarrollado por el proyecto con el mismo nombre que está dirigido y coordinado por Moodle HQ, una compañía australiana de 30 desarrolladores apoyada financieramente por una red de 60 empresas de servicios de Moodle en todo el mundo. Reporta miles de instalaciones a nivel mundial desde que se lanzó su primera versión en el 2002. (Moodle, 2014)

4.1 Conceptos de la computación en la Nube

Con el fin de introducir la computación en la Nube se presentan las siguientes definiciones, conceptos y terminología. Se inicia con el concepto de la computación en la Nube en sí, el cual cuenta con una variedad de opiniones así como de definiciones. La Comisión Europea (2012) considera por ejemplo que "la computación en nube dispone de una gama de características definitorias (lo que hace difícil dar una definición general". Algunos consideran que el término es sencillamente el resultado de una estrategia de mercadeo que suele ser utilizada con fines publicitarios para modernizar las ofertas existentes dentro de un nuevo envoltorio (Leimeister, Riedl, Böhm, & Krcmar, 2010). No obstante, importantes instituciones como por ejemplo la Unión Europea y el Instituto Nacional de Normas y Tecnología de los Estados Unidos (NIST por sus siglas en inglés) han abordado el tema y han contribuido con sus propias definiciones.

Para la NIST (2011), la computación en la Nube es un modelo que proporciona de forma cómoda un acceso de red bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y liberados rápidamente con una gestión mínima o con poca interacción del proveedor de servicios. Otra definición, quizá más sencilla, es la siguiente: "el almacenamiento, tratamiento y utilización de datos en ordenadores a distancia a los que se tiene acceso a través de Internet" (Comisión Europea, 2012). En la definición de la NIST (2011) el modelo de la computación en la Nube está compuesto por cinco características esenciales, cuatro modelos de implementación y tres tipos de servicios.

Según ISACA (2009) las cinco características de la computación en la Nube con sus respectivas descripciones:

1. **Autoservicio a solicitud**: "El proveedor de la nube debe poder suministrar capacidades de computación, tales como el almacenamiento en servidores y redes, según sea necesario sin requerir interacción humana con cada proveedor de servicios". (ISACA, 2009)
2. **Acceso a redes de banda ancha**: "De acuerdo con el NIST, debe ser posible acceder a la red en la nube desde cualquier lugar y por medio de cualquier dispositivo (por ejemplo, teléfono inteligente, laptop, dispositivos móviles, PDA)". (ISACA, 2009)
3. **Agrupación de recursos**: Los recursos informáticos son agrupados con independencia geográfica para ofrecerlos a una diversidad de clientes utilizando un modelo de usuarios múltiples y diferentes tipos de recursos físicos y virtuales que son asignados y reasignados de manera dinámica según la demanda. (ISACA, 2009)
4. **Rápida elasticidad**: Posibilidad de suministrar las capacidades, para el cliente pueden parecer ilimitadas, de manera rápida y elástica y en ocasiones automatizada para una pronta expansión o contracción. (ISACA, 2009)
5. **Servicio medido**: De forma automática el uso de recursos puede ser monitoreado controlado y optimizado utilizando una capacidad de medición (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). (ISACA, 2009)

El modelo de servicios de la implementación de la Nube permite soluciones en base a tres tipos de servicios sobre los cuales ofrecer los recursos. ISACA (2009) describe estos tres servicios de la siguiente manera:

1. **Infraestructura como un servicio (IaaS):** "Capacidad para configurar procesamiento, almacenamiento, redes y otros recursos de computación fundamentales, ofreciendo al cliente la posibilidad de implementar y ejecutar

software arbitrario, el cual puede incluir sistemas operativos y aplicaciones." (ISACA, 2009)

2. **Plataforma como un servicio (PaaS):** "Capacidad para implementar en la infraestructura de la nube aplicaciones creadas o adquiridas por el cliente que se hayan creado utilizando lenguajes y herramientas de programación que estén respaldados por el proveedor." (ISACA, 2009)
3. **Software como un servicio (SaaS):** "Capacidad para utilizar las aplicaciones del proveedor que se ejecutan en la infraestructura de la nube. Se puede acceder a las aplicaciones desde diferentes dispositivos cliente a través de una interfaz de cliente ligero (thin client), como un explorador web..." (ISACA, 2009)

Un modelo de implementación define el tipo de operación y la disponibilidad de la Nube y sus servicios. La descripción de cada uno de los cuatro modelos de implementación según ISACA (2009) es:

1. **Nube privada**: "Operada únicamente para una organización. Puede ser manejada por la organización o un tercero. Puede existir dentro o fuera de las instalaciones." (ISACA, 2009)
2. **Nube comunitaria**: "Compartida por varias organizaciones. Respalda una comunidad específica que haya compartido su misión o interés. Puede ser manejada por las organizaciones o un tercero. Puede residir dentro o fuera de las instalaciones." (ISACA, 2009)
3. **Nube pública**: "Está disponible para el público en general o un grupo industrial grande. Pertenece a una organización que vende servicios en la nube." (ISACA, 2009)
4. **Nube híbrida**: "Una composición de dos o más nubes (privada, comunitaria o pública) que continúan siendo entidades únicas, pero que están unidas mediante tecnología estandarizada o propietaria que permite la portabilidad de datos y aplicaciones..." (ISACA, 2009)

La aplicación del paradigma de la computación en la Nube tiene una serie de beneficios, pero también conlleva ciertos riesgos que deben ser considerados para luego

poder eliminarlos, reducirlos y/o mitigarlos. La Comisión Europea (2012) identifica los siguientes beneficios:

La propiedad del equipo físico (ordenadores, dispositivos de almacenamiento): El equipo físico es propiedad del proveedor de la computación en nube y no del usuario, que interactúa con ellos a través de la red. (Comisión Europea, 2012)

La optimización de los equipos físicos: La utilización de los equipos físicos está optimizada de manera dinámica a través de una red de ordenadores, de manera que el emplazamiento exacto de los datos, los procesos, la información sobre qué elemento está siendo utilizado y por qué usuario, no tienen en principio que afectar al usuario. (Comisión Europea, 2012)

El traslado de tareas para la optimización de los recursos: "los proveedores en nube suelen trasladar las tareas de sus usuarios (por ejemplo, de un ordenador a otro o de un centro de datos a otro) para optimizar la utilización de los equipos disponibles" (Comisión Europea, 2012)

Trasparencia para los usuarios: una empresa puede utilizar la computación en la Nube exactamente del mismo modo que los consumidores ya utilizan ahora sus cuentas de correo en red porque los equipos físicos a distancia almacenan y tratan los datos y los hacen accesibles, por ejemplo mediante aplicaciones. (Comisión Europea, 2012)

Acceso ubicuo: "las organizaciones y los particulares pueden acceder a sus contenidos, y utilizar sus programas informáticos cuando y donde lo necesiten, por ejemplo en ordenadores de sobremesa, ordenadores portátiles, ordenadores pizarra (tabletas) y teléfonos inteligentes" (Comisión Europea, 2012)

Ampliación del modelo de negocios: la disposición de un modelo de la Nube consta de capas; la normalización es importante, especialmente en la capa donde hay soportes intermedios o plataformas y programas informáticos de aplicaciones, porque amplia la variedad de clientes potenciales para los creadores

y la variedad de posibilidades para los usuarios sobre las cuales elegir. (Comisión Europea, 2012)

Modelo de pago por uso: "los usuarios normalmente pagan por uso, evitando tener que sufragar los elevados costes iniciales y fijos necesarios para organizar y explotar unos equipos informáticos complejos" (Comisión Europea, 2012)

Rápida elasticidad: "los usuarios pueden modificar muy fácilmente la cantidad de equipos físicos que utilizan (por ejemplo, es posible introducir nueva capacidad de almacenamiento en línea en cuestión de segundos con unas cuantas pulsaciones de ratón)." (Comisión Europea, 2012)

Otros beneficios que señala ISACA (2012) sobre la computación en la Nube son:

Agilidad: "las empresas se mueven a la velocidad de Internet." (ISACA, 2012)

Contención de costos: "la computación en la nube ofrece a las empresas la posibilidad de utilizar recursos financieros con máxima eficiencia." (ISACA, 2012)

Arquitectura común multiempresa (multitenancy): "la naturaleza compartida de la computación en la nube distribuye los costos y las capacidades entre una mayor cantidad de usuarios." (ISACA, 2012)

Confiabilidad: Las características de la computación de la Nube, incluyendo el acceso a un conjunto de recursos compartidos por medio de una red de banda ancha y la rápida flexibilidad según las necesidades, hacen más confiables las soluciones en la Nube. (ISACA, 2012)

Escalabilidad: "la computación en la nube tiene el potencial para atender las necesidades de la pequeña y mediana empresa, de los gobiernos y de las corporaciones Fortune 100." (ISACA, 2012)

Por otro lado están los riesgos de la computación en la Nube. Es importante también identificar los puntos de tensión donde existe una diferencia entre la computación en la nube y los servicios de TI que se suministran a nivel interno o

mediante acuerdos de externalización (ISACA, 2012). La causa subyacente de muchos de los riesgos y desafíos asociados a la computación en la Nube es que los usuarios traspasan la responsabilidad de los datos y la aplicación al proveedor de la Nube, quién tiene un entorno multi-usuario en donde los recursos son compartidos (ETSI, 2013). Dentro de los riesgos que ETSI (2013) menciona están:

- La no disponibilidad, la no integridad y la pérdida del control de los servicios y/o los datos
- La ausencia de mecanismos para la clasificación de los datos
- Las preocupaciones de confidencialidad
- El incumplimiento de la normativa
- El repudio y la falta de capacidad forense
- La ambigüedad de la responsabilidad
- La falta de responsabilidad de los proveedores en caso de incidentes de seguridad
- El costo y la dificultad de la migración a la nube
- La dependencia de un proveedor.

Existen diferentes actores implicados en las soluciones de la Nube desde la perspectiva de un negocio. Iyer (2011) señala que las funciones y las responsabilidades de cada uno así como las relaciones entre ellos son definidas por medio de la asignación de roles a las partes implicadas. Los roles en un servicio de la nube son para este autor los siguientes: usuario, proveedor, socio y autoridad gubernamental. En tanto que las partes son individuos y/u organizaciones que ocupan uno o varios roles. El autor también indica que algunos roles se pueden identificar a partir de la perspectiva de la implementación y la operación de la Nube, como por ejemplo los siguientes: el administrador, el usuario, el gestor y el administrador del equipo. Los roles también pueden tener sub-roles, como lo sugiere ETSI (2013).

Los conceptos que seguidamente se incorporan, sirven para definir las características de una solución en la Nube e introducir algunos procesos y técnicas relacionados. De esta manera se amplia sobre los temas de la *escalabilidad* y la

elasticidad, las *unidades de escalabilidad* y *los contextos de elasticidad*. Además se seleccionan y describen los procesos de aprovisionamiento, asignación y despliegue de recursos así como las técnicas de *clustering* y de balanceo de carga.

4.2 Escalabilidad

La *escalabilidad* es un atributo deseable de una red, sistema o proceso. El concepto denota la capacidad de un sistema para dar cabida a un número creciente de elementos u objetos, para procesar crecientes volúmenes de trabajo correctamente o para ser susceptible a la ampliación (Bondi, 2000).

Escalabilidad vertical o *scaling up*: requiere volver a desplegar la solución utilizando un hardware diferente. En un entorno de la Nube la plataforma de hardware es típicamente un entorno virtualizado donde escalar verticalmente implica la provisión de recursos más potentes para este entorno y mover el sistema a estos nuevos recursos. La escalabilidad vertical es a menudo un proceso disruptivo que requiere poner el sistema temporalmente como no disponible mientras se reasigna. Es posible mantener el sistema original funcionando mientras se aprovisiona el nuevo hardware y se pone en línea pero es probable que haya alguna interrupción durante las transiciones de procesamiento del ambiente viejo al nuevo. Es común utilizar la **auto escalabilidad** para implementar una estrategia de escalamiento vertical. (microsoft.com, 2014)

Escalabilidad horizontal: o *scaling out*: requiere desplegar el sistema con recursos adicionales. El sistema puede seguir funcionando sin interrupción mientras que estos recursos son provisionados. Cuando termine el proceso de aprovisionamiento las copias de los elementos que componen el sistema se despliegan en estos recursos adicionales y se ponen como disponibles. Si la demanda disminuye, los recursos adicionales son recuperados después de que los elementos que los hayan utilizado hayan sido apagados correctamente. Muchos sistemas basados en la Nube, incluyendo Microsoft Azure, apoyan esta forma de auto escalabilidad. (microsoft.com, 2014)

4.3 Elasticidad

La *elasticidad* es el grado en que un sistema es capaz de adaptarse a los cambios en la carga de trabajo aumentando o reduciendo los recursos de forma autónoma, de tal manera que en cada momento los recursos disponibles coincidan lo más cercano posible con la demanda existente (Herbst, Kounev, & Reussner, 2012).

El servicio aumenta o disminuye sus capacidades según la demanda de los consumidores a la velocidad de una plena automatización (son segundos para algunos servicios y horas para otros). La elasticidad es un rasgo de un conjunto de recursos compartidos mientras que la escalabilidad es una característica de las plataformas de infraestructura y software subyacentes. Elasticidad se asocia no sólo a escalar, sino también a un modelo económico que posibilita el escalar en ambas direcciones en forma automatizada. Esto significa que los servicios escalan bajo demanda para agregar o quitar recursos según sea necesario (Pettey & Goasduff, 2009).

En otras palabras la elasticidad es la característica que tiene un servicio soportado en un conjunto de recursos computacionales para adaptarse dinámicamente en algún grado, aumentando o disminuyendo automáticamente el número o la capacidad de esos recursos para satisfacer la demanda según criterios preestablecidos.

Un *contexto de escalabilidad* es un conjunto de elementos o recursos a escalar según un criterio basado en la función compartida por cada uno de ellos. Identificar los contextos de escalabilidad ayuda a responder la pregunta de qué componentes deben ser capaces de incrementarse o reducirse automáticamente y en qué forma para así obtener un grado de elasticidad.

En un evento de auto escalabilidad se busca agregar o eliminar automáticamente una cantidad de CPU, memoria RAM, almacenamiento, redes para obtener una capacidad adecuada según la demanda. No obstante esos recursos no son gestionados individualmente, se encuentran agrupados en una entidad que es un servidor o una instancia. Entonces se puede establecer una unidad diferenciada para medir cuanta capacidad se está agregando o eliminando durante los eventos de auto escalamiento.

Debe ser definida de acuerdo al criterio de elasticidad considerado para la arquitectura y los servicios que ofrece. En otras palabras, la *unidad de escalamiento* es el tamaño de la instancia que se utilizará como estándar a la hora de agregar servidores o eliminarlos para obtener el grado de elasticidad estimado.

Los siguientes términos ayudan a comprender ciertos procesos o tareas comunes en una solución para la Nube:

Aprovisionamiento de los recursos: Es el proceso que ofrece al cliente la posibilidad de asignar los recursos descubiertos y seleccionados. (Siddiqui & Fahringer, 2010).

Asignación de un recurso: Ante la solicitud de un recurso, se asigna una porción de la capacidad de un nodo por medio de una función para que las restricciones para las solicitudes coincidan con las condiciones ofrecidas (Siddiqui & Fahringer, 2010).

Despliegue de la arquitectura: Es la instalación y configuración de la arquitectura lógica a un entorno físico. El entorno incluye los nodos de computación en una ambiente de red, CPU, memoria, dispositivos de almacenamiento y otros componentes presentes en el diseño de la arquitectura. (Sun Microsystems, Inc., 2004)

Clustering: Consiste en crear un grupo de computadores completos interconectados, trabajando al mismo tiempo como un recurso unificado que se puede identificar como una única máquina (Stallings, 2006).

Balanceo de carga: Es una técnica para distribuir las cargas de trabajo entre un grupo de recursos según una o varias reglas con alguna finalidad como la obtención de un mejor rendimiento o la optimización de los recursos. También sirve para dar una única identidad a un *cluster*.

La *alta disponibilidad* es otro concepto relacionado que se puede considerar en una solución para la Nube. Weygant, (Clusters for High Availability: A Primer of HP

Solutions, 2001), amplia los términos relacionados con este concepto con el objetivo de introducir las implicaciones del mismo. Lo hace de la siguiente manera:

> **Disponibilidad**: Esta característica en un sistema describe un requerimiento a nivel de servicio. En el ámbito de la computación generalmente se entiende como el periodo de tiempo que un servicio está disponible. En otras palabra, el tiempo que se requiere que el sistema responda a los usuarios. Si ese tiempo se interrumpe, de forma planeada o no, se considera como un corte del servicio. El tiempo de interrupción es la duración del corte medida en una unidad de tiempo. (Weygant, 2001)
>
> **Alta disponibilidad**: caracteriza a un sistema que está diseñado para evadir la pérdida del servicio reduciendo o gestionando las fallas así como minimizando las interrupciones planeadas. Se espera que un servicio sea altamente disponible cuando una vida, la salud o el bienestar, incluido el de una organización, dependan de él. (Weygant, 2001)
>
> **Computación con alta disponibilidad**: Sistemas computacionales diseñados y gestionados para operar con solo una pequeña cantidad de interrupciones planeadas y no planeadas. (Weygant, 2001)

Otro concepto relacionado es *la tolerancia a fallas;* es más un medio para alcanzar niveles muy altos de disponibilidad que un grado de disponibilidad. Un sistema con esta característica la habilidad de continuar ofreciendo el servicio a pesar de una falla de *hardware* o de *software*. Para ello incorpora la redundancia en la mayoría de sus componentes de hardware, incluido el CPU, la memoria, los subsistemas de E/S así como otros elementos. Sin embargo, este tipo de sistemas siguen estando sujetos a los errores humanos. (Weygant, 2001)

Un término importante sobre la tolerancia a fallas es el *punto único de fallo*, SPOF por sus siglas en inglés. El término se refiere a un componente de un sistema que, al experimentar un fallo, deja el sistema completo no disponible o inestable. Al diseñar un sistema, estos deben ser identificados y luego se debe investigar como mitigarlos.

Dos técnicas para mitigarlos son el uso de la redundancia y el *clustering*. (Oracle Corp, 2010)

4.4 Gestores de la Nube y sus servicios

Una componente importante en una implementación de la Nube es la Plataforma de Gestión de la Nube (CMP por sus siglas en inglés). En ella se definen la mayoría de servicios ofrecidos a nivel de gestión. Algunos de esos servicios son característicos de la computación en la Nube y se encuentran disponibles en las diferentes plataformas de gestión mientras que otros son específicos de cada una.

Un CMP tiene productos integrados que permiten la gestión de entornos públicos, privados e híbridos de la Nube. Los requisitos mínimos para ser incluidos en esta categoría son: incorporar interfaces de autoservicio, aprovisionar imágenes de sistema, habilita la medición y la facturación y proporciona un cierto grado de optimización de la carga de trabajo a través de las políticas establecidas. Las ofertas más avanzados también se integran con los sistemas de gestión empresarial externos, incluyen catálogos de servicios, apoyan la configuración de los recursos de almacenamiento y de red, ofrecen una gestión de recursos mejorada a través de los reguladores de servicio y proporcionan un monitoreo avanzado para un mejor rendimiento y disponibilidad de los "huéspedes" (Gartner, 2013).

OpenStack es un CMP que controla grandes conjuntos de recursos computacionales, cómputo, almacenamiento y redes, dentro de un centro de datos, todos administrados a través de un panel que proporciona control a los administradores al tiempo que le da poder a sus usuarios para provisionar recursos a través de una interfaz web. Además integra otros servicios compartidos adicionales que pueden interactuar entre ellos y con sistemas externos como por ejemplo los servicios de Identidad, Imagen, Telemetría, Orquestación y Bases de Datos. Es un proyecto de código abierto bajo licencia Apache 2.0 (openstack.org, 2014).

La implementación de la Nube cuenta con una variedad de servicios a nivel de gestión sobre los cuales ofrece su modelo de servicios IaaS, PaaS y SaaS. Cualquier

despliegue en la Nube hace uso de algunos de ellos, pero es importante conocerlos para identificar cuáles son necesarios en una implementación de una arquitectura dinámica. Los siguientes son servicios de la plataforma de la Nube a considerar para construir la arquitectura con un grado de elasticidad. Cada uno cuenta con una interfaz de programación de aplicaciones (API por sus siglas en ingles) y otras interfaces web. Los siguientes son algunos de los principales:

Computo: Gestión y aprovisionamiento por demanda de recursos computacionales en grandes redes de máquinas virtuales. (openstack.org, 2014)

Almacenamiento: Almacenamiento en bloque o por objetos para el uso en servidores o aplicaciones. (openstack.org, 2014)

Redes: Sistema de administración de redes y direcciones IP que también es escalable y compatible. Puede ofrecer redes definidas por software (SDN por sus siglas en ingles). En el caso particular de OpenStack, tiene disponibles dos implementaciones del servicio de red: Nova Network y OpenStack Networking (Neutron), la segunda ofrece completamente SDN (openstack.org, 2014).

Identidad: Sistema de autenticación común que puede ser integrado con otros servicios de directorio como LDAP. El sistema mantiene un directorio central de usuarios asociado a los servicios a los que se les da acceso (openstack.org, 2014).

Imagen: Proporciona servicios de descubrimiento, registro y entrega para imágenes de servidores y discos. Ofrece crear copiar de instantáneas del estado de un servidor y la opción de almacenarlas en otro lugar para luego ser utilizadas como respaldos o como plantillas (openstack.org, 2014). También es posible crear imágenes personalizadas en diferentes formatos para luego subirlas y utilizarlas con el servicio de imágenes.

Telemetría: Agrega datos de uso y rendimiento en todos los servicios desplegados. Esta poderosa función proporciona una visión detallada del uso de la Nube entre decenas de datos y ofrece a los administradores la posibilidad de

consultar métricas de forma global o de forma individual por cada recurso desplegado (openstack.org, 2014).

Orquestación: Motor basado en plantillas que permite a los desarrolladores de aplicaciones describir y automatizar el despliegue de la infraestructura. Este lenguaje de plantillas es flexible y ofrece la posibilidad de especificar configuraciones de cómputo, almacenamiento y redes así como un obtener un detalle de la actividad posterior al despliegue para utilizarlo en la automatización del completo aprovisionamiento de la infraestructura, servicios y aplicaciones. Se integra con el servicio de Telemetría para realizar un auto escalamiento de ciertos elementos de la infraestructura (openstack.org, 2014).

Metadatos: Acceso de red a datos sobre los recursos. Estos datos facilitan la configuración y/o gestión dinámica de los recursos (Amazon, 2015).

5 Diseño propuesto para la arquitectura de la Nube

En este capítulo se detalla la construcción del diseño iniciando con la arquitectura base, la definición de los elementos de alta disponibilidad como por ejemplo el uso de zonas de disponibilidad o la afinidad de las instancias en diferentes hipervisores así como el uso de capas. Luego se pasa al tema de la continuidad del servicio y la tolerancia a fallas donde se aborda el tema de la identificación de puntos únicos de fallos y las consideraciones para eliminarlos o mitigarlos desde el diseño de la solución. Posteriormente se desarrolla el tema de la elasticidad y la definición de contextos donde se produce esta característica. Por último se hace referencia a todos los otros componentes del diseño y los conceptos por los cuales se incorporan.

El sistema que se va a diseñar es una plataforma elástica para un LMS, con elementos de alta disponibilidad y de tolerancia a fallas. Está basado en Moodle, el cual requiere un servicio web que interprete el lenguaje PHP, un servicio de base de datos SQL y un servicio de almacenamiento para su código fuente y para los datos que genera tal y como se observa en la Figura 4. Cada uno de estos servicios tiene una serie de requerimientos y consideraciones en cuanto a rendimiento, seguridad e integración con otros sistemas y servicios. El servicio web utiliza un *cluster* de Apache; el de base de datos, un *cluster* de Postgres; mientras que los datos de la aplicación se almacenan en un punto compartido y conectado en red.

