UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSTGRADO

MAESTRÍA EN INGENIERÍA DE SOFTWARE

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

Trabajo de Investigación previo a la obtención del Título de Magíster en Ingeniería de Software.

Director:
Ing. Freddy Mauricio Tapia León MSc.

Autor:
Ing. Alexis Fernando Saransig Chiza.

IBARRA - ECUADOR

2018

APROBACIÓN DEL TUTOR

UNIVERSIDAD TÉCNICA DEL NORTE
Resolución N° 001-073 CEAACES -2013-13
INSTITUTO DE POSTGRADO

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Grado, presentado por el maestrante Alexis Fernando Saransig Chiza, para optar por el grado de Magister en Ingeniería de Software, doy fé de que dicho trabajo reúne los requisitos y méritos suficientes para ser sometido a presentación (pública o privada) y evaluación por parte del jurado examinador que se designe.

En la ciudad de Ibarra a los 2 días del mes de agosto de 2018.

Ing. Freddy Tapia MSc.

APROBACIÓN DEL ASESOR

UNIVERSIDAD TÉCNICA DEL NORTE
Resolución N° 001-073 CEAACES -2013-13
INSTITUTO DE POSTGRADO

APROBACIÓN DEL ASESOR

"Análisis de rendimiento entre una Arquitectura Monolítica y una Arquitectura de Microservicios – Tecnología Basada en Contenedores."

Por: Alexis Fernando Saransig Chiza.

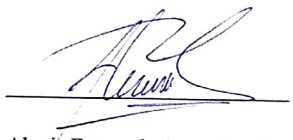
Trabajo de Grado de Maestría aprobado en nombre de la Universidad Técnica del Norte, por el aseosr, a los 30 días del mes de julio de 2018.

Ing. Diego Trejo MSc.

AUTORÍA

AUTORÍA

Yo, Alexis Fernando Saransig Chiza, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentada para ningún grado, ni calificación profesional, que he consultado referencias bibliográficas que se incluyen en este documento y que todos los datos presentados son resultado de mi trabajo.



Alexis Fernando Saransig Chiza

C.C. 1003043872

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. Identificación de la obra

La Universidad Técnica del Norte dentro del proyecto Repositorio Digital Institucional, determinó la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
Cédula de ciudadanía:	1003043872
Apellidos y nombres:	Saransig Chiza Alexis Fernando
Dirección:	Monserrath – Otavalo
Email:	afsaransigc@utn.edu.ec
Teléfono:	0985528967
DATOS DE LA OBRA	
Título:	"Análisis de rendimiento entre una Arquitectura Monolítica y una Arquitectura de Microservicios – Tecnología basada en Contenedores"
Autor:	Saransig Chiza Alexis Fernando
Fecha:	02/08/2018
SÓLO PARA TRABAJOS DE GRADO	
Programa:	Pregado ( ) Postgrado (X)
Título por el que opta:	Magister en Ingeniería de Software
Tutor:	Ing. Freddy Tapia MSc.

2. Autorización de uso a favor de la Universidad Técnica del Norte

Yo, Alexis Fernando Saransig Chiza, con cédula de ciudadanía número 1003043872, en calidad de autor y titular de los derechos patrimoniales del trabajo de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

3. Constancia

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que se asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 2 días del mes de agosto del 2018.

EL AUTOR

Alexis Fernando Saransig Chiza

C.C. 1003043872

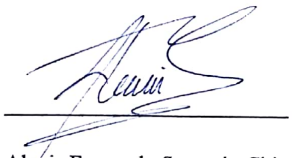
CESIÓN DE DERECHOS DEL AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DEL AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Alexis Fernando Saransig Chiza, con cédula de ciudadanía número 1003043872, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5 y 6, en calidad de autor del trabajo de grado denominado: "**Análisis de rendimiento entre una Arquitectura Monolítica y una Arquitectura de Microservicios – Tecnología basada en contenedores**", trabajo de investigación elaborado para optar por el título de Magister en Ingeniería de Software en la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.



Alexis Fernando Saransig Chiza

C.C. 1003043872

DEDICATORIA

A MI FAMILIA, que ha crecido en número, a cada uno con sinceridad.

A MI ESPOSA Y MI HIJA, a quienes amo con todo mi ser y son parte de mis logros.

A MI AMIGO, quien se adelantó a un mejor lugar, con quien teníamos el sueño de obtener un título de maestría y más, sinónimo de superación personal y profesional. Mi dedicación Ismael Castro, Sensei como lo conocía.

AGRADECIMIENTO

A Dios, por la vida, por estar presente en las bendiciones que me dan mis padres, por ser parte de mis logros y ayudarme a superar los fracasos.

A mi familia, por el apoyo, por estar pendientes de mi bienestar e incentivarme a culminar mis estudios. En especial, a mis padres Mercedes Chiza y José Saransig que harían posible lo imposible por verme feliz, a mi hermano Suri que lo quiero tanto, a mis padrinos Andrea y Luis por sus consejos, mis sobrinas Aylin y Akemi son mi tesoro, mis suegros Luzmila Quinchuquí y José de la Torre por la esposa maravillosa que tengo.

A mi esposa y mi hija, Gabriela y Amaia que sin duda son mi mayor bendición y que con su amor llenan de felicidad mi vida. Gracias por la paciencia y por comprenderme en las situaciones difíciles, con todo mi amor para ustedes.

A la Universidad Técnica del Norte y el Instituto de Postgrado, por abrirme las puertas y darme la oportunidad de instruirme en una carrera de maestría de alto nivel.

A los Docentes, por compartir sus conocimientos y su valioso tiempo.

Al Ing. Freddy Tapia MSc., por la tutoría en el trabajo de grado, por la paciencia, el tiempo, pero sobretodo por la amistad que vale la pena.

A mis compañeros, por la amistad y la unión desde la primera clase. Por formar un grupo de amigos con quienes se compartió gratos momentos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD TÉCNICA DEL NORTE
INSTITUTO DE POSGRADO
PROGRAMA DE MAESTRÍA

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

Autor: Ing. Alexis Fernando Saransig Chiza.
Tutor: Ing. Freddy Mauricio Tapia León MSc.
Año: 2018

RESUMEN

La evolución tecnológica, apunta a ser más eficiente en el uso de los recursos. La producción de Software, con el tiempo ha manejado diferentes tipos de arquitecturas con el fin de que cada producto cumpla los objetivos funcionales y sea eficiente en el uso de los recursos. Este es el caso de la Arquitectura Monolítica, muy reconocida en la producción de Software, su fusión con las Máquinas Virtuales, la ha convertido en una fórmula exitosa y efectiva para proyectos pequeños y de gran escala. La innovación ha dado lugar a nuevas arquitecturas que proponen óptimas soluciones para mejorar el proceso de producción de Software. La Arquitectura de Microservicios, va ganando terreno e incuestionablemente será parte en la toma de decisiones de los DevOps para futuros proyectos por las ventajas que esta presenta. La tecnología de Contenedores es aún poco conocida en nuestro entorno, sin embargo, las tendencias muestran que hay más acogida por esta tecnología que brinda un manejo más eficiente de los recursos en comparación a las Máquinas Virtuales. En esta investigación, se hace un análisis comparativo de rendimiento entre una Aplicación con Arquitectura Monolítica ejecutándose sobre una Máquina Virtual contra la misma Aplicación, pero esta vez basada en una Arquitectura de Microservicios y usando Contenedores, ambas combinaciones corriendo sobre un equipo con las mismas características. Se somete cada ambiente a pruebas de estrés y se analiza posteriormente los datos en bruto almacenados en archivos de logs, el resultado con la comparación correspondiente permite hacer una toma de decisiones directamente enfocada en el manejo eficiente de los recursos y la eficiencia de la producción de Software. Además, se exponen ventajas relacionadas a las nuevas metodologías de desarrollo que no se detallan en profundidad en la presente investigación, sin embargo, quedan abiertas para posteriores investigaciones.

Palabras clave: arquitectura, monolítica, microservicios, contenedores, rendimiento.

UNIVERSIDAD TÉCNICA DEL NORTE
INSTITUTO DE POSGRADO
PROGRAMA DE MAESTRÍA

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

Autor: Ing. Alexis Fernando Saransig Chiza.
Tutor: Ing. Freddy Mauricio Tapia León MSc.
Año: 2018

ABSTRACT

Technological evolution aims to be more efficient in the use of resources. The production of Software, over time has handled different types of architectures in order that each product meets the functional objectives and is efficient in the use of resources. This is the case of the Monolithic Architecture, very recognized in the production of Software, its fusion with the Virtual Machines, has turned it into a successful and effective formula for small and large-scale projects. Innovation has given rise to new architectures that propose optimal solutions to improve the Software production process. The Microservices Architecture is gaining ground and will unquestionably be part of the DevOps decision making for future projects due to the advantages it presents. Container technology is still little known in our environment; however, trends show that there is more acceptance for this technology that provides a more efficient management of resources compared to Virtual Machines. In this research, a comparative performance analysis is made between an application with Monolithic Architecture running on a Virtual Machine against the same application, but this time based on a Microservices Architecture and using Containers, both combinations running on a computer with the same characteristics. Each environment is subjected to stress tests and the raw data stored in log files is subsequently analyzed, the result with the corresponding comparison allows making a decision directly focused on the efficient management of resources and the efficiency of the production of Software. In addition, advantages related to the new development methodologies that are not detailed in the present investigation and that are open for further investigations are exposed.

Keywords: architecture, monolithic, microservices, containers, performance.

CAPÍTULO I

EL PROBLEMA

Tema

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

1.1 Contextualización del problema

La evolución tecnológica no da tregua, siendo esta una ventaja para la innovación e investigación, donde los límites llegan hasta donde el conocimiento así lo permite. Es así que en la actualidad, dichas innovaciones tienden a hacerse más eficientes y un verdadero soporte al desarrollo de las sociedades. Por mencionar un ejemplo, una de las mayores preocupaciones es el impacto que las nuevas tecnologías puedan causar al medio ambiente, ya en el año de 1992 aparece la Tecnología de la Información – TI verde[1], un término usado para hacer referencia al gran impacto ambiental que genera el hacer mal uso de las TI (a nivel social y empresarial). Siendo esta tendencia aplicable a diferentes escenarios como por ejemplo: Centros de Datos, Computación Móvil, Sistemas Embebidos y Aplicaciones Web (Basar Bener, Morisio, Miranskyy, & Akinii Kocak, 2014).

Así también, la producción de Software se ha incrementado de forma exponencial, lo que exige un desarrollo en la infraestructura físico-tangible para su debida operación. Sin embargo, si de optimizar recursos se trata, no se puede limitar a crear Software o incrementar Hardware a la par, sino más bien pensar en la forma de reutilizar recursos y reducir costos. En la década de los 60's, surgió el mismo planteamiento por parte de International Business Machine, IBM[2] por sus siglas en inglés, dando como resultado el desarrollo del primer método de Virtualización, el cual consiste en hacer particiones lógicas, donde cada partición trabaja independientemente y haga uso de los recursos

[1] Uso de la Tecnología de la Información con aplicabilidad del concepto de Desarrollo Sostenible.
[2] Empresa multinacional estadounidense de tecnología y consultoría con sede en, Nueva York – EEUU.

compartidos provistos por una supercomputadora, esto implicaría una optimización y un mejor manejo de los recursos (Meier, Virun, Blumert, & Jones, 2008). Desde entonces, la virtualización ha sido parte de las innovaciones tecnológicas que hasta la actualidad conocemos y utilizamos masivamente.

Ahora bien, las Aplicaciones Web son tendencias por sus potentes funcionalidades, adaptabilidad y buena aceptación por parte de las empresas que desarrollan Software, siendo estas Aplicaciones, en su mayoría diseñadas con una Arquitectura Monolítica[3] – AM (Ver sección 2.4.1.1), como base para su implementación e implantación. Comúnmente esta Arquitectura se compone de tres capas: interfaz de usuario, lógica de la aplicación y el sistema gestor de datos, todo esto se ejecuta en un único servidor. Esta última, es la parte clave donde se limita la escalabilidad y existen altos riesgos de que un fallo tenga un impacto crítico a nivel global en el sistema. Ahora, conociendo las capacidades de un servidor, imaginemos que hay varias aplicaciones con tecnología diferente ejecutándose sobre este, y en un intento de optimizar recursos se utilizan Máquinas Virtuales – MV para emular el Hardware subyacente. Aquí se genera la interrogante ¿Se han optimizado suficiente y adecuadamente los recursos?, más aún cuando todo profesional conoce las consecuencias de virtualizar una AM en una MV. Según (Prashant, 2016), previo a cualquier proceso de virtualización, se deben considerar los siguientes recursos como factores importantes de análisis de desempeño:

- CPU
- La Memoria RAM
- La Red
- Disco
- Aplicaciones o Servicios

Con estas consideraciones, podemos hacer mención de algunos puntos problemáticos que conlleva virtualizar una AM usando MV:

[3] Modelo tradicional unificado para diseñar Software.

- **Gestión eficiente de Recursos (Hardware)**. Las MV requieren en su mayoría una instalación completa del Sistema Operativo – SO, atribuyendo esto al consumo excesivo de recursos de Hardware, pues el sistema en ejecución no diferencia una máquina física de una virtual, y dependerá mucho de la configuración para mejorar o limitar su rendimiento.

- **Portabilidad de servicio**. Algunos expertos en Desarrollo de Software optan por hacer una documentación detallada para poder reproducir las mismas condiciones en otra computadora o servidor, mientras otros expertos optan por exportar la MV preconfigurada con la AM, lo que resulta en generar un archivo de gran tamaño difícil de llevar en un disco o memoria USB.

- **Escalabilidad de Servicio**. Es limitado y aunque depende principalmente de los recursos disponibles, una AM no es lo suficientemente flexible como para escalar y su jerarquía vertical la vuelve incluso más complicada de ejecutar este proceso. Posiblemente se piense en hacer una clonación de la MV, sin embargo, esto significa mayor consumo de recursos y podría terminar ralentizando procesos y deteriorando el desempeño alcanzado.

- **Versión de la Aplicación o Servicio**. Las actualizaciones son en su mayoría un problema ya sea por temas de compatibilidad con las versiones anteriores y tiempo de ejecución. Además de que es difícil tener varias versiones de una Aplicación en el mismo sistema, donde el proceso de instalar y desinstalar Software hasta encontrar la versión correcta es algo tedioso y estresante.

- **Compatibilidad Multiplataforma**. Aunque en la actualidad la mayoría de las Aplicaciones presentan versiones multiplataforma, algunas requieren de Software o librerías extras para poder funcionar, esto no asegura tener un ambiente íntegro o libre de errores a futuro.

- **Trabajo en equipo**. Significa que todos los involucrados y usuarios del servicio virtualizado deben de compartir un mismo ambiente, lo cual sería ideal, sin embargo, para ello se necesitaría contar con equipos similares en cuanto a

Hardware se refiere. Se lo podría suplir usando MV's, pero con los inconvenientes mencionados sabemos que no es una solución óptima.

El Gráfico 1, muestra los orígenes del problema y sus consecuencias, referente a la Arquitectura Monolítica.

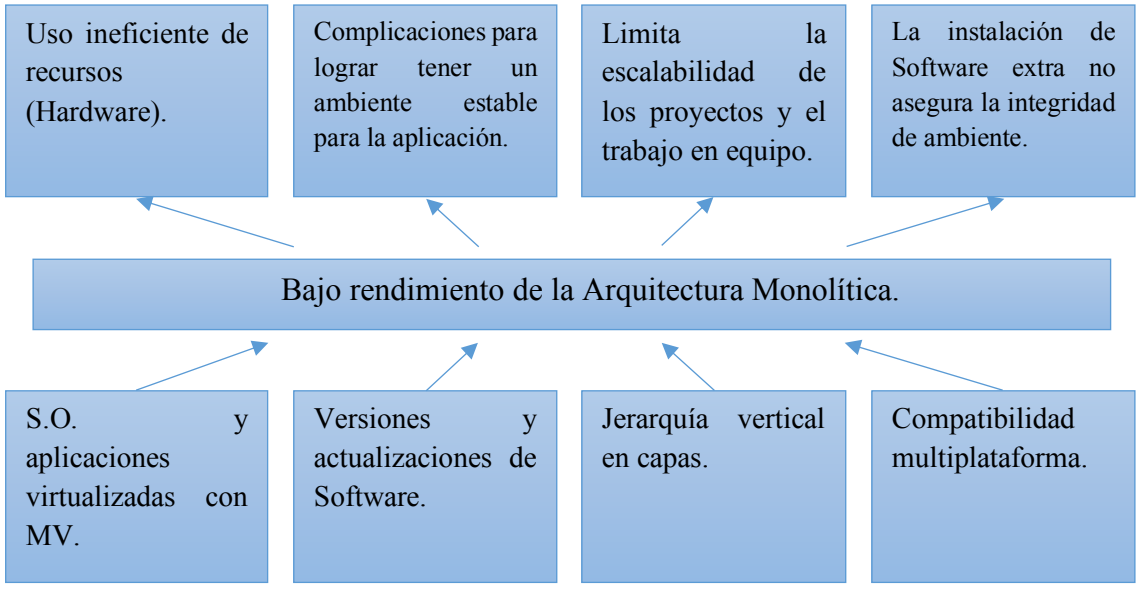


Gráfico 1: Árbol de Problemas.
Elaborado por: Investigador.

1.2 Planteamiento del problema

En el área informática la innovación es constante, siendo el Software un elemento decisorio y clave al momento de implementar propuestas dinámicas, mismas que faciliten y optimicen la Toma de Decisiones. Donde, un modelo adecuado y estable de desarrollo permite una puesta en producción eficiente asegurando resultados y optimización de recursos.

Así también, para la Virtualización de Servicios, tanto el Hardware como el Software deben cumplir con cierto nivel de compatibilidad, pues el rendimiento dependerá mucho de las características que cada uno tenga, además de que en relación al tiempo de ejecución no deberían tener mayor diferencia el uno del otro.

Como se ha evidenciado el tema de rendimiento y adaptabilidad (Arquitectura Monolítica – AM/Máquina Virtual – MV), es la hipótesis a validar y objetivos de la

presente investigación, mismos que influyen directamente en la producción y ejecución de Software, así como de los servicios a ofrecer e impacto que estas producen.

Con todos estos antecedentes, se hace necesario incursionar en estas áreas, con el objetivo de proveer indicadores y aspectos que faciliten una toma de decisiones más óptima y basada en datos reales, finalmente se deja de manifiesto cuáles podrían ser posibles trabajos futuros o trabajos complementarios a la presente investigación.

1.3 Formulación del problema

Incidencia en el rendimiento de los recursos de Hardware con una Arquitectura Monolítica o una Arquitectura de Microservicios basada en Contenedores.

1.4 Justificación de la investigación

Los cambios tecnológicos en algunos escenarios permiten la masificación de servicios, así como también su optimización y fácil acceso, donde la reducción de costos y mejora en los servicios son producto de una adecuada implementación y/o ejecución de los mismos. Uno de estos servicios es la Virtualización basada en Contenedores, la cual toma a las Máquinas Virtuales – MV y Arquitecturas Monolíticas – AM, como objetos de estudio en términos de realizar comparativas que ayuden a plantear y conseguir los objetivos definidos en esta investigación.

La presente investigación propone analizar el uso de Contenedores y la Arquitectura de Microservicios – AMS (Ver sección 2.4.1.9), tendencias relativamente nuevas, que aparecen en el año 2013, y que desde su lanzamiento hasta la presente fecha, han sido ampliamente aceptadas no sólo por empresas que desarrollan Software, sino por gigantes tecnológicos como: Google, Microsoft y Amazon, empresas que han decidido migrar y hacer uso de estas tecnologías para ofrecer servicios más óptimos y acorde a las nuevas exigencias tecnológicas.

1.5 Objetivos de la investigación

1.5.1 Objetivo general

Hacer un análisis comparativo de rendimiento entre la Arquitectura de Microservicios basada en Contenedores y la Arquitectura Monolítica.

1.5.2 Objetivos específicos

- Estudiar el contexto evolutivo de la Arquitectura de Software, enfatizando en la Arquitectura Monolítica y la de Microservicios, para obtener un Marco Teórico que sustente la presente investigación.
- Desarrollar una investigación diagnóstica que permita conocer los factores que influyen en el rendimiento de los recursos de Hardware.
- Realizar pruebas de desempeño y rendimiento referenciales.
- Analizar y exponer los resultados del análisis comparativo de rendimiento obtenidos.

1.6 Pregunta de Investigación

¿Cómo incide la Arquitectura de Microservicios basada en Contenedores en el rendimiento de los recursos de Hardware?