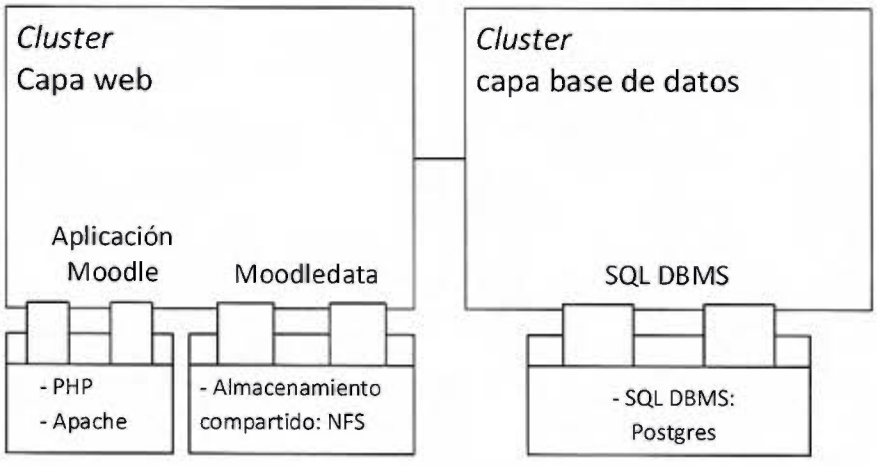


Figura 4. Diagrama de la arquitectura base a diseñar

5.1 Alta disponibilidad y tolerancia a fallas

Los principales componentes deben estar diseñados para garantizar la continuidad del servicio. La implementación de la Nube puede ofrecer redes, datos, procesamiento, almacenamiento y energía redundante lo cual ciertamente ayuda a este objetivo. Sin embargo no todos los usuarios tienen acceso a estos detalles de la implementación de la Nube provista. Es factible entonces incorporar estos conceptos en la misma arquitectura a desplegar sobre la Nube. Para ello se definen las zonas y las capas necesarias para proveer alta disponibilidad

El proveedor puede ofrecer una separación de los recursos que va desde ubicar las instancias en diferentes hipervisores a ubicar los componentes en diferentes zonas o regiones geográficamente distintas, diseñadas para seguir funcionando independientemente entre ellas. A lo primero se le conoce como afinidad para los servidores mientras que a lo segundo como zonas de disponibilidad y regiones. La propuesta para este diseño es utilizar dos zonas de disponibilidad diferentes para separar los componentes. En la Figura 5 se observan estas zonas: la zona de disponibilidad A y la zona de disponibilidad B, cada zona demarcada por líneas punteadas de color negro.

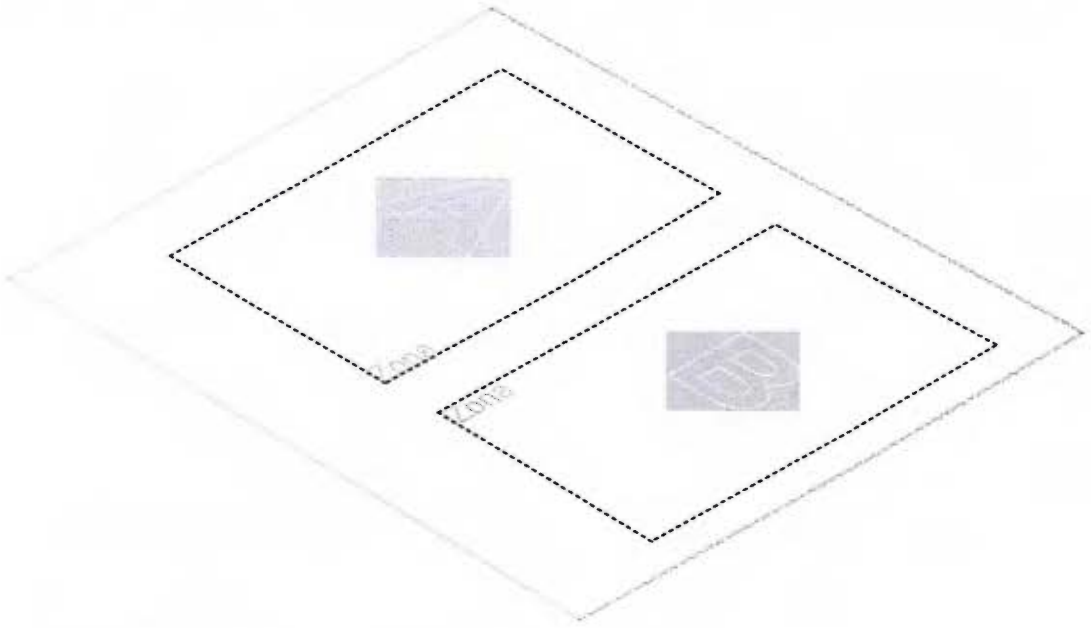


Figura 5. Zonas de disponibilidad en el diseño inicial de la solución

Los componentes pueden ser agrupados en capas independientemente de si están en zonas de disponibilidad distintas. La definición de capas ofrece la posibilidad de separar, agrupar e identificar los componentes según funciones primordiales requeridas por la arquitectura. Es una estrategia que facilita la escalabilidad ya que permite monitorear, mantener y configurar los recursos de forma separada para agregar más capacidad a la capa que lo necesite. También ofrece la posibilidad de realizar configuraciones según la función que cumple cada capa e integrar conceptos de alta disponibilidad y tolerancia a fallas de acuerdo a lo naturaleza de esa función compartida. Algunos ejemplos de capas son: base de datos, aplicación, servidores web, almacenamiento. En el caso de la propuesta se mantienen las siguientes capas: 1) web; 2) base de datos. En la Figura 6 se observan estas dos capas delimitas cada una por líneas punteadas.

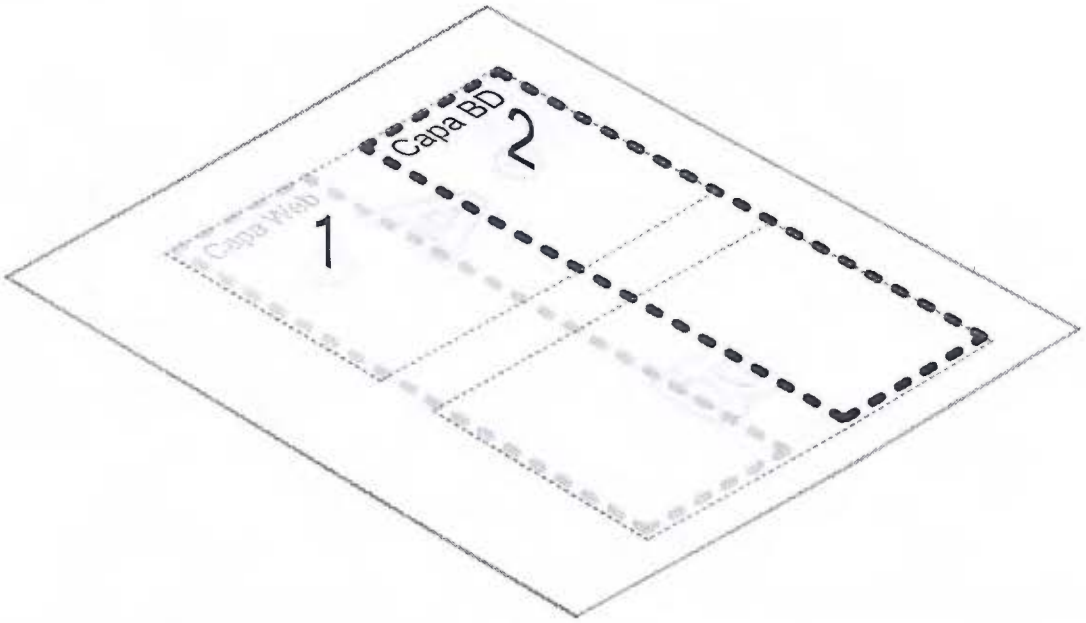


Figura 6. Capas en el diseño inicial de la solución

Para obtener tolerancia a fallas es necesario identificar y eliminar los puntos donde una falla haría que el servicio se detenga. En la Figura 7 se identifican estos puntos por medio de cruces rojas. El primer punto (1) corresponde a la conectividad entre las capas y otros componentes. Se asume que las redes (conectividad) es un servicio provisto por la Nube y estos cuentan con mecanismos para tolerancia de fallas

como "*Bonding*" en el caso de las redes y RAID y *clusters* en el caso del almacenamiento. Los otros dos puntos corresponden a las funciones provistas por cada una de las dos capas establecidas: el servicio web (2) y la base de datos (3). Para eliminar estos dos puntos se deben implementar estrategias de tolerancia a fallas según los elementos desplegados para proveer el servicio de cada capa. El último punto (4) corresponde al almacenamiento de Moodledata, un requerimiento específico del LMS Moodle. Por si mismo represente un punto único de falla. Al igual que la conectividad, se asume que el almacenamiento de Moodledata es un servicio de almacenamiento en la Nube con redundancia.

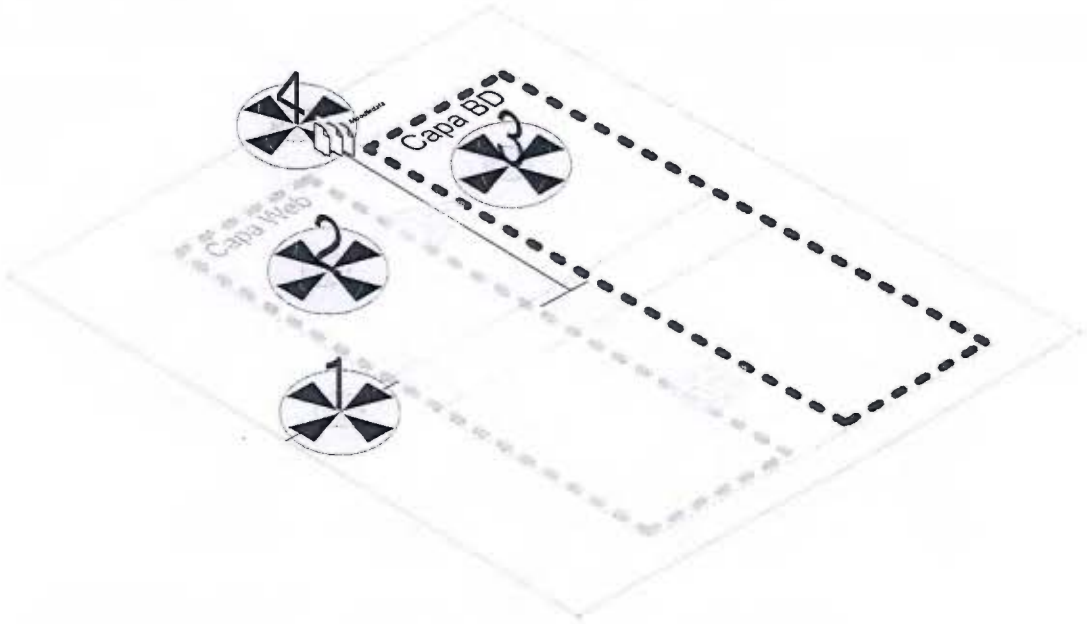


Figura 7. Puntos únicos de fallos identificados en el diseño inicial de la solución

5.2 Contextos de elasticidad

Se proponen dos contextos donde llevar a cabo la auto escalabilidad del LMS: 1) en la capa web y 2) en la capa de base de datos. En la Figura 8 se observa como estos dos contextos corresponden a las dos capas definidas. Con esto se busca definir estrategias de auto escalabilidad diferenciadas para cada contexto tomando en consideración el tráfico a soportar y sus características así como la naturaleza del servicio ofrecido por la capa y los elementos que la componen.

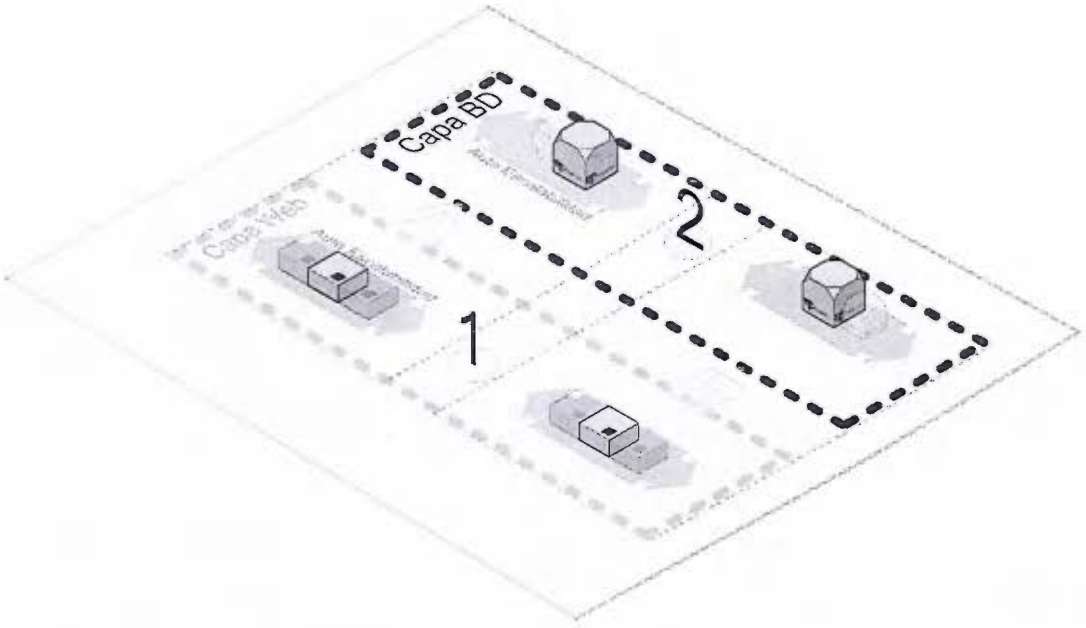


Figura 8. Contextos de escalabilidad en el diseño inicial de la solución

Además de considerar los servicios a los que se debe conectar el LMS, también se pueden integrar otros componentes buscando obtener una mejora en el rendimiento, balancear la carga e incrementar la seguridad. Estos componentes son: 1) un caché compartido para los servidores de la capa web; 2) un servicio de balanceo para cada capa que ayude a la escalabilidad, la gestión del tráfico y la alta disponibilidad; 3) grupos de seguridad para controlar el tráfico TCP desde, hacia y entre las capas; y 4) otros servicios o sistemas relacionados. La Figura 9 contiene estos componentes y servicios e ilustra la integración con los conceptos anteriores dentro del diseño.

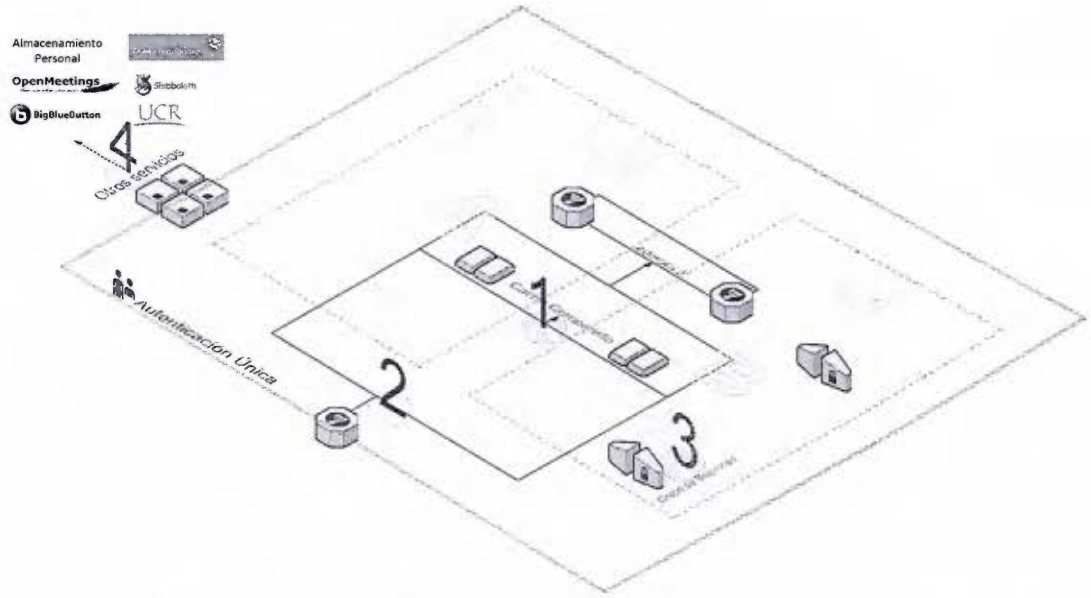


Figura 9. Componentes adicionales para seguridad, rendimiento e interconexión en el diseño inicial de la solución

Para integrar todas las consideraciones hechas anteriormente se presenta la Figura 10. En ella se resume el diseño de la solución propuesta con todos sus componentes. Se pueden observar las capas y los componentes necesarios para proveer alta disponibilidad; los componentes que eliminan los puntos donde una falla haría que el servicio se detenga; y las capas donde se debe dar la auto escalabilidad.

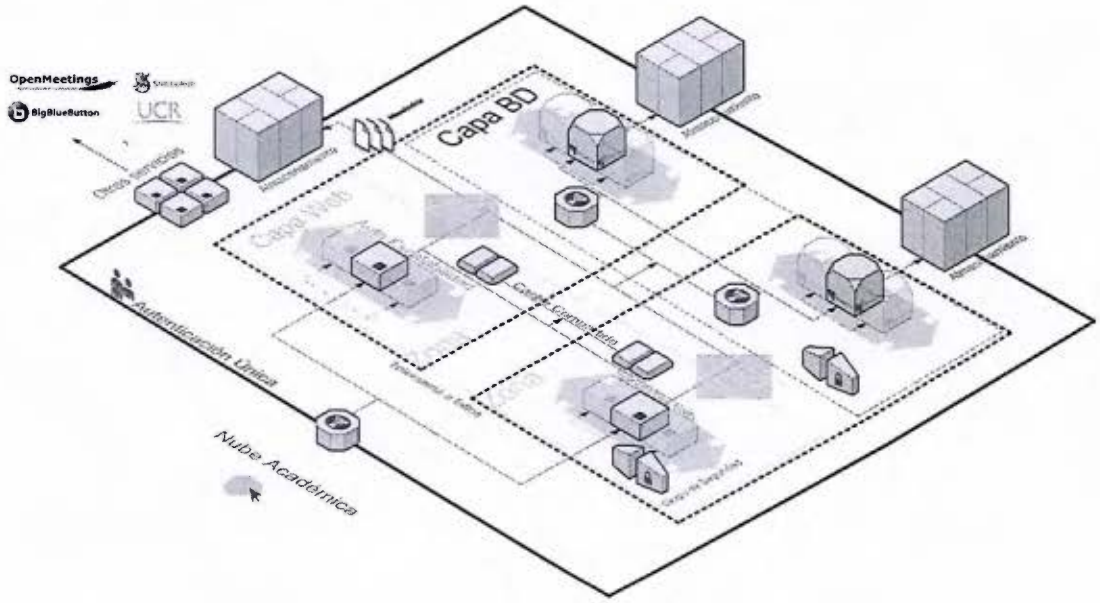


Figura 10. Diagrama del diseño inicial de la solución.

6 Implementación del prototipo elástico

En este apartado se detalla la implementación del prototipo a partir del diseño propuesto anteriormente. Para ello primero se describe el entorno de Nube utilizado junto con sus particularidades de configuración. Este ámbito contiene al CMP y a los servicios necesarios para soportar el procesamiento, la memoria, el almacenamiento, la interconexión y la auto escalabilidad del prototipo. Posteriormente se especifica la implementación del prototipo en sí, los recursos que lo componen y los elementos de configuración que utiliza. De esta manera se incluyen los temas sobre la creación y utilización de imágenes personalizadas y la programación de la arquitectura por medio de plantillas de orquestación.

Antes de poder desplegar un prototipo elástico es necesario contar con un entorno de Nube básico que satisfaga ciertos requerimientos sobre el CMP, el subconjunto de servicios disponibles y la configuración de cada uno de ellos. En el caso específico de esta investigación, en el entorno de Nube se utiliza el CMP *OpenStack* en su versión *Kilo* junto con un subconjunto de sus servicios principales. La configuración de algunos de esos servicios depende de otros componentes que también constituyen el entorno de Nube. Por ejemplo uno de los principales componentes es el o los hipervisores para el servicio de procesamiento. OpenStack soporta una variedad de hipervisores, tanto propietarios como de código abierto. En este trabajo se utiliza el hipervisor de VMWare ESX en su versión 5.5. Vale mencionar que el prototipo podría funcionar también en el hipervisor de código abierto KVM. Para conocer más en detalle los requerimientos y el tipo de configuraciones del CMP y el hipervisor véase el Anexo 1.

6.1 Plataforma y servicios de la Nube

El servicio de procesamiento es uno de los principales servicios del modelo de la Nube. Existe una variedad de servicios adicionales, como por ejemplo el almacenamiento y las redes, sobre los cuales se pueden construir soluciones. El CMP ofrece un catálogo con su propia implantación de esos servicios y de otros propios. Los

servicios pueden ser habilitados, configurados y personalizados. En el caso de esta investigación solo se habilitaron en el entorno aquellos servicios que son indispensables para el prototipo elástico. Esos servicios seleccionados se listan en el Cuadro 1 junto con el nombre respectivo según la implementación del CMP OpenStack. Varios de ellos requieren configuraciones específicas que dependen de otros componentes del entorno de la Nube, por ejemplo el servicio de procesamiento necesita un controlador específico según el hipervisor a configurar. En el Anexo 2 se puede encontrar un mayor detalle de las particularidades de configuración realizadas al CMP según el entorno de la Nube para esta investigación.

Cuadro 1. Servicios necesarios de la Nube

TIPO DE SERVICIO	SERVICIO DE OPENSTACK
Computo	Nova Compute
Almacenamiento	Cinder
Redes	Nova Network
Imagen	Glance Image
Telemetría	Ceilometer
Orquestación	Heat
Identidad	Keystone
Metadatos	Metadata API

Un caso particular dentro de los servicios seleccionados es el servicio de red. OpenStack ofrece dos implementaciones de este servicio: Nova-Network y Nova-networking (Neutron). En este entorno se utiliza la primera implementación: Nova-Network. El motivo inicial ha sido la facilidad para integrarlo al utiliza el hipervisor ESX de VMWare.

Usar Nova-network sobre Nova-networking tiene ciertas desventajas y también algunas ventajas. Entre las desventajas esta que ha sido señalado como un servicio heredado y también que no cuenta con *Load-Balancing-as-a-Service* (LBaaS), haciendo necesario implementar el balanceo como parte de la arquitectura a desplegar. Lo cual

funciona como una desventaja y una ventaja a la vez: desventaja por la complejidad adicional al prototipo y ventaja por la posibilidad de obtener un mayor control y personalización en la implementación del servicio de balanceo. Otra ventaja de Nova-network es que tiene una menor sobrecarga en el tráfico y por ende un menor impacto al rendimiento de red. También facilita el uso de IP virtuales gestionadas por las mismas instancias. Algo necesario en ambas capas del prototipo. Sin embargo esto también podría ser posible con Nova-networking utilizando "allowed-address-pairs" (Rosen, 2014) o usando IP flotantes y acceso al API desde las máquinas virtuales para hacer el cambio en caso de un fallo.

Los componentes que completan el entorno son las imágenes personalizadas y las plantillas de orquestación. El Anexo 3 contiene un detalle técnico de las herramientas necesarias para usar estos dos componentes. El uso de imágenes del sistema previamente personalizadas requiere a menudo un mayor uso de almacenamiento en el servicio Glance pero da la posibilidad de acortar el tiempo de despliegue, algo que resulta importante durante un evento de auto escalamiento. Las plantillas por otro lado se encargan de los ciclos de vida de la arquitectura y la automatización de la configuración durante cada ciclo.

Ambos componentes, las imágenes personalizadas y las plantillas de orquestación, se utilizan para aplicar configuraciones de los servidores de la arquitectura, ofreciendo ambas la opción de automatizar esas configuraciones. Sin embargo cada componente aplica esas configuraciones en momentos diferentes. Las imágenes personalizadas permiten tener una configuración antes de instancias un servidor. Las plantillas pueden aplicar la configuración luego de instanciado el servidor. Por configuración se refiere tanto a la instalación como a la configuración de software específico y/o los sistemas subyacentes como el sistema operativo. Para sacar el mejor provecho de la Nube es importante tomar en consideración las implicaciones de programar las configuraciones en alguna de estas dos fases por medio de alguno de estos dos componentes: plantillas de orquestación e imágenes personalizadas.

La configuración previa al despliegue puede realizarse utilizando *snapshots* o bien utilizando imágenes personalizadas. Para obtener un *snapshot* se instancia una máquina virtual, se realizan las configuraciones necesarias y se guarda el *snapshot*. Luego se utiliza para iniciar una instancia desde ella. En tanto las imágenes personalizas se crean a través de herramientas externas a la Nube y también a partir de *snapshots*. Una de las ventajas de usar una herramienta para la creación de imágenes personalizadas son las funciones que ofrece para automatizar todo el proceso de creación y para reutilizar los elementos disponibles según la herramienta utilizada.

Se pueden incluir en esta fase de creación de la imagen toda o parte de la configuración de cada instancia. Esto afectara el tamaño de la imagen y la flexibilidad para reutilizarla, probablemente haciendo mayor el número de imágenes requeridas por la arquitectura. La mayoría de imágenes para la Nube buscan ser lo más compactas posibles debido al modelo de cobro de servicios utilizado. Lo que significa que no cuentan con mucha configuración. Un argumento para aumentar la configuración en esta fase es la disminución de la configuración requerida durante el despliegue y por ende del tiempo para incorporar un nuevo recurso dentro de una pila.

La configuración posterior al despliegue puede automatizarse por medio de la programación de plantillas de Heat y las herramientas de automatización. También puede hacerse de forma manual pero es ineficiente, no se puede reutilizar y se pierde al finalizar la instancia. Es mucho mejor automatizar toda o la mayor parte de la configuración necesaria. Heat se puede integrar con varias herramientas de automatización. Algunas de ellas tienen la capacidad de utilizarse en entornos de miles de servidores y por ende se necesita instalar toda otra arquitectura para ellas. Esto se puede convertir en un requerimiento adicional para el prototipo. Utilizar los scripts de Shell parece una mejor opción para ajustar las dependencias y obtener un prototipo compatible con un número mayor de entornos. Para que Heat inyecte la configuración programada se requiere que la imagen tenga instaladas previamente ciertas herramientas, como por ejemplo: **Heat-config** y **CloudInit**. Dependiendo de la cantidad de configuración a realizar en esta fase, el tiempo de despliegue, la descarga de datos y el

uso de la red pueden verse afectados. No obstante una de las ventajas de automatizar la configuración durante el despliegue es la posibilidad de utilizar el servicio de metadata y los recursos de Heat, entre ellos las imágenes personalizadas, para realizar configuraciones dinámicas.

6.2 Imágenes personalizadas

Las imágenes son copias de un disco de arranque con programas previamente instalados y listos para ser instanciados en un servidor de la Nube. Las imágenes personalizas tienen instalaciones y configuraciones personalizadas dentro de esa copia para iniciar las instancias y reducir el tiempo de auto escalamiento. Se utiliza la herramienta *Disk Image Builder* para programar los elementos que componen las imágenes personalizadas. La programación es algo así como una receta: los elementos tienen fases donde se incluirán los scripts o ingredientes, luego en el proceso de creación se escoge qué elementos adicionales incluir o se dejan solo las dependencias previamente establecidas. Para el prototipo se hace uso de dos imágenes personalizadas: 1) una imagen base con configuraciones comunes de las instancias de la arquitectura; 2) y otra imagen basada en la anterior con configuraciones adicionales para utilizar en el contexto de elasticidad. Las instalaciones y configuración en la imagen base son por ejemplo la distribución del Sistema Operativo, los agentes para HEAT, las herramientas en común, el usuario por defecto, el envío de correos de las tarea administrativas, la configuración del servicio de hora, de la ubicación regional, de la zona horaria, y de los espejos por defecto, entre otros. En el caso de la segunda imagen personalizada, se usa en la auto escalabilidad según un contexto de elasticidad para disminuir el tiempo de preparación de las instancias añadidas durante los eventos de auto escalabilidad. Esta segunda imagen tiene preparado por ejemplo el repositorio de la aplicación, las herramientas de sincronización y todos los paquetes específicos del servicio web para que no sea necesario realizar descargas e instalaciones al instanciar desde la imagen. El detalle de que contienen estas dos imágenes personalizadas se pude consultar en el Anexo 4.

6.3 Orquestación

Una plantilla de orquestación es un archivo de texto donde se puede definir la arquitectura por medio de un conjunto de recursos de la Nube (pila). Además permite aprovisionar, desplegar y configurar la pila y gestionarla durante su ciclo de vida: creación, actualización, suspensión, reanudación y eliminación. Las plantillas de orquestación se utilizan entre otras cosas para:

- Definir y utilizar recursos como zonas de disponibilidad, redes, imágenes personalizadas, tipos de máquinas virtuales (sabores), grupos de seguridad, metadata, scripts de configuración y automatización personalizados.
- Realizar el auto escalamiento
    - Grupos de auto escalamiento
    - Políticas de auto escalamiento con su respectivo *webhook*, este último sirve para activar una acción enviando una solicitud a la respectiva URL.
    - Alarmas según métricas de Ceilometer
    - Unidades de escalamiento
- Realizar manualmente el escalamiento vertical.

La arquitectura del prototipo esta descrita por medio de una serie de plantillas en el formato HOT para HEAT, el servicio de orquestación de OpenStack. En la Figura 11 se puede observar de forma global el prototipo implementado, el cual está basado en el diseño del capítulo anterior; incorpora aquellos componentes relacionados directamente con la elasticidad, la tolerancia a fallas y la alta disponibilidad. Otros elementos que no son parte de los objetivos planteados no se integran en el prototipo implementado.

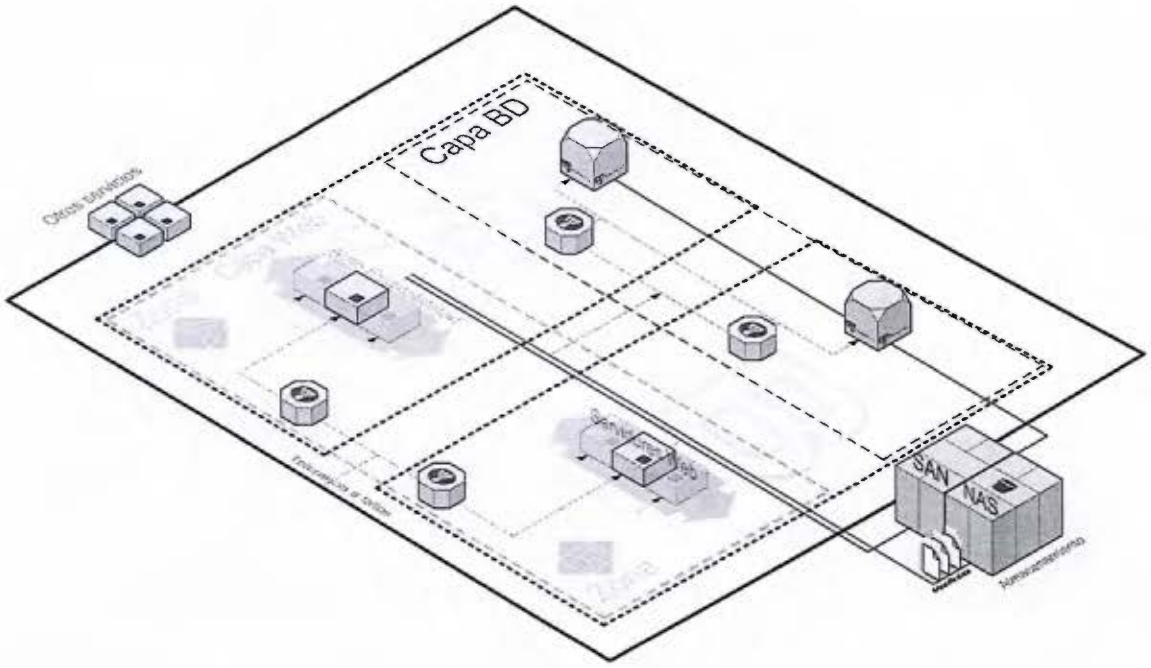


Figura 11. Diagrama del prototipo

6.3.1 Capa web

La capa web es donde se recibe, balancea y procesa el tráfico web y además se produce un auto escalamiento basado en un par de métricas y estadísticas de la aplicación del prototipo. En la Figura 12 se ilustra el detalle programado por las plantillas de orquestación para esta capa. No se utiliza LBaaS sino que esta implementado un servicio en la misma arquitectura utilizando HAProxy y Keepalived. Se incorpora cónsul como sistema de detección de nuevos servicios para ayudar en la configuración dinámica durante los eventos de auto escalamiento. La sincronización del código de la aplicación y los archivos de configuración de los programas de servicios web son sincronizados automáticamente gracias a Csync2 y Lsycnd

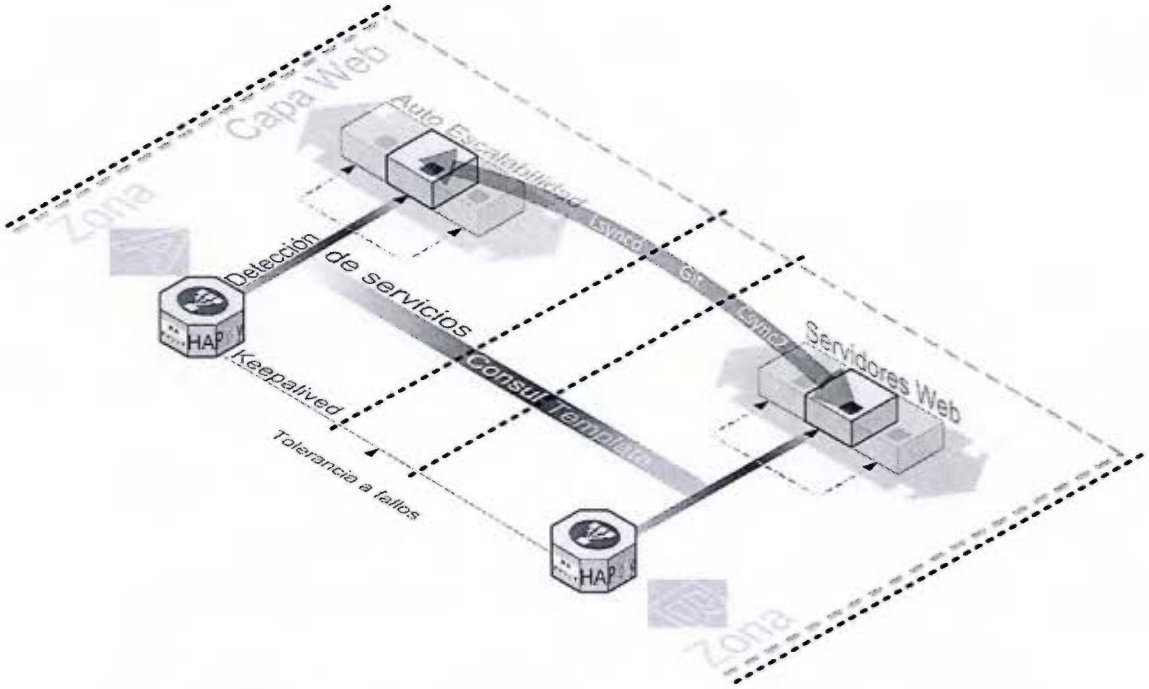


Figura 12. Diagrama de la capa web en el prototipo

Balanceadores

Los balanceadores son un componente de gran importancia para la alta disponibilidad y el auto escalamiento horizontal. Existe un punto de falla único en la capa web. Por eso es necesario incorporar junto con HAProxy el programa Keepalived y ubicar al menos un balanceador y un servidor web en cada una de las dos zonas de disponibilidad. La detección de servicios y la generación dinámica de archivos de configuración con Consul y Consul-Template respectivamente se utilizan para la gestión de los miembros del grupo de balanceadores, algo que ofrece nativamente el LBaaS de OpenStack junto con Heat pero que debía implementarse al no estar disponible ese servicio en Nova-network. Dos ventajas en el prototipo sobre utilizar LBaaS es que 1) se utilizan programas agnósticos a la implementación de la Nube y 2) se gestiona de una manera más considerada la eliminación automática de instancias en el grupo de auto escalamiento.

Eliminar instancias amigablemente

Un problema que se busca solucionar con esta implementación son los errores mostrados al usuario en el momento que se lleva a cabo la elasticidad hacia la baja. Utilizando el mecanismo de auto escalamiento junto con LBaaS se observó que era posible recibir errores en las consultas HTTP durante un evento de este tipo. Una razón podría ser que las instancias son eliminadas de manera abrupta antes de ser notificado o actualizado el servicio de balanceo. No se profundizó sobre la causa, pero al no contar con LBaaS de todos modos y habiendo experimentado los errores, se consideró como parte del problema a solucionar. Con consul, consul-template y la configuración automática durante alguno de los ciclos de vida que ofrece HEAT, HEAT ejecuta el comando de salida en el mismo servidor que está por eliminarse para eliminarlo del grupo de balanceo lo que genera una actualización y una recarga del servicio HAProxy activo. Luego HEAT continúa con la eliminación del servidor.

Auto escalamiento de la capa web

El proceso de auto escalamiento está compuesto por dos estrategias de auto escalamiento, una unidad de escalamiento, varias políticas de auto escalamiento, las alarmas para monitorear las métricas seleccionadas y los *scripts* de automatización ante un evento de este tipo.

Para la primera estrategia se define una (1) unidad de escalamiento. Si el CPU o el uso de memoria del conjunto de los servidores web actuales son mayores a 50% en promedio durante el último minuto para cuando se realiza un sondeo en el servicio de telemetría entonces se agrega la cantidad de unidades definida (1). Si el uso de CPU es menor al 1% en promedio durante los últimos 10 minutos en combinación con un promedio de bytes por segundo menor a 1024 en el tráfico de entrada a los servidores web entonces se elimina una unidad de escalamiento (1). La combinación de estos dos criterios, CPU y tráfico de red, es para asegurar que la reducción se produzca en momentos que hay o ninguna solicitud web (usuarios) y evitar así cualquier afectación. Tanto el agregar como quitar unidades de auto escalamiento se da de forma gradual en esta estrategia. Una posibilidad y un problema es que la capacidad requerida para

satisfacer la demanda sea mayor a la unidad de escalamiento que se está por agregar y está no pueda ser estimada por el uso actual de CPU o memoria RAM.

Auto escalabilidad según estadísticas de HAProxy

La segunda estrategia se establece para atender el problema anterior. Se consulta a la aplicación de balanceo cada cierto tiempo sobre las solicitudes recientes y se agregan tantas unidades de auto escalamiento como se haya estimado necesarias para satisfacerlas. La ventaja está en que se puede agregar una capacidad de recurso computacional mayor en un solo evento de auto escalamiento según el tráfico actual. Está es una estrategia más acorde a lo recomendado por expertos: estirarse pronto y encogerse de forma lenta (Orzell & Becker, 2012).

Por debajo lo que sucede es que se utiliza consul para realizar un chequeo. El chequeo es un *script* que consulta las estadísticas de HAProxy cada cierto tiempo por medio de un intervalo sobre las últimas 1024 solicitudes y el número de miembros activos. Cada uno de ellos representa una unidad de escalamiento según las estimaciones y consideraciones previas. La suma de ellos representa la capacidad actual. El número de solicitudes representa la capacidad por cubrir. Luego se calcula el número de unidades de auto escalamiento faltantes a partir de esas dos variables. De ser necesario se invoca el *webhook* correspondiente a la política de auto escalamiento que ofrece ese número de unidades.

Es importante estimar bien el número de solicitudes que puede atender una unidad de escalamiento (dimensionar bien los servidores de la capa web) y ajustar lo mejor posible sus componentes internos relacionados con el procesamiento de las solicitudes. Como por ejemplo: haproxy, apache, php-fpm, pgpool y postgres. Esto último para obtener el mejor rendimiento posible de los recursos básicos como CPU y memoria RAM que componen la unidad de escalamiento.

6.3.2 Capa de base de datos

En la capa de base de datos se almacenan procesan y devuelven datos a partir de solicitudes iniciadas en la capa web. La Figura 13 muestra los elementos de esta capa programados por medio de las plantillas de orquestación. Se utiliza Postgres como Sistema de Gestión de Bases de Datos (DBMS por sus siglas en inglés) y se complementa con PGPool para balanceo de las consultas, tolerancia a fallas, *online recovery* y un novedoso servicio de *pool* de conexiones entre otras posibles funcionalidades. El esquema de tolerancia a fallas es similar al de capa web: servidores, zonas de disponibilidad y un programa que se encargue de la tolerancia a fallas. En este caso, y como ya se mencionó, el que se encarga de la tolerancia a fallas son los programas PGPool y Postgres en el modo *Streaming Replication*.

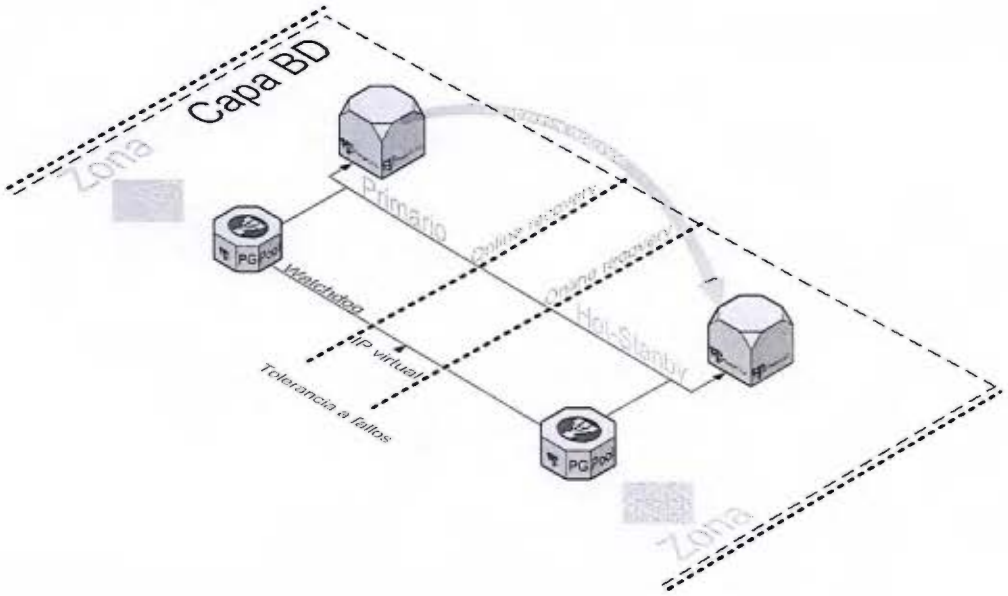


Figura 13. Diagrama de la capa de base de datos en el prototipo

La capa de base de datos es uno de los contextos de auto escalamiento identificados. Sin embardo auto escalar esta capa utilizando la implementación anterior es de una complejidad aún no resuelta. Es posible agregar y quitar instancias del mismo modo que en la capa web gracias a los servicios de la Nube. La dificultad se encuentra en el manejo a nivel de Postgres y PGPool ante esos eventos para lograr un mecanismo

eficiente que mantenga la disponibilidad, la integridad y la confiabilidad de los datos. Por ejemplo: eliminar un nodo PGPool no es tan transparente como lo puede ser eliminar un nodo de HAProxy. Se dispara por defecto un evento de tolerancia a fallas. Agregar un nodo requiere reiniciar el servicio. Algo que puede afectar por algunos segundos la disponibilidad de los datos.

7 Evaluación del prototipo

En este apartado se procede a detallar el diseño, la ejecución y los resultados de las pruebas para la evaluación de la adaptabilidad del prototipo, sus estrategias de elasticidad y el impacto de esa característica sobre el hardware del entorno de la Nube utilizado. Además se evalúa la diferencia en el rendimiento entre diferentes implementaciones de la solución para una plataforma LMS, incluía la versión tradicional y la solución de la Nube, entre otras.

Se proponen entonces dos tipos de prueba: una para la evaluación de la adaptabilidad y otra para medir y comparar el rendimiento. En ambas pruebas se requiere simular el tráfico, generar planes y cursos de prueba y extraer estadísticas sobre el rendimiento, además de caracterizar e identificar el prototipo respecto de las otras implementaciones a evaluar. Para realizar las tres primeras tareas se utilizan las herramientas Jmeter, Moodle y Moodle Perfomance Comparison (MPC). Para la cuarta tarea se definen perfiles para representar cada implementación.

Un perfil es una arquitectura de implementación factible para una plataforma LMS; caracteriza el entorno según el tipo de recursos (físico/Nube), el tipo de asignación (estático/elástico) y el modelo base de la arquitectura (local/capas). Se han definido en total cinco perfiles, lo cuales tienen algunas diferencias entre sí pero mantienen similitudes para hacerlos comparables. Por ejemplo, los perfiles comparten una cantidad similar de recursos asignados (CPU, memoria, disco y red); tienen la misma línea base (base de datos, código de aplicación, datos); y comparten la misma elementos de la plataforma: distribución de Linux, el mismo sistema de gestión de la base de datos y el mismo servidor web. En el Cuadro 2 se presenta está caracterización y se indica si el perfil está implementando la alta disponibilidad.

Cuadro 2. Caracterización de los Perfiles definidos para las pruebas

		Perfil					
		1	2	3	4	5	Prototipo
Tipo de recursos	Físicos	X					
	Nube		X	X	X	X	
Tipo de asignación de recursos	Estáticos	X	X	X	X		
	Elásticos					X	
Tipo de modelo	Local		X				
	Capas	X		X	X	X	
Alta disponibilidad		X		X			

El Cuadro 3 contiene la cantidad de cores, memoria y almacenamiento asignados en cada perfil. Se debe tomar en cuenta que el tipo de recursos puede variar de un perfil a otro como se muestra en el Cuadro 2. Por ejemplo el perfil #1 y el perfil #3 tienen las mismas cantidades de cores y memoria asignados pero en el perfil #3 los recursos son virtuales. El perfil correspondiente al prototipo de la investigación es el #5. Se diferencia respecto a los otros en la asignación elástica de recursos descrita por medio de un rango con un mínimo y un máximo como se observa en el Cuadro 3. La asignación elástica de recursos dependerá de las unidades de escalabilidad disponibles según la demanda y las estrategias de elasticidad. La cantidad mínima está por debajo respecto a tres de los otros cuatro perfiles buscando una optimización de los recursos, y junto con la cantidad máxima permiten observar el grado de adaptabilidad del prototipo respecto a los recursos. En los Anexos 5 y 6 se pueden encontrar más detalles de cada perfil, incluyendo diagramas para ilustrar las arquitecturas de cada uno.

Cuadro 3. Cantidad de recursos asignados en los Perfiles definidos para las pruebas

	Perfil						
	1	2	3*	4*	5* Min.	5* Max.	Prototipo
CPU	48	24**	48**	48**	32**	48**	
RAM	64GB	64GB	64GB	64GB	40GB	80GB	
Disco	480GB	60GB	2104GB	120GB	120GB	320GB	

* No se cuentan los recursos de las instancias para los balanceadores implementados.
** Son CPU virtuales o vCPU

7.1 Prueba 1 – auto escalabilidad

Esta prueba permitirá evaluar si el prototipo es capaz de adaptarse a la demanda y las ventajas o desventajas de esa capacidad. Además ayudará a establecer mejores estrategias o ajustar las que han sido propuestas. Se espera obtener datos que permitan medir el grado de adaptabilidad. En esta prueba se utiliza únicamente el perfil #5 correspondiente al prototipo elástico. La prueba tiene dos ejecuciones, una primera que se identifica como **NO_ASG** (*sin autoscaling group*) con una cantidad estática de recursos; y la otra que se identifica como **ASG** (*con autoscaling group*) con una asignación de recursos elástica. La prueba utiliza el hardware físico de forma dedicada durante sus ejecuciones.