1.7 Hipótesis

La Arquitectura de Microservicios basada en Contenedores mejora el rendimiento en un 15% en comparación a la Arquitectura Monolítica.

1.8 Preguntas directrices

- ¿Qué factores influyen en el rendimiento de las Arquitecturas de Software?
- ¿Cuáles son las áreas de influencia de las Arquitecturas de Software?
- ¿Existen investigaciones o trabajos relacionados con la Arquitectura de Software que den soporte a la presente experimentación?

- ¿Cuál es el nivel de conocimientos y aceptación de los Microservicios basados en Contenedores?
- ¿Es posible reemplazar o migrar las Máquinas Virtuales por Contenedores?

1.9 Variables e indicadores

Independiente: Arquitectura de Microservicios basada en Contenedores.
Dependiente: Rendimiento.

Independiente: Arquitectura de Microservicios basada en Contenedores.

Conceptualización	Dimensiones	Indicadores	Ítems básicos	Técnica o instrumento
Está por delante de las virtualizaciones tradicionales en términos de usabilidad, pero esta también extiende la utilización de recursos y por lo tanto reduce el gasto en general de crear nuevos procesos de virtualización. **(Adufu, Choi, & Kim, 2015)**	Virtualización	Nivel de virtualización: Kernel, Sistema Operativo, red.	¿A qué nivel virtualizan los Contenedores?	Análisis de archivos
	Contenedores	Adaptabilidad multiplataforma: si, no.	¿Los Contenedores se adaptan a diferentes plataformas?	Análisis de archivos
	Recursos (Hardware)	Gestión eficiente de recursos: bajo, medio, alto.	¿Cuál es el nivel de eficiencia en el manejo de recursos?	Análisis de archivos
	Costo - Beneficio	Costos de implantación: bajo, medio, alto.	¿El costo de implementar una AMS es retribuida una vez puesta en producción?	Análisis de archivos

Tabla 1: Indicadores de la variable Independiente.
Elaborado por: Investigador.

Dependiente: Rendimiento.

Conceptualización	Dimensiones	Indicadores	Ítems básicos	Técnicas e instrumentos
Es la proporción entre el producto o el resultado obtenido y los medios utilizados. (Asale, 2014)	Hardware	Uso de recursos	¿Es posible optimizar el uso de los recursos implementando una AMS?	Análisis de archivos
	Software	Compatibilidad Actualizaciones Estabilidad Portabilidad	¿Con una AMS basada en contenedores, se puede obtener ambientes íntegros para producir Software?	Análisis de archivos

Tabla 2: Indicadores de la variable dependiente.
Elaborado por: Investigador.

CAPÍTULO II

MARCO REFERENCIAL

2.1. Marco Teórico

2.1.1. Antecedentes Investigativos

Como antecedente investigativo, el término de virtualización aparece en los años 60 por parte de IBM, donde la necesidad de optimizar y aprovechar los recursos existentes para esos años era imperiosa. El objetivo de virtualizar, es crear una representación virtual, en lugar de realizar varias instalaciones físicas, dependiendo siempre de las características del equipo (VMWare, 2017). En la actualidad es posible virtualizar: servidores, aplicaciones, almacenamiento y redes. Según (Amaral et al., 2015), las Máquinas Virtuales – MV no gestionan eficientemente los recursos y esa ha sido una de las principales desventajas de esta tecnología. Una alternativa actual son los Contenedores.

Con la propuesta de los Contenedores viene inherente el uso de la Arquitectura de Microservicios. Para el análisis, (Felter, Ferreira, Rajamony, & Rubio, 2015) en su investigación, acuden al uso de trabajos de carga de estrés[4] para la: CPU, memoria y red. Para hacer la comparativa por el lado de las Máquinas Virtuales lo más sobresaliente es la tecnología Kernel Virtual Machine - KVM[5] (Máquina Virtual de Núcleo, en español), y en cuanto a los Contenedores existe la tecnología llamada Docker[6]. Los mismos autores hacen hincapié en el uso de los Contenedores por ser prometedor y con mayor tendencia de uso a largo plazo. La Arquitectura de Microservicios, es una técnica relativamente nueva de virtualizar a nivel de Sistemas Operativos para ejecutar múltiples programas (Contenedores) en aislamiento sobre una simple computadora (Vaucher, 2015).

En un estudio realizado por (Villamizar et al., 2015), muestra una comparativa realizada de una Aplicación Web en versión Arquitectura Monolítica y otra en versión Arquitectura de Microservicios, donde somete a pruebas de rendimiento y da como

[4] Son pruebas que por lo general superan los límites determinados con el fin de analizar las respuestas y comportamiento de un Sistema Informático.

[5] Es una máquina virtual de tipo Hipervisor I, desarrollada por Sun Microsystems.

[6] Proyecto de código abierto usado para automatizar el despliegue de aplicaciones usando Contenedores.

resultado mejoras con el uso de la Arquitectura de Microservicios. A esto se suma una reducción en el costo de infraestructura indicando que puede ser beneficioso tener una Aplicación Web de alto nivel en un servidor en la nube.

En su publicación (Dragoni et al., 2016), nos habla de los grandes cambios que han tenido tecnológicamente la Arquitectura Monolítica – AM y la Arquitectura de Microservicios – AMS en el tiempo. Se consideran factores como: recursos, dependencias, despliegue, escalabilidad, entre otros, mismos que de ser un problema en la AM han hallado finalmente una solución en la AMS. El mismo autor, enfatiza la aceptación de la AMS en el mercado, donde grandes compañías han decidido formar parte de esta revolución tecnológica, también aclara que esta arquitectura es relativamente nueva y existe poca literatura relacionada, para lo cual deja temas abiertos en los que se puede contribuir con más investigación.

Los Microservicios, según (Amaral et al., 2015), se los considera actualmente como un refinamiento y simplificación de la Arquitectura Orientada a Servicios – SOA (Ver sección 2.4.1.8), donde la idea principal es reemplazar las Aplicaciones Monolíticas por servicios individuales (Microservicios), facilitando la habilidad para actualizar y escalar servicios de manera independiente. Así mismo, propone el uso de Microservicios como una opción efectiva, señalando que son: livianos, rápidos de iniciar y pueden agrupar dependencias y configuraciones dentro de sí mismas.

En las lecturas efectuadas, se señala un punto que marca la transición de una arquitectura a la otra, es decir, la aparición de una tecnología innovadora que merece ser analizada y utilizada. Si bien, los autores dan su punto de vista, es importante señalar que estas tecnologías han tenido una evolución drástica en los últimos años, lo cual motiva a continuar investigando.

2.1.2. Fundamentación Legal

El día jueves 10 de Abril del 2008, se emitió el decreto 1014 por parte de la presidencia del Eco. Rafael Correa Delgado, que promueve el uso de Software Libre en las instituciones públicas del Ecuador.

Decreto Ejecutivo 1014.

Art. 1.-: Establecer como política pública para las entidades de administración Pública central la utilización del Software Libre en sus sistemas y equipamientos informáticos.

Art. 2.-: Se entiende por Software libre, a los programas de computación que se pueden utilizar y distribuir sin restricción alguna, que permitan el acceso a los códigos fuentes y que sus aplicaciones puedan ser mejoradas.

Estos programas de computación tienen las siguientes libertades:

- Utilización de programa con cualquier propósito de uso común.
- Distribución de copias sin restricción alguna
- Estudio y modificación de programa (Requisito: código fuente disponible)
- Publicación del programa mejorado (Requisito: código fuente disponible)

Art. 3.-: Las entidades de la administración pública central previa a la instalación del Software Libre en sus equipos, deberán verificar la existencia de capacidad técnica que brinde el soporte necesario para este tipo de Software.

Art. 4.-: Se faculta la utilización de Software propietario (no libre) únicamente cuando no exista una solución de Software Libre que supla las necesidades requeridas, o cuando esté en riesgo de seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno.

Tabla 3: Decreto Ejecutivo 1014
Realizado por: Investigador

Es necesario mencionar adicionalmente al Reglamento de Régimen Académico con fines de sustentación de la presente investigación.

Reglamento de Régimen Académico.
Artículo 3. **Literal d.** articular la formación académica y profesional, la investigación científica, tecnológica y social, y la vinculación con la colectividad en un marco de calidad, innovación y pertinencia. **Literal h.** Impulsar el conocimiento de carácter multi, inter y trans disciplinarios en la formación de grado y postgrado, la investigación y la vinculación con la colectividad. **Literal j.** Desarrollar la educación superior bajo la perspectiva del bien público social, aportando a la democratización del conocimiento para la garantía de derechos y la reducción de inequidades.

Tabla 4: Reglamento de Régimen Académico
Realizado por: Investigador

En la Universidad Técnica del Norte - UTN se cuenta con el Plan de Desarrollo Informático, como fuente de apoyo y aseguramiento para que esta investigación sea parte contributiva en otras investigaciones relacionadas.

Plan de Desarrollo Informático UTN 2013 – 2017.
Justificación: Al contar con este instrumento estratégico, se reducirán muchos problemas en el uso de las TICs, evitar inversiones obsoletas, y dar buen uso de las TICs e innovaciones

que ayudarán en la calidad de la información que brinda a la UTN el Sistema Integrado de Información de Gestión Universitaria y que descansa en tres elementos:

- Exactitud: Información libre de errores, clara, fácil de entender.
- Oportunidad: Información disponible cuando y donde se la necesite.
- Relevancia: Brindar a cada usuario la información que realmente necesita su trabajo dentro de la Institución.

La misión de la Dirección de Informática

Dentro de lo más importante:

- Asegurar la adecuada circulación interna y externa de la información en materia d computación; informando, capacitando y asesorando a funcionarios, docentes y estudiantes de la UTN.
- Proponer y desarrollar proyectos que involucren tecnologías computacionales y de información capaces de elevar la parre académica y de asegurar su competitividad tecnológica a nivel nacional e internacional.
- Asesoría previa a la adquisición de Hardware y Software de los servicios de Ciencias de Informática, Computación y Comunicaciones, dando orientación y apoyo a la comunidad universitaria en la necesidad de adquirir equipos y componentes computacionales.
- Captar el avance tecnológico informático y aplicarlo adecuadamente a la Universidad en base a los requerimientos, así como sentar las bases para el desarrollo futuro.

Tabla 5: Plan de Desarrollo Informático UTN 2013 – 2017.
Realizado por: Investigador

2.2. Esquema del Marco Teórico de la Investigación

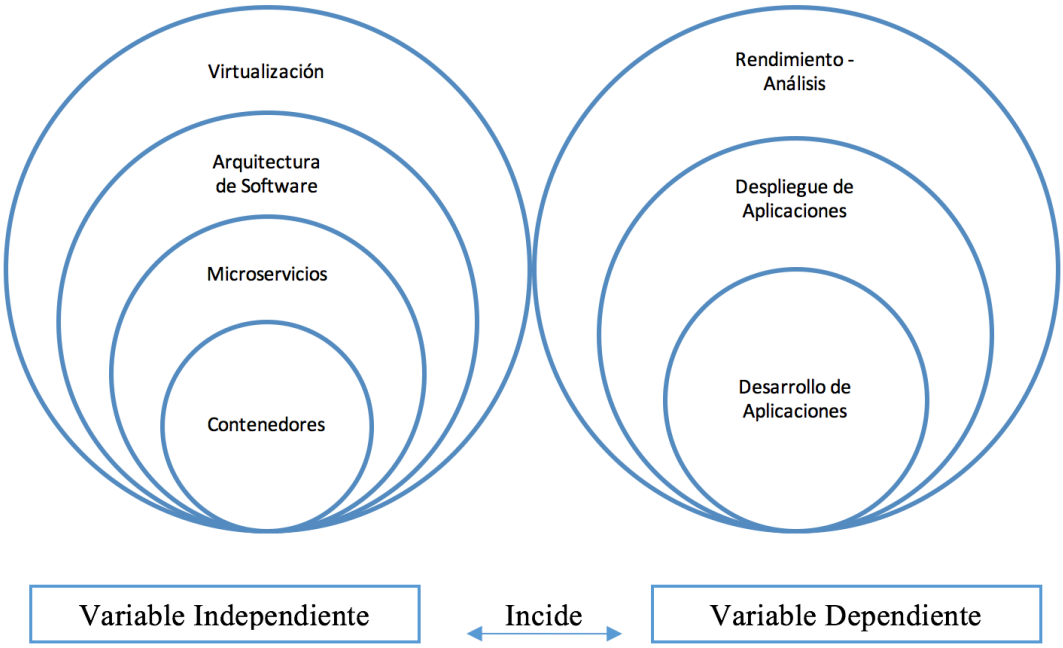


Gráfico 2: Categorías fundamentales.
Elaborado por: Investigador.

Los temas a desarrollar en la presente sección serán:

- Virtualización
- Máquinas Virtuales y Contenedores Linux
- Arquitectura de Software
- Arquitecturas Monolítica y de Microservicios
- Desarrollo y despliegue de Aplicaciones
- Rendimiento

2.3. Virtualización

La virtualización es el proceso de crear una representación basada en Software (o virtual), en lugar de una física (VMWare, 2017).

El mismo autor indica que, la virtualización se puede aplicar a servidores, aplicaciones, almacenamiento y redes, y es la manera más eficaz de reducir los costos de TI y aumentar la eficiencia y la agilidad de los negocios de cualquier tamaño.

2.3.1. Tipos de virtualización

A continuación, se listan los tipos de virtualización más conocidos y usados.

Tipo de Virtualización	Detalle
Servidores	Es una de las más comunes por la rapidez en la carga de trabajo, el buen rendimiento de las aplicaciones que se ejecutan sobre estas y su disponibilidad. Según (VMWare, 2017), con este tipo de virtualización, se abordan ineficiencias mediante la ejecución de varios sistemas operativos como Máquinas Virtuales en un único servidor físico. Esto se debe a que cada una de las Máquinas Virtuales tiene acceso a los recursos de procesamiento del servidor subyacente.
Aplicaciones	Para (UDSEnterprise, 2017), es la forma de aislar las aplicaciones del Sistema Operativo subyacente y de otras aplicaciones para conseguir una mayor compatibilidad y movilidad de las aplicaciones. Este tipo de virtualización permite reducir costos y aumentar el servicio mediante el suministro rápido y sencillo de escritorios y aplicaciones virtualizadas.

Almacenamiento	Este tipo de virtualización aparece debido a las exigencias de los volúmenes grandes de datos y las aplicaciones en tiempo real. Técnicamente es la agrupación de almacenamiento físico desde varios dispositivos de almacenamiento, simulando ser un simple dispositivo de almacenamiento el cual puede ser centralmente administrado (Siddiqui & Mahdy, 2015). Principalmente se separan los discos y las unidades flash en los servidores, se los combina para formar depósitos de almacenamiento de alto rendimiento y se los suministra como Software. A esto se lo conoce como Almacenamiento Definido por Software (SDS, Software-Defined Storage en inglés) (VMWare, 2017).
Redes	Es el proceso de combinar Hardware y recursos de red y funcionalidad de Software en una única red virtual (Gartner, 2017). La virtualización de redes brinda dispositivos y servicios de red lógicos, es decir, puertos lógicos, switches, enrutadores, firewalls, balanceadores de carga, Virtual Private Network[7] - VPN por sus siglas en inglés, entre otros, que funcionan igual a un dispositivo físico.

Tabla 6: Tipos de virtualización.
Elaborado por: Investigador.

[7] Tecnología de red de computadoras que permite una extensión segura de la red de área local (LAN) sobre una red pública o no controlada como Internet.

2.3.2. Beneficios de la virtualización

El utilizar virtualización supone varios beneficios para un negocio. La empresa VMWare en su artículo "Virtualization: Overview", proporciona una lista de dichos beneficios, de los cuales a continuación se detallan:

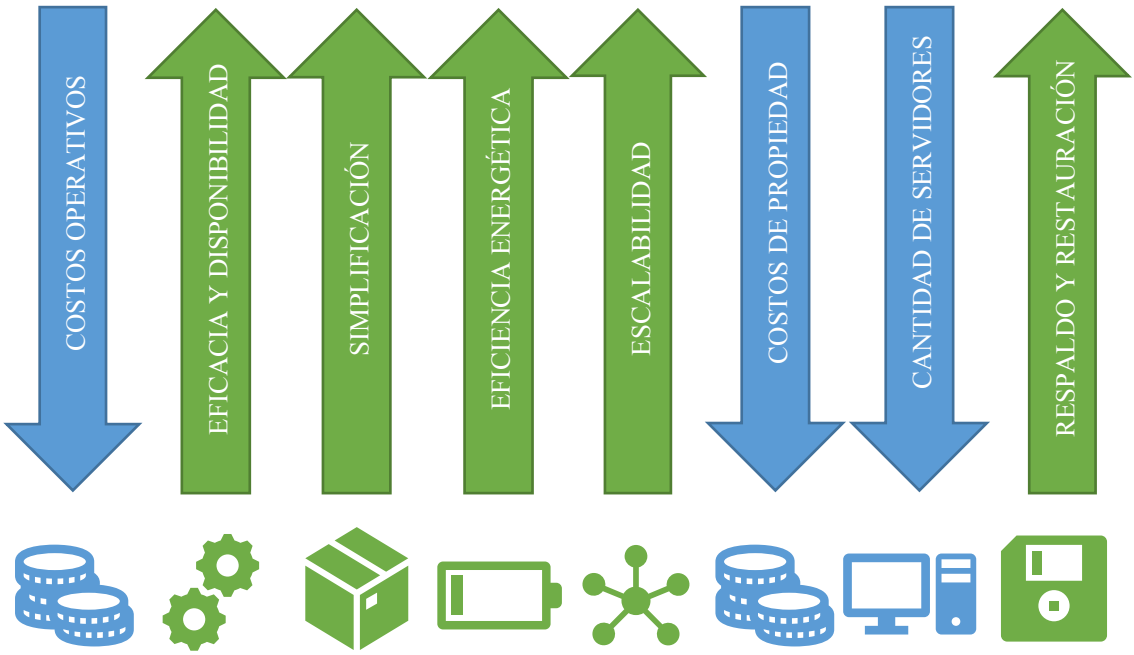


Gráfico 3: Virtualización, ventajas y beneficios (flecha hacia abajo: menor, flecha hacia arriba: mayor).
Elaborado por: investigador.

Tal como se muestra en el Gráfico 3, una virtualización adecuada puede ofrecer mejoras en diferentes aspectos, a continuación, más detalles de cada uno:

- **Disminuye costos operativos y de propiedad**, debido a que se utiliza menos infraestructura física y el equipamiento existente se aprovecha durante toda su vida útil.
- **Mejora la eficacia**, el uso, flexibilidad y disponibilidad de los recursos y aplicaciones, cuando los cambios del mercado exigen una rápida reacción.
- **Simplificación**, debido a la reducción de servidores físicos.
- **Eficiencia energética**, pues los entornos virtuales usan menos recursos ambientales.

- **Mayor escalabilidad**, cuando se usa Data Centers, los entornos virtualizados aumentan su capacidad de almacenamiento y distribución.
- **Gestión y control centralizado de recursos,** ya que controla el aumento desmedido de servidores.
- **Procedimientos de respaldo y restauración sencillos,** garantizando la continuidad del negocio y tolerancia a fallos.

2.3.3. Terminología

Diferentes factores han permitido la aparición de términos y conceptos variados que difuminan el entendimiento de los mismos. Cada autor por su lado trata de darle el significado más adecuado a cada término. (Campbell & Jeronimo, 2006), en uno de sus artículos nos presentan la siguiente lista de términos con su breve definición:

Máquina anfitriona (Host). Es la máquina física ejecutando el Software de virtualización. Por lo general, una máquina anfitriona cuenta con Hardware especializado: CPU, RAM, Disco duro, Acceso a la Red, entre otros componentes, necesarios para poder virtualizar.

Máquina Virtual. Es un Software que simula un sistema de computación y puede ejecutar como si fuese una computadora real, además, está perfectamente aislado, con un Sistema Operativo y aplicaciones en el interior. Lo cual permite tener más de una Máquina Virtual ejecutándose en una computadora/servidor.

Software de virtualización. Un término genérico que denota Software que permite a un usuario ejecutar Máquinas Virtuales en una máquina anfitriona. Entre las más conocidas están VMWare, Virtual Box, QEMU, entre otros.

Disco virtual. Se refiere a la representación física de la Máquina Virtual en el disco de la Máquina anfitriona. Un disco virtual se compone de un único archivo o conjunto de archivos relacionados. Siendo la portabilidad la principal característica de la MV entre máquinas físicas, aunque se debe considerar el tamaño de dicho archivo que generalmente requiere de un dispositivo de almacenamiento de gran capacidad.