Objetivos: Los objetivos de la prueba son:

- Probar el grado de adaptación del prototipo y sus estrategias para agregar y/o quitar recursos automáticamente cuando se produce un incremento gradual o esporádico en el tráfico de usuarios o bien una disminución.
- Documentar los requerimientos, dependencias y resultados de las pruebas.

Métricas: Para el análisis y la representación de resultados se utilizarán las métricas que se observan en el Cuadro 4. Se seleccionaron dentro de un conjunto más amplio de mediciones que recolectan las herramientas utilizadas.

Cuadro 4. Métricas consideradas para el análisis de la prueba de auto escalabilidad

Métricas Jmeter		
Métricas de la herramienta Jmeter utilizadas en la prueba		
Etiqueta	**Unidad**	**Descripción**
# samples	Número	Total de muestras
Average	ms	Promedio de tiempo de respuesta
Std. Dev.	ms	Desviación estándar
Error %	%	Porcentaje de errores registrados
Throughput	Número/tiempo	Muestras por unidad de tiempo
Active Threads	Número	Usuarios activos
Métricas vSphere PowerCLI		
Métricas de la herramienta vSphere PowerCLI utilizadas en la prueba		
Etiqueta	**Unidad**	**Traducido**
cpu.usage.average	%	Promedio de uso del CPU
mem.usage.average	%	Promedio de uso de memoria
disk.usage.average	KBps	Promedio de uso de disco
net.usage.average	KBps	Promedio de uso de red
Otras métricas		
Etiqueta	**Unidad**	**Traducido**
Unidad de escalamiento	Número	Número en uso de unidades de auto escalamiento

Otra documentación: El Anexo 5 tiene la documentación completa de esta prueba. En él se documentan:

- Los requerimientos de la prueba
- La definición del plan de pruebas JMETER
- El procedimiento para realizar la prueba
- Las características técnicas del ambiente de pruebas, incluido el detalle del equipo que hardware desde donde se realiza la prueba y el hardware objeto de la prueba.

7.1.1 Prueba 1 – resultados

En la Figura 14 se observa la respuesta del prototipo ante la simulación de variaciones en la demanda. Los resultados para el prototipo **con** la característica de elasticidad (**ASG**) se presentan en la parte superior y los resultados del prototipo **sin** esa característica (**NO_ASG**) en la parte inferior. La simulación de la demanda se puede identificar en el gráfico por medio de la línea de color anaranjado que representa el número de usuarios activos y que se registra en el eje vertical secundario. Los tiempos de respuesta a las solicitudes hechas por esos usuarios se registran en milisegundos en el eje secundario primario. El color en los tiempos de respuesta indica si fue exitoso o fallido, azul y rojo respectivamente. El eje horizontal primario contiene el tiempo relativo trascurrido en el formato mm:ss (minutos:segundos).

De la Figura 14 se rescatan los aspectos sobre la disponibilidad del servicio y sobre los tiempos de respuesta exitosos registrados ante las variaciones de la demanda. El prototipo en la prueba NO_ASG presenta afectaciones en la prestación del servicio. Esto se puede observar por las incidencias de tiempos de respuesta fallidos, particularmente luego de incrementos en la demanda. Respecto a los tiempos de respuesta exitosos, ambas ejecuciones tienen algunos muy altos durante los incrementos graduales y los picos esporádicos de solicitudes. Sin embargo se evidencia también que para el prototipo en la ejecución ASG los tiempos son menores respecto a la ejecución NO_ASG. No hay respuestas fallidas a pesar de los incrementos en la demanda,

logrando el prototipo elástico mantener la disponibilidad del servicio en la ejecución ASG.

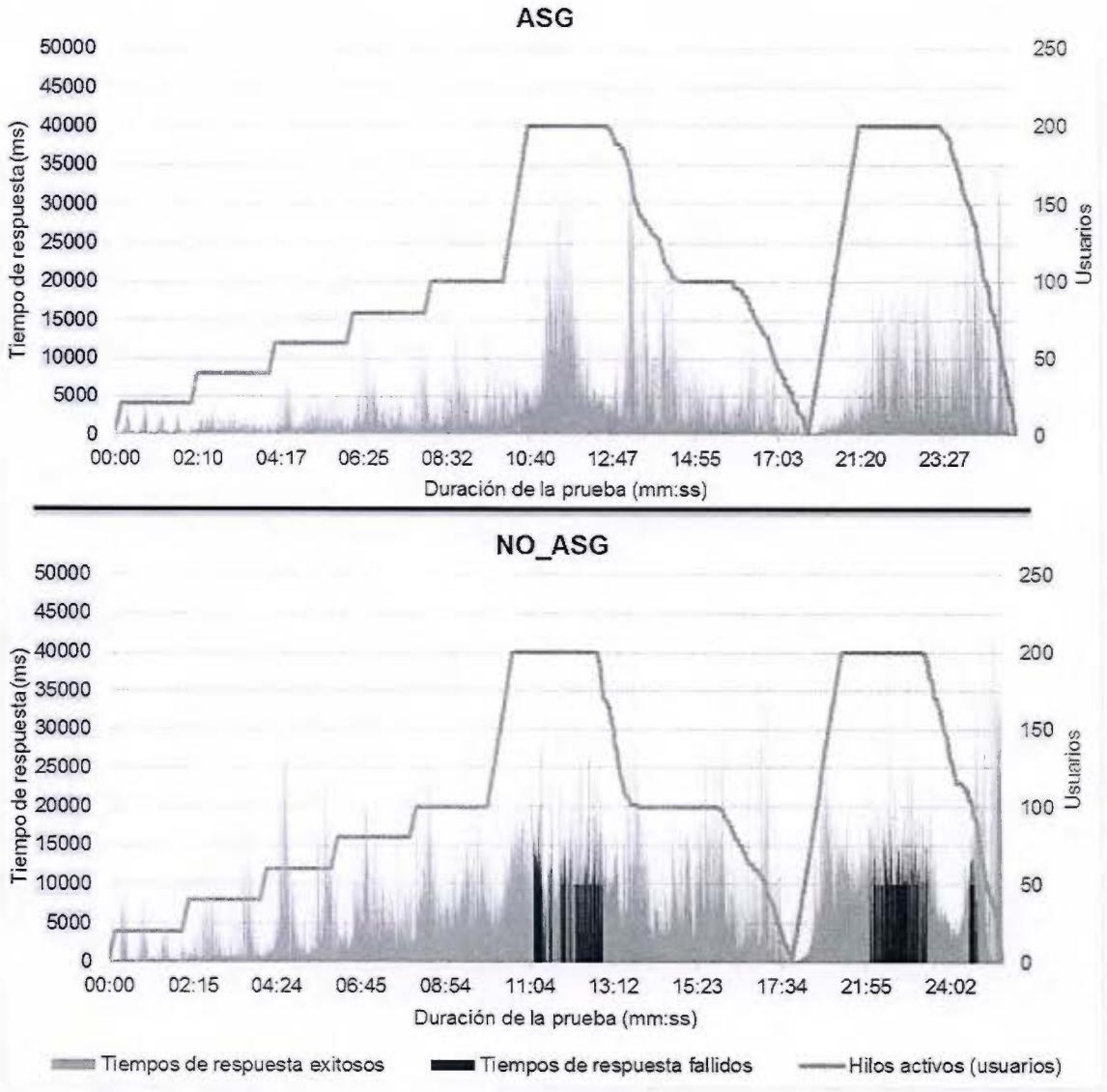
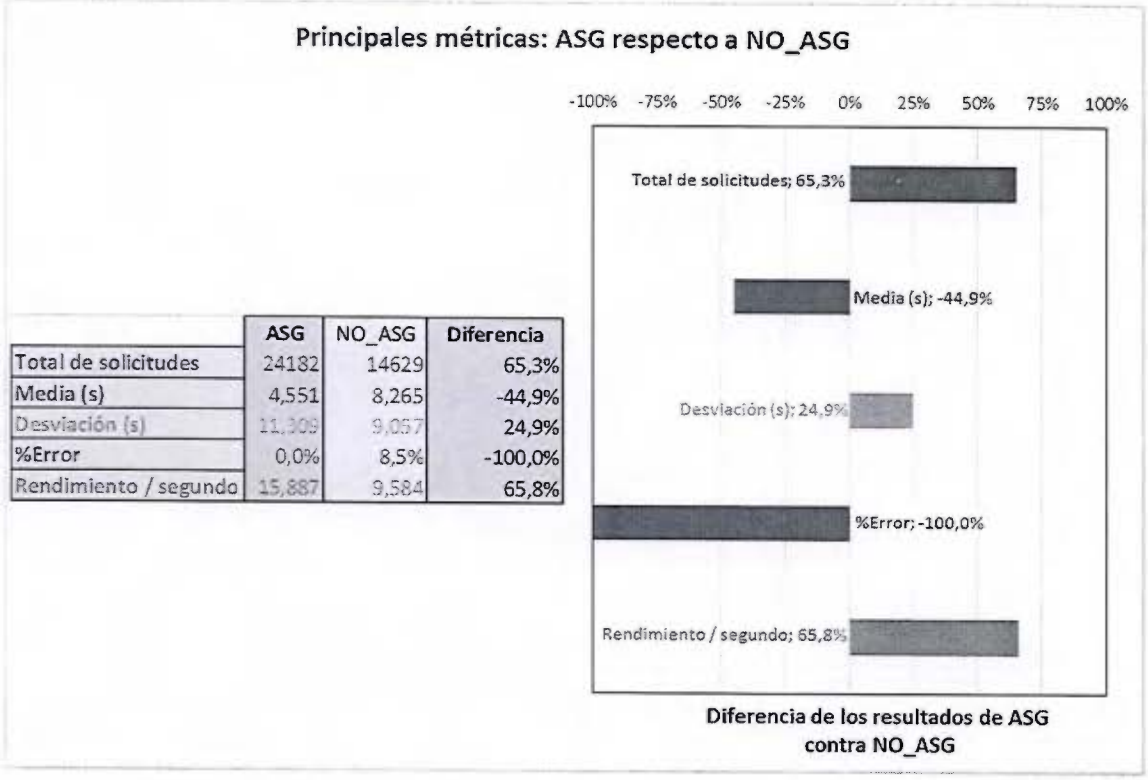


Figura 14. Gráfico de resultados. Usuarios y tiempos de respuesta con y sin auto escalabilidad

La Figura 15 presenta una tabla con los resultados de las principales métricas registradas en ambas ejecuciones y a un lado un gráfico para visualizar la diferencia correspondiente al prototipo **con** la característica de elasticidad (ASG) en comparación al prototipo **sin** ella (**NO_ASG**). Este gráfico ayuda a identificar en cuales métricas y en qué porcentaje el primero tuvo una ganancia (porcentaje positivo) o pérdida (porcentaje

negativo) respecto al segundo. Por ejemplo el primero tiene una mayor capacidad de procesar solicitudes que su contraparte. La evidencia de esto está en los registros del total de solicitudes y del rendimiento promedio, con alrededor de un 65% más que en la segunda ejecución (NO_ASG). Además el prototipo elástico no registra errores y presenta un tiempo de respuesta promedio inferior. Sin embargo la desviación estándar presenta un incremento de alrededor de un 24% respecto al segundo.



	ASG	NO_ASG	Diferencia
Total de solicitudes	24182	14629	65,3%
Media (s)	4,551	8,265	-44,9%
Desviación (s)	11,309	9,057	24,9%
%Error	0,0%	8,5%	-100,0%
Rendimiento / segundo	15,887	9,584	65,8%

Figura 15. Gráfico de resultados. Comparación de las principales métricas entre las ejecuciones de la prueba de auto escalabilidad

En la Figura 16 se puede comparar el uso de CPU, de memoria, de red y de disco a nivel del hipervisor para conocer el impacto de ambas ejecuciones sobre los recursos físicos disponibles. Una vez más los resultados para del prototipo **con** la característica de elasticidad (**ASG**) se presentan en la parte superior y los resultados del prototipo **sin** esa característica (**NO_ASG**) en la parte inferior. El porcentaje de consumo de CPU y de memoria se consulta en el eje vertical primario (izquierda) por medio de las líneas de tonalidad azul; el primero con marcadores circulares en tonalidad oscura y el segundo

con marcadores cuadrados en tonalidad clara. En el eje vertical secundario (derecha) se ubica el consumo de red y de disco en KBps (*Kilobytes* por segundo), utilizando áreas de tonalidad verde oscuro y verde claro respectivamente. El eje horizontal primario (abajo) indica el tiempo transcurrido antes (-), durante, y después (+) de la prueba. El formato de este eje es en mm:ss (minutos:segundos).

Al comparar los dos gráficos es posible observar que el hipervisor presenta incrementos y decrementos en el uso de recursos en momentos similares de ambas ejecuciones. Sin embargo en la ejecución de prueba para el prototipo elástico el hipervisor hace un consumo considerablemente más alto, llegando incluso a utilizar en algún momento el 100% del CPU. Adicionalmente al uso de CPU, el de disco y de red registran consumos más altos. El consumo de memoria sin embargo se mantiene en niveles similares en ambas ejecuciones.

En el prototipo elástico el hipervisor tiene una carga adicional debido a por lo menos dos factores, 1) la preparación de nuevas unidades de auto escalamiento y 2) las solicitudes adicionales que el prototipo elástico atiende. En el proceso de añadir unidades de escalamiento, el hipervisor recibe del CMP las instrucciones de asignar, iniciar y preparar nuevos recursos para instanciar unidades adicionales. Cada una de estas tareas en este proceso consume recursos. Con la capacidad adicional por los recursos de esas nuevas unidades, el prototipo elástico atiende un mayor número de solicitudes según los picos en la demanda, como se observó en la Figura 15. Esto también implica un mayor consumo de recursos en el hardware físico subyacente.

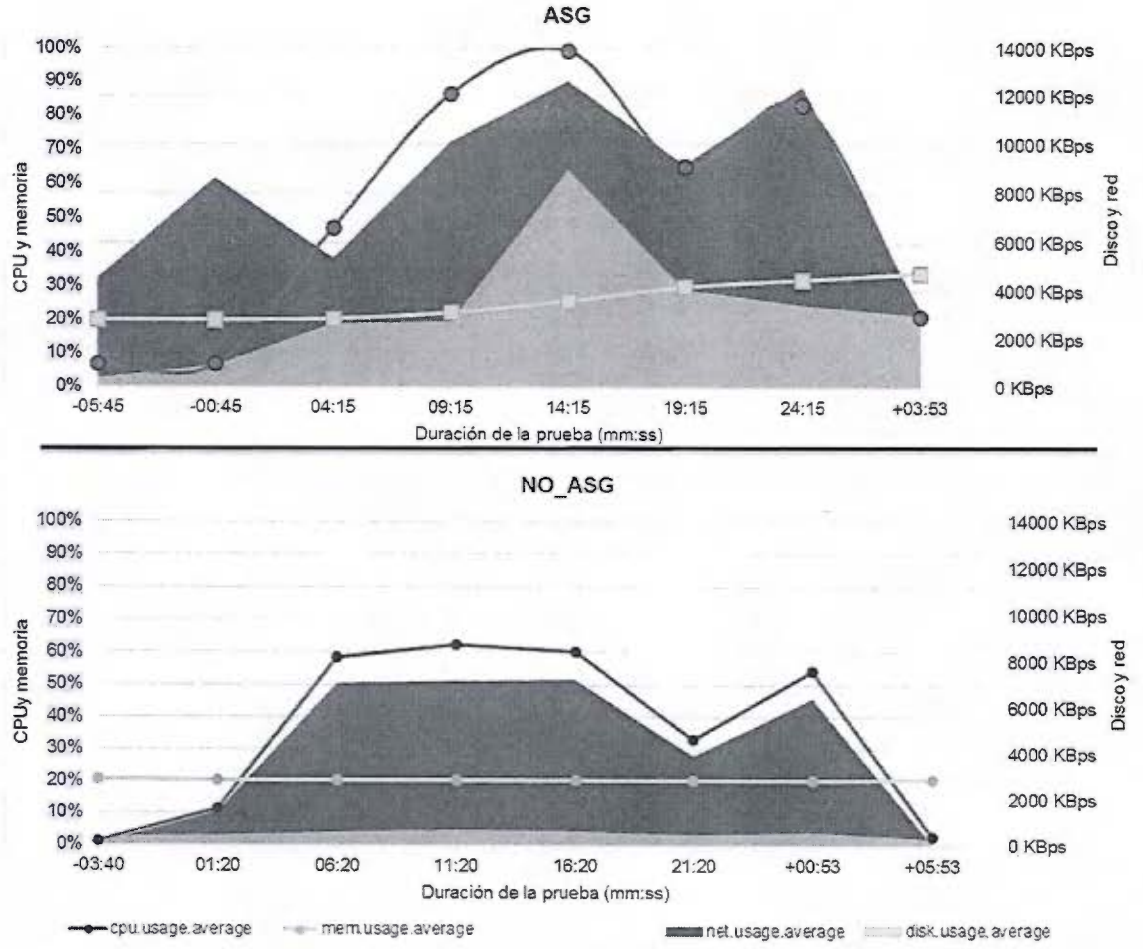


Figura 16. Gráfico de resultados. Monitoreo del consumo de recursos en el hipervisor durante la prueba auto escalabilidad

Resultados específicos para la ejecución ASG

La Figura 17 contiene una serie de gráficos relacionados con la ejecución **ASG**. En ella se muestran la adaptabilidad del prototipo ante la simulación de variaciones en la demanda y el impacto de la ejecución sobre el consumo de recursos del hipervisor. El primer punto se puede observar en el primer gráfico, en la parte superior. En él se presenta la cantidad de unidades de auto escalamiento a las que el prototipo crece o decrece antes, durante y después de la ejecución. El número de unidades de auto escalamiento se ubica en el eje vertical primario, a la izquierda de ese gráfico. Para los incrementos se puede diferenciar a que estrategia de elasticidad, de las dos utilizadas, pertenece la adición. Las incidencias de reducción por otro lado son de un solo tipo y se pueden identificar en ese mismo eje. De esta manera es posible observar cuándo, cuánto y cómo creció y decreció el prototipo (número de unidades de auto escalamiento en un momento en el tiempo). El gráfico tiene también un eje vertical secundario que sirve para identificar por medio de una línea anaranjada el número de hilos activos (usuarios), cantidad que varía durante la ejecución para simular picos y reducciones en la demanda. La hora del día se ubica en el eje horizontal primario del gráfico, en la parte inferior, en formato h:mm:ss a.m. / p.m.

Los siguientes cuatro gráficos de la Figura 17 se utilizan para representar el impacto de la ejecución **ASG** sobre el consumo de recursos en el hipervisor; los primeros dos de ellos respecto a las variaciones en las unidades de auto escalamiento disponibles; y los otros dos respecto a la demanda. Para diferenciar estos dos temas se utiliza el eje vertical primario de los cuatro gráficos. En cada par, un gráfico corresponde al porcentaje de consumo del CPU y la memoria; y el otro al consumo de Disco y Red en KBps. Para lo cual se utiliza el eje vertical secundario. En el caso de estos cuatro gráficos, el eje horizontal primario contiene la duración de la ejecución en formato mm:ss.

A partir de la Figura 17 se evidencia que el prototipo es elástico y que el proceso de adaptación junto con la capacidad añadida impacta en alguna medida el consumo de recursos en el hardware físico. La elasticidad le permite al prototipo crecer ante

incrementos en la demanda, utilizando alguna de las dos estrategias, y decrecer de forma más conservadora cuando el tráfico es menor, en este caso particular por haber finalizado la ejecución. La primera estrategia se presenta cuando hay incrementos graduales, añadiendo una unidad a la vez; la segunda, cuando hay un pico más significativo, añadiendo un número de unidades que puede variar según el volumen de solicitudes recibidas. Si la segunda estrategia no se utilizará, el tiempo para llegar a un número de unidades de auto escalabilidad óptimo sería probablemente mayor, comprometiendo la capacidad de respuesta y la prestación del servicio en el prototipo.

Es importante rescatar también a partir de la Figura 17 que los incrementos en el número de unidades suceden con una diferencia de tiempo de uno o dos minutos respecto a los incrementos en el número de usuarios. Esto sirve para considerar que la adición de nuevas unidades no sucede inmediatamente después de dispararse una solicitud de auto escalabilidad. Es un proceso que lleva un tiempo variable, donde participan el CMP, sus servicios, en particular el de orquestación, y el hipervisor. Para que el proceso se dé por finalizado, cada unidad de auto escalamiento debe ser antes instanciada, configurada e integrada. Esto significa que la capacidad adicional no se podrá utilizar para atender solicitudes hasta que termine el proceso de adición y la nueva unidad esté lista.

En cuanto al consumo de recursos en el hardware físico, sí existe un impacto sobre el uso de CPU, red y disco. En los gráficos correspondientes de la Figura 17 se observa que el consumo de CPU y de red está directamente relacionado con el incremento en la demanda. Su utilización por parte del hipervisor aumenta o disminuye al mismo tiempo que la demanda. El consumo de disco tiene además una relación directa con el incremento de las unidades de auto escalamiento atendiendo nuevas solicitudes. Cuando hay más unidades atendiendo un mayor número de solicitudes, el consumo de disco se incrementa; sin embargo, cuando las solicitudes bajan, el consumo de disco también lo hace. La utilización de la memoria contrasta con lo anterior, al presentar apenas un incremento leve durante la ejecución. A partir de los resultados no se observa

una relación en el consumo de memoria con la demanda ni con el número de unidades de auto escalamiento disponibles.

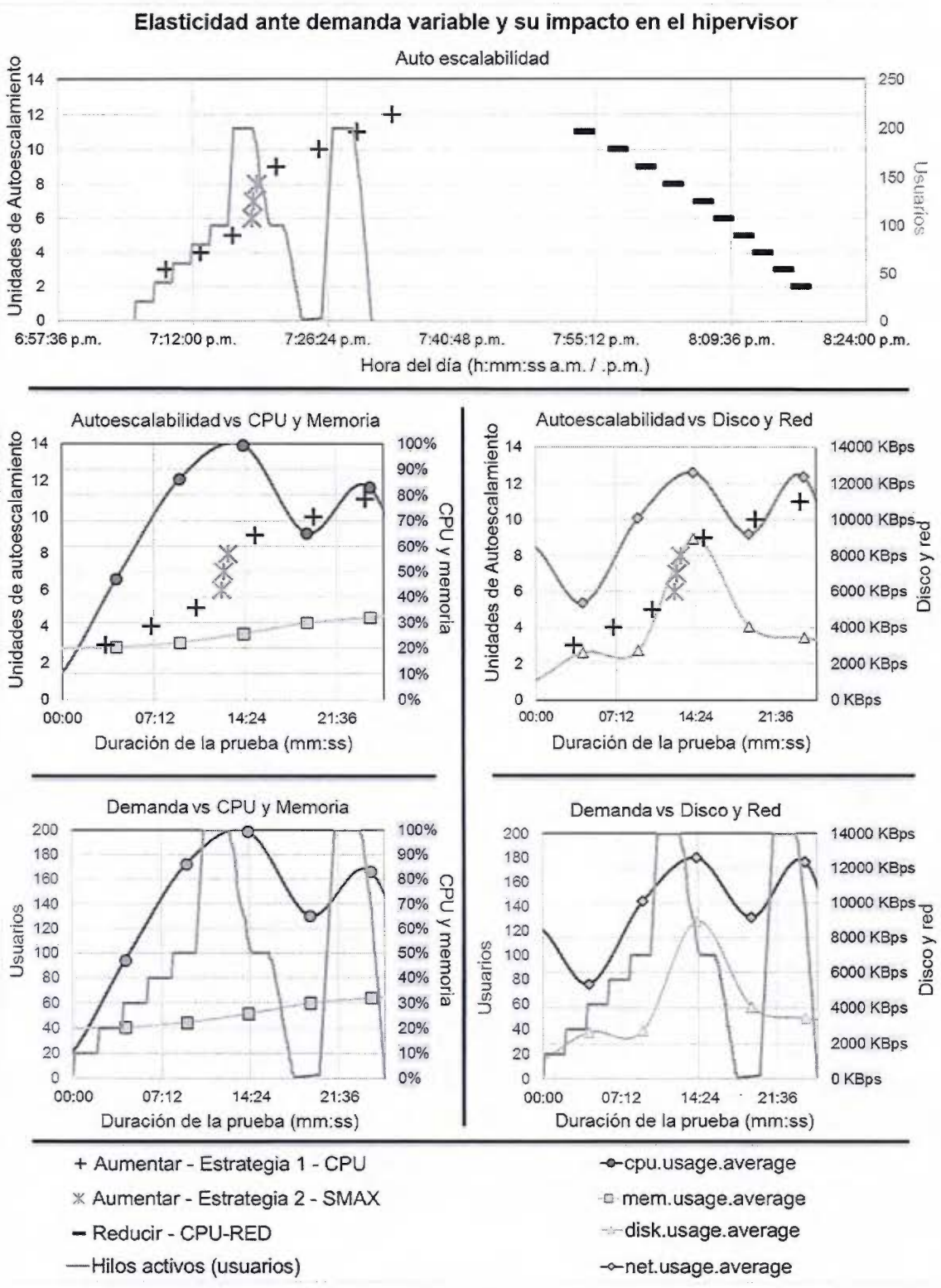


Figura 17. Gráficos de resultados. Elasticidad ante demanda variable y su impacto en el hipervisor durante la prueba de auto escalabilidad

7.2 Prueba 2 – rendimiento

Esta prueba permitirá evaluar el prototipo respecto a otras soluciones conocidas. Además de que facilita el análisis y la recolección de datos para establecer parámetros aceptables de rendimiento así como posibles ajustes. Se utilizarán los cinco perfiles descritos al inicio de este capítulo, a los cuáles se les ejecutaran diferentes tamaños de prueba. Los tamaños tienen en un común las acciones a simular (pasos) pero difieren en el volumen de solicitudes. Para la prueba se recolectan una serie de métricas a partir de dos diferentes fuentes. La prueba utiliza el hardware físico de forma dedicada durante sus ejecuciones.

Objetivos: Los objetivos de la prueba son:

- Establecer líneas base del rendimiento del prototipo en diferentes entornos y compararlas con respecto al entorno auto escalable.
- Documentar los requerimientos, dependencias y resultados de las pruebas.

Tamaños de la prueba: Para cada perfil se correrán cuatro tamaños de prueba definidos por la herramienta MPC. Como se observa en el Cuadro 5 estos tamaños de prueba se componen de una serie de variables como: número de usuarios, bucles o repeticiones y el periodo de inicialización.

Cuadro 5. Tamaños para las pruebas de rendimiento

		Usuarios	**Bucles**	**Subida**	**Capacidad**
Parámetros:*		*# de usuarios a simular*	*Iteraciones de los pasos*	*Ventana de tiempo para iniciar los usuarios*	*# muestras por minuto*
Tamaños:**	XS	1	5	1	120.0
	S	30	5	6	120.0
	M	100	5	40	120.0
	L	1000	6	100	120.0
* Documentación de Jmeter					
** Tamaños predefinidos por la herramienta "Moodle Perfomance Comparison" (MPC)					

Métricas: Para el análisis se utilizarán **dos tipos** de métricas descritas en el Cuadro 6. EL primer grupo son generadas por Moodle, recolectadas por MPC, y el segundo por JMETER. En el primer caso, las métricas son generadas en el servidor que atiende y responde la solicitud, en el *footer* de la página enviada como respuesta. La herramienta MPC analiza y extrae cada métrica a partir de esa respuesta. En el segundo caso las métricas son calculadas desde el servidor JMeter que ejecuta prueba.

Cuadro 6. Métricas consideradas de MPC y JMETER para el análisis de las pruebas de rendimiento.

Métricas MPC por cada paso		
Métricas reportadas por Moodle a MPC		
Etiqueta	**Unidad**	**Traducido**
tiempoBD	ms	Tiempo de las consultas a BD para servir URL
memoriaUsada	MB	Memoria utilizada para servir URL
cargaServidor	Número	Promedio de la carga del servidor al servir URL
tiempoCarga	ms	Tiempo de carga de la página
Std. Dev.	ms	Desviación estándar
Métricas Jmeter		
Métricas de la herramienta Jmeter utilizadas en la prueba		
Etiqueta	**Unidad**	**Descripción**
Average	ms	Promedio de tiempo de respuesta
Std. Dev.	ms	Desviación estándar

Pasos: Los pasos son las acciones que realizan los usuarios (hilos). Son parte de la simulación del plan de pruebas generado por la Herramienta Moodle. El Cuadro 7 contiene el detalle de los pasos. El paso 11, "Enviar respuesta al foro", no está disponible en las métricas MPC. El tema personalizado utilizado no publica la información de ese paso para que sea recolectado por esa herramienta. Las métricas de JMETER en cambio, si lo toman en cuenta.

Cuadro 7. Pasos del plan de pruebas de rendimiento con la herramienta MPC

Plan de prueba de MPC***			
Pasos		**Etiqueta**	**Traducido**
Calentamiento	1	Frontpage not logged	Ver página principal sin autenticarse
	2	Login	Autenticarse
	3	Frontpage logged	Ver página principal autenticado
	4	View course	Ver un curso
	5	View a page activity	Ver una pagina
	6	View course again	Ver el curso de nuevo
	7	View a forum activity	Ver un foro
	8	View a forum discussion	Ver tema del foro
	9	Fill a form to reply a forum discussion	Responder en el foro
	10	Send the forum discussion reply****	Enviar respuesta al foro****
	11	View course once more	Ver el curso una vez mas
	12	View course participants	Ver los participantes del curso
	13	Logout	Salir
Iteración	1	Frontpage not logged	Ver página principal sin autenticarse
	2	Login	Autenticarse
	3	Frontpage logged	Ver página principal autenticado
	4	View course	Ver un curso
	5	View a page activity	Ver una pagina
	6	View course again	Ver el curso de nuevo
	7	View a forum activity	Ver un foro
	8	View a forum discussion	Ver tema del foro
	9	Fill a form to reply a forum discussion	Responder en el foro
	10	Send the forum discussion reply****	Enviar respuesta al foro****
	11	View course once more	Ver el curso una vez mas
	12	View course participants	Ver los participantes del curso
	13	Logout	Salir
*** Pasos predefinidos en herramienta "Moodle Perfomance Comparison" (MPC)			
**** No se considera este paso ni se incluye en los resultados de las métricas MPC.			

Otra documentación: El Anexo 6 tiene la documentación completa de esta prueba. En él se documentan:

- Las dependencias
- La definición del plan de pruebas JMETER
- El procedimiento para realizar la prueba

- Las características técnicas del ambiente de pruebas, incluido el detalle del equipo que hardware desde donde se realiza la prueba y el hardware objeto de la prueba.

7.2.1 Prueba 2 – resultados según JMETER

La Figura 18 contiene dos gráficos sobre el tiempo de carga según JMeter. El primero, en la parte superior, muestra el tiempo promedio de carga en cada tamaño de prueba para cada uno de los cinco perfiles; el segundo, en la parte inferior, la desviación estándar del primero. En ambos gráficos el eje vertical se utiliza para presentar los tiempos promedios en milisegundos (ms); el eje horizontal contiene los cuatro tamaños de la prueba (XS, S, M, L); y la profundidad ubica a cada uno de los perfiles.

Según la Figura 18 el prototipo presenta un mejor rendimiento que los demás entornos. Los tiempos de respuesta del perfil #5, correspondiente al prototipo, son menores respecto a los otros cuatro perfiles en todos los tamaños. En cambio el perfil #1, el cual cuenta con recursos físicos, tiene en el tamaño L el resultado más alto de los 5 perfiles. En cuanto a la desviación estándar los resultados son muy similares. En ese apartado el prototipo tiene resultados un poco inferiores respecto a los otros perfiles en todos los tamaños excepto en uno: el tamaño L. En ese tamaño el prototipo comparte uno de los resultados más altos, siendo el perfil #3 el que tiene una menor desviación estándar que los demás. Esto se puede interpretar como una mayor estabilidad para ese perfil durante periodos con un volumen de tráfico mayor. La razón de ello puede estar en aquello en lo que difiere el perfil #3 respecto a los demás: está implementado en capas y desplegado sobre dos hipervisores independientes.

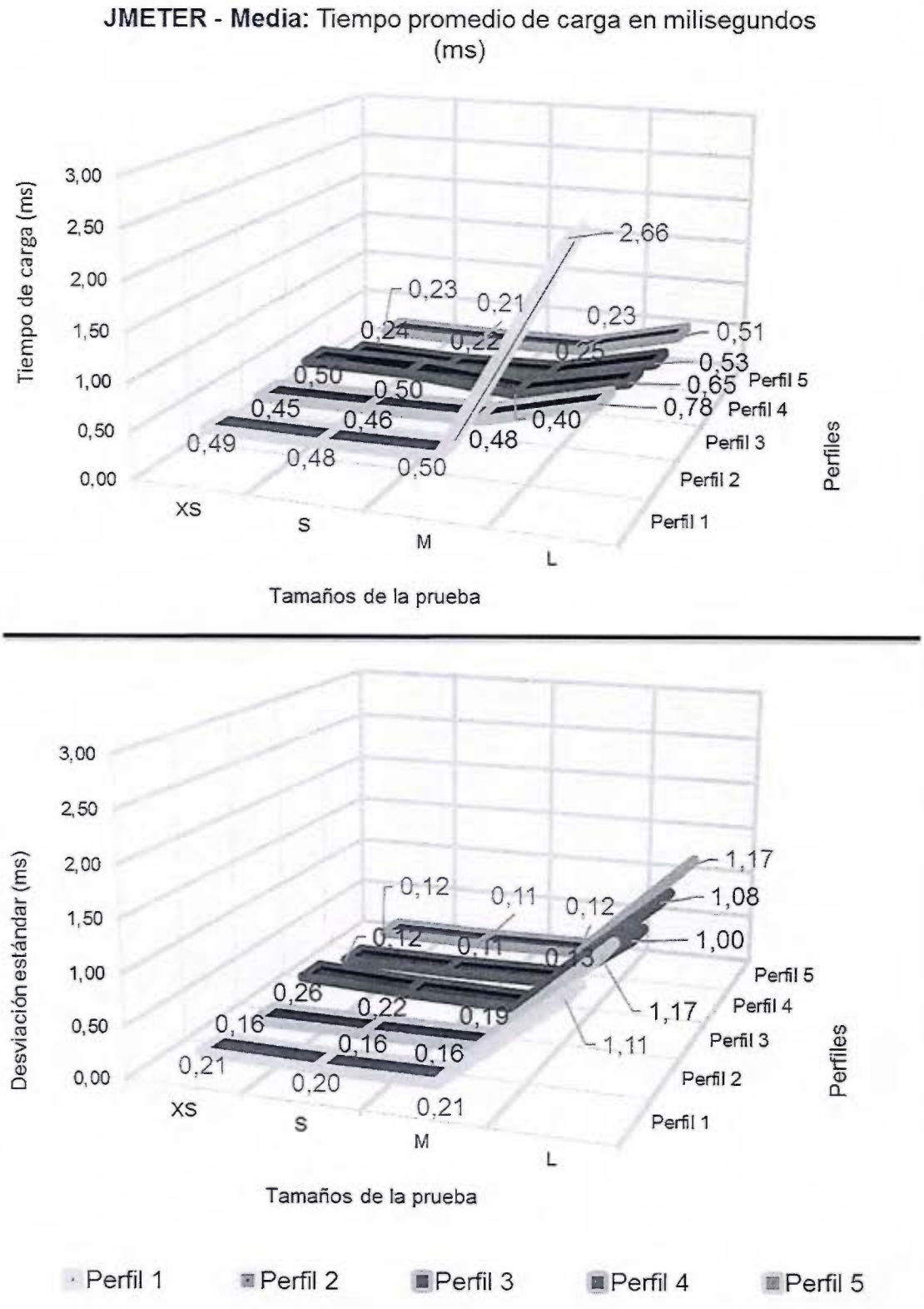


Figura 18. Gráfico de resultados. Tiempo promedio de las pruebas de rendimiento según JMETER

7.2.2 Prueba 2 – resultados según MPC

A continuación se incluyen los resultados de las métricas según las estadísticas de Moodle recolectadas por la herramienta MPC. No se han incluido todos los tamaños de la prueba; se ha seleccionado para mostrar únicamente el tamaño L, el de mayor número de usuarios. Lo anterior como una manera de satisfacer el objetivo de la prueba sin cargar demasiado los gráficos, buscando así darles una mejor lectura.

Las figuras de la 19 a la 23 tienen elementos en común y solo varían en la métrica que representa cada una. Por ejemplo, cada uno de los cinco perfiles se puede identificar con el mismo color y el mismo marcador en todas estas figuras. Además cada una tiene un gráfico para presentar la métrica; y otro, para su desviación estándar. Ambos elementos se ubican en el eje vertical. Las métricas MPC son sobre los tiempos de las consultas a la base de datos, la memoria utilizada, la carga del servidor y el tiempo de carga, para cada solicitud generada durante la prueba. En el eje horizontal, por otro lado, se encuentran los pasos que se simulan en la prueba.

La Figura 19 muestra los resultados que obtuvieron cada uno de los perfiles en la métrica "tiempoBD" sobre los tiempos en promedio y en milisegundos (ms) para las consultas a la base de datos. El perfil #2 tiene los mejores registros en cada uno de los pasos. Ese perfil se caracteriza por su implementación de todo en un solo servidor de manera que la instalación y el acceso a los recursos son locales. Lo cual explicaría en parte los menores tiempos de las consultas hacia la base de datos. Al considerar el perfil del prototipo, el #5, sus resultados lo ubican como el segundo con mejores tiempos, y el primero entre los que tienen una implementación en capas. Otro aspecto interesante a observar en la figura son los pasos con incremento en los tiempos de la consulta, como por ejemplo, los pasos relacionados con los foros. Éstos presentan el mayor incremento respecto a los otros pasos en todos los perfiles, incluso en la desviación estándar. Lo cual evidencia que los tiempos se ven afectados por otros factores aparte de los tipos de recursos, el tipo de asignación o el modelo de implementación. Entre estos podrían estar

el tamaño de la base de datos, la programación del mismo Moodle o bien su configuración.

La Figura 20 presenta, para cada perfil, el uso promedio de memoria en megabytes (MB) en cada uno de los pasos. En este caso los resultados son muy similares, excepto para el perfil #2. Ese perfil tiene un uso de memoria significativamente mayor respecto a los demás perfiles. En el Anexo 6 se puede consultar como todos ellos utilizan Apache, mod-fastcgi y PHP-fpm excepto el perfil #2, el cual utiliza la configuración por defecto para Apache y PHP. Esta parece ser la clave en el mayor consumo de memoria para este perfil y el bajo consumo en el resto, dentro de los cuales se encuentra el prototipo. Por otro lado, al igual que en la métrica anterior, sobresale el incremento en los pasos relacionados con los foros: el consumo y su desviación estándar es mayor en estos pasos respecto a los demás.

La Figura 21 corresponde a los resultados de la métrica sobre la carga promedio reportada al servir cada paso. En esta métrica el prototipo tiene menores cargas registradas y una menor desviación estándar. La diferencia con los perfiles #3, #4, e incluso #2, es poca. No obstante, la diferencia respecto al #1 si es considerable, especialmente en los pasos relacionados a los foros. Estos resultados son similares al consultar y comparar la desviación estándar del perfil #1 respecto a los demás.

La Figura 22 contiene los resultados obtenidos sobre el tiempo de carga promedio en cada paso. Una vez más el perfil #1 llama la atención sobre los demás porque, aunque no se observan variaciones importantes entre ellos, el paso "ver tema del foro" de ese perfil tiene el mayor incremento, tanto en el tiempo de carga promedio con en su desviación estándar. Mientras que en el caso particular del prototipo elástico, los registros permiten observar tiempos de carga similares a los otros tres perfiles estáticos.

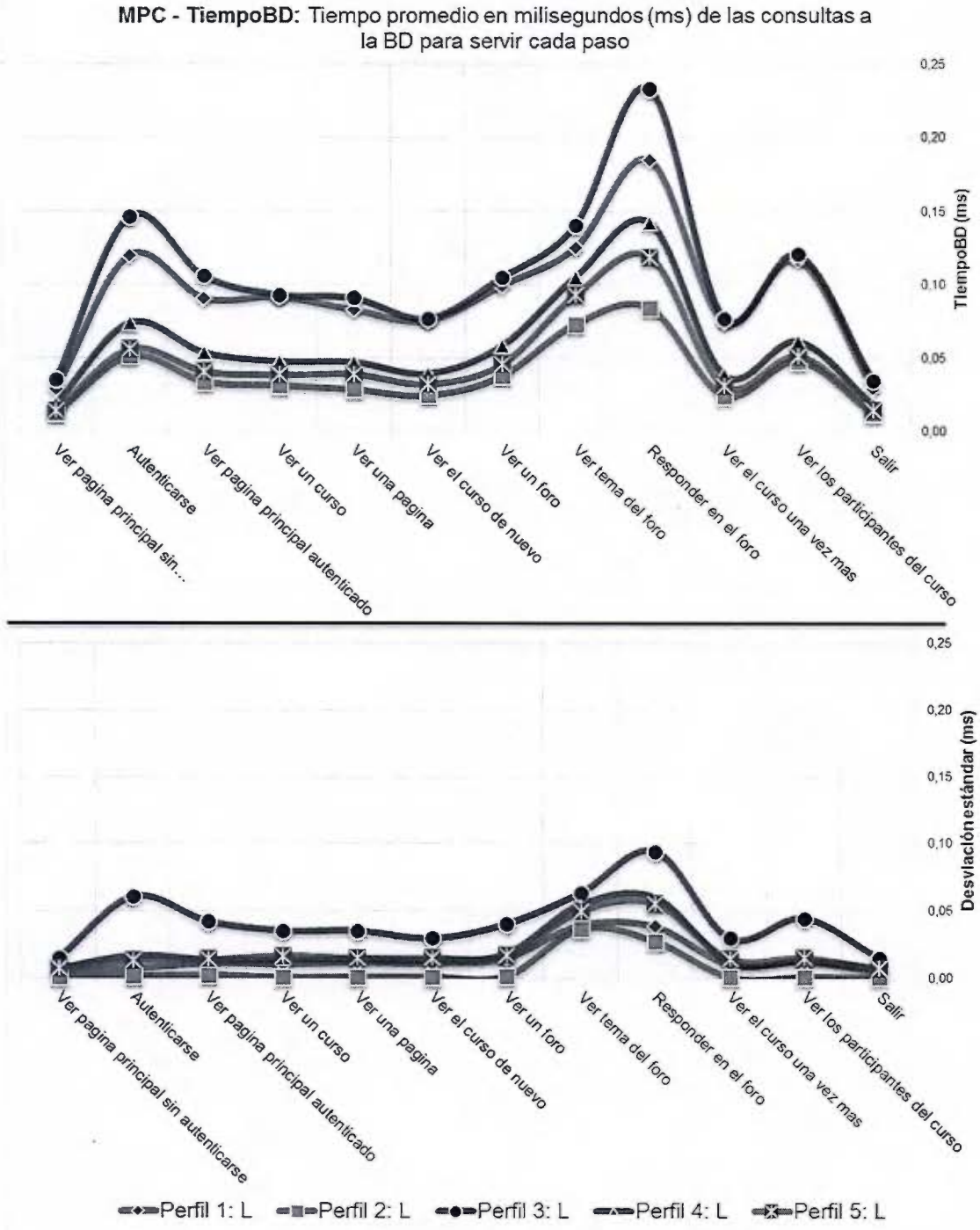


Figura 19. Gráfico de resultados. Tiempo promedio de las consultas a la base de datos en las pruebas de rendimiento según la herramienta MPC

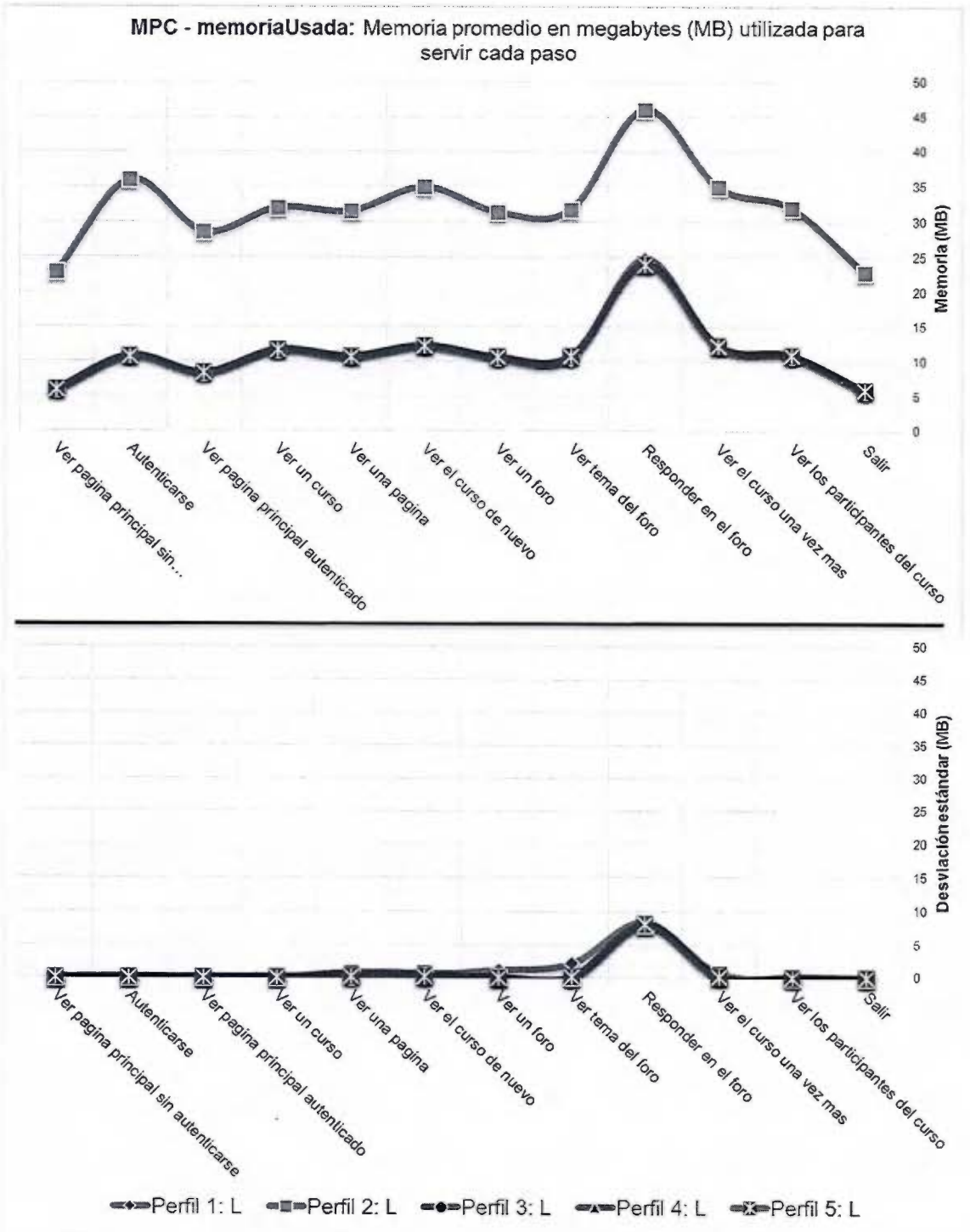


Figura 20. Gráfico de resultados. Memoria utilizada en las pruebas de rendimiento según la herramienta MPC