Adiciones de Máquina Virtual. son componentes que incrementan el rendimiento del Sistema Operativo alojado. Por ejemplo, acceso a dispositivos USB, a dispositivos de CD/DVD, resolución de video, entre otros. Una de las características más sobresalientes es la habilidad de enfocar naturalmente el cursor.

Carpetas compartidas. Es un recurso que sirve para que una Máquina Virtual – MV pueda tener acceso a archivos, ya sea de otra MV o de la Máquina Host. Normalmente se permite el uso de carpetas compartidas, tras la instalación de las adiciones de la MV.

Monitor de Máquina Virtual. VMM por sus siglas en inglés. Es una solución de Software que implementa la virtualización para ejecutarse en conjunto con el Sistema Operativo anfitrión. Con el VMM se puede virtualizar ciertos recursos de Hardware como: CPU, memoria y discos físicos, y crear dispositivos emulados para MV que se ejecutan sobre una máquina anfitrión.

Hipervisor, Es un Software que maneja las interacciones entre cada Máquina Virtual y el Hardware que todos los sistemas alojados comparten (Portnoy, 2012). El Hipervisor es una capa de Software que reside debajo de las MV y encima del Hardware. Sin un Hipervisor, las diferentes MV intentarían controlar simultáneamente el Hardware, lo cual resultaría en un caos. Se clasifica en tipo I y tipo II.

- **Hipervisor tipo I,** También se lo conoce como Hipervisor Bare-metal. Básicamente es un Sistemas de computadora o red en la cual una Máquina Virtual se instala directamente sobre el Hardware en vez de instalarse sobre el SO anfitrión. El mismo autor indica que el Hipervisor tipo I corre directamente en el Hardware del servidor sin un SO debajo de él. Tiene mejor rendimiento y es considerado más seguro que el Hipervisor tipo II, ya que un sistema invitado no puede afectar al Hipervisor ni a otros sistemas invitados, lo que se traduce en que una MV sólo puede dañarse a sí misma.

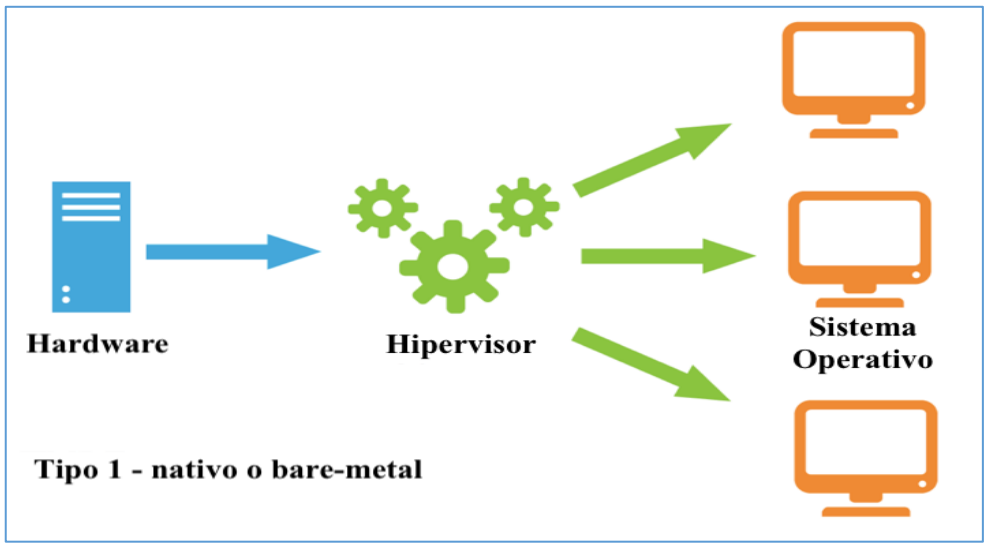


Gráfico 4: Hipervisor Tipo I.
Elaborado por: (Bligh, 2014).

- **Hipervisor tipo II,** Continúa el mismo autor describiendo que, un Hipervisor tipo II se ejecuta sobre un SO tradicional, entre los más usados tenemos: Windows, Linux y MacOS. Además, se destaca la capacidad de soportar una gran cantidad de Hardware, porque es heredado del SO que usa. Este Hipervisor es menos eficiente, debido a la capa extra existente entre el Hipervisor y el Hardware; también es menos confiable, ya que cualquier cosa que afecte la disponibilidad del SO anfitrión, también puede impactar en el Hipervisor y los sistemas invitados soportados.

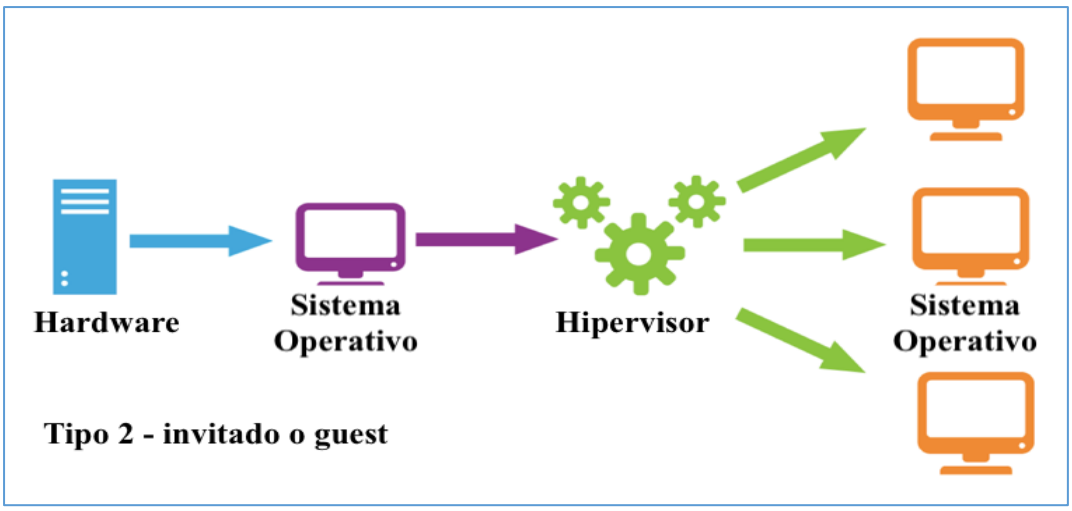


Gráfico 5: Hipervisor Tipo 2.
Elaborado por: (Bligh, 2014).

2.4.Arquitectura de Software

Según la recomendación (1471-2000 IEEE, 2000), define a una Arquitectura basada en Sistemas de Software, como la organización fundamental de un sistema incorporado en sus componentes, sus relaciones con los demás, y con el ambiente, y los principios que guían su diseño y evolución.

En términos generales la Arquitectura de Software es, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Destacando que por componentes se refiere a las aplicaciones que forman parte de un sistema completo, y que ayudan a cumplir la funcionalidad correcta del mismo.

Este término difiere conceptualmente de la Ingeniería de Software, lo expuesto en el estándar (IEEE 610.12-1990, 1990), el cual lo define como la aplicación de una estrategia sistemática, disciplinada y cuantificable al desarrollo, aplicación y mantenimiento del Software; esto es, la aplicación de la Ingeniería al Software.

En el Gráfico 6, se muestra un estado actual de la evolución de las TI, señalando las metodologías de desarrollo y la infraestructura usada, que evidencia además, las tecnologías a considerarse en esta investigación.

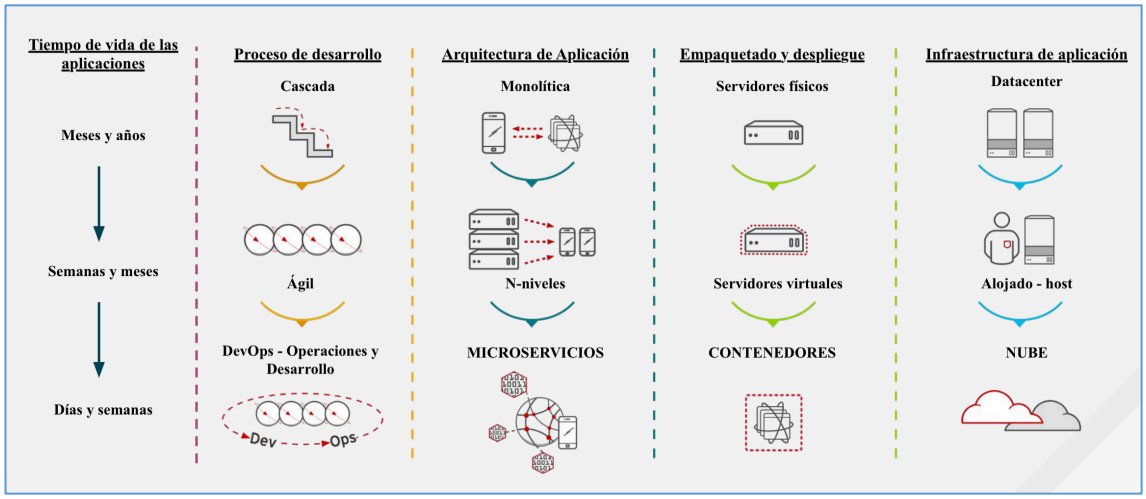


Gráfico 6: Evolución de las TI.
Elaborado por: (Sayed & Khan, 2017).

2.4.1. Tipos de Arquitecturas de Software

Algunos autores como (Fowler & Lewis, 2014), definen a una Arquitectura de Software más bien como el diseño de un Sistema Informático, el cual es único para cada Arquitectura propiamente dicha está definida como única para cada sistema implementado, y de ser el caso se le puede dar una definición diferente o similar dependiendo el punto de vista. En el Gráfico 7 se listan en secuencia de evolución algunos tipos de Arquitectura de Software más conocidos.

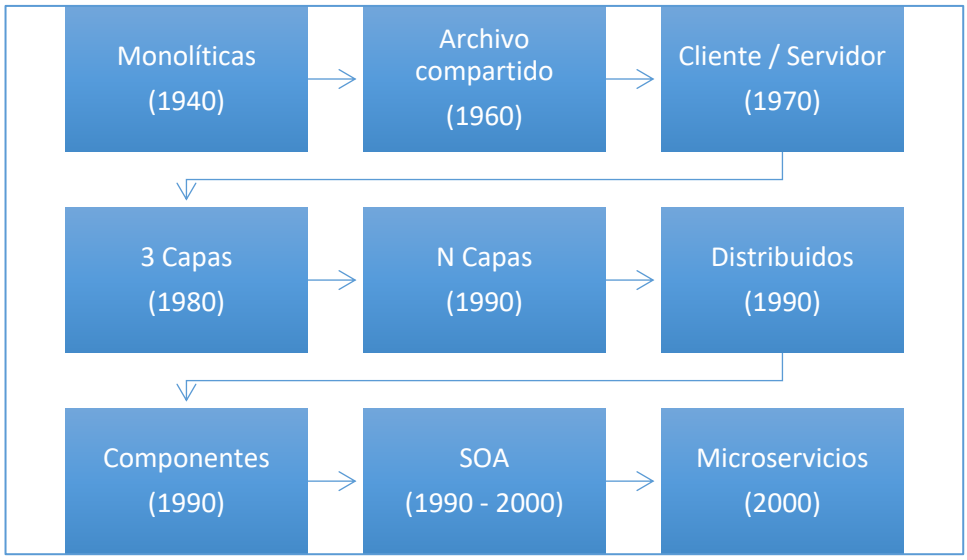


Gráfico 7: Secuencia evolutiva de los tipos de Arquitecturas de Software.
Elaborado por: Investigador

2.4.1.1. Monolítica

Según (Nielsen, 2015) lo define como un único ejecutable lógico. Es decir, que cualquier cambio realizado en una parte de un sistema de este tipo implica la construcción y despliegue de una nueva versión de todo el sistema.

En una Arquitectura Monolítica únicamente se emplea una tecnología de desarrollo, lo cual limita la disponibilidad de una herramienta adecuada para cada tarea que debe ejecutar el sistema.

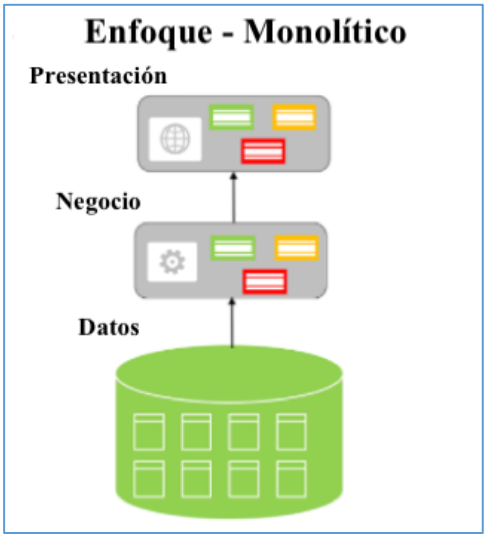


Gráfico 8: Ejemplo de un sistema con Arquitectura Monolítica.
Elaborado por: (MSfussell, 2017).

En el Gráfico 8, se muestra un ejemplo de un sistema construido con Arquitectura Monolítica.

Por lo general, las Aplicaciones con esta Arquitectura involucran aspectos de presentación, procesamiento y almacenamiento de información dentro de un sólo componente de Software. Además, son elaboradas con un supuesto de que serán ejecutadas en único servidor.

La eficiencia es un punto sobresaliente en esta Arquitectura, debido a los mínimos cambios que se producen en su contexto, por otro lado se conoce que se dificulta para realizar tareas de depuración, escalado, distribución e implantación.

Ejemplos de Aplicaciones que funcionan con esta Arquitectura: Firmware, Kernel, virus, keylogger, entre otros.

2.4.1.2. Intercambio de Archivos o File Sharing

Nace por la popularidad de las Redes Locales, donde el servidor descarga en su espacio local, archivos de otro servidor y la solicitud del cliente se procesa en el espacio de este mismo. Esta arquitectura funciona cuando es baja la demanda de contenido, actualización

y transferencia, un ejemplo de esto puede ser un sistema desarrollado en Visual Basic[8] con Access[9].

2.4.1.3. Cliente/Servidor

En la patente (US6336137 B1, 2002), explica que esta arquitectura tiene un servidor configurado para recibir solicitudes del cliente, enviar respuestas al cliente, y donde el servidor internamente procesa la información tanto de entrada como de salida.

2.4.1.4. Tres Capas

En esta arquitectura cada capa es un proceso separado y bien definido ejecutándose en plataformas separadas. En el primer nivel esta la capa de presentación, que independiente del navegador, este se encarga de dar formato y mostrar la información. El siguiente nivel, son aplicaciones que se ejecutan en el servidor, tales como: Apache, Tomcat, entre otros. Finalmente está el nivel de Datos, donde un servidor genera los datos necesarios que se procesarán en la segunda capa.

2.4.1.5. N – Capas

Es un referente de Seguridad de Datos. Aquí, cada proceso se distribuye en diferentes capas que pueden ser lógicas o físicas, los procesos se ejecutan en diferentes equipos e incluso usan diferentes plataformas/Sistemas Operativos para su ejecución. Cada equipo puede tener diferente configuración siempre con el fin de optimizar el desempeño de la función asignada tanto del proceso asignado como la eficiencia global del sistema.

[8] Lenguaje de programación creado por Microsoft.
[9] Sistema de administración de base de datos creado por Microsoft incorporado en su paquete de Office.

2.4.1.6. Distribuidos

En esta arquitectura no hay distinción entre cuál es el proveedor de servicios y cuál es el usuario de estos. Se lo puede considerar como un conjunto de objetos que interactúan y cuya localización es irrelevante. Entre las consideraciones más importantes destacan:

- Elimina cuellos de botella.
- Mantenimiento de los datos cerca de donde se utiliza.
- Minimiza la necesidad de personal con alto nivel de experiencia.
- Agrupa datos afines en la misma ubicación.
- Alta tolerancia a fallos y minimización de latencia.

2.4.1.7.Basada en Componentes

Para (Peláez, 2017), este tipo de arquitectura describe una aproximación de ingeniería de Software al diseño y desarrollo de un sistema. El mismo autor señala que, esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que exponen interfaces de comunicación bien definidas. Entre sus principales características tenemos:

- Usan aplicaciones compuestas de componentes individuales.
- Los componentes lógicos o funcionales tienen interfaces bien definidas.
- Los componentes se comunican a través de interfaces que contienen métodos, eventos y propiedades.
- Usa el principio de usabilidad, escalabilidad y encapsulado.
- Cada componente tiene dependencia mínima con los demás componentes.

Es una de las primeras arquitecturas de fácil implementación y que relativamente redujo costos en las etapas de desarrollo de un producto de Software.

2.4.1.8. Service Oriented Arquitecture - SOA

En español, Arquitectura Orientada a Servicios. Según Erl en su patente (US9213526 B1, 2015) lo define como un enfoque para desarrollar Software que admite el desacoplamiento de las funciones principales del negocio desde sus implementaciones físicas.

Todo el Software es descompuesto en pequeños programas llamados Servicios, los cuales por su lógica de encapsulamiento pueden ser genéricos, multipropósito y reutilizables.

Para (Dhara, Dharmala, & Sharma, 2015), el SOA viene de un proceso evolutivo que inicia en los 80's con los Modelos Orientados a Objetos, luego pasó por el Modelo de Desarrollo basado en Componentes hasta tener en la actualidad el Modelo Orientado a Servicios.

2.4.1.9. Microservicios

Este tipo de Arquitecturas es relativamente nueva y aún no existe una definición formal, sin embargo, tomamos un par de conceptos referenciales.

El primero viene de (Kratzke, 2015), quien dice que este tipo de arquitectura es usada para construir aplicaciones grandes, complejas y horizontalmente escalables compuestas de procesos pequeños, independientes y altamente desacoplables, comunicándose cada uno usando API's - Application Programming Interface[10] (Interfaz de Programación de Aplicaciones en español).

Por otro lado, (Fowler & Lewis, 2014), definen una arquitectura de Microservicios como una propuesta innovadora para producir Software basándose en la agrupación de servicios individuales, que tienen su propio proceso o hilo de proceso en el sistema principal y mantienen una comunicación a través de mecanismos ligeros. La comunicación entre cada uno de estos servicios se la hace comúnmente usando el

[10] Son funciones o procedimientos que brinda una librería a otro Software para que haga uso de estos.

protocolo HTTP - Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto en español).

La características más destacadas para (MSfussell, 2017) son:

- Encapsulan un escenario de cliente o negocio. ¿Cuál es el problema que va a solucionar?
- Desarrollados por un pequeño equipo de ingenieros.
- Escritos en cualquier lenguaje de programación y usando cualquier marco.
- Constan de código y, opcionalmente, de un estado, ambos de los cuales se controlan para versiones, implementan y escalan de forma independiente.
- Interactúan con otros Microservicios a través de protocolos e interfaces bien definidas.
- Tienen nombres únicos (direcciones URL) que se usan para resolver su ubicación.
- Siguen siendo coherentes y estando disponibles en caso de errores.

En el Gráfico 9, nos muestra un ejemplo de una Arquitectura de Microservicios, donde la principal característica es el aislamiento de los Servicios y las Bases de Datos.

Gráfico 9: Ejemplo de Arquitectura de Microservicios.
Elaborado por: (MSfussell, 2017)

Los Microservicios manejan ciertos principios, (Newman, 2015), durante el trabajo que ha realizado con muchas empresas que implementan Microservicios, ha establecido una lista de principios que han ayudado a dichas empresas, a construir Microservicios exitosos:

- Modelado en torno al dominio del negocio.
- Cultura de automatización.
- Detalles de implementación ocultos.
- Descentralización de todas las cosas.
- Despliegues independientes.
- Consumidor primero.
- Aislamiento de fallos.
- Altamente observable.

Tal como cualquier otro estilo de Arquitectura, los Microservicios tienen ventajas y desventajas. Estas se deben tener claro al momento de seleccionar la mejor elección. La Tabla 7 muestra algunas ventajas y desventajas que presenta este tipo de Arquitectura:

Ventajas	Desventajas
Tecnología heterogénea	Complejidad en pruebas unitarias, modulares o del sistema
Alto grado de tolerancia	Requiere de Software para monitoreo debido a su complejidad
Escalable	Complejidad operacional
Alineamiento horizontal (organizacional)	Necesario que un experto maneje los temas de configuración inicial y de actualizaciones.
Optimización para cargas de estrés	No existe documentación formal que ayude en la configuración o el seguimiento de un patrón.

Tabla 7: Ventajas y desventajas de los Microservicios.
Elaborado por: Investigador.

Entre la terminología más común tenemos:

Edge service. Es el servicio de enrutamiento el cual permite a los clientes de una Aplicación de Microservicios, acceder a servicios individuales, por un único punto de entrada. Según (Richardson & Smith, 2016), un servicio de enrutamiento es una puerta de enlace para un único punto de entrada, gestiona todas las peticiones de los clientes, y realiza un re direccionamiento al servicio o servicios apropiados.

Balanceador de carga y registro de servicios. Según (Nginx, 2017), se refiere a la distribución eficiente del tráfico que ingresa a través de un grupo de servidores Backend[11], también conocido como granja de servidores o grupo de servidores. En tanto que el registro de servicios para (Richardson & Smith, 2016), es una Base de Datos de instancias de servicios disponibles.

Circuit Breaker. Para (Fowler, 2014) la idea de un circuit breaker es simple, manejas una llamada a una función protegida en un objeto de circuit breaker, el cual monitorea el proceso por posibles fallas. En el Gráfico 10, se puede ver un ejemplo de un proceso que previene los Timeout (tiempo de espera, en español) en una llamada que se demora en responder.

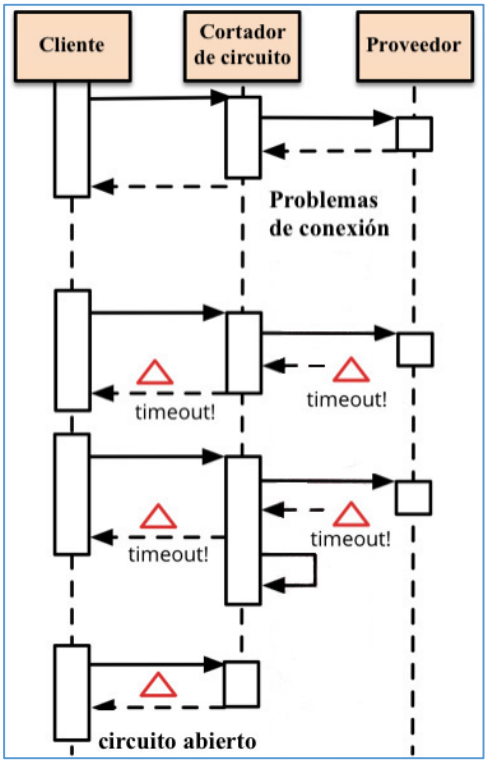


Gráfico 10: Ejemplo de un comportamiento para prevenir un Timeout.
Elaborado por: (Fowler, 2014)

[11] Es la parte que procesa la entrada desde el Frontend, usualmente es un cliente.

Logs Centralizados. Permiten obtener en uno o varios archivos relacionados un registro de los logs o eventos que suceden durante la ejecución de ciertos procesos. Un experto lo puede ver útil desde el punto de vista técnico cuando trata de detectar fallas o identificar el rendimiento de un proceso o el sistema global.

Configuración centralizada. Permite tener integridad de configuración referente al Software requerido y variables de configuración para que el sistema de Microservicios funcione adecuadamente. La intercomunicación exitosa entre los Microservicios dependerá de esta configuración que contiene la información fundamental para que funcione en diferentes ambientes.

Orquestación. La complejidad de manejar los Microservicios cuando estos tienden a ser de grandes cantidades ha dado lugar a diferentes soluciones para automatizar tal administración. La orquestación de Microservicios para (Xalambrí, 2017), significa manejar de forma automatizada múltiples servicios y sistemas. Actualmente ya existen proveedoras de este servicio, entre las más conocidas están: Kubernetes y Docker Swarm.

Despliegue o Deployment. Desplegar una Aplicación de Microservicios es desafiante debido a la complejidad que conlleva por ser una Arquitectura Distribuida. (Richardson & Smith, 2016), mencionan un caso puntual, asegurando que una Aplicación de Microservicios consiste en muchos servicios y que los mismos pueden estar escritos en una variedad de lenguajes y Framework's, y si cada uno tiene requerimientos de despliegue en específico, tornan desafiante la etapa de despliegue de un sistema, pues podría terminar con errores de configuración y mayores costos.

2.5.Arquitectura Monolítica y Arquitectura de Microservicios

La Arquitectura de Software ha evolucionado en términos de mejorar la producción de Software, reduciendo costos, tiempo y ser más eficientes en sus procesos.

Una Arquitectura Monolítica, según (Flygare & Holmqvist, 2017), es una simple unidad, donde las aplicaciones están construidas en tres partes: una Interfaz de Usuario Cliente, Aplicación del lado del Servidor y una Base de Datos.

Una Arquitectura de Microservicios, para (Fowler & Lewis, 2014), es un conjunto de pequeños servicios. Cada uno de estos servicios se ejecutan en su propio proceso y se comunican con los otros servicios usando una interfaz ligera como un HTTP API.

De forma general, en el Gráfico 11 se expone básicamente cómo se vería una Aplicación en base a cada Arquitectura.

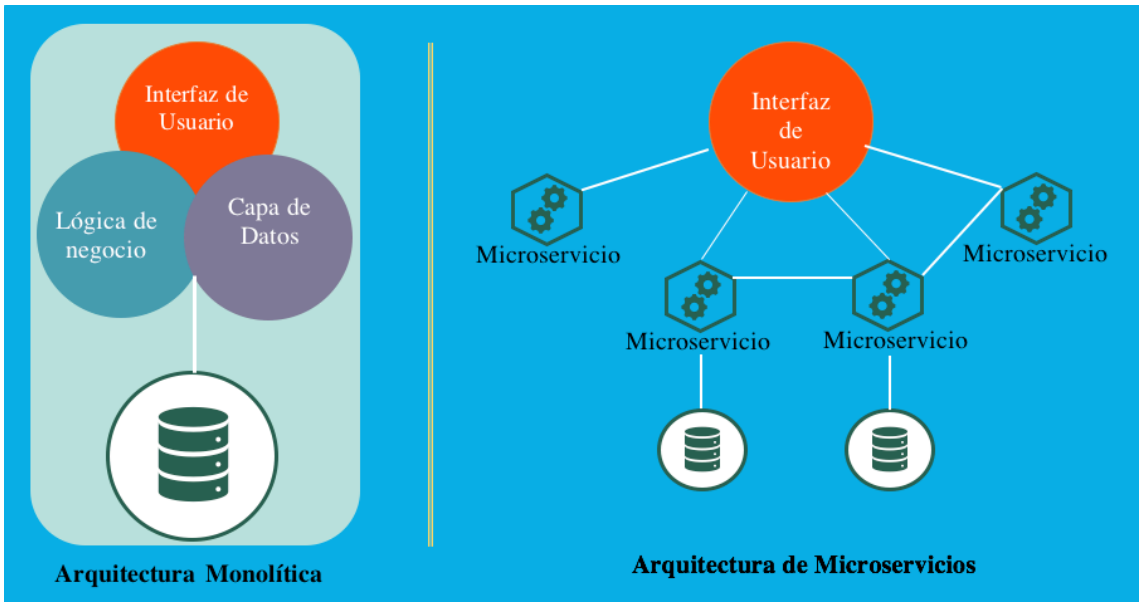


Gráfico 11: Arq. Monolítica y de Microservicios.
Elaborado por: Investigador.

A continuación, se listan características de una Arquitectura Monolítica y una Arquitectura de Microservicios que nos propone (Dragoni et al., 2016):

Monolítica	Microservicios
Su gran tamaño y complejidad dificulta el mantenimiento.	Implementa una cantidad limitada de funcionalidades, lo cual es pequeño reduce el número de errores posteriores.
Existe estricta dependencia entre las partes que componen el sistema.	Posibilita planear la migración a nuevas versiones de una tecnología.

Cambios en un módulo implica reiniciar un sistema completo.	El manejo modular de los Microservicios permite evitar el reinicio completo de un sistema, y enfocarse únicamente en una parte.
El despliegue de un Sistema Monolítico no siempre es exitoso debido a conflicto de requerimiento de recursos.	Los Microservicios se prestan para el uso de Contenedores. Esto le permite a un DevOps poder adecuar un ambiente de desarrollo compartido y único.
Limita la escalabilidad del sistema	La escalabilidad no implica la duplicidad de sus componentes, sino, a conveniencia se pueden crear nuevas instancias de servicios dependiendo de la carga de trabajo.
Representa una tecnología cerrada para los desarrolladores, quienes están obligados a usar los mismos lenguajes y Framework's[12] de la Aplicación original.	Su comunicación en red entre componentes se restringe al uso de protocolos como HTTP. A parte de eso, un desarrollador es libre de elegir el lenguaje y Framework que quiera usar.
Complejidad para aplicar la Integración / Despliegue Continuo.	Va de la mano con la Integración / Despliegue Continuo, fácil integración.
El Integrated Development Environment – IDE (Ambiente de Desarrollo Integrado, en español), se vuelve lento cuando carga incluso lo que nunca se utiliza para desarrollar la Aplicación.	El enfocarse en una parte de la Aplicación le permite al desarrollador tener en su IDE solamente las herramientas necesarias para esta parte en desarrollo.

Tabla 8: Características de una Arquitectura Monolítica y una de Microservicios. Elaborado por: (Dragoni et al., 2016).

[12] Es un término usado para identificar un conjunto estandarizado de conceptos, prácticas y criterios, se enfoca en un problema y ayuda a resolverlo.

Existe un paradigma actual, donde una aplicación ya no sólo debe ser funcional para una PC, sino también para tabletas y teléfonos inteligentes, si vamos más allá también podemos mencionar el Internet of Things - IoT[13] (Internet de las Cosas, en español). Entonces se lanza la interrogante ¿cómo lograr que una sola aplicación sea útil en multiplataforma, compatible con cualquier dispositivo o navegador? La respuesta es simple, Microservicios.

A esto se suma el análisis de (Salah, Zemerly, Yeun, Al-Qutayri, & Al-Hammadi, 2016), quienes listan características de cuatro Sistemas Distribuidos y hacen la siguiente comparativa:

Característica	Arquitectura Monolítica			Arquitectura de Microservicios
	Cliente / Servidor	Agentes móviles	SOA	
Uso de Red	NO	SI	SI	SI
Comunicación de proceso interno	NO	SI	NO	SI
Escalamiento elástico	NO	NO	NO	SI
Comunicación más ligera	NO	NO	SI	SI
Resistencia	NO	NO	SI	SI
Entrega rápida de Software	NO	NO	SI	SI
Integración de servicio autónomo	NO	NO	SI	SI
Servicio Portable	NO	SI	SI	SI
Reusabilidad de servicios	NO	NO	SI	SI
Migración de servicios	NO	SI	NO	SI
Servicios Monolíticos	SI	SI	SI	NO
Perdida de acoplamiento	NO	NO	SI	SI
Mayormente basado en la nube	NO	NO	NO	SI
Desarrollo multifuncional	NO	NO	SI	SI
Separación funcional	NO	NO	SI	SI
Requiere Middleware	NO	NO	SI	NO
Alta tolerancia a Fallos	NO	NO	NO	SI

[13] Se refiere a la interconexión digital de objetos cotidianos con Internet.

Manejo descentralizado	NO	NO	NO	SI
Bajo costo de recursos	NO	NO	NO	SI
Servicio independiente de despliegue	NO	NO	NO	SI

Tabla 9: Comparación de características de Arquitecturas de Sistemas Distribuidos. Elaborado por: (Salah et al., 2016).

La Tabla 9, nos explica de las grandes limitantes que se presentan con una Arquitectura Monolítica, entre ellas es la preocupación por producir un Software funcional para cualquier dispositivo y aplicación. Hoy en día lo primero son los teléfonos inteligentes, pues para los usuarios es más fácil portar un dispositivo pequeño que brinde servicios que antes se las podían realizar únicamente en una computadora portátil o personal. Lograr algo semejante con una Arquitectura Monolítica, tan sólo de pensarlo, genera intriga y seguramente no es algo que un DevOps moderno pueda tener en mente.

Por su lado, los Microservicios van ganado terreno, pues se ajusta a la demanda actual, debido a su flujo de trabajo y cultura de entendimiento, los Microservicios dan solución a largo plazo a esas limitantes que en una Aplicación Monolítica eran difíciles de manejar.

El mismo autor agrega que los Microservicios incluso son amigables con los Desarrolladores de Software, esto se debe a que en vez de hacer una sola Aplicación se trabaja en partes definidas y desacopladas, para posteriormente cuando estén terminadas se las pueda integrar en un sistema único.

2.6. Máquina Virtual basada en el Núcleo – KVM

(Felter et al., 2015) lo define como una característica de Linux que le permite al mismo actuar como un Hipervisor de Tipo I, ejecutando un Sistema Operativo invitado sin modificaciones dentro de un proceso de Linux. Microsoft se une a esta tendencia con su Sistema Operativo Windows, presentando su propia versión de Hipervisor llamado Microsoft Hyper-V.

Por lo general, la forma de interactuar con una KVM es a través de una aplicación llamada Command Line Interface - CLI[14] (Interfaz de Línea de Comandos en español), esto ha limitado su aceptación debido a la dificultad que conlleva la administración de Máquinas Virtuales. Por ello, Desarrolladores de Terceros[15] han contribuido con aplicaciones que tienen interfaces gráficas amigables que ayudan a administrar las Máquinas Virtuales basadas específicamente en esta tecnología. (Vaucher, 2015) nos da como ejemplo a libvirt, que es una API compatible con diferentes Hipervisores incluyendo las KVM.

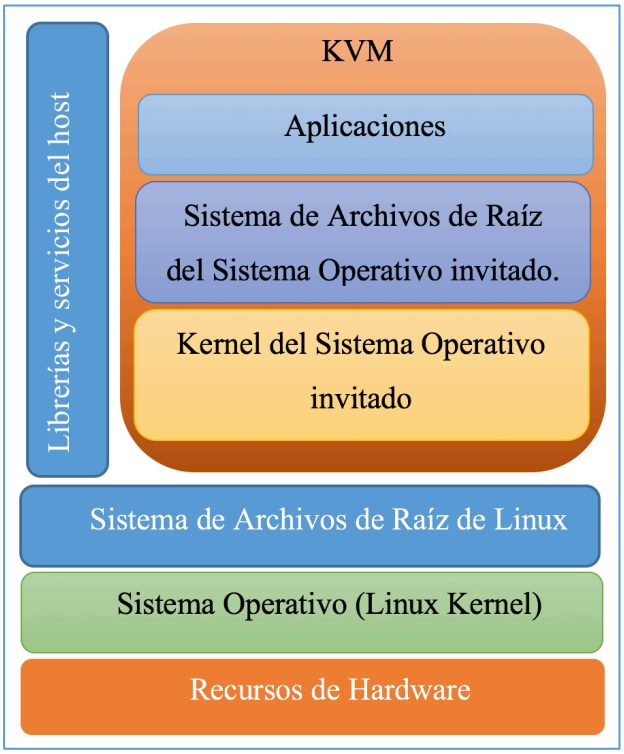


Gráfico 12: Estructura de una KVM.
Elaborado por: Investigador.

La KVM es parte de Linux y fue lanzada en el año 2006, el hecho de ser parte del mencionado Sistema Operativo hace que aproveche todo en cuanto a funcionalidades, soluciones y avances sin necesidad de procesos de ingeniería adicional.

[14] Aplicación con limitaciones gráfincas en su interfaz que generalmente se usa para ejecutar comandos usando únicamente un teclado.
[15] Empresas independientes que desarrollan Software y que contribuyen con o sin fines de lucro.

Generalmente una KVM necesita ciertos componentes a nivel del sistema para su funcionamiento, tales como: administrador de memoria, planificador de procesos, pila de entrada / salida, controladores de dispositivos, gestión de seguridad, entre otros.

Cada Máquina Virtual, se ejecuta como un proceso de Linux, y adhiere Hardware Virtual como por ejemplo: tarjeta de red, CPU, memoria, discos y manejador de gráficos.

Las características más sobresalientes de una KVM son:

- **Seguridad**, Usa una combinación de Security-Enhanced Linux (SELinux) (Mejora de Seguridad, en español) y Virtualización Segura (sVirt) para la seguridad y aislamiento de la Máquina Virtual.
- **Almacenamiento**, Puede usar cualquier almacenamiento compatible con Linux, incluye discos locales o almacenamiento conectado a la red.
- **Compatibilidad de Hardware**, Si un Hardware es compatible con Linux, por ende, será también compatible con la KVM.
- **Rendimiento y escalabilidad,** Una KVM hereda el rendimiento de Linux, por lo que puede ser escalable en cuanto a número de Máquinas Virtuales se refiere.
- **Gestión de memoria,** Usa las mismas funciones de Linux para la gestión de memoria, además, tiene soporte de grandes volúmenes para mejoras en el rendimiento.
- **Migración en caliente**, Es la capacidad de mover una Máquina Virtual en ejecución desde un Host físico a otro sin interrumpir el servicio.
- **Latencia baja,** El Kernel de Linux cuenta con extensiones en tiempo real que permiten que las aplicaciones basadas en Máquinas Virtuales se ejecuten con una latencia baja.

2.6.1. QEMU

Se la define como un emulador de procesador rápido que utiliza la traducción dinámica para lograr una buena velocidad de emulación (Bartholomew, 2006). En su sitio Web (QEMU, 2017), explica 3 principales formas de su uso:

- Full System Emulation (Emulación del Sistema Completo), ejecuta un Sistema Operativo para cualquier máquina, sobre una máquina soportada.
- User Mode Emulation (Emulación en Modo Usuario), ejecuta programas para otras distribuciones Linux, sobre cualquier arquitectura soportada.
- Virtualización, ejecuta un KVM o una Máquina Virtual Xen con un cercano rendimiento nativo.

2.7. Contenedores

Para (Scott, 2017), los Contenedores habilitan cada carga de trabajo para tener acceso exclusivo a recursos como procesador, memoria, cuenta de servicios y librerías, lo cual es esencial para el proceso de desarrollo. El mismo autor señala que los Contenedores corren como un grupo de procesos aislados dentro de un Sistema Operativo, lo cual los vuelve rápidos de iniciar y dar mantenimiento.

Esta terminología de Contenedor es genérica para cualquier partición virtual que no sea una Máquina Virtual basada en un Hipervisor. Los Contenedores empaquetan y aíslan aplicaciones con todas las librerías y dependencias que necesitan para poder ejecutarse.

2.7.1. Técnicas de partición de recursos no basadas en Hipervisores

Según (Intel, 2014) los enfoques alternativos para la partición de recursos que no se basen en Hipervisores, fueron desarrollados por Unix y su crecimiento dio lugar a que a principios de 1980 se impulsaran por diferentes organizaciones que incluían laboratorios, tales como: Bell, AT&T, Digital Equipment Corporation, Sun Microsystems e instituciones académicas.

2.7.1.1. Llamada al Sistema Chroot

Este término chroot (abreviación de change root), es usado raramente, pues el nivel de conocimiento debe ser profundo y extenso en ambientes Linux. Según (Felter et al., 2015) lo nombra como Unix chroot, y menciona que ha sido muy usado para implementar características de tipo jails (cárceles en español) y su extensión BSD jails. En el Sistema

Operativo Solaris[16] lo presenta y promociona como una moderna implementación de Contenedores.

El mecanismo chroot, redefine el directorio raíz para cualquier proceso que se esté ejecutando, previniendo que el programa sea capaz de nombrar o acceder recursos fuera del árbol del directorio principal.

2.7.1.2. FreeBSD jails

FreeBSD jails, si bien su concepto es el mismo que chroot, este tiene mejoras en temas de seguridad. Según el mismo autor, las FreeBSD jails restringen explícitamente el acceso fuera del entorno de espacio aislado por entidades tales como archivos, procesos, y cuentas de usuario.

2.7.1.3. Contenedores en Solaris - Solaris zones

Es la versión mejorada de chroot y FreeBSD jails. Continúa el mismo autor definiendo a las Zones, como instancias de servidores individuales que pueden coexistir dentro de una sola instancia de Sistema Operativo.

Según la reseña histórica, Sun Microsystems introduce esta característica con el nombre de Solaris Containers como parte de Solaris 10 en el año 2005, luego Oracle oficialmente cambio el nombre a Solaris Zones con el lanzamiento de Solaris 11 en el año 2011.

2.7.1.4. Contenedores Linux – LxC

Se hace su lanzamiento en el año 2007. Su principal característica es la virtualización basada en Sistema Operativo. (Jurenka, 2015), lo define como una combinación de las características que ofrecen los “namespaces” y “control groups” para crear un ambiente completamente aislado, el cual es fácil de manejar. Su estructura se caracteriza por no

[16] Es un Sistema Operativo de tipo Unix, desarrollado en 1992 por Sun Microsystem. Actualmente pertenece a Oracle Corporation.

tener un Sistema Operativo invitado, tal como se muestra en el Gráfico 13, eso significa mayor velocidad de procesamiento y respuesta.