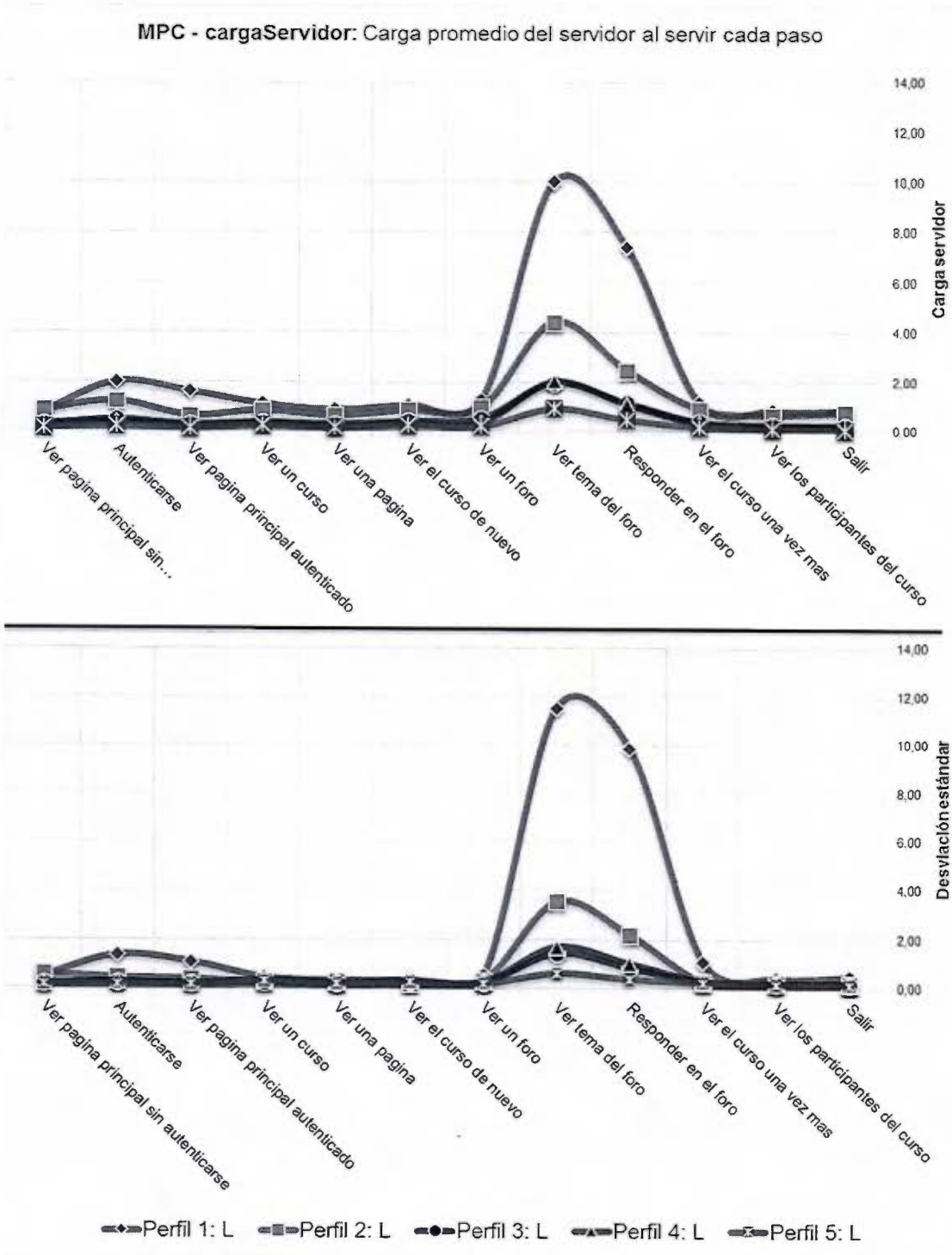


Figura 21. Gráfico de resultados. Carga al servidor en las pruebas de rendimiento según la herramienta MPC

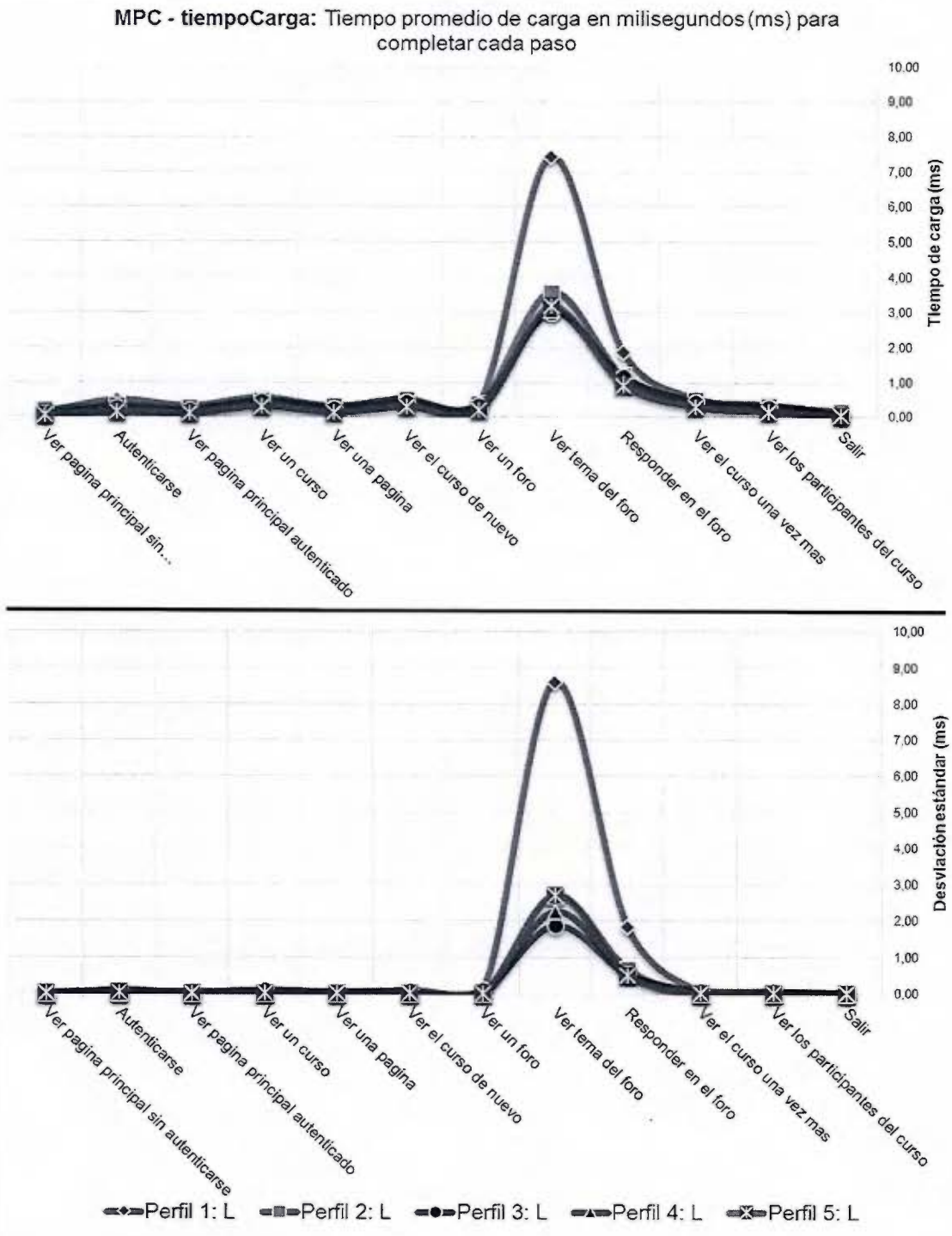


Figura 22. Gráfico de resultados. Tiempo de carga en las pruebas de rendimiento según la herramienta MPC

8 Conclusiones

Se comparte a continuación las conclusiones a las que se ha llegado luego del desarrollo de esta investigación y con base en los objetivos planteados. Además se incluye un apartado sobre algunos temas a desarrollar en trabajos futuros. Las conclusiones son las siguientes.

El prototipo diseñado, implementado y evaluado cuenta con un grado de adaptación según la demanda y un buen rendimiento en comparación con otras soluciones viables. Así lo evidencian los resultados de las pruebas propuestas, dando por satisfechos los objetivos de esta investigación. En la prueba de auto escalabilidad el prototipo creció cuando se presentaron incrementos en la demanda, aumentando su tamaño hasta alcanzar un máximo predefinido. Posteriormente cuando la demanda disminuyó, el tamaño y la asignación de recursos en el prototipo volvieron a sus valores iniciales. Esto le permitió utilizar más recursos para atender los picos en el tráfico y liberarlos cuando ya no eran necesarios. Esta capacidad no implica una afectación al rendimiento, considerando los resultados de la prueba correspondiente. Por el contrario, el prototipo registró buenos resultados en comparación con los otros contextos definidos.

El grado de adaptación del prototipo puede ser ajustado según el contexto o el presupuesto. El servicio de orquestación permite modificar fácilmente el tamaño y/o el número de las unidades de escalamiento en cualquiera de los ciclos de vida del prototipo. Adicionalmente el grado se puede ajustar revisando y modificando las reglas de las estrategias de auto escalabilidad. Es relevante considerar el impacto que pueden tener las tareas de auto escalamiento en el hardware subyacente y en los costos. Para el usuario de la Nube podría ser más importante tomar en cuenta los costos económicos de la estrategia para ajustarlas a un presupuesto mientras que para el propietario de la implementación de la Nube podrían interesarle más los efectos sobre los equipos físicos buscando obtener las mejores condiciones según su infraestructura. De cualquier forma es imperativo para ambas partes dimensionar las unidades de auto escalamiento de manera óptima, automatizar eficazmente su despliegue, y prever y controlar la incidencia de incrementos y decrementos.

Añadir una estrategia que prevea los cambios en la demanda antes de que sucedan ayuda a mitigar la carga que tiene auto escalar sobre el hardware y los servicios subyacentes, evitando que se dispare durante picos importantes en la demanda. Tener estadísticas del comportamiento de los usuarios y del uso de un sistema puede ayudar a establecer ese tipo de estrategia. De esta manera se puede establecer el mejor momento para agregar o quitar unidades de forma anticipada. No obstante, el implementar las estrategias dependerá de los servicios y herramientas disponibles en el CMP o la capacidad para desarrollar soluciones en ese sentido.

El tiempo de despliegue alcanzado gracias a la automatización y el uso de recursos abstraídos por medio del modelo de la Nube es, junto con la elasticidad, una de las mayores ventajas del prototipo. Listo el prototipo, este puede ser desplegado, actualizado, o eliminado en minutos. El tiempo que eso puede llevar en una solución tradicional puede ser días, meses o incluso años, considerando que es probable que se requiera adquirir nuevo equipo físico, instalarlo y configurarlo, y desplegar sobre el la aplicación, ya sea de forma manual o automatizada.

La definición de perfiles como soluciones viables para una plataforma LMS ayuda a visualizar, comparar y ajustar el potencial rendimiento entre el entorno que cada uno representa. De esta manera se tiene una base sobre la cual se puede configurar y ajustar correctamente los diferentes componentes de un sistema para alcanzar un mejor rendimiento. Por otro lado puede ayudar a eliminar el estigma de que la virtualización y la Nube tienen un menor rendimiento que las soluciones tradicionales porque solo permiten el uso de recursos compartidos. Otro aspecto relacionado a tomar en cuenta respecto al rendimiento es que también influyen el código y la programación de la aplicación, un tema que no se definió como parte del alcance de las pruebas.

El uso de recursos en el hipervisor aumenta tanto por incrementarse la demanda como por ejecutar las estrategias de elasticidad para atender lo primero. Esto se observa en los resultados de las pruebas. Los eventos de auto escalamiento producen más carga de trabajo para el hipervisor quién además de procesar la demanda según la capacidad del prototipo en un momento dado, debe soportar el inicio, la preparación y la

integración de una o varias unidades de auto escalamiento, dependiendo de la estrategia de auto escalabilidad a la que pertenecen. Por otro lado cuando esas unidades están listas, el sistema tendrá más capacidad para atender carga de trabajo adicional. Si la demanda continua significa que habrá otro incremento en el consumo de recursos en el hipervisor y que nuevas solicitudes de adición pueden ser disparadas, generando un mayor consumo. En caso contrario, si no hay solicitudes de auto escalamiento por procesar, las solicitudes que el sistema no es capaz de atender serán rechazadas, generando errores, pero no se agregara más carga al hipervisor. Se deja para un trabajo futuro cuánto del uso de recursos en el hipervisor corresponde a la carga de iniciar, preparar e incluir nuevas unidades; y cuánto corresponde a una mayor capacidad de procesar la demanda debido a las nuevas unidades añadidas.

Ciertas decisiones a la hora de configurar los componentes de la plataforma LMS afectan el uso de recursos. Un ejemplo es la diferencia en el consumo de memoria para el perfil #2 respecto a los otros. Ese incremento, o más bien la reducción en los otros perfiles, puede deducirse a partir de los detalles técnicos de cada perfil en el Anexo 6. El perfil #2 utiliza la configuración por defecto para los servicios de la plataforma, incluido para PHP. Los otros perfiles en cambio, utilizan FPM-PHP y Apache Fastcgi, además de ajustes adicionales al kernel, Apache y PHP-FPM.

Basado en las pruebas de rendimiento y la comparación de los resultados se concluye que el prototipo elástico, el perfil #5, cuenta con mejores tiempos de respuesta y una menor carga en los servidores. El tiempo de respuesta es una métrica importante porque es el tiempo que experimenta el usuario. Por otro lado, la métrica sobre la carga reportada también es relevante porque indica que tan ocupado está el CPU durante los picos en la demanda. Además una de las estrategias de auto escalabilidad está directamente relacionada con el consumo de CPU. La métrica y sus resultados permiten evaluar si las unidades están bien dimensionadas. Si el hardware subyacente tiene la capacidad suficiente, esto se traduce en una mejor distribución de la carga entre un número de unidades de trabajo óptimo que se ajusta según la estrategia de elasticidad.

En el modelo de la Nube, cuando se factura por demanda, pero también por un tema de optimización, los usuarios obtienen mayores beneficios con estrategias que utilizan recursos solo cuando son necesarios, liberándolos rápidamente cuando ya no lo son. Sin embargo es recomendable atender el principio de crecer rápidamente y decrecer de manera paulatina. Aun cuando esto pueda significar mantener recursos subutilizados por algún tiempo incrementando así los costos. No obstante, se pueden afectar los costos incluso en estrategias muy ajustadas donde los recursos se agregan o eliminan tan pronto sea posible. En especial cuando la adición y la remoción de recursos conlleva el uso de servicios adicionales que también son facturados, por ejemplo los servicios de almacenamiento y la transmisión de datos. Encontrar el balance para añadir y liberar recursos es una de los retos más importantes que se esperan solventar en la definición de mejores estrategias de elasticidad.

8.1 Trabajo futuro

Se han mencionado ya varios aspectos que son candidatos a desarrollar en trabajos futuros. Por ejemplo: investigar cuánto en el consumo de recursos del equipo físico corresponde a la demanda y cuánto al auto escalamiento. Adicionalmente se incluyen los siguientes temas.

La infraestructura y la plataforma son dos temas dentro de varios que pueden afectar el rendimiento. También lo puede ser la aplicación y cómo este desarrollada e implementada. Un ejemplo en la investigación es el incremento significativo en ciertos pasos de la prueba de rendimiento. Aquellos pasos relacionados con la consulta y la respuesta de foros arrojaron resultados con tiempos más altos respecto a los otros pasos en cada uno de los perfiles y para cada una de las métricas. Lo anterior deja entrever que es un aspecto de la funcionalidad de la aplicación y su implementación a nivel del software LMS.

El prototipo requiere una evaluación de la seguridad de sus componentes. Si bien se utilizan ciertas medidas facilitas por la implementación de la Nube, están no son suficientes. El protocolo para uso, transmisión, y almacenamiento de llaves ssh y/o otras

credenciales debe ser revisado y modificado según corresponda. Al hacer uso de sistemas adicionales como Consul, Csync2, HAProxy, entre otros, se introducen nuevos elementos a resguardar. Además es de gran importancia definir el "hardening" a nivel general de cada uno de los componentes.

Las estrategias de elasticidad pueden ser ajustadas, mejoradas y comparadas según diferentes contextos y objetivos. También se pueden generar nuevas estrategias. En cuanto a las dos utilizadas, de la primera se pueden analizar los parámetros y las métricas a monitorear de manera que se modifiquen los criterios o se incluyan otros adicionales. En cuanto a la segunda estrategia de elasticidad a partir de métricas a nivel de la plataforma, en este caso según las estadísticas de HAProxy, se puede considerar su integración dentro de las métricas de Ceilometer. Sería una alternativa al servicio de *LBaaS*. Incluso se puede mejorar la estrategia incluyendo otras estadísticas de HAProxy y utilizando la lectura y escritura de registros de eventos.

Un tema muy interesante que debe ser considerado como trabajo futuro es la solución que permita un contexto de elasticidad en la base de datos. Entre los temas que se deben resolver es revisar si PGPool y Postgres en particular pueden ser considerados para ese fin o si bien que implementar otros sistemas. Para ello se recomienda considerar la compatibilidad que tienen los sistemas candidatos para manejar la adición y remoción en caliente de nuevos nodos. También hay que considerar los servicios de la implementación de la Nube a utilizar y poner mucha atención en la integridad de los datos durante los eventos de auto escalabilidad.

9 Bibliografía

Agüero, M. (4 de abril de 2014). *Centro de Informática*. Obtenido de http://ci.ucr.ac.cr/node/319

Amazon. (2015). *AWS Documentation*. Obtenido de Instance Metadata and User Data: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

Ascar, B. (2014). *OpenStack Summit Presentations*. Recuperado el 16 de 08 de 2014, de Filling the Management Gap: Cloud Management Platforms for Managing OpenStack and Other Cloud Infrastructures: https://www.openstack.org/summit/openstack-summit-atlanta-2014/session-videos/presentation/filling-the-management-gap-cloud-management-platforms-for-managing-openstack-and-other-cloud-infrastructures

Bernardez, M. (2007). *Diseño, producción e implementación de e-learning: Metodología, herramientas y modelos.* Indiana, Estados Unidos: AuthorHouse.

Bione, A., & de Souza, R. (2014). Integrating Moodle with a Postgres-XC cluster database providing high availability and high performance at a low cost. *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, (págs. 212-217). New Orleans.

Bondi, A. (2000). Characteristics of scalability and their impact on performance. *Proceedings of the 2nd international workshop on Software and performance*, 195-203.

Chacón, S. (2009). *CMAPS METICS.* Obtenido de http://cmaps.ucr.ac.cr/rid=1GCF32QHS-1GPF4KC-13J/Resumen%20para%20CRUSA.doc

CITIC. (s.f.). *Centro de Investigaciones en Tecnologías de la Información y Comunicación*. Obtenido de http://citic.ucr.ac.cr/proyecto/nac

Comisión Europea. (2012). *Liberar el potencial de la computación en nube en Europa.* Bruselas.

Coutinho, E., Moreira, L. O., Paillard, G. A., & Maia, J. R. (2014). How to deploy a virtual learning environment in the cloud? En ACM (Ed.), *Proceedings of the 7th Euro American Conference on Telematics and Information Systems.* New York.

ETSI. (2013). *Cloud Standards Coordination* .

Gartner. (2013). *Gartner IT Glossary.* Recuperado el 16 de 8 de 2014, de Cloud Management Platforms: http://www.gartner.com/it-glossary/cloud-management-platforms

Guo, X., Shi, Q., & Zhang, D. (2013). A Study on Moodle Virtual Cluster in Cloud Computing. En IEEE (Ed.), *Internet Computing for Engineering and Science (ICICSE), 2013 Seventh International Conference on* , (págs. 15 - 20). Shanghai.

He, P., Qiu, J., & Zhai, B. (2015). Study on the Integration of Cloud Computing and Moodle Learning Platform. En IEEE (Ed.), *Communication Software and Networks (ICCSN), 2015 IEEE International Conference on*, (págs. 367 - 371). Chengdu.

Herbst, N. R., Kounev, S., & Reussner, R. (2012). Elasticity in Cloud Computing: What It Is, and What It Is Not. *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013).*

ISACA. (2009). Computación en la nube: Beneficios de negocio con perspectivas de seguridad, gobierno y aseguramiento.

ISACA. (2012). Principios rectores para la adopción y el uso de la computación en la nube.

Iyer, S. (11 de 03 de 2011). *ibm.com.* Obtenido de Cloud Computing Central - Chapter 12 - Cloud Users & Roles: https://www.ibm.com/developerworks/community/blogs/c2028fdc-41fe-4493-8257-33a59069fa04/entry/chapter_124?lang=en

Learning Systems Architecture Lab at Carnegie Mellon. (2003). *The SCORM Best Practices Guide for Content Developers.* Pennsylvania: Carnegie Mellon University. Obtenido de http://faculty.coehd.utsa.edu/pmcgee/LearningObjects/guide-v1p0-20030228.pdf

Leimeister, S., Riedl, C., Böhm, M., & Krcmar, H. (2010). The Business Perspective of Cloud Computing: Actors, Roles and Value Networks. *ECIS.*

microsoft.com. (2014). *msdn.microsoft.com.* Recuperado el 17 de 08 de 2014, de Autoscaling Guidance: http://msdn.microsoft.com/en-us/library/dn589774.aspx

Moodle. (2014). *moodle.org.* Obtenido de Moodle Documentation: https://docs.moodle.org/27/en/About_Moodle

Morgado, E., & Schmidt, R. (2012). Increasing Moodle resources through Cloud Computing. En IEEE (Ed.), *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*, (págs. 1 - 4). Madrid.

NIST. (Setiembre de 2011). *The NIST Definition of Cloud Computing.* Obtenido de Information Technology Laboratory: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

openstack.org. (2014). *OpenStack Software.* Recuperado el 23 de 08 de 2014, de OpenStack: The Open Source Cloud Operating System: http://www.openstack.org/software/

Oracle Corp. (2010). *Oracle Help Center.* Recuperado el 26 de marzo de 2016, de Availability and Single Points of Failure: https://docs.oracle.com/cd/E19693-01/819-0992/fjdch/index.html

Orzell, G., & Becker, J. (18 de 1 de 2012). *The Netflix Tech Blog.* Obtenido de Auto Scaling in the Amazon Cloud: http://techblog.netflix.com/2012/01/auto-scaling-in-amazon-cloud.html

Pettey, C., & Goasduff, L. (23 de 06 de 2009). *gartner.com*. Recuperado el 17 de 08 de 2014, de Gartner Highlights Five Attributes of Cloud Computing: http://www.gartner.com/newsroom/id/1035013

Piovesan, S., Hoff do Amaral, E., Arenhardt, C., & Medina, R. (2012). Adaptive Virtual Learning Environment. En IEEE (Ed.), *Global Engineering Education Conference (EDUCON), 2012 IEEE*, (págs. 1 - 6). Marrakech.

Rosen, A. (24 de 5 de 2014). *Aaron's Blog*. Recuperado el 8 de 6 de 2016, de Implementing High Availability Instances with Neutron using VRRP: http://blog.aaronorosen.com/?p=320

Salazar, E. (2011). Resultados del primer curso de traduccióncompletamente virtual e interactivoen Costa Rica. *Revista de Lenguas Modernas*, 337-355.

Siddiqui, M., & Fahringer, T. (2010). *Grid Resource Management: On-demand Provisioning, Advance Reservation, and Capacity Planning of Grid Resources.* Springer.

Stallings, W. (2006). *Organización y arquitectura de computadores* (7a ed. ed.). Madrid: Pearson Educación, S.A.

Sun Microsystems, Inc. (2004). *docs.oracle.com*. Obtenido de Designing a Deployment Architecture : https://docs.oracle.com/cd/E19199-01/817-5759/dep_architect.html

Szabo, M., & Flesher, K. (2002). CMI Theory and Practice: Historical Roots of Learning Management Systems. *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, 929–936.

Tarreau, W. (25 de 05 de 2008). *HAProxy Architecture Guide.* Obtenido de haproxy.org: http://www.haproxy.org/download/1.2/doc/architecture.txt

Tarreau, W. (01 de 02 de 2015). *HAProxy Configuration Manual* . Obtenido de cbonte.github.io: http://cbonte.github.io/haproxy-dconv/configuration-1.5.html

Universidad de Costa Rica. (2010). *Oficio CI-DIR-I-020.* Universidad de Costa Rica, Centro de Informática.

Universidad de Costa Rica. (2011). *Oficio CI-1640-2011.* Universidad de Costa Rica, Centro de Informática.