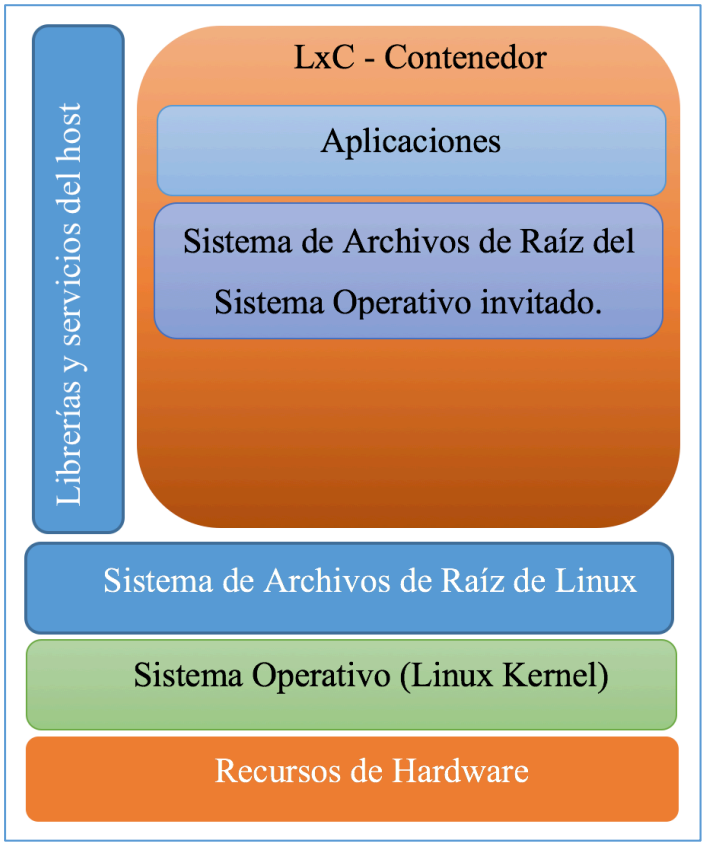


Gráfico 13: Estructura de un Contenedor
Elaborado por: Investigador.

Prácticamente, el LxC negocia las operaciones básicas que se realizan sobre los Contenedores: crear, iniciar, detener, listar y remover.

Como parte del LxC tenemos:

a) Grupos de Control – Cgroups

También conocidos como Kernel Control Groups, es una característica del núcleo de Linux. Según (Jurenka, 2015), originalmente inicia en el 2006 con la participación de los ingenieros de Google quienes habilitaron administradores para la restricción y/o limitación del uso de recursos para grupos de procesos.

La capacidad de poder configurar a cada grupo manualmente o de forma programada, permite tener mayor control sobre la distribución, priorización, denegación y monitoreo de los recursos del sistema.

b) Espacio de Nombres o Namespaces

Esta es una pieza clave que distingue al núcleo de Linux para soportar virtualizaciones ligeras. El propósito de cada Namespace es el de agrupar un recurso de sistema global en particular en una abstracción que hace que los procesos dentro del Namespace tenga su propia instancia aislada del recurso global (Kerrisk, 2013).

Los Namespaces permiten a cada Contenedor tener su propia interfaz y dirección IP. Además, limita la visibilidad que cada "cgroup" con respecto a los recursos del sistema.

2.7.2. Docker

Uno de los autores más influyentes en hacer un uso adecuado de los Contenedores (Turnbull, 2016), describe a Docker como un motor de Código Abierto que automatiza el despliegue de Aplicaciones en Contenedores.

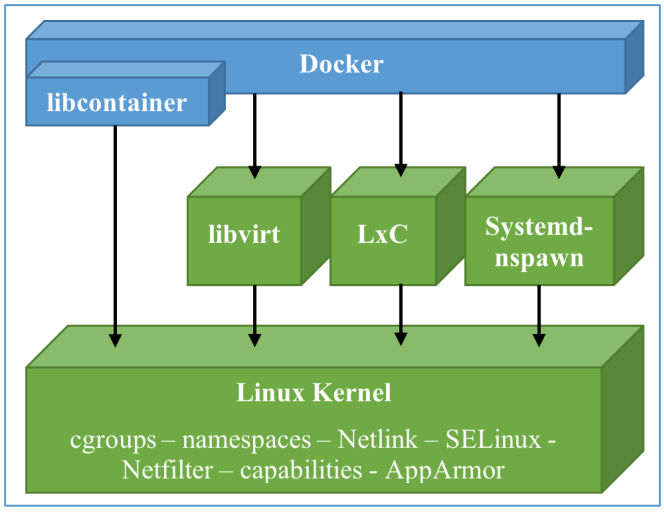


Gráfico 14: Ambiente de ejecución de Docker.
Elaborado por: (Vaduva, 2015).

Docker fue desarrollado por la empresa del mismo nombre Docker Inc., anteriormente conocida como dotCloud Inc.

Docker es un Software de Código Abierto que pretende automatizar el método de implementación de Aplicaciones dentro de Contenedores Linux. Ofreciendo una capa de abstracción sobre LxC (Vaduva, 2015). Ver el Gráfico 14.

Inicialmente Docker usaba LxC como ambiente de ejecución por defecto, sin embargo, el cambio continuo de las distribuciones representaba un problema al tratar de estandarizar una versión genérica. Docker en su versión 0.9, incluye su propio ambiente propio de ejecución, llamado "libcontainer", lo cual le ayudó a convertirse en un estándar, además de convertirlo a Docker en una tecnología multiplataforma.

2.7.2.1. Evolución de Docker

Los ingenieros Solomon Hykes y Sébastien Pahl, son los cofundadores de este proyecto, quienes inicialmente lanzan la primera versión en el 2013 como un proyecto OpenSource de dotCloud.

La empresa pudo recaudar cerca de once millones de dólares de inversores como Jerry Yang cofundador de Yahoo, además de firmas como Trinity Ventures y Benchmark. Esto no duró mucho tiempo y fue decayendo hasta cuando Ben Golub (presidente de la empresa Plaxo), se unió a la compañía apoyando a Hykes. Juntos arriesgaron esta gran idea poniéndola gratis a disposición de cualquier persona. A un mes ya se habían inscrito más de 10000 desarrolladores interesados en conjunto con compañías como RedHat y Amazon.

En el 2013, en Santa Clara – California, se anuncia el proyecto Docker. Para finales del mismo año, su popularidad impresionó tanto a nuevos inversores, pudiendo recaudar $15 millones. Para junio del 2014 el motor de Docker se descargó 2.75 millones de veces. En la actualidad ese número sobrepasa los 100 millones de descargas. La tendencia de su

acogida se la puede ver en el Gráfico 15 generado por Google Trends[17]. Desde entonces, Docker se ha venido expandiendo, y ya cuenta con versiones para Linux, Windows y Mac, además de servicios empresariales.

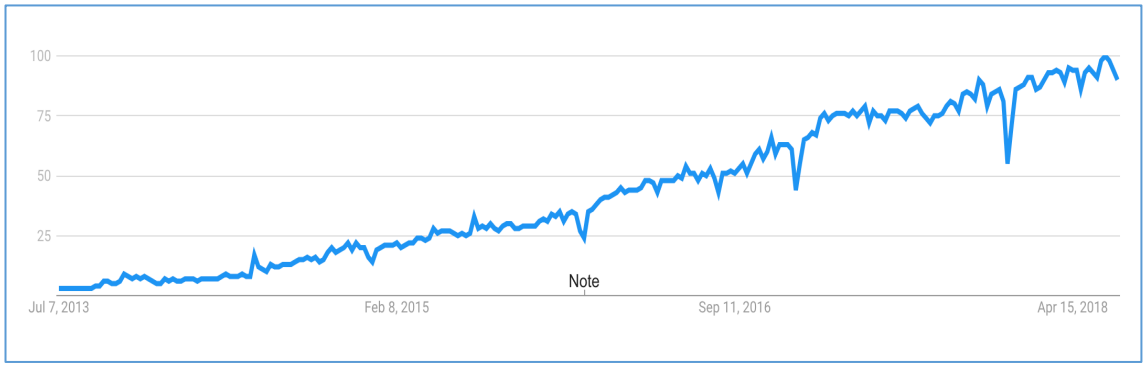


Gráfico 15: Tendencia mundial de popularidad de Docker del 2012 al 2018.
Elaborado por: Investigador usando Google Trends.

En el Gráfico 16, se ven claramente los países donde más se investiga sobre Docker. Los mismos países que aparecen con más densidad (tonos de azul, claro es menos densidad mientras oscuro significa mayor densidad), tienden a ofertar soluciones con tecnologías modernas y de bajo costo, esto significa que una tecnología como Docker es de uso libre y presta mejoras en temas de virtualización. De esta forma esta tendencia es considerada tecnológicamente como una oportunidad para evolucionar proyectos y por ende el conocimiento.

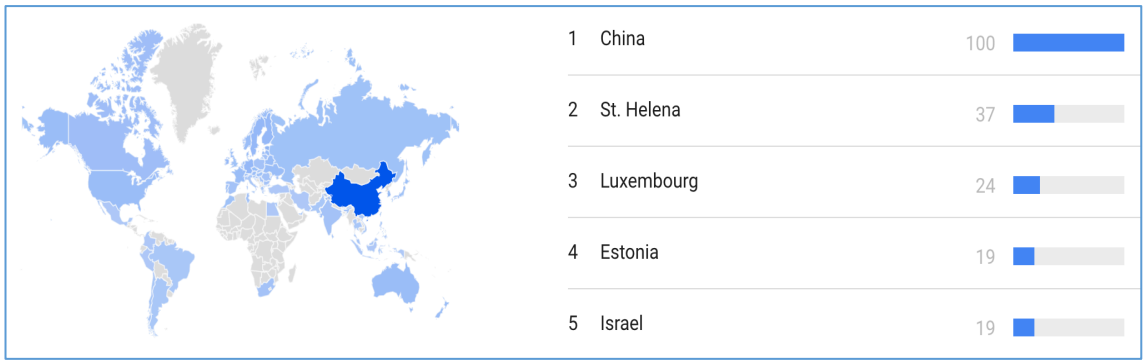


Gráfico 16: Países donde más popular es Docker del 2012 al 2018.
Elaborado por: Investigador usando Google Trends.

[17] Herramienta de acceso libre y gratuito que brinda Google, permite obtener datos estadísticos de tendencia de palabras o frases.

2.7.2.2. Componentes Base

A continuación, se describen los componentes núcleo de Docker, estos son el servidor y el cliente, los cuales en conjunto se les conoce como Docker Engine.

a) Cliente y Servidor de Docker

A esta combinación también se la conoce como Docker Engine (Motor de Docker en español). (Turnbull, 2016), nos explica que Docker en sí es una Aplicación de tipo cliente – servidor, donde el lado cliente habla con el servidor o demonio (daemon[18] en inglés), el mismo que por turnos realiza todo el trabajo. Docker utiliza una interfaz de tipo "RestFul" para comunicarse entre cliente y servidor, así también el cliente usa los flujos de trabajo de Docker para la comunicación con otros servidores remotos. En el Gráfico 17, se muestra la Arquitectura que propone Docker.

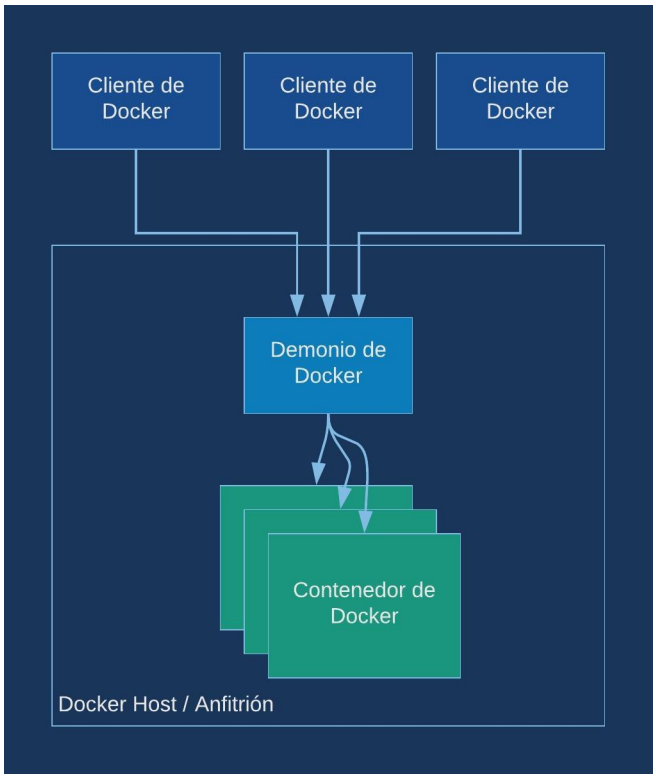


Gráfico 17: Arquitectura de Docker.
Elaborado por: (Turnbull, 2016).

[18] Proceso que se ejecuta todo el tiempo a espera de una llamada específica a través de un puerto usando cierto protocolo.

b) Imágenes de Docker

El mismo autor, considera a las imágenes como bloques de construcción del mundo de Docker. Para ejecutar un Contenedor se lo hace a partir de una imagen, para ello se entiende que las imágenes son la parte de la construcción en el ciclo de vida de Docker. Los pasos más comunes para crear una imagen son:

- Crear un archivo
- Ejecutar un comando
- Exponer un puerto

Son prácticamente el código fuente de los Contenedores, además esto les permite ser portables y pueden ser compartidos, almacenados y actualizados con facilidad.

c) Registros

Por defecto, las imágenes se almacenan en registros. Estos registros pueden ser públicos o privados tal como nos lo menciona (Turnbull, 2016). Docker, Inc., ofrece este servicio de registro de manera gratuita en su Docker Store[19] para imágenes públicas y una versión pagada para privatizar imágenes con contenido o funcionalidad de carácter confidencial.

d) Contenedores de Docker

Docker tiene como objetivo ayudar en la construcción y despliegue de Contenedores, dentro de los cuales es permitido empaquetar aplicaciones y servicios. El autor continúa indicando que un Contenedor es un formato de una imagen, que contiene un conjunto de operaciones estándar, dando como resultado un ambiente de ejecución.

[19] Servicio de registro de imagenes de Docker, también se la conoce como Repositorio de imágenes, pues hace un control de versionamiento de cada imagen.

El término de Contenedor viene de un parecido a los contenedores que transportan bienes, pero en este caso se transporta Software, permitiendo ejecutar un conjunto de operaciones como: Crear, Iniciar, Detener, Reiniciar y Destruir.

2.7.2.3. Orquestadores: Docker Compose y Swarm

En el ecosistema de Docker existen dos grandes herramientas:

a) **Docker Compose,** Permite ejecutar una pila de Contenedores (como servicios), los cuales representan una pila de aplicaciones, por ejemplo: Servidor Web, servidor de Base de Datos, entre otros. Los cuales hacen de soporte para la correcta funcionalidad de una o más aplicaciones específicas.

b) **Docker Swarm**, Permite crear Clusters[20] de Contenedores, a este proceso se lo denomina Swarm, el cual permite ejecutar cargas fuertes de trabajo y escalables.

2.7.2.4. Beneficios

Tanto organizaciones como autores de textos relacionados al tema de Docker (Fowler & Lewis, 2014) o (Turnbull, 2016), dan su versión de los beneficios de hacer uso de esta tecnología, coincidiendo en su mayoría en los siguientes aspectos:

- Rapidez en el despliegue de aplicaciones.
- Portabilidad a diferentes Hosts y compatibilidad multiplataforma.
- Control de versiones y reúso de componentes.
- Permite el trabajo compartido.
- Un modelo ligero y un mínimo de gastos.
- Mantenimiento simplificado.

[20] Es un grupo de ordenadores unidos entre si por redes informáticas, simulando ser un único ordenador potenciando la capacidad del mismo.

Según (Bashari Rad, Bhatti, & Ahmadi, 2017), el hecho de que Docker no use Hipervisores es la principal clave de su eficiencia, además explica que, el rendimiento mejora por la alta densidad y el no sobregasto de los recursos.

2.8. Computación en la nube

Este término se ha vuelto tendencia, (IBM, 2017) dice que la Computación en la Nube, consiste en el suministro de recursos informáticos bajo demanda[21]. Desde Aplicaciones hasta Centros de Datos, a través de Internet y basado en un modelo de pago por uso.

En contraste, (Salesforce, 2017), una de las empresas más grandes en brindar este tipo de servicios lo define como: una tecnología que permite acceso remoto a servicios informáticos como: el almacenamiento de archivos y uso de Internet para procesar datos.

Entre otras definiciones, todos los proveedores de este servicio coinciden en que la Computación en la Nube se refiere a la entrega bajo demanda de recursos informáticos y aplicaciones a través de Internet, cada una en competencia, tratan de brindar costos asequibles por el uso de los servicios ofertados.

2.8.1. Tipos de Computación en la nube

Sea por temas de seguridad o eficiencia, los tipos de Computación en la Nube son:

a) Nube Privada

Este tipo de nube tiende aprovechar al máximo la eficiencia de la nube, además que permite mantener el control sobre los recursos y evitar la multitenencia[22].

Consta de una única organización con un grupo de servidores que crean su Nube y el Software que se limita a uso interno restringiendo el acceso público (Salesforce, 2017).

[21] Término informático usado por empresas que brindan servicios bajo demanda o personalizados.
[22] Significa la ejecución de una única instancia en un servidor que procesa las solicitudes de múltiples clientes.

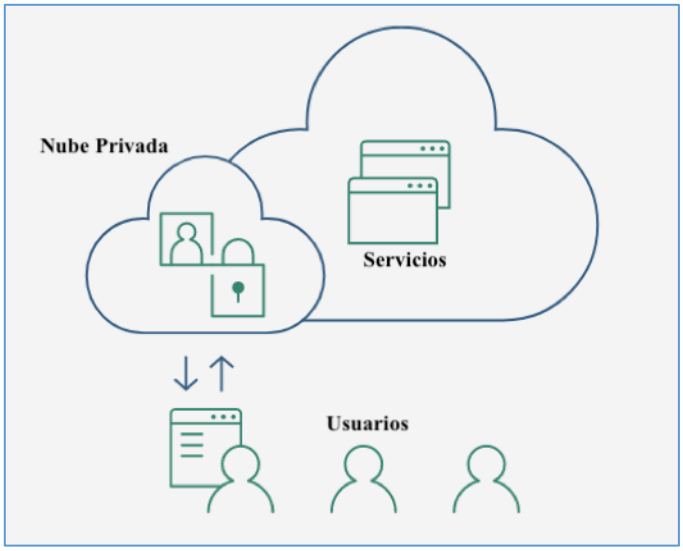


Gráfico 18: Nube privada.
Elaborado por: (Salesforce, 2017)

Según el Gráfico 18, generalmente los usuarios pueden acceder a una Nube Privada utilizando credenciales especiales, esta puede ser propia de una organización donde la confidencialidad de su información es estricta. Cualquier usuario fuera de la organización o sin las credenciales no podría públicamente acceder a la información almacenada en esta Nube Privada.

b) Nube Pública

En una Nube Pública, se evita la adquisición de Hardware, Software e infraestructura de Soporte, pues es un servicio que brindan los proveedores y son ellos mismos quienes lo gestionan.

Al igual que la Nube Privada, el mismo autor indica que diversas empresas pueden usar de manera simultánea los servicios. La responsabilidad tanto del mantenimiento y la seguridad de esta nube recae sobre el proveedor.

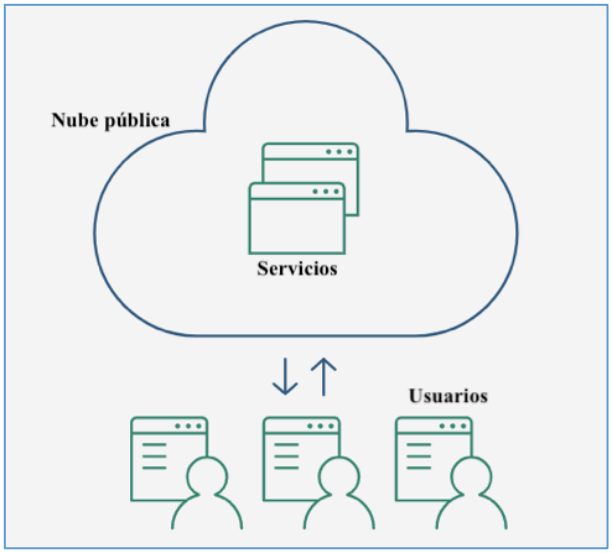


Gráfico 19: Nube pública.
Elaborado por: (Salesforce, 2017)

En este tipo de nube, el nivel de seguridad es menor respecto a la Nube Privada, en este caso podrían incluso omitirse credenciales para el acceso a la información que esta nube pueda contener, eso representa el Gráfico 19.

c) Nube Híbrida

A este punto, las empresas tienden a evolucionar para gestionar adecuadamente las cargas de trabajo en todos los Centros de Datos, Nubes Privadas y Nubes Públicas, resultando de esto una Nube Híbrida.