Universidad de Costa Rica. (2012). Arquitectura propuesta para el sistema METICS. *Instalación del Cluster de Base de Datos de METICS.*

Universidad de Costa Rica. (2012). *Oficio CI-ADR-037-2012.* Universidad de Costa Rica, Centro de Informática.

Universidad de Costa Rica. (2015). *Docencia multiversa.* Obtenido de http://vd.ucr.ac.cr/multiversa/

Universidad de Costa Rica. (2015). *Vicerrectoría de Docencia.* Obtenido de http://vd.ucr.ac.cr/index.php/profesor-invitado/docencia-multiversa

Weygant, P. (2001). *Clusters for High Availability: A Primer of HP Solutions.* New Jersey: Prentice Hall Professional.

10 Anexos

Anexo 1. Componentes de la Nube a configurar

Componente	Requerimiento	Descripción	Configuración
OpenStack	>= Kilo	Gestor de la Nube. El desarrollo del prototipo de trabajo en basado en las funcionalidades de este gestor de Nube.	Entorno básico[1] Servicios de OpenStack (ver Anexo 2)
Hipervisor	ESX[2] >= 5.5 ó KVM	Virtualización de recursos.	vCenter • Datastore(s) • *Cluster(s)* • Host(s)

[1] Manual de instalación de Kilo: http://docs.openstack.org/kilo/install-guide/install/apt/content/

[2] VMWare requiere ciertas licencias para vCenter y ESX. Solo con licencia se tiene acceso al API para uso con Ceilometer y de momento (a partir de OpenStack Icehouse) solo permite utilizar ciertas métricas. Por ejemplo: uso del CPU, uso de disco, memoria y uso de red.

Anexo 2. Servicios de OpenStack a configurar

Tipo de Servicio	Servicio de OpenStack	Configuración	Personalización
Computo	Nova Compute	Controlador para VMWare VMwareVCDriver[3]	Tipos (sabores) personalizados. Llaves ssh para acceso Separación de máquinas virtuales por medio de Zonas de disponibilidad, Agregados o Grupos sin Afinidad Cuotas de CPU, memoria, disco según el grado de adaptabilidad
Almacenamiento	Cinder	Controlador VMwareVcVmdkDrive[4]	
Redes	Nova Network	Controlador FlatDHCPManager Archivo de configuración de dnsmasq[5] Manejo de IP virtuales por parte de las instancias	Red pública Red privada Grupos de seguridad[6]
Imagen	Glance Image	Almacén vmware_datastore.Store[7]	Formato qcow2. Si se utiliza ESX

[3] Si se utiliza VMWare

[4] Si se utiliza VMWare

[5] Se utilizó para sobrescribir e insertar gateway y rutas por defecto

[6] Si se utiliza VMWare, los grupos de seguridad solo son soportados si se utiliza Neutron y el plugin NSX.

[7] Si se utiliza VMWare

			como hipervisor se debe convertir a vmdk.
Telemetría	Ceilometer	Inspector Vsphere[8] Intervalo de sondeo	Acceso a las siguientes métricas[9]: Uso de CPU Uso de memoria Uso de red
Orquestación	Heat	Domain Domain user Sin instance_user	Uso del formato nativo para plantillas: HOT
Identidad	Keystone		Usuario y proyecto en el gestor de la nube.
Metadatos	Metadata API	Acceso por parte de todas las máquinas virtuales al servicio de metadata	

[8] Si se utiliza VMWare

[9] Varían según el tipo de hipervisor: http://docs.openstack.org/admin-guide-cloud/content/section_telemetry-compute-metrics.html

Anexo 3. Otros componentes del entorno de la Nube a configurar

Componente	Requerimientos	Descripción
Imágenes personalizadas	Glance DiskImageBuilder[10] Qemu-img[11] heat-config-script[12] cloud-init[13]	Imágenes previamente configuradas y personalizadas para iniciar las instancias y reducir el tiempo de auto escalamiento.
Plantillas de orquestación	Heat (HOT) Versión de plantillas 2014-10-16	Plantillas HOT para desplegar la arquitectura elástica utilizando el servicio de orquestación HEAT

[10] https://github.com/openstack/diskimage-builder

[11] Esta herramienta convierte de qcow2 a vmdk, este último es el formato que utiliza VMWare

[12] https://github.com/openstack/heat-templates/tree/master/hot/software-config/elements/heat-config-script

[13] https://launchpad.net/cloud-init

Anexo 4. Imágenes personalizadas con Disk Image Builder

			Fase		
	Elemento	**Paquetes**	**root.d**	**pre-install.d**	**install.d**	**post-install.d**
Imagen auto escalamiento / Imagen base	**DEBOOTSTRAP**		▪Jessie		▪dhcp eth1	
	DEBIAN	▪cloud-init ▪heat-config ▪csync2 ▪exim4 ▪ntp		▪Espejos locales ▪Timezone ▪Locale	▪Usuario ▪Cloud-init ▪Correo ▪vm-tools ▪ntp ▪csync2	
	DEBIAN-CONSUL			▪Consul		
	DEBIAN-CONSUL-TEMPLATE			▪Consul-template		
	DEBIAN-WB	▪Apache ▪PHP-FPM ▪lsync2 ▪git				▪git ▪apache mods
	MOODLE	▪graphviz ▪clamav ▪php5-ldap ▪php5-xsl			▪Repo de Moodle	▪Directorio Moodledata ▪Cron

Anexo 5. Prueba 1 - Auto escalabilidad

Introducción
Documentos de referencia
Jmeter Manual 1.2 README MPC Borrador MPC
Descripción
Esta prueba permitirá evaluar si el prototipo es capaz de adaptarse a la demanda y las ventajas o desventajas de esa capacidad. Además ayudará a establecer mejores estrategias o ajustar las que han sido propuestas. Se espera obtener datos que permitan medir el grado de adaptabilidad. En esta prueba se utiliza únicamente el perfil #5 correspondiente al prototipo elástico. La prueba tiene dos ejecuciones, una primera que se identifica como NO_ASG (sin autoscaling group) con una cantidad estática de recursos; y la otra que se identifica como ASG (con autoscaling group) con una asignación de recursos elástica. La prueba utiliza el hardware físico de forma dedicada durante sus ejecuciones.
Objetivos
Los objetivos de la prueba son:
Probar el grado de adaptación del prototipo y sus estrategias para agregar y/o quitar recursos automáticamente cuando se produce un incremento gradual o esporádico en el tráfico de usuarios o bien una disminución.
Documentar los requerimientos, dependencias y resultados de las pruebas.
Enfoque
Se enfocará en la capacidad para adaptarse, las estrategias de auto escalabilidad propuestas y el impacto de auto escalar.
Realizado por:
Job Natán Céspedes Ortiz
Universidad de Costa Rica

		Perfil					
		1	2	3	4	5	Prototipo
Tipo de recursos	Físicos	X					
	Nube		X	X	X	X	
Tipo de asignación de recursos	Estáticos	X	X	X	X		
	Elásticos					X	
Tipo de modelo	Local		X				
	Capas	X		X	X	X	
Alta disponibilidad		X		X			

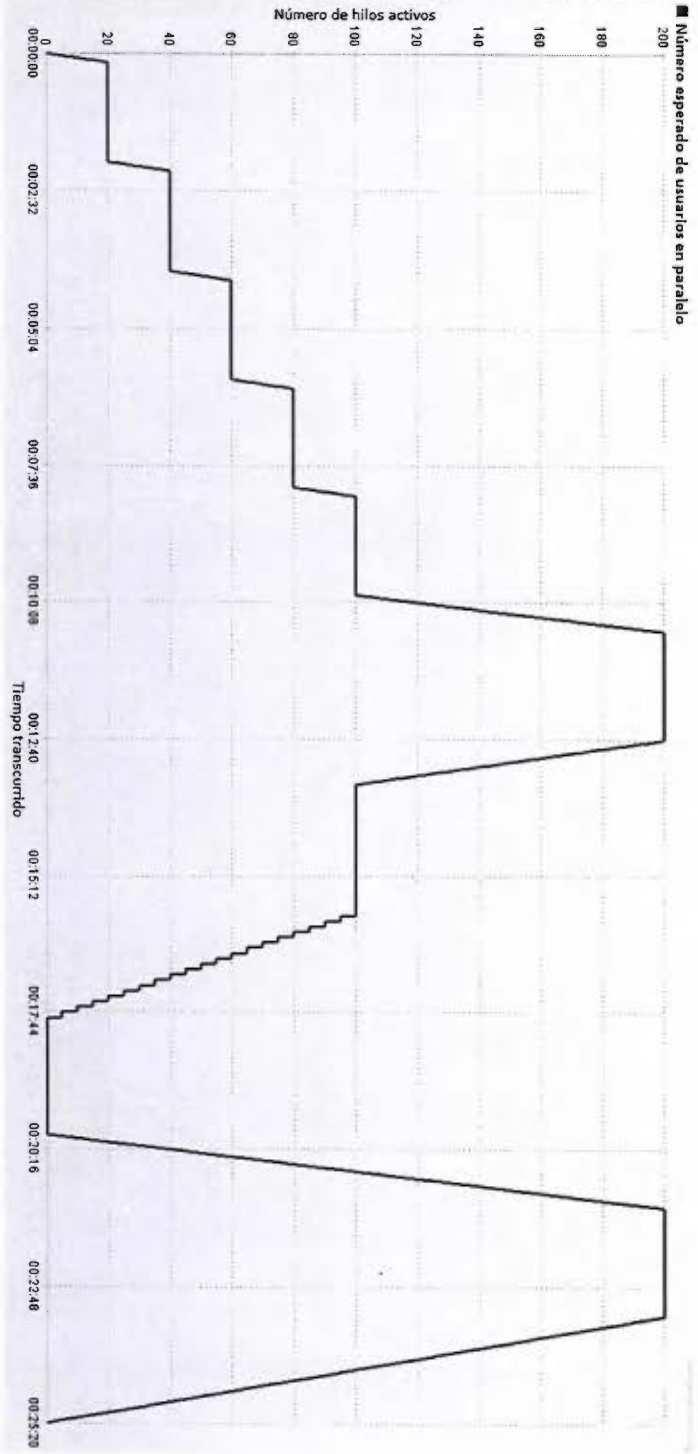
	Perfil						
	1	2	3*	4*	5* Min.	5* Max.	Prototipo
CPU	48	24**	48**	48**	32**	48**	
RAM	64GB	64GB	64GB	64GB	40GB	80GB	
Disco	480GB	60GB	2104GB	120GB	120GB	320GB	

* No se cuentan los recursos de las instancias para los balanceadores implementados.
** Son CPU virtuales o vCPU

Dependencias				
En el servidor de pruebas				
Herramientas:	Nombre	Jmeter	Java JRE	Moodle Perfomance Comparison
	Versión	2.12	1.6	N/A
	Licencia	Apache License, Versión 2.0	N/A	GNU GENERAL PUBLIC LICENSE Versión 2
	Descripción	Automatización de pruebas de carga y rendimiento	Máquina virtual para ejecución de programas java	Creación y ejecución de plan de pruebas jmeter para moodle.
Otros:	Nombre	Plan de pruebas Moodle Jmeter (Basado en el plan de pruebas de la herramienta de Moodle)		
	Descripción	* Archivo de usuarios * Plan de pruebas		
En el perfil a probar				
Otros:	Nombre	Línea base de aplicación funcional y sus dependencias		
	Descripción	* Base de datos (dump) * Moodledata (tar) * Aplicación (git commit)		

Tipo de prueba **Rendimiento-Línea base**		
Definición: Prueba de auto escalabilidad a partir de línea base		
Plan de prueba basado en plan de jmeter de Moodle* **		
Pasos	**Etiqueta**	Traducido
Calentamiento 1	Frontpage not logged	Ver página principal sin autenticarse
Calentamiento 2	Login	Autenticarse
Calentamiento 3	Frontpage logged	Ver página principal autenticado
Calentamiento 4	View course	Ver un curso
Calentamiento 5	View a page activity	Ver una pagina
Calentamiento 6	View course again	Ver el curso de nuevo
Calentamiento 7	View a forum activity	Ver un foro
Calentamiento 8	View a forum discussion	Ver tema del foro
Calentamiento 9	Fill a form to reply a forum discussion	Responder en el foro
Calentamiento 10	Send the forum discussion reply	Enviar respuesta al foro
Calentamiento 11	View course once more	Ver el curso una vez mas
Calentamiento 12	View course participants	Ver los participantes del curso
Calentamiento 13	Logout	Salir
Bucle 1	Frontpage not logged	Ver página principal sin autenticarse
Bucle 2	Login	Autenticarse
Bucle 3	Frontpage logged	Ver página principal autenticado
Bucle 4	View course	Ver un curso
Bucle 5	View a page activity	Ver una pagina
Bucle 6	View course again	Ver el curso de nuevo
Bucle 7	View a forum activity	Ver un foro
Bucle 8	View a forum discussion	Ver tema del foro
Bucle 9	Fill a form to reply a forum discussion	Responder en el foro
Bucle 10	Send the forum discussion reply	Enviar respuesta al foro
Bucle 11	View course once more	Ver el curso una vez mas
Bucle 12	View course participants	Ver los participantes del curso
Bucle 13	Logout	Salir
*** Generador de plan de pruebas para jmeter en Moodle		

Número de usuarios esperado en paralelo (JMETER)



Métricas Jmeter		
Métricas de la herramienta Jmeter utilizadas en la prueba		
Etiqueta	**Unidad**	**Descripción**
# samples	Número	Total de muestras
Average	ms	Promedio de tiempo de respuesta
Std. Dev.	ms	Desviación estándar
Error %	%	Porcentaje de errores registrados
Throughput	Número/tiempo	Muestras por unidad de tiempo
Active Threads	Número	Usuarios activos
Métricas vSphere PowerCLI		
Métricas de la herramienta vSphere PowerCLI utilizadas en la prueba		
Etiqueta	**Unidad**	**Traducido**
cpu.usage.average	%	Promedio de uso del CPU
mem.usage.average	%	Promedio de uso de memoria
disk.usage.average	KBps	Promedio de uso de disco
net.usage.average	KBps	Promedio de uso de red
Otras métricas		
Etiqueta	**Unidad**	**Traducido**
Unidad de escalamiento	Número	Número en uso de unidades de auto escalamiento

Procedimiento para la prueba

Pre-pruebas (única vez):

1. **Línea base**

   Crear una línea base de aplicación para realizar pruebas. Esto incluye respaldar:

   a. Código de repositorio de la aplicación(rama git de línea base)
      1.Crear rama git
      2.En config.php agregar:
         a. Contraseña de usuarios de prueba
            ```
            $CFG->tool_generator_users_password = 'contraseña';
            ```
         b. Deshabilitar envío de correo a usuarios
            ```
            $CFG->noemailever = true;
            ```
      3. Commit de rama git. Este commit será el que se use para las pruebas

   b. Base de datos
      1.En el sitio de moodle:
         a. Crear cursos de prueba. Ej uno mediano y un grande
         b. Cambiar configuración de:

Messaging -> no
debug -> desarrollador
Perfdebug -> si

      2. Respaldar base de datos. Este respaldo es el que se usa para las pruebas

   c. Moodledata
      1.Cambiar nombre de tamaños de prueba en lang si se usa es_mx
      2.Respaldar moodledata. Este respaldo es el que se usa para las pruebas

Prueba:

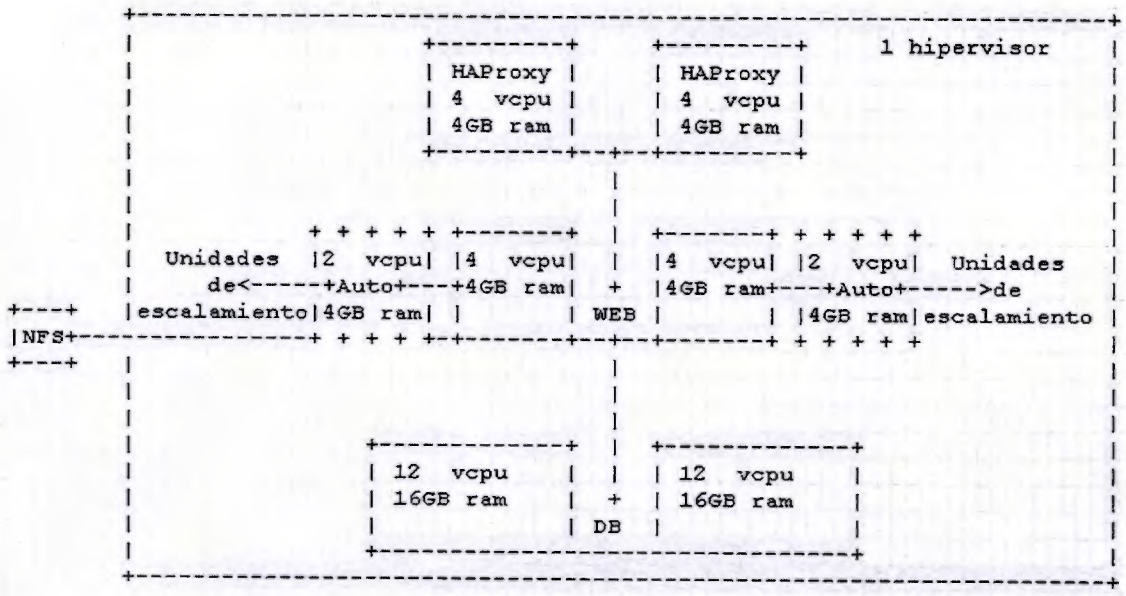
2. **En perfil a probar**
   a. Instalar línea base de pruebas
      i. Clonar código de moodle. Rama de pruebas en mpc/moodle. Revisar que los permisos de mpc/moodle estén bien.
      ii. Base de datos.
      iii. Moodledata.
      iv. Limpiar cache de Moodle

3. **En servidor de pruebas jmeter**
   a. Descargar jmeter en cualquier ruta
   b. Descargar los archivos de plan de prueba y usuarios de prueba
   c. Configurar jmeter_config.properties

Características del servidor de pruebas	
Cpu	8 cores (4 virtual sockets y 2 cores)
Memory	16GB
Disco	100GB
SO	Ubuntu 14.04
Ethernet	1GB
Recursos	Nube/estáticos/locales
Hipervisor	ESX 5.5
Configuración	sysctl -w net.core.somaxconn = 1024
Configuración Jmeter	HEAP="-Xms10000m -Xmx10000m"

Características del perfil 5	
Cpu	min 36 vcpu - max 56 vcpu
Memory	min 48 GB - max 88 GB
Disco	min 160 GB - max 360 GB
SO	Debian Jessie 8.2
Ethernet	1 GB
Recursos	**Nube/elásticos/capas**
Hipervisores	1x ESX 5.5
	Cluster Vcenter, OpenStack con Heat
	Todas las VM en ESX 01
	ESX 5.5 -> disk: thick provisioned
Configuración SO	Utilidades estándar
	Dependencias aplicación
	Ajustes sysctl, apache, php5-fpm
	apache 2.4.10
	libapache2-mod-fastcgi 2.4.10
	php5-fpm 5.6.13
	pgpool 3.3.4
	postgres 9.4
	haproxy 1.5.8
Moodle	moodle 2.7.3+
Rama	(nube-metics) rendimeinto_master
Commit	1942473f62a2ec811657b8ae939a2de7d0059765
Opciones internas de moodle	messaging -> no
	Debugging -> Extra
	perfdebug -> yes
	tema -> elegance (personalizado)
	themedesignermode -> no

Linea base


1 hipervisor
HAProxy
4 vcpu
4GB ram
HAProxy
4 vcpu
4GB ram
Unidades de escalamiento
2 vcpu
4GB ram
Auto
4 vcpu
4GB ram
WEB
4 vcpu
4GB ram
Auto
2 vcpu
4GB ram
Unidades de escalamiento
NFS
12 vcpu
16GB ram
DB
12 vcpu
16GB ram

Anexo 6. Prueba 1 – Rendimiento

Introducción
Documentos de referencia
Jmeter Manual 1.2 README MPC Borrador MPC
Descripción
Esta prueba permitirá evaluar el prototipo respecto a otras soluciones conocidas. Además de que facilita el análisis y la recolección de datos para establecer parámetros aceptables de rendimiento así como posibles ajustes. Se han creado 5 perfiles que se consideran de alguna manera soluciones viables para este tipo de sistemas. Estos perfiles tienen algunas diferencias entre ellos pero mantienen similitudes para hacerlos comparables: una cantidad similar de recursos asignados (CPU, memoria, disco y red), misma línea base (base de datos, código de aplicación, datos), distribución de Linux, sistema de gestión de base de datos. En tanto la categorización que se busca es: el tipo de recursos (físico/Nube), el tipo de asignación (estático/elástico) y el modelo base de la arquitectura (local/capas):
Perfil 1: Recursos físicos, estáticos y en capas con alta disponibilidad.
Perfil 2: Recursos de la Nube y locales
Perfil 3: Recursos de la Nube y en capas con alta disponibilidad.
Perfil 4: Recursos de la Nube y en capas.
Perfil 5: Recursos de la Nube, elásticos y en capas.
Objetivos
Los objetivos de la prueba son:
Establecer líneas base del rendimiento del prototipo en diferentes entornos y compararlas con respecto al entorno auto escalable
Medir y comparar el rendimiento de las variantes del prototipo en entornos habituales para un LMS
Probar y comparar diferentes ajustes a diferentes niveles.
Documentar los requerimientos, dependencias y resultados de las pruebas.
Enfoque
Se enfocará primeramente en el rendimiento de versiones alternativas del protipo según los perfiles. Habrá al menos un perfil donde se cuente con un grado de adaptabilidad por medio de la característica de la elasticidad.
Realizado por:
Job Natán Céspedes Ortiz
Universidad de Costa Rica

		Perfil					
		1	2	3	4	5	Prototipo
Tipo de recursos	Físicos	X					
	Nube		X	X	X	X	
Tipo de asignación de recursos	Estáticos	X	X	X	X		
	Elásticos					X	
Tipo de modelo	Local		X				
	Capas	X		X	X	X	
Alta disponibilidad		X		X			

	Perfil						
	1	2	3*	4*	5* Min.	5* Max.	Prototipo
CPU	48	24**	48**	48**	32**	48**	
RAM	64GB	64GB	64GB	64GB	40GB	80GB	
Disco	480GB	60GB	2104GB	120GB	120GB	320GB	

* No se cuentan los recursos de las instancias para los balanceadores implementados.
** Son CPU virtuales o vCPU

Dependencias				
En el servidor de pruebas				
Herramientas:	Nombre	Jmeter	Java JRE	Moodle Perfomance Comparison
	Versión	2.12	1.6	N/A
	Licencia	Apache License, Versión 2.0	N/A	GNU GENERAL PUBLIC LICENSE Versión 2
	Descripción	Automatización de pruebas de carga y rendimiento	Máquina virtual para ejecución de programas java	Creación y ejecución de plan de pruebas jmeter para moodle.
En el perfil a probar				
Herramientas:	Nombre	Moodle Perfomance Comparison (MPC)		
	Versión	N/A		
	Licencia	GNU GENERAL PUBLIC LICENSE Versión 2		
	Descripción	Creación del curso y del plan de pruebas en moodle utilizando la herramienta MPC.		
Otros:	Nombre	Línea base de aplicación funcional y sus dependencias		
	Descripción	Copia de: * Base de datos (dump) * Moodledata (tar) * Aplicación (git commit)		

Tipo de prueba **Rendimiento-Línea base**					
Definición: Prueba de rendimiento a partir de línea base					
		Usuarios	Bucles	Subida	Capacidad
Parámetros:*		*# de usuarios a simular*	*Iteraciones de los pasos*	*Ventana de tiempo para iniciar los usuarios*	*# muestras por minuto*
Tamaños:**	XS	1	5	1	120.0
	S	30	5	6	120.0
	M	100	5	40	120.0
	L	1000	6	100	120.0
* Documentación de Jmeter					
** Tamaños predefinidos por la herramienta "Moodle Perfomance Comparison" (MPC)					