Continúa el mismo autor, indicando que esta Nube podría constituirse por dos o más infraestructuras de nubes distintas, controladas por una tecnología propietaria.

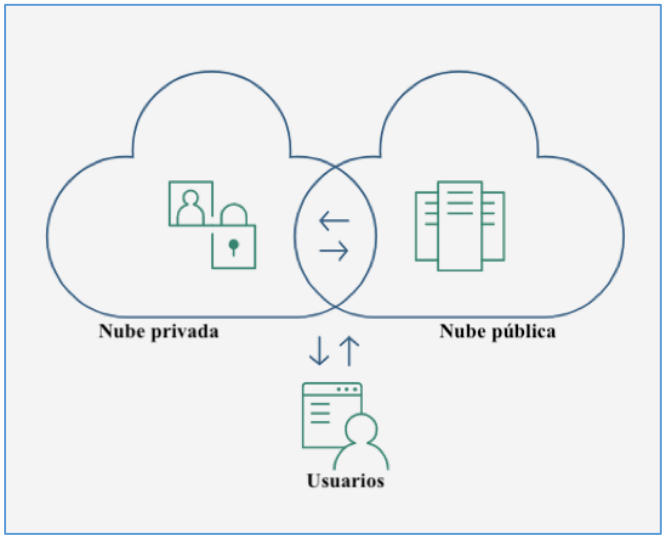


Gráfico 20: Nube híbrida.
Elaborado por: (Salesforce, 2017)

En el Gráfico 20, muestra una Nube Privada y una Nube Pública trabajando en conjunto, este caso se da en situaciones en las que cierta información es de carácter confidencial y requiere de niveles de seguridad más avanzados, mientras que también existe información que puede ser de acceso público o compartida entre organizaciones afines.

2.8.2. Modelos de computación en la nube

Dependiendo de los recursos que exige el usuario para poner su producto de Software en funcionamiento, existen estos tres modelos:

a) Software as a Service – SaaS

Software como Servicio en español, según (Salesforce, 2017), torna fácil el acceso al Software, es así que el usuario interactúa con dicho servicio a través de una interfaz gráfica.

Características:

- Rápido inicio de sesión e inmediatez para utilizar las innovadoras aplicaciones de negocio.

- Aplicaciones y datos accesibles desde cualquier sistema conectado.
- Tolerancia a fallos de sistema, los datos están más seguros y con respaldo.
- Escalamiento dinámico dependiendo de la demanda.

b) Platform as a Service – PaaS

Plataforma como Servicio en español, la definición de (IBM, 2017) nos dice que este modelo proporciona un entorno basado en la Nube con todos los requisitos necesarios para dar soporte a todo el ciclo de vida de creación y puesta en marcha de aplicaciones (Cloud) basadas en la Web. En la actualidad las aplicaciones van más allá de la Web, también se destaca la tendencia Internet of Things - IoT[23] (Internet de las Cosas en español), para lo cual ya se ofrecen servicios asociados. El mismo autor indica que al hacer uso de este modelo se lo hace sin el costo y la complejidad de comprar y gestionar el Hardware, Software, Aprovisionamiento y Alojamiento de servicios.

Características:

- Mejora la productividad en el desarrollo y la comercialización de aplicaciones.
- Despliegue inmediato de nuevas versiones de aplicaciones web en la nube.
- Reduce la complejidad con Middleware como servicio.

c) Infrastructure as a Service – IaaS

Infraestructura como Servicio en español, este modelo, según (IBM, 2017) prácticamente la gestión principal de Software y la Plataforma la hace la empresa responsable y el proveedor brinda los recursos sobre los cuales se ejecutarán las aplicaciones.

Características:

- Los costos de inversión en Hardware ya no son necesarios.
- Escalabilidad dinámica dependiendo la demanda del cliente.
- Alta disponibilidad y soporte técnico.

[23] Término relacionado a una red de dispositivos electrónicos con capacidad de conectarse a Internet.

2.8.3. Beneficios y Proveedores

Los beneficios son cambiantes en pro de mantener un cliente satisfecho y con el menor número de quejas posible. Entre los beneficios más destacados y comunes entre quienes ofrecen estos servicios tenemos:

- Usabilidad
- Flexibilidad
- Rentabilidad
- Confiabilidad
- Escalabilidad y rendimiento
- Seguridad

A los proveedores de estos servicios también se los conoce como Cloud Service Provider[24] - CSP (Proveedor de Servicio en la Nube, en español). Los más conocidos según su servicio son:

SaaS	PaaS	IaaS
Oracle On Demand	Engine Yard	CloudSigma
SalesForce	Windows Azure	Linode
NetSuite	Lunacloud	Amazon Web Services
AppDynamics	Qt Cloud Services	Dimension Data
Aprenda	Skyvia	Rackspace
Cloud9 Analytics	CloudForge	Digital Ocean
Cumulux	Openstack	IBM Cloud
Intact	Amazon Elastic Beanstack	CloudForge

Tabla 10: Proveedores de Servicio en la Nube.
Elaborado por: Investigador.

[24] Son compañías que ofrecen componentes de computación en la nube, generalmente, Infraestructura como Servicio (IaaS), Software como Servicio (SaaS) o Plataforma como Servicio (PaaS), de tipo de negocios o individuales.

2.9. Desarrollo y Despliegue de Aplicaciones

2.9.1. Development and Operations - DevOps

Desarrollo y Operaciones en español. Para el autor (Scott, 2017), el termino DevOps, viene a ser una técnica de programación ágil que enfatiza modularidad, liberaciones frecuentes y un ciclo de retroalimentación constante. La definición a este término relativamente nuevo, va en concordancia con la definición de (Sharma, 2014), quien lo ve como un enfoque basado en principios de agilidad y eficiencia el mismo que podría traducirse al uso de metodologías ágiles de programación como Scrum. Continua el mismo autor indicando que los propietarios de empresas y los departamentos de desarrollo, operaciones y control de calidad colaboran para ofrecer Software de forma continua y de calidad, es decir, se aplica Integración Continua y/o Despliegue Continuo en el proceso de la producción de Software, lo que permite a las empresas sacar partido de las oportunidades del mercado de una forma más rápida y en menor tiempo, lo que permitiría incluir respuestas de clientes.

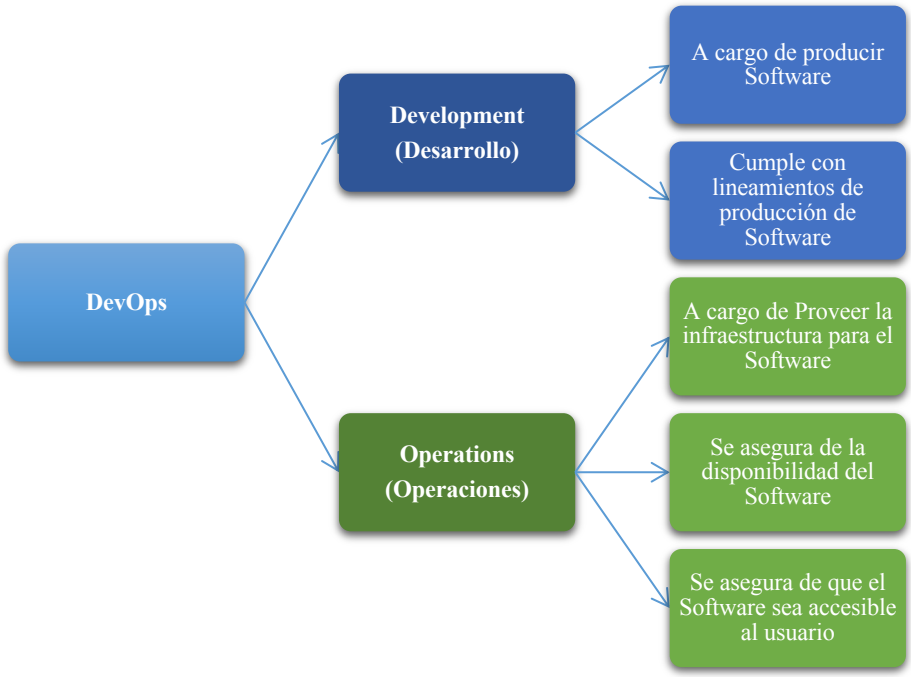


Gráfico 21: Estructura de DevOps.
Elaborado por: Investigador.

A diferencia de las metodologías ágiles como Scrum o XP, DevOps está más centrado en fomentar una cultura diferente y moderna aplicable en la producción de Software, por lo que no se la debe considerar como una metodología. Revisando el Gráfico 21, a esa forma de producir Software, muchos expertos lo llaman Cultura de Desarrollo, pues para su cumplimiento, no es suficiente la experiencia, sino también, la disciplina de seguir los lineamientos que se definen al inicio de un proyecto.

2.9.2. Formas de integrar DevOps

Según las necesidades de quien produce Software, a DevOps se la puede adoptar analizando cuál de las siguientes opciones se integra mejor (Sharma, 2014):

a) Planificación y medición

Se enfoca en la planificación empresarial continua. Establece objetivos de negocio y los ajusta en base a la retroalimentación de los clientes (feedback). Se requiere del uso de Metodologías Ágiles y de ser proactivos. Es importante realizar las planificaciones necesarias con el fin de maximizar la habilidad de crear valor.

b) Desarrollo y pruebas

Establece el núcleo de las funciones de desarrollo colaborativo y control de calidad. Esto permite al personal involucrado trabajar en forma conjunta, ofreciendo un proceso habitual de prácticas y una plataforma común para producir y entregar Software. Las pruebas continuas, son pruebas anticipadas y constantes durante todo el ciclo de vida de producción de Software.

Un proyecto puede estar dividido en componentes que a su vez son asignados a un equipo encargado de su producción y funcionalidad. Esto facilita a que los equipos funcionen incluso cuando están trabajando remotamente, sabiendo que esa parte del producto final viene probado y completamente funcional.

c) Lanzamiento de versiones y despliegues

Se crea un canal de distribución que facilita el despliegue continuo de Software a control de calidad y después a producción, todo esto de forma eficiente y automatizada. El objetivo del lanzamiento de versiones y de la implementación continua es presentar avances frecuentes a los clientes y usuarios.

d) Supervisión y optimización

Ayudan a las empresas a monitorizar los resultados de aplicaciones desplegadas en el ambiente de producción y la retroalimentación de los clientes. La supervisión ofrece datos y métricas al personal de líneas de negocios, desarrollo, control de calidad y operaciones, así como de otras partes interesadas, sobre aplicaciones en distintas fases del ciclo de entrega. Quien hace la entrega del producto de Software recibe información de cómo los clientes usan el Software y más información en la retroalimentación que da el cliente, permitiendo hacer una optimización continua.

2.9.3. Modelos de DevOps

Al igual que los Microservicios y los Contenedores, los modelos de DevOps son relativamente nuevos, a continuación, se mencionan los más conocidos:

a) Continuous Integration - CI

Integración Continua, en español. Este modelo fue propuesto por (Fowler, 2006), e indica que esto ocurre cuando todos los desarrolladores empujan sus códigos a una línea principal al menos una vez al día, con pruebas automatizadas para detectar errores de integración.

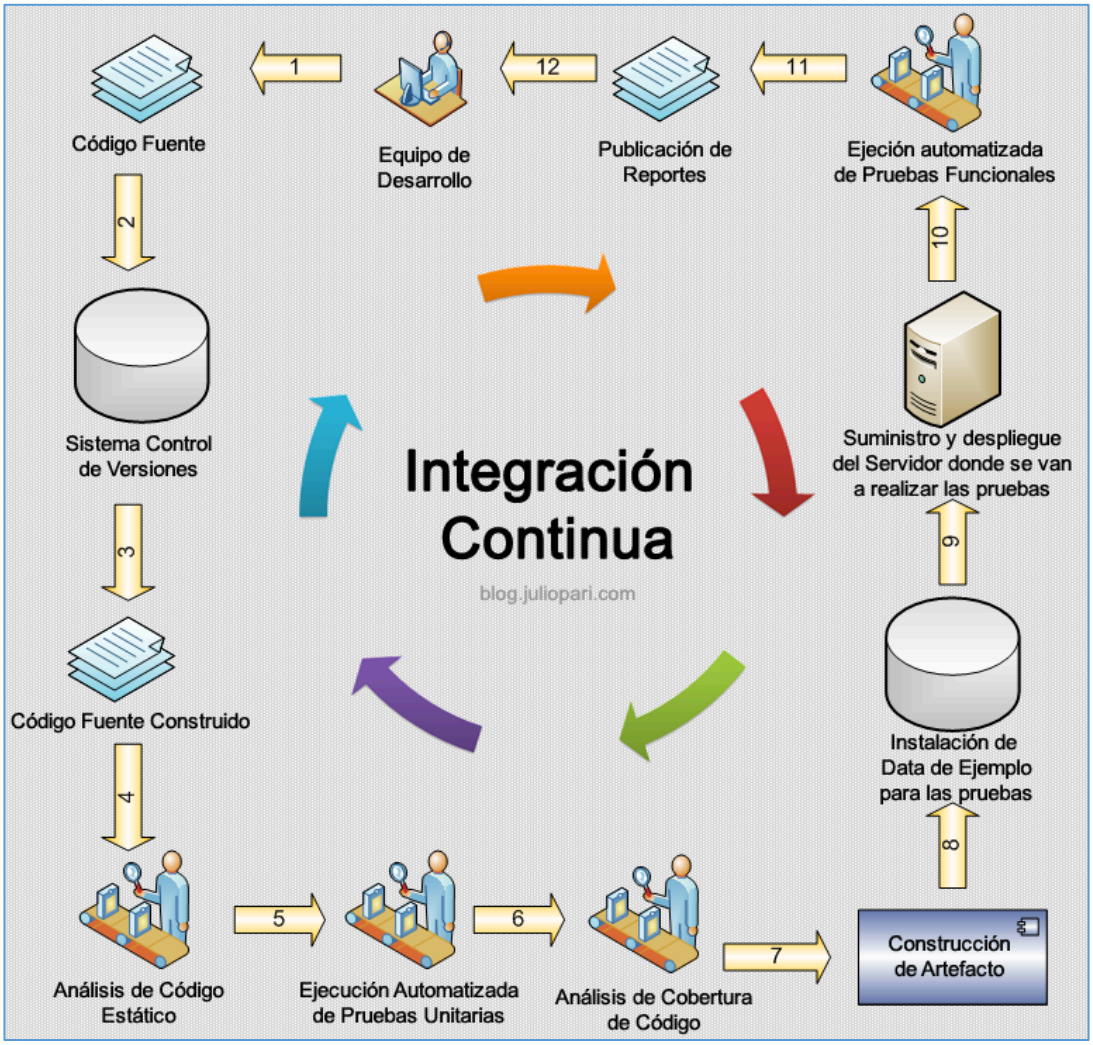


Gráfico 22: Integración Continua.
Elaborado por: (Faura, 2014).

b) Continuous Delivery - CD

Entrega Continua, en español. En este modelo, según (Sundman, 2013), la aplicación pasa por la etapa de pruebas automáticas, pero mantiene a decisión si debe o no ser desplegada, este último proceso se lo hace manualmente. De forma inherente la Integración Continua implica la práctica de la entrega continua.

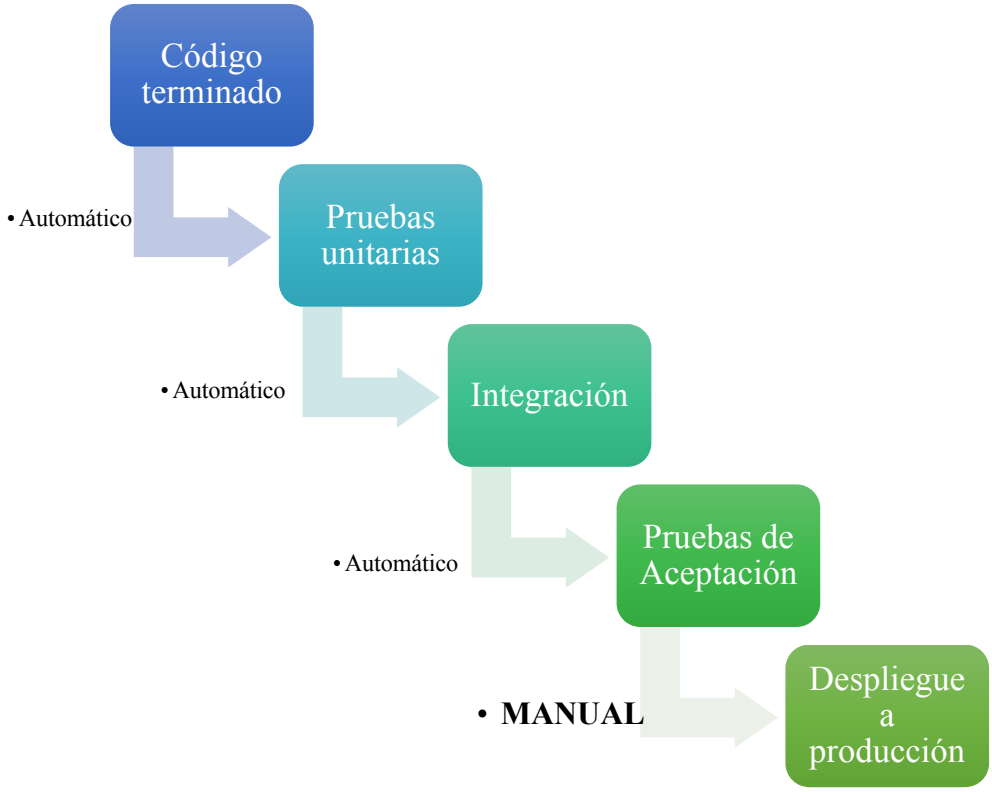


Gráfico 23: Entrega Continua.
Elaborado por: Investigador.

c) Continuous Deployment - CD

Despliegue Continuo, en español. El mismo autor nos define que a diferencia de la Entrega Continua, el paso de hacer el despliegue a producción en este caso es automático. Este modelo se complementa con la Entrega Continua, pues el resultado del mencionado permite hacer un despliegue exitoso. Este quizá es el objetivo primordial para un DevOps, pues con esto tiene controlado el proceso completo de producción de Software.

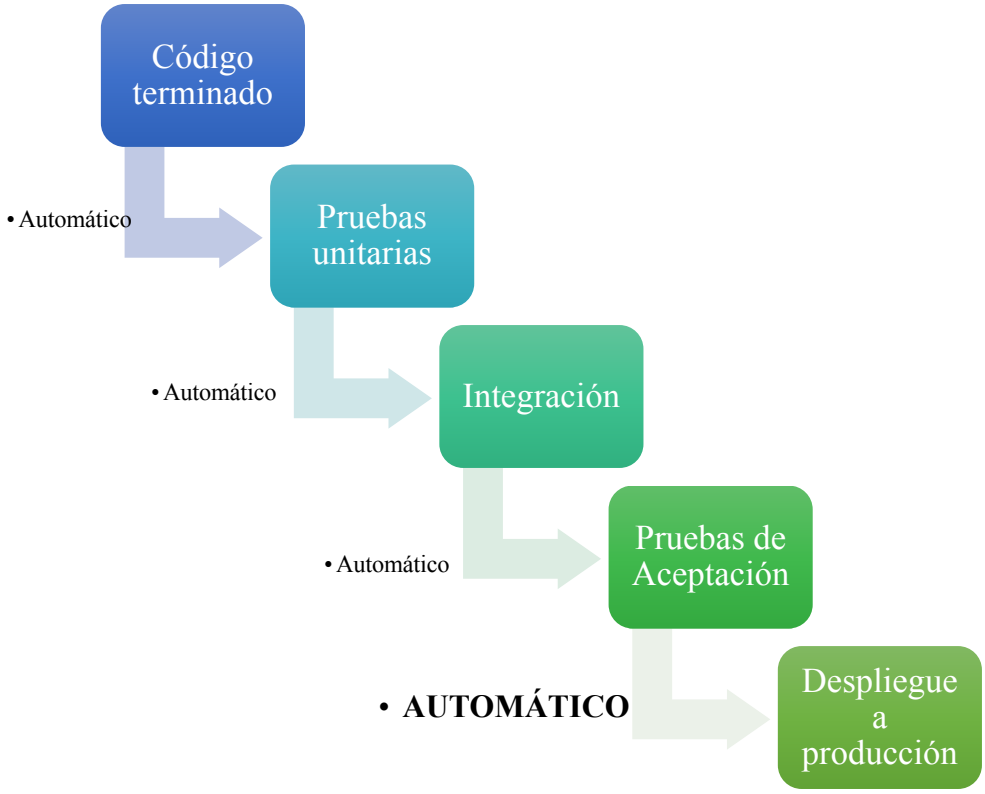


Gráfico 24: Despliegue Continuo.
Elaborado por: Investigador.