Plan de prueba de MPC***

Pasos		Etiqueta	Traducido
Calentamiento	1	Frontpage not logged	Ver página principal sin autenticarse
	2	Login	Autenticarse
	3	Frontpage logged	Ver página principal autenticado
	4	View course	Ver un curso
	5	View a page activity	Ver una pagina
	6	View course again	Ver el curso de nuevo
	7	View a forum activity	Ver un foro
	8	View a forum discussion	Ver tema del foro
	9	Fill a form to reply a forum discussion	Responder en el foro
	10	Send the forum discussion reply****	Enviar respuesta al foro****
	11	View course once more	Ver el curso una vez mas
	12	View course participants	Ver los participantes del curso
	13	Logout	Salir
Iteración	1	Frontpage not logged	Ver página principal sin autenticarse
	2	Login	Autenticarse
	3	Frontpage logged	Ver página principal autenticado
	4	View course	Ver un curso
	5	View a page activity	Ver una pagina
	6	View course again	Ver el curso de nuevo
	7	View a forum activity	Ver un foro
	8	View a forum discussion	Ver tema del foro
	9	Fill a form to reply a forum discussion	Responder en el foro
	10	Send the forum discussion reply****	Enviar respuesta al foro ****
	11	View course once more	Ver el curso una vez mas
	12	View course participants	Ver los participantes del curso
	13	Logout	Salir
		*** Plantilla de plan de pruebas de "Moodle Perfomance Comparison" (MPC)	

**** No se considera este paso ni se incluye en los resultados.

Métricas MPC por cada paso		
Métricas reportadas por Moodle a MPC		
Etiqueta	**Unidad**	**Traducido**
tiempoBD	ms	Tiempo de las consultas a BD para servir URL
memoriaUsada	MB	Memoria utilizada para servir URL
cargaServidor	Número	Promedio de la carga del servidor al servir URL
tiempoCarga	ms	Tiempo de carga de la página
Std. Dev.	ms	Desviación estándar
Métricas Jmeter		
Métricas de la herramienta Jmeter utilizadas en la prueba		
Etiqueta	**Unidad**	**Descripción**
Average	ms	Promedio de tiempo de respuesta
Std. Dev.	ms	Desviación estándar

Procedimiento para la prueba

Pre-pruebas (única vez):

2. Línea base

Crear una línea base de aplicación para realizar pruebas. Esto incluye respaldar:

b. Código de repositorio de la aplicación(rama git de línea base)

1.Crear rama git

2.En config.php agregar:

a. Contraseña de usuarios de prueba

```
$CFG->tool_generator_users_password = 'contraseña';
```

c. Deshabilitar envío de correo a usuarios

```
$CFG->noemailever = true;
```

4. Commit de rama git. Este commit será el que se use para las pruebas

c. Base de datos

1.En el sitio de moodle:

a. Crear cursos de prueba. Ej uno mediano y un grande

b. Cambiar configuración de:

Messaging -> no
debug -> desarrollador
Perfdebug -> si

3. Respaldar base de datos. Este respaldo es el que se usa para las pruebas

d. Moodledata

1.Cambiar nombre de tamaños de prueba en lang si se usa es_mx

2.Respaldar moodledata. Este respaldo es el que se usa para las pruebas

Prueba:

3. En perfil a probar

a. Instalar MPC en raíz de sitio apache (Asegurar el acceso)

i. Descargar herramienta mpc

ii. Configurar el virtual host de apache para mpc y mpc/moodle

1. Ej:

```
DocumentRoot /var/www/mpc/moodle
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

<Directory /var/www/mpc/moodle>
    Options Indexes FollowSymLinks
MultiViews
    AllowOverride None
    order deny,allow
```

```
        deny from all
        allow from 10  172 127
    </Directory>
```

iii. Reiniciar apache

b. Instalar línea base de pruebas
   i. Instalar llave ssh con acceso a repo de repositorio de la aplicación
   ii. Clonar código de moodle. Rama de pruebas en mpc/moodle. Revisar que los permisos de mpc/moodle estén bien.
   iii. Base de datos.
   iv. Moodledata.

c. Crear plan actualizando contraseñas

```
size="XS"
sudo php /var/www/mpc/moodle/admin/tool/generator/cli/maketestplan.php --size="${size}" --shortname="PruebaMed" --updateuserspassword
```

d. Configurar al archivo webserver_config.properties

e. Crear archivo site_data.properties desde mpc/

```
./create_site_data_file.sh
```

4. En servidor de pruebas jmeter

a. Descargar jmeter en cualquier ruta
b. Descargar mpc en raíz de sitio apache
c. Asegurar sitio mpc en virtualhost de apache
d. Configurar jmeter_config.properties
e. Descargar los archivos de plan de prueba y usuarios de prueba
f. Descargar el archivo site_data.properties
g. Correr la prueba con el script "test_runner.sh" en mpc/

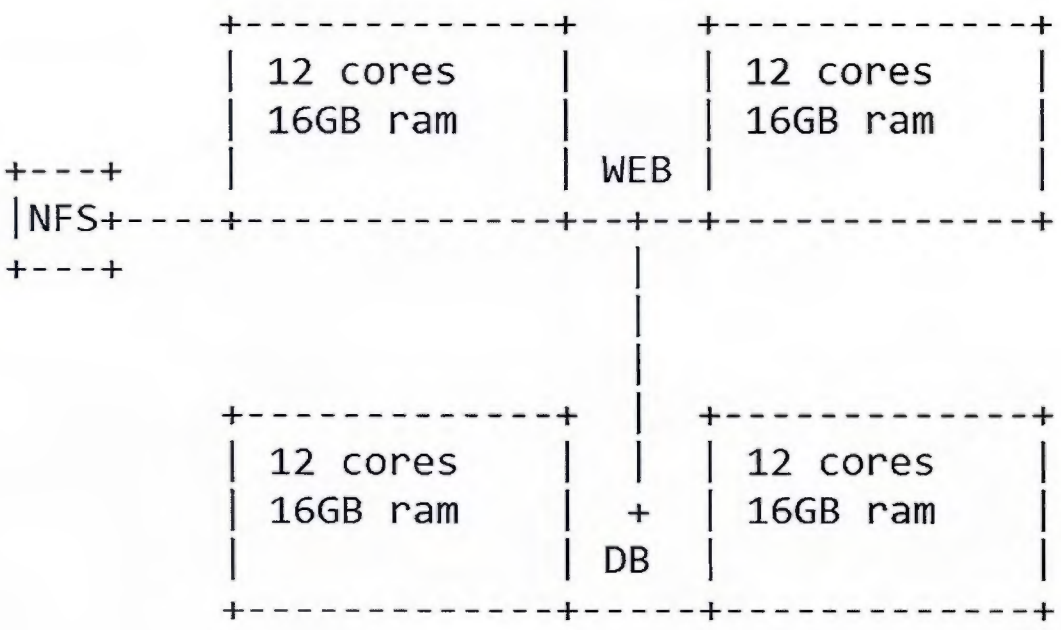
Servidor de pruebas

Características del servidor de pruebas	
Cpu	16 cores (4 virtual sockets y 4 cores)
Memory	64GB
Disco	30GB
SO	Debian Wheezy 7.8
Ethernet	1GB
Recursos	Nube/estáticos/locales
Hipervisor	ESX 5.5
Opciones de instancia	ESX 5.5 -> memorylocked (64gb)
	ESX 5.5 -> thick provisioned
Configuración	sysctl -w kernel.shmmax=51539607552
	sysctl -w kernel.shmall=4294967295
	echo 4096 > /proc/sys/vm/nr_hugepages
Configuración Jmeter	HEAP="-Xms49152m -Xmx49152m"

Perfil #1

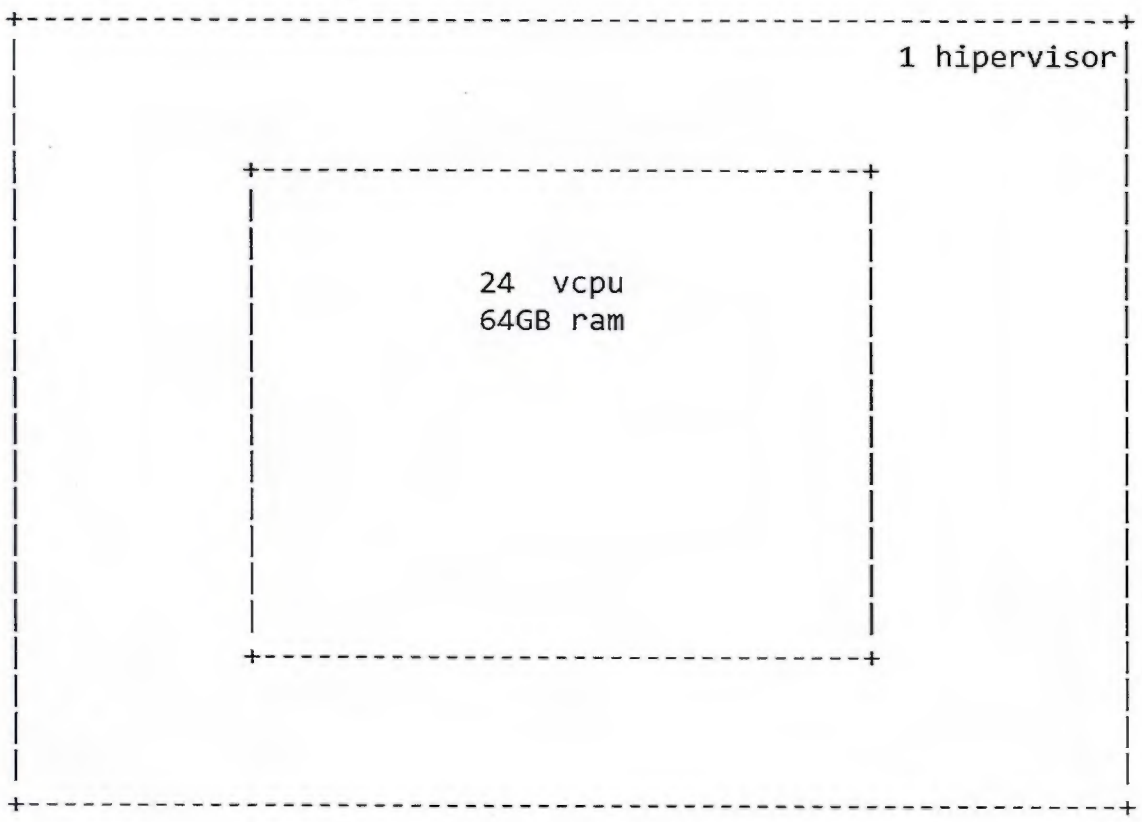
Características del perfil 1	
Cpu	48 cores
Memory	64GB
Disco	60GB
SO	Debian Wheezy 7.8
Ethernet	1GB
Recursos	**Físicos/estáticos/capas/HA**
Configuración SO	Utilidades estándar
	Dependencias aplicación
	apache 2.2.22
	libapache2-mod-fastcgi 2.4.7
	php5-fpm 5.5.21
	pgpool 3.3.4
	postgres 9.3
Moodle	moodle 2.7.3+
Rama	(nube-metics) rendimeinto_master
Commit	316c65bb02055213865e80794ca4449533f056d6
Opciones internas de moodle	messaging -> no
	Debugging -> Extra
	perfdebug -> yes tema -> elegance (personalizado)
	themedesignermode -> no

Línea base


12 cores
16GB ram
12 cores
16GB ram
WEB
NFS
12 cores
16GB ram
DB
12 cores
16GB ram

Perfil #2

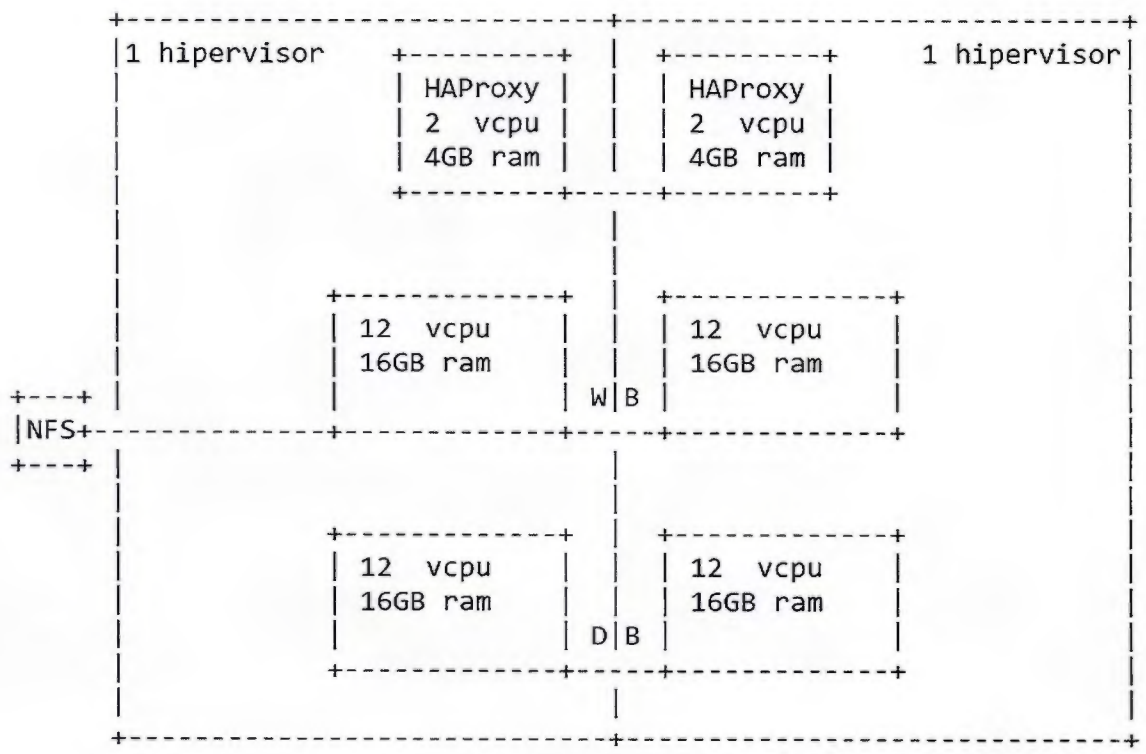
Características del perfil 2		
Cpu	24 cores (6 virtual sockets y 4 cores)	
Memory	64GB	
Disco	60GB	
SO	Debian Wheezy 7.8	
Ethernet	1GB	
Recursos	**Nube/locales**	
Hipervisor	ESX 5.5	
Opciones de instancia	ESX 5.5 -> memorylocked (64gb)	
	ESX 5.5 -> thick provisioned	
Configuración SO	Utilidades estándar	
	Dependencias de la aplicación	
	apache 2.2.22	
	php5 5.4	
	postgres 9.1	
Moodle	moodle 2.7.3+	Linea base
Rama	(nube-metics) rendimeinto_master	
Commit	316c65bb02055213865e80794ca4449533f056d6	
Opciones internas	messaging -> no	
	Debugging -> Extra	
	perfdebug -> yes tema -> elegance (personalizado)	
	themedesignermode -> no	


1 hipervisor
24 vcpu
64GB ram

Perfil #3

Características del perfil 3	
Cpu	52 cores
Memory	72 GB
Disco	2144 GB
SO	Debian Jessie 8.2
Ethernet	1 GB
Recursos	**Nube/capas/HA**
Hipervisores	2x ESX 5.5
	Cluster VCenter
	RDS
	Rol master ESX 01
	Rol backup ESX 02
	ESX 5.5 -> disk: thin provisioned
Configuración SO	Utilidades estándar
	Dependencias aplicación
	Ajustes sysctl, apache, php5-fpm
	apache 2.4.10
	libapache2-mod-fastcgi 2.4.10
	php5-fpm 5.6.13
	pgpool 3.3.4
	postgres 9.4
	haproxy 1.5.8
Moodle	moodle 2.7.3+
Rama	(nube-metics) rendimeinto_master
Commit	316c65bb02055213865e80794ca4449533f056d6
Opciones internas de moodle	messaging -> no
	Debugging -> Extra
	perfdebug -> yes
	tema -> elegance (personalizado)
	themedesignermode -> no

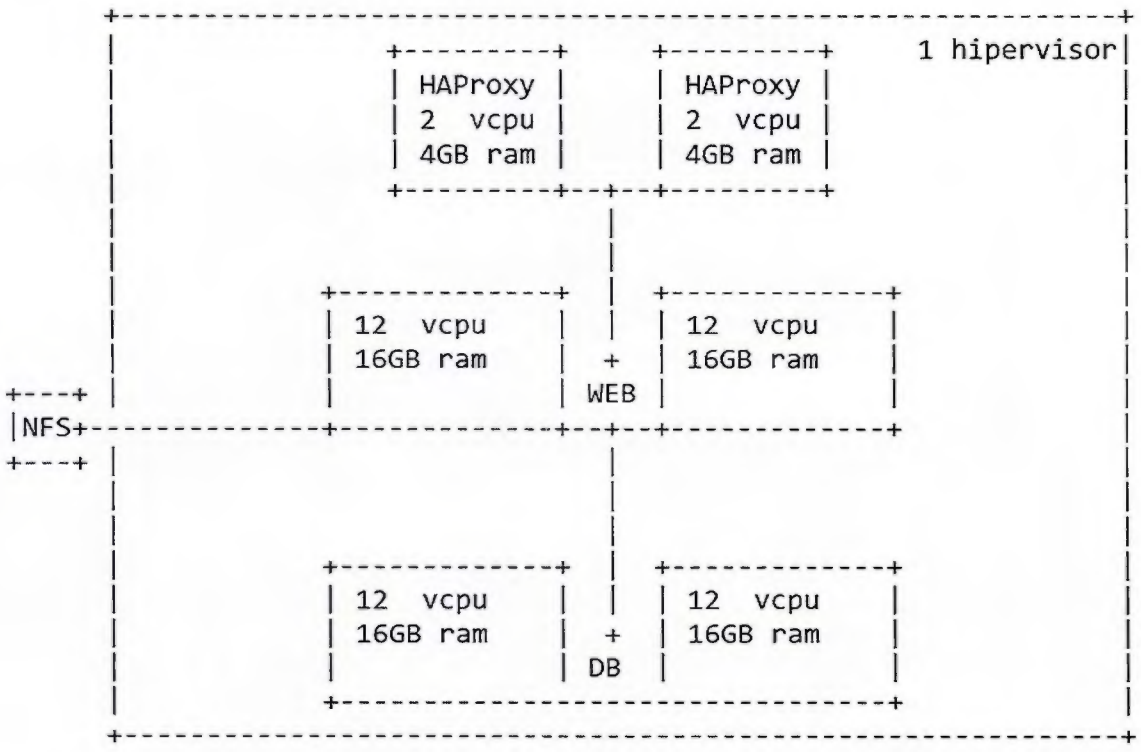
Línea base


1 hipervisor
HAProxy
2 vcpu
4GB ram
HAProxy
2 vcpu
4GB ram
1 hipervisor
12 vcpu
16GB ram
12 vcpu
16GB ram
W
B
NFS
12 vcpu
16GB ram
12 vcpu
16GB ram
D
B

Perfil #4

Características del perfil 4	
Cpu	52 cores
Memory	72 GB
Disco	160 GB
SO	Debian Jessie 8.2
Ethernet	1 GB
Recursos	**Nube/capas**
Hipervisores	1x ESX 5.5
	Cluster VCenter
	Todas las VM en ESX 01
	ESX 5.5 -> disk: thick provisioned
Configuración SO	Utilidades estándar
	Dependencias aplicación
	Ajustes sysctl, apache, php5-fpm
	apache 2.4.10
	libapache2-mod-fastcgi 2.4.10
	php5-fpm 5.6.13
	pgpool 3.3.4
	postgres 9.4
	haproxy 1.5.8
Moodle	moodle 2.7.3+
Rama	(nube-metics) rendimeinto_master
Commit	316c65bb02055213865e80794ca4449533f056d6
Opciones internas de moodle	messaging -> no
	Debugging -> Extra
	perfdebug -> yes
	tema -> elegance (personalizado)
	themedesignermode -> no

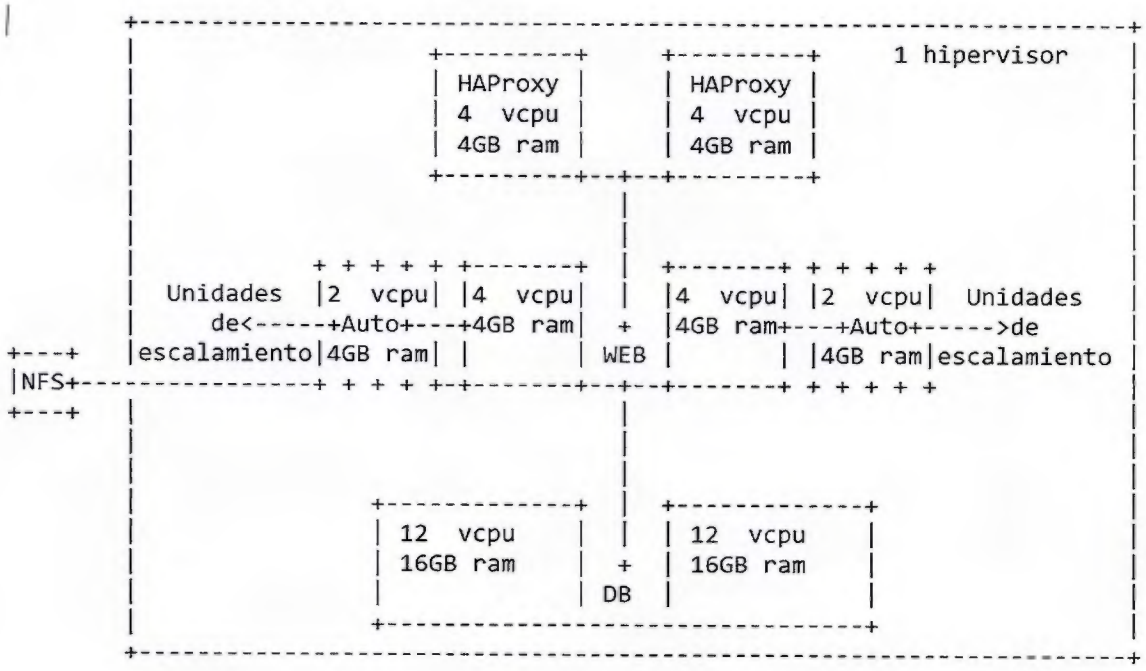
Línea base


1 hipervisor
HAProxy
2 vcpu
4GB ram
HAProxy
2 vcpu
4GB ram
12 vcpu
16GB ram
WEB
12 vcpu
16GB ram
NFS
12 vcpu
16GB ram
DB
12 vcpu
16GB ram

Perfil #5

Características del perfil 5	
Cpu	min 40 vcpu - max 56 vcpu
Memory	min 48 GB - max 88 GB
Disco	min 160 GB - max 360 GB
SO	Debian Jessie 8.2
Ethernet	1 GB
Recursos	**Nube/elásticos/capas**
Hipervisores	1x ESX 5.5
	Cluster Vcenter, OpenStack con Heat
	Todas las VM en ESX 01
	ESX 5.5 -> disk: thick provisioned
Configuración SO	Utilidades estándar
	Dependencias aplicación
	Ajustes sysctl, apache, php5-fpm
	apache 2.4.10
	libapache2-mod-fastcgi 2.4.10
	php5-fpm 5.6.13
	pgpool 3.3.4
	postgres 9.4
	haproxy 1.5.8
Moodle	moodle 2.7.3+
Rama	(nube-metics) rendimeinto_master
Commit	1942473f62a2ec811657b8ae939a2de7d0059765
Opciones internas de moodle	messaging -> no
	Debugging -> Extra
	perfdebug -> yes
	tema -> elegance (personalizado)
	themedesignermode -> no

Línea base


1 hipervisor
HAProxy
4 vcpu
4GB ram
HAProxy
4 vcpu
4GB ram
Unidades de escalamiento
2 vcpu
4GB ram
Auto
4 vcpu
4GB ram
WEB
4 vcpu
4GB ram
Auto
2 vcpu
4GB ram
Unidades de escalamiento
NFS
12 vcpu
16GB ram
DB
12 vcpu
16GB ram