2.10. Rendimiento

En su investigación, (Prashant, 2016) señala que la abundancia de Computación en la Nube en la última década ha llevado a un crecimiento significativo en las técnicas de virtualización utilizadas. Esto da paso a un tema que muchas veces pasa desapercibido, pero es de gran importancia para que un producto de Software funcione adecuadamente. El rendimiento, viene dado por diferentes factores, cada uno con parámetros únicos que permiten conocer su efectividad realizando el procesamiento de datos.

Algunos investigadores optan por la virtualización como parte fundamental para mejorar el rendimiento, otros autores como (Khazaei, Barna, Beigi-Mohammadi, & Litoiu, 2016) tienen tendencias de investigación distintas pero el mismo enfoque en mejoras de rendimiento, pues los autores señalan que las arquitecturas modernas como los Microservicios, han iniciado una nueva tendencia para aplicaciones de desarrollo/despliegue en la nube, debido a su flexibilidad, escalabilidad, mantenibilidad y rendimiento.

(Amaral et al., 2015) coincide y agrega que usar nuevas tecnologías como los Microservicios reduce la complejidad de los sistemas informáticos y si se le aplica la tecnología de los Contenedores prácticamente se está potenciando por la ligereza de estos, reduciendo tiempos y siendo más eficientes al producir Software.

En torno a esto, a continuación, se pone a consideración los factores y sus respectivos parámetros que permiten medir el rendimiento.

a) Rendimiento de la Aplicación.

Parámetro de Medición: Rendimiento.

Mide el tiempo de respuesta desde que ingresan las solicitudes a ser procesadas por la aplicación hasta que terminan con una respuesta desde el servidor. No necesariamente se la considera como parte de la medición de eficiencia de los recursos de Hardware, sin embargo, el tiempo total que dure la ejecución de las pruebas pueden servir para hacer la comparativa de eficiencia en relación con el tiempo que pueda durar la ejecución en cada arquitectura.

b) Rendimiento de la CPU.

Parámetro de Medición: Rendimiento (Throughput por sus siglas en inglés).

Se mide la salida de la carga de trabajo a la que la CPU está sujeta, a este proceso se lo conoce como High Performance Computing – HPC (Cálculo de Alto Rendimiento en español).

c) Rendimiento de la Memoria RAM.

Parámetro de Medición: Ancho de Banda (Bandwidth por sus siglas en inglés).

Permite la medición de la velocidad de acceso y operación de la memoria. Por lo general, a la memoria se la somete a pruebas de estrés accediendo a la misma de forma secuencial o aleatoria, tratando de sobrecargar poniendo al límite su capacidad y analizando su comportamiento. Durante la ejecución de las pruebas de estrés, la mayoría

de los errores están relacionados al desbordamiento de memoria (Overflow en inglés), que se da cuando la capacidad de memoria llega a su límite y el sistema envía una alerta del error o en el peor de los casos el sistema completo deja de funcionar.

d) Rendimiento de la Red.

Parámetro de Medición: Ancho de Banda (Bandwidth por sus siglas en inglés).

Mide el rendimiento de la comunicación de red. El escenario de comunicación es transfiriendo una gran cantidad de datos sobre una conexión simple de Protocolo de Control de Transmisión – TCP[25] (Transmission Control Protocol en inglés) como un modelo cliente – servidor. En este caso, depende de la configuración de red para su evaluación, por lo cual se debe tomar muy en cuenta los escenarios y dicha configuración.

e) Rendimiento del Disco.

Parámetro de Medición: Rendimiento (Throughput por sus siglas en inglés).

Permite medir la eficiencia de las operaciones del disco. Usualmente se realizan pruebas de escritura y lectura de forma secuencial o aleatoria para medir los tiempos de respuesta.

Para (Prashant, 2016), es factible hacer medidas de las aplicaciones en específico para calcular su rendimiento sobre un ambiente en particular, sin embargo, las mediciones deberían ser comparativas respecto a otra tecnología con similar enfoque, por ejemplo de Bases de Datos o Servidores Web, pues, cada aplicación tiene demandas diferentes de recursos de Hardware.

[25] Es un protocolo usado para la comunicación a través de la red física o virtual.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Tipo de Investigación

Esta investigación es de tipo cuantitativa y cualitativa, porque se va a hacer un análisis de los procesos que intervienen en la experimentación y se utilizarán parámetros para la medición del rendimiento en experimentos de tiempo real sobre ambientes virtuales.

3.2 Diseño de la Investigación

3.2.1 Modalidad de Investigación

Investigación Documental: Por el uso de fuentes como libros, documentos, artículos, revistas, entre otros medios, para la construcción del Marco Teórico (variables dependiente e independiente). Además, permite realizar un proceso de abstracción científica, generalizando sobre los fundamentos que se plantean.

Investigación Bibliográfica: Apoya a la investigación evitando emprender investigaciones ya existentes o similares, determinando el conocimiento real sobre el tema de este proyecto y generar criterios de fundamentación científica, filosófica y legal. Además, permite la reutilización de experimentos ya realizados en otras investigaciones cuando sea necesario.

Investigación Experimental: El investigador realizará actividades intencionales en ambientes controlados (virtuales), con el fin de describir la causa – efecto que produce la investigación en particular.

3.2.2 Tipos o niveles de Investigación

Investigación Correlacional: Para medir el grado de relación entre las variables que se manejan en esta investigación: Arquitectura de Microservicios basada en Contenedores y Rendimiento.

3.3 Métodos

Deductivo: "La deducción es un proceso que parte de un principio general ya conocido para inferir de él, consecuencias particulares" (Gutiérrez M., 2006). Nos permitirá partir de modelos generales (en el caso de las Arquitecturas de Software) para diseñar las comparativas que permitan el análisis respectivo.

Inductivo: "Este Método utiliza el razonamiento para obtener conclusiones que parten de hechos particulares aceptados como válidos, para llegar a conclusiones cuya aplicación sea de carácter general." (Bernal Torres, 2010). Permitirá hacer un análisis profesional de los datos obtenidos, con el fin de determinar: estrategias, recursos, materiales y medios que puedan intervenir en la interpretación de dichos datos.

Analítico – Sintético: "El análisis consiste en descomponer en partes algo complejo, en desintegrar un hecho o una idea en sus partes, para mostrarlas, describirlas, numerarlas y para explicar las causas de los hechos o fenómenos que constituyen el todo" (Leiva Zea, 2010). Permitirá exponer conceptos fundamentales de una Arquitectura de Software y fases del ciclo de vida, esto con el fin de tener un panorama claro que permita correlacionar lo esperado con los resultados obtenidos.

3.4 Estrategias Técnicas

Se utilizará la técnica de Análisis de archivos, debido a que es una investigación experimental y los resultados de la experimentación se registrarán en registros (archivos) de logs para su posterior análisis.

3.5 Instrumentos de investigación

- Registros de logs.
- Dispositivos Móviles: Como equipo de comunicación, recolector de evidencia y multimedia.
- Computadora(s) Portátil(es): Para la realización de pruebas, análisis de datos y la documentación de la presente investigación.

CAPÍTULO IV

MARCO ADMINISTRATIVO

4.1 Vialidad

El desarrollo de la presente investigación toma a consideración lo siguiente:

Factibilidad Operativa: Se cuenta con las herramientas necesarias para la investigación, incluyendo el conocimiento y el ambiente adecuado.

Factibilidad Tecnológica: Los recursos a utilizarse son considerados como Software Libre, y están disponibles para ser utilizados en todo momento.

Factibilidad Económica: El financiamiento de la investigación está a cargo del investigador.

4.2 Valor Práctico

Beneficiario Directo, Son los expertos en Desarrollo de Software y empresas dedicadas a la producción de Software, que hagan uso de los resultados de este análisis para la toma de decisiones.

Beneficiario Indirecto, Son las empresas que proveen los servicios de Hosting, pues el mejorar el rendimiento incide en una mejor gestión de recursos y reducción de costos.

4.3 Presupuesto

PRESUPUESTO DETALLADO					
1	**EQUIPOS, SOFTWARE Y SERVICIOS**	VALOR	2	**RECURSOS HUMANOS, TRANSPORTE**	VALOR
	COMPUTADOR	$1,000.00		ASESORAMIENTO	$400.00
	SERVICIOS	$300.00		SALIDAS Y VISITAS	$300.00
				TRANSPORTE	$700.00
	SUBTOTAL 1	$1,300.00		SUBTOTAL 2	$1,400.00
3	**MATERIALES Y SUMINISTROS**	VALOR	4	**MATERIAL BIBLIOGRAFICO**	VALOR
	PAPEL RESMA	$30.00		THE DOCKER BOOK	$10.00
	FOTOCOPIAS	$20.00			
	IMPRESIONES	$200.00			
	SUBTOTAL 3	$250.00		SUBTOTAL 4	$10.00
PRESUPUESTO GLOBAL					
1	EQUIPOS, SOFTWARE Y SERVICIOS				$1,300.00
2	RECURSOS HUMANOS, TRANSPORTE Y SALIDAS DE CAMPO				$1,400.00
3	MATERIALES Y SUMINISTROS				$250.00
4	MATERIAL BIBLIOGRÁFICO				$10.00
SUBTOTAL					$2,960.00
(+) 10% IMPREVISTOS					$296.00
(=) VALOR TOTAL					**$3,256.00**

Tabla 11: Presupuesto.
Elaborado por: Investigador.

4.4 Cronograma de actividades del Plan de Investigación

Las actividades señaladas en el cronograma se las ejecutarán durante los años 2017 y 2018.

Actividades	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. Recopilación bibliográfica																								
2. Elaboración del Marco Teórico																								
3. Elaboración de instrumentos																								
4. Prueba de instrumentos																								
5. Recolección de datos																								
6. Procesamiento de datos																								
7. Análisis de los datos																								
8. Redacción del borrador																								
9. Revisión y corrección del borrador																								
10. Presentación del informe																								

Tabla 12: Cronograma de Actividades
Realizado por: Investigador.

CAPÍTULO V

EXPERIMENTO: RESULTADOS Y ANÁLISIS

5.1 Arquitectura de Software, Virtualización y Contenedores.

5.1.1 Descripción

Esta investigación sigue lineamientos propuestos en investigaciones previas por otros autores, y que tienen un mismo fin en común que es el demostrar cuan eficiente puede resultar el hacer uso de tecnología moderna en nuestros proyectos de investigación o desarrollo, en este caso comparando tipos de Arquitectura de Software y Virtualización.

Se presentan dos escenarios, el primero es la Aplicación basada en una Arquitectura Monolítica ejecutándose sobre una KVM (ver sección 5.1.4.1). El segundo escenario es la misma Aplicación basada en una Arquitectura de Microservicios que se ejecuta usando Contenedores (Ver sección 5.1.4.2).

Hay un tercer escenario, es el Host o máquina física sobre la cual se ejecutan: la KVM y los Contenedores, también, desde la misma se ejecutan las pruebas de estrés y se recolectan los datos para su posterior análisis.

5.1.2 Hardware utilizado

En un principio, para la implementación de la Aplicación base y la configuración de las pruebas de estrés se utilizó una computadora con las características presentadas en la Tabla 13:

Característica	Detalle
Nombre dispositivo	MacBook Pro (Retina, 13-inch, Late 2013)
CPU	2.4 GHz Intel Core i5
RAM	8 GB 1600 MHz DDR3
Disco	Macintosh HD – APPLE SSD 250GB
Gráficos	Intel Iris 1536 MB

Sistema Operativo	macOS High Sierra 10.13.3

Tabla 13: Características de la primera computadora.
Realizado por: Investigador.

Posteriormente, para la ejecución de pruebas de estrés y recolección de datos de los archivos o logs, se utilizó una computadora con las características señaladas en la Tabla 14:

Característica	Detalle
Nombre dispositivo	Dell Inspiron 5567
CPU	Intel Core i7-7500U 2.7 GHz 7ma Generación
RAM	16GB DDR3
Disco	HD 1TB
Gráficos	AMD RADEON R7-M445 HD Graphics
Sistema Operativo	Linux Ubuntu 16.04 LTS

Tabla 14: Características de la segunda computadora.
Realizado por: Investigador.

5.1.3 Software utilizado

El Software utilizado para el experimento es de Código Abierto, con excepción de NewRelic para lo cual se hizo uso de una versión de prueba. A continuación, la lista:

- **Linux Ubuntu**, versión: Desktop y Server, 16.04 LTS 64-bit, Kernel 4.4.0-119-generic. A la fecha es la distribución de Linux estable y con soporte continuo. (Ubuntu, 2018) en su sitio Web lo define como un Software de Sistema Operativo de código abierto que se ejecuta desde en una computadora de escritorio, hasta la nube y hasta las cosas conectadas a Internet. Es característico por ser sencillo de instalar, utilizar y cuenta con una amplia lista de aplicaciones disponibles desde su tienda de Software.

  Ubuntu Desktop, se usa en este proyecto como el Sistema Operativo base y sobre el cual se ejecutarán las diferentes pruebas necesarias para llevar a cabo esta investigación.

Ubuntu Server, es la versión que se ejecutará sobre la KVM que a su vez está instalada en Ubuntu Desktop. Esto con el fin de poder ejecutar pruebas de estrés y poder recolectar la información de performance que se genere.

- **QEMU,** versión: 2.10.2. (QEMU, 2017) se define como una máquina emuladora y virtualizador genérico y de código abierto. Este es el Software que nos permitirá crear la KVM con Ubuntu Server. Por defecto, esta aplicación se la usa mediante comandos para la generación de las Máquinas Virtuales, por ello es necesario utilizar una aplicación extra llamada Virtual Machine Manager – VMM (Manejador de Máquinas Virtuales en español).

- **Virtual Machine Manager,** versión: 1.3.2 powered by libvirt. (Berrangé, 2017) lo describe como, una interface de usuario de escritorio para el manejo de Máquinas Virtuales a través de libvirt.

- **Docker,** versión: Cliente y Servidor 18.03.0-ce. En el sitio Web Oficial (Docker, 2017) lo define como una plataforma abierta tanto para desarrolladores como para administradores de sistemas, tal que puedan crear, transportar y ejecutar sus sistemas informáticos. Esto puede hacerse en computadoras portátiles, Centros de Datos de Máquinas Virtuales e inclusive en la Nube. En esta investigación se enfoca para administrar los Contenedores que se utilizarán para ejecutar los distintos servicios.

- **Docker Compose,** versión: 1.14.0. (Docker, 2017), lo define como una herramienta para definer y ejecutar aplicaciones de Docker multi-Contenedores. Básicamente esta herramienta se usa en este experimento para definir las configuraciones del ambiente de Microservicios.

- **Sublime,** versión: 3.0. (Sublime, 2018), es un sofisticado editor de texto para código, marcado y prosa. En esta investigación, se lo utiliza para editar el código de la aplicación y configuraciones que se realizarán en la adecuación de los ambientes.

- **CLI – Command Line Interface,** es la Interfaz de Linea de Comandos la cual utilizamos para hacer la instalacion de aplicaciones via comandos o ejecutar comandos personalizados con el fin de procesar y desplegar nuestra aplicación, de igual forma se usa para la ejecución de las pruebas de estrés debido a que con esta interfaz es mas eficiente con el uso de recursos.

- **Git,** versión:2.7.4. (GIT, 2017) es un sistema de control de versiones libre y distribuido como código abierto diseñado para manejar todo desde pequeños hasta proyectos muy grandes con rapidez y eficiencia. En la investigación se usa para mantener actualizada la aplicación (código, archivos, recursos) en todos los ambientes manteniendo así la integridad y asegurando que tanto procesos como resultados sean los adecuados.

- **Java 8 (JRE),** versión: 1.8.0_162. (Java, 2017), es la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en Web y Software de empresa. En la presente investigación se la usa únicamente el entorno de ejecución de Java para levantar el Servidor Agente y ejecutar JMeter.

- **NodeJS,** versión: 6.14.1. (Foundation, 2018), es un ambiente de ejecución de JavaScript construido sobre el motor de Chrome V8 JavaScript. Usa un modelo de Entrada/Salida no bloqueado y controlado por eventos que lo hace liviano y eficiente. La Aplicación toma a esta tecnología como base para su funcionamiento. Es mundialmente conocida y aceptada, muy usada por grandes companias para proyectos de alto nivel.

- **NPM,** versión: 3.10.10. Es el manejador de paquetes de JavaScript, se usa en esta investigación básicamente para manejar las dependencias de librerías utilizadas para desplegar la aplicación.

- **MySQL,** versión: 5.7.21. (Oracle, 2018), es un sistema de administración de Bases de Datos relaciones de código abierto. Muy conocido y usado a nivel

mundial para proyectos de diferentes tamaños. Se usa en esta aplicación para almacenar las Publicaciones generadas en las pruebas de estrés.

- **MongoDB,** versión: 3.4.10. (MongoDB, 2018) es una Base de Datos basada en documentos escalable y flexible para consultas e indexación. En este proyecto se usa para almacenar la información generada para los Hilos y Usuarios.

- **Apache JMeter,** versión: 3.3. (Apache, 2018) es un software de código abierto, una aplicación Java diseñada para cargar el comportamiento funcional de la prueba y medir el rendimiento. Esta herramienta se usa tanto para configurar las pruebas de estrés como para ejecutarlas y recolectar información del comportamiento y respuesta de la aplicación en tiempo real al someterla a dichas pruebas.

- **Server Agent,** versión: 2.2. (JMeter, 2013), componente de JMeter para recolectar información del performance. En este proyecto se lo usa como servicio que ayuda en la recolección e interpretación de datos generados durante las pruebas de estrés.

- **NewRelic,** (NewRelic, 2018) brinda un servicio de análisis de rendimiento profundo para cada parte del entorno del Software. Permite ver y analizar grandes cantidades de datos y obtener información útil en tiempo real. En la aplicación, se usa al igual que JMeter, pero en este caso se enfoca más bien en la recolección de información sobre el rendimiento del ambiente donde se ejecuta la aplicación, ideal para recolectar información durante las pruebas de estrés y posteriormente analizarlas en gráficas en tiempo real.

Para la configuración de las pruebas de estrés, se utiliza la aplicación JMeter 2.0 por medio de una Interfaz Gráfica de Usuario (Graphic User Interface - GUI[26] por sus siglas

[26] Es la interfaz que utiliza un usuario para interactuar con una aplicación sin necesidad de ejecutar comandos manualmente.

en inglés), y para la ejecución de dichas pruebas de estrés se lo realiza mediante la ejecución de comandos utilizando una aplicación de Línea de Comandos.

Para la recolección de datos resultantes de las pruebas de estrés, se ha recurrido al uso de dos herramientas muy avanzadas con el fin de verificar que la interpretación de dichos resultados sea fehaciente, estas herramientas son: JMeter y NewRelic, cada uno se lo detalla en la sección 5.1.3.

5.1.4 La Aplicación – App

La Aplicación utiliza tecnología de programación y desarrollo Web modernas, tenemos: Node.js para la generación de servicios y Bases de Datos SQL (MySQL) y NoSQL (MongoDB) para almacenar los datos. Se presentan dos escenarios (Ver sección 6.1.5) que muestran la misma App usando diferentes arquitecturas. Posteriormente se detallan los ambientes virtuales donde se levantan los servicios y bajo los mismos se ejecutan las pruebas de estrés para cada versión de la App.

Para su implementación se tomó como base el ejemplo publicado por Amazon Web Services - AWS (Servicios Web de Amazon, en español) en su repositorio público (https://github.com/awslabs/amazon-ecs-nodejs-microservices), y se realizaron las modificaciones necesarias con el fin de acoplar la App para fines experimentales de esta investigación.

Para obtener únicamente la App en su versión final, sin virtualización, básicamente se sigue un flujo de trabajo como se representa en el Gráfico 25, permitiendo obtener una Aplicación de Backend[27] simple y capaz de soportar las cargas de estrés que simulen situaciones que podrían suceder en la realidad.

[27] Es un tipo de programación enfocado en temas de funcionalidad y servicios que se ejecutan en el lado del servidor mayormente y que no tiene mucha relación con la estética de la interfaz con la cual un usuario interactúa.

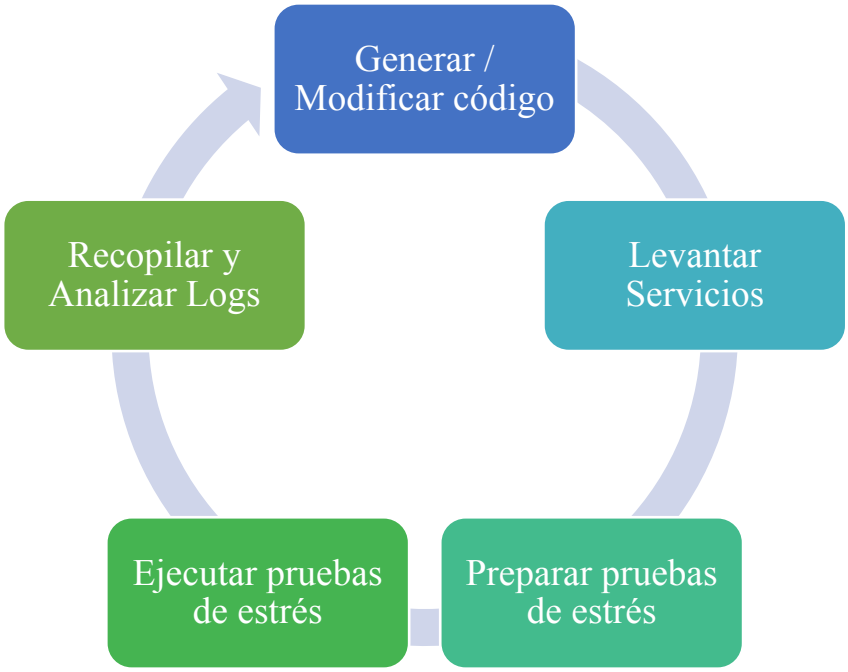


Gráfico 25: Flujo de trabajo para implementar la Aplicación.
Elaborado por: Investigador.

En general, se representa la evolución de una App basada en una Arquitectura Monolítica y el resultado que es la misma App usando una Arquitectura de Microservicios. A esto se suma la adecuación de los ambientes virtuales donde se levantarán los servicios y se ejecutarán las pruebas.

En esencia, la Aplicación representa el diseño básico - en BackEnd[28] - para el manejo de foros, chats, comentarios o notificaciones. Para ello se identifican tres elementos principales que se describen en el Gráfico 26

[28] En programación, es un término usado para identificar procesos que se llevan a cabo en el lado del servidor.

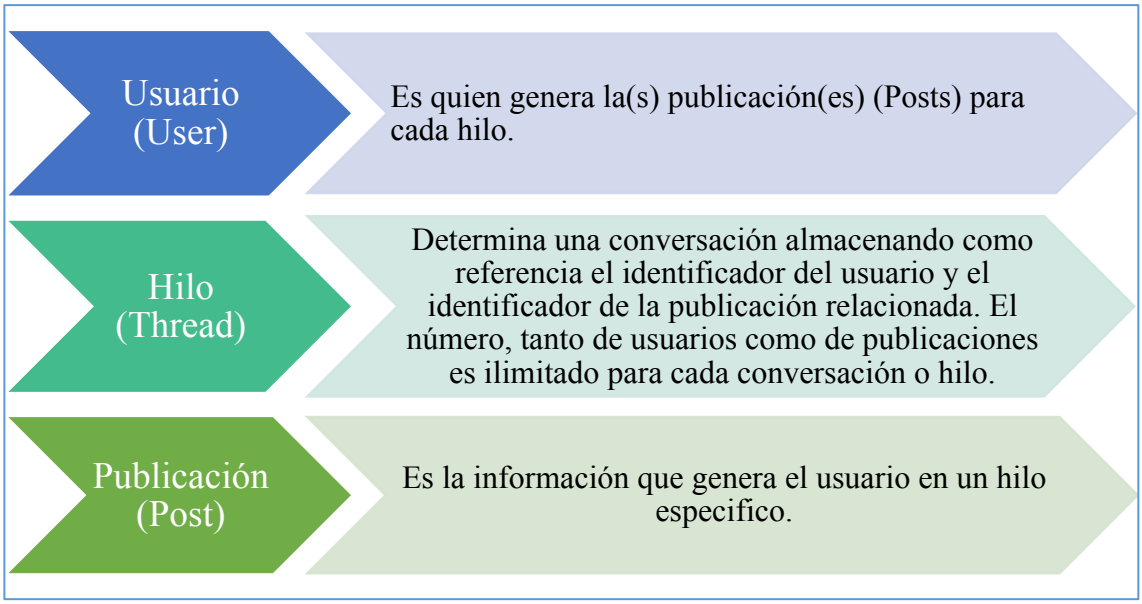


Gráfico 26: Elementos básicos de la Aplicación en general.
Elaborado por: Investigador.

5.1.4.1 Versión 1: Aplicación Monolítica

La Aplicación Monolítica, por definición viene con una estructura de directorio sencilla y con un número reducido de archivos, debido a que no existe segmentación de código y tampoco lo exige este tipo de Arquitectura, aunque es opcional. En el Gráfico 27 se muestra la estructura del directorio.

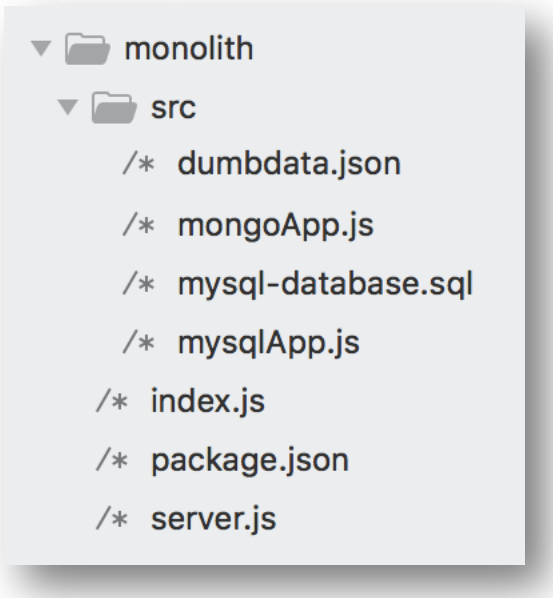


Gráfico 27: Estructura del directorio de la Aplicación con Arq. Monolítica.

Elaborado por: Investigador.

A continuación, en la Tabla 15 se listan los archivos y el detalle acerca del contenido o detalle que tiene cada uno. Se han omitido algunos archivos que no forman parte del proyecto en sí y pueden ser ignorados, estos pueden ser: archivos de dependencia, archivos ocultos de configuración, archivos usados para capturar los logs de la Aplicación o archivos temporales usados por el investigador.

Archivo	Detalle
src/dumbdata.json	Contiene datos de ejemplo en formato JSON, y se usa al momento de generar contenido en los tres servicios: usuarios, hilos y publicaciones.
src/mongoApp.js	Contiene el código fuente para conectarse a la Base de Datos en MongoDB, generar contenido aleatorio y almacenarlo.
src/mysql-database.sql	Contiene en código, las sentencias en SQL para crear la Base de Datos y Tabla en MySQL que se usan en este experimento.
src/mysqlApp.js	Contiene el código fuente para conectarse a la Base de Datos en MySQL, generar contenido aleatorio y almacenarlo.
index.js	Contiene código fuente para inicializar y levantar los servicios.
package.json	Describe las dependencias de librerías requeridas para poder ejecutar esta aplicación. Además, en este archivo se indica mediante un Script el archivo que se debe ejecutar para inicializar los servicios.
server.js	Contiene el código fuente que implementa los endpoints para cada servicio, tanto para generar los datos como para hacer llamadas de consulta.

Tabla 15: Archivos de la Aplicación Monolítica.
Elaborado por: Investigador.

Para la ejecución de esta Aplicación se requiere tener instalado: Navegador, CLI, Git, NodeJS, MongoDB y MySQL.

Los pasos para ejecutar la Aplicación son los siguientes:

1) Independiente del Sistema Operativo, instalar y configurar: NodeJS, MongoDB y MySQL.
2) Clonar el proyecto desde el repositorio en GitHub. Usando el CLI ejecutar el comando: 'git clone git@github.com: alexissaransig/monolith-microservice-performance-test.git'
3) Usando el CLI, ingresar hasta la carpeta raíz de la Aplicación Monolítica. Considérese /monolith como la carpeta raíz.
4) Manejo de dependencias, para ello se debe ejecutar el comando: 'npm install'.
5) Levantar los servicios, se ejecuta el comando: 'npm start'.
6) Para generar usuarios, hilos o publicaciones, en el navegador seguir las siguientes URL:
   - localhost:3000/api/generate/users
   - localhost:3000/api/generate/threads
   - localhost:3000/api/generate/posts
7) Para seleccionar usuarios, hilos o publicaciones, en el navegador seguir las siguientes URL:
   - localhost:3000/api/users
   - localhost:3000/api/threads
   - localhost:3000/api/posts

Se destaca que la Aplicación, en momentos puede demorarse en dar una respuesta, eso es debido a la gran cantidad de datos que debe procesar antes de presentarlo en el navegador. Además, este no es el enfoque que se sigue para realizar las pruebas de carga de estrés, sino, una de las formas como el investigador pudo comprobar que la Aplicación funciona correctamente.

Los pasos del paso 1 al 5, son los mismos que se deben seguir para levantar los servicios cuando la KVM esté disponible.

5.1.4.2 Versión 2: Aplicación con Microservicios

En esta versión, la Aplicación tiene segmentados sus archivos en una estructura de directorio más organizada por funcionalidades. En este caso en particular, se considera a cada elemento señalado en el Gráfico 26 de la Aplicación, como una parte que puede ser segmentada y ejecutada como un servicio independiente.

Como se mencionó antes, la estructura del directorio y localización de archivos cambia tal como se muestra a continuación en el Gráfico 28.

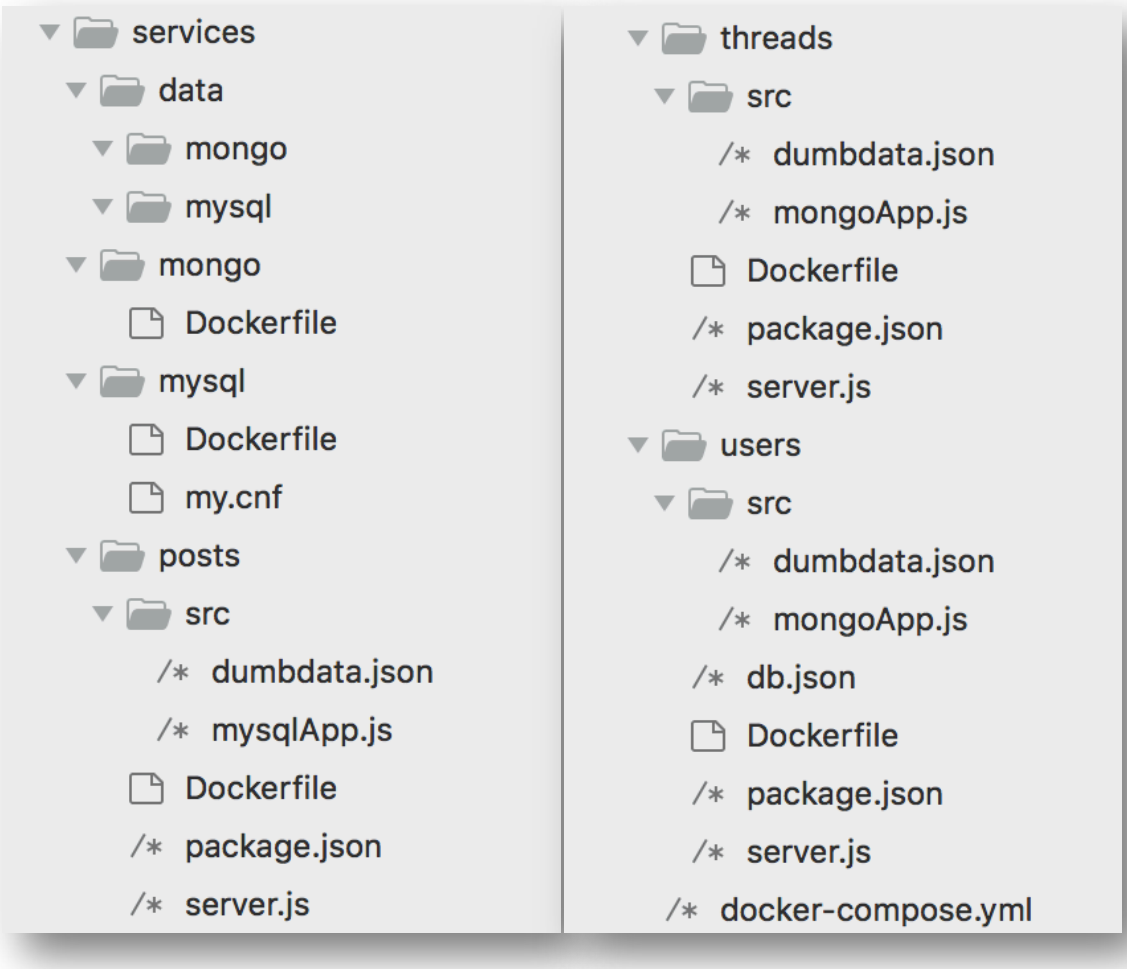


Gráfico 28: Estructura del directorio de la Aplicación con Arq. De Microservicios.
Elaborado por: Investigador.

En la Tabla 16 se listan los archivos y el detalle acerca del contenido o detalle que tiene cada uno. Se han omitido algunos archivos que no forman parte del proyecto en sí y pueden ser ignorados, estos pueden ser: archivos de dependencia, archivos ocultos de

configuración, archivos usados para capturar los logs de la Aplicación o archivos temporales usados por el investigador.

Archivo	Detalle
services/data/mongo **services/data/mysql**	Contienen configuraciones y archivos de las Bases de Datos correspondientes. Docker sincroniza este directorio con el existente dentro del Contenedor para mantener la información en el Host y no en el Contenedor. Docker permite implementar volúmenes, y cada vez que inicie un Contenedor se reutilizara el estado de la última versión de la Base Datos. En este caso en particular la configuración esta especificada en el archivo docker-compose.yml.
services/mongo/Dockerfile **services/mysql/Dockerfile** **services/mysql/my.cnf**	El Dockerfile, en el directorio correspondiente, contiene el Script que se ejecutará para generar la Imagen del Contenedor a utilizarse en cada caso. En el caso del archivo mysql/my.cnf, por buenas prácticas se recomienda tener un archivo de configuración aparte que será agregado automáticamente al momento de generar la imagen de MySQL.
services/ [post, thread, user] /Dockerfile	Para el caso de los tres servicios: Usuario, Hilo y Publicación, se usa la misma configuración de Dockerfile. Es decir, usan la misma imagen base para instanciar sus Contenedores. El archivo básicamente configura NodeJS y lo ejecuta como servidor de cada servicio.
services/ [post, thread, user] /src/dumbdata.json	Contiene datos de ejemplo en formato JSON, y se usa al momento de generar contenido en los tres servicios: usuarios, hilos y publicaciones.
services/ [thread, user] /src/mongoApp.js	Contiene el código fuente para conectarse a la Base de Datos en MongoDB, generar contenido aleatorio y almacenarlo.

services/post/src/mysqlApp.js	Contiene el código fuente para conectarse a la Base de Datos en MySQL, generar contenido aleatorio y almacenarlo.
services/ [post, thread, user] /package.json	Describe las dependencias de librerías requeridas para poder ejecutar el servicio individual. Se indica también mediante un Script el archivo que se debe ejecutar para inicializar el servicio.
services/ [post, thread, user] /server.js	Contiene el código fuente que implementa los endpoints para cada servicio, tanto para generar los datos como para hacer llamadas de consulta. En cada servicio se especifica el endpoint únicamente para el servicio independiente.
services/docker-compose.yml	Contiene la configuración principal para levantar los tres servicios individualmente, así también los servicios de Bases de Datos y volúmenes que mantendrán sincronizadas las configuraciones y datos.

Tabla 16: Archivos de la Aplicación con Microservicios.
Elaborado por: Investigador.

Para la ejecución de esta Aplicación se requiere tener instalado: Navegador, CLI, Git, Docker y Docker-Compose.

Los pasos para ejecutar la Aplicación son los siguientes:

1) Dependiendo el Sistema Operativo, seguir los pasos especificados en el sitio oficial de Docker (www.docker.com) para instalar Docker CE[29] y Docker-Compose.
2) Clonar el proyecto desde el repositorio en GitHub. Usando el CLI ejecutar el comando: 'git clone git@github.com: alexissaransig/monolith-microservice-performance-test.git'

[29] Es la abreviación de Docker Community Edition, es la versión gratuita de Docker.

3) Usando el CLI, ingresar a la carpeta services/post. Aquí se encuentra el archivo Dockerfile que contiene el script para generar nuestra imagen de Docker para usarlo en cada servicio.
4) Genera la imagen de Docker para los servicios. Ejecutar el comando 'docker build -t node-exp1 .' (con el punto al final).
5) Usando el CLI, ingresar hasta la carpeta raíz de la aplicación con Microservicios. Considérese /microservices/services como la carpeta raíz. En este directorio se encuentra el archivo docker-compose.yml, el cual contiene las configuraciones principales para iniciar los servicios de la aplicación.
6) Ejecuta la aplicación y los servicios. Ejecutar el siguiente comando: 'docker-compose up'. La primera vez puede demorarse unos minutos hasta descargar las imágenes y luego configurarlas. Las siguientes ejecuciones deberían tomar breves segundos.
7) Para generar usuarios, hilos o publicaciones, en el navegador seguir las siguientes URL:
   - localhost:3001/api/generate/users
   - localhost:3002/api/generate/threads
   - localhost:3003/api/generate/posts

8) Para seleccionar usuarios, hilos o publicaciones, en el navegador seguir las siguientes URL:
   - localhost:3001/api/users
   - localhost:3002/api/threads
   - localhost:3003/api/posts

De los pasos anteriores, se puede notar que para este experimento se exponen a cada servicio en un puerto diferente. Al igual que en la Aplicación en su version Monolítica, al momento de generar o seleccionar datos puede demorarse en dar una respuesta debido a la gran cantidad de datos que se manejan cada vez q se hace una llamada a un Enpoint. Estos pasos son únicamente para hacer pruebas de la Aplicación y su correcto funcionamiento.

Los pasos del 1al 4, son los mismos que se deben seguir para levantar los servicios en el ambiente de pruebas usando Contenedores.

5.1.5 Experimento

Las características de los escenarios fueron adecuadas de tal manera que la Aplicación en ambas versiones de Arquitectura, tenga los mismos recursos y configuraciones, esto se lo hace porque los Scripts de las pruebas de estrés son iguales para ambos escenarios, (Ver Anexo 1). En este experimento se proponen dos escenarios, el primero esta adecuado con tecnología tradicional, la misma que se la identifica en el Capítulo I como el problema a resolver, en tanto que el segundo escenario, presenta una propuesta con tecnología moderna y también se la menciona en el Capítulo I como la solución en cuanto a eficiencia de recursos tiene que ver. Los escenarios de prueba se los puede revisar en el Anexo 2, donde se presentan capturas de pantalla con los escenarios en ejecución.

5.1.5.1 Escenario 1: Aplicación con Arquitectura Monolítica y KVM.

A este escenario se lo puede considerar como una versión tradicional que se usa hasta la actualidad en proyectos tecnológicos de diferentes tamaños y fines. Para la KVM se tiene definida su estructura base como la expuesta en el Gráfico 12 (Ver sección 2.6). En cuanto a la Aplicación, tiene una estructura diferente a la expuesta en el Gráfico 11 (Ver sección 2.5) para la Arquitectura Monolítica. A continuación, se presenta el enfoque tomado para la Aplicación. Ver Gráfico 29.

La KVM contiene las aplicaciones y librerías necesarias para que la Aplicación pueda ejecutarse y exponer 2 puertos que permiten hacer llamadas desde el Host.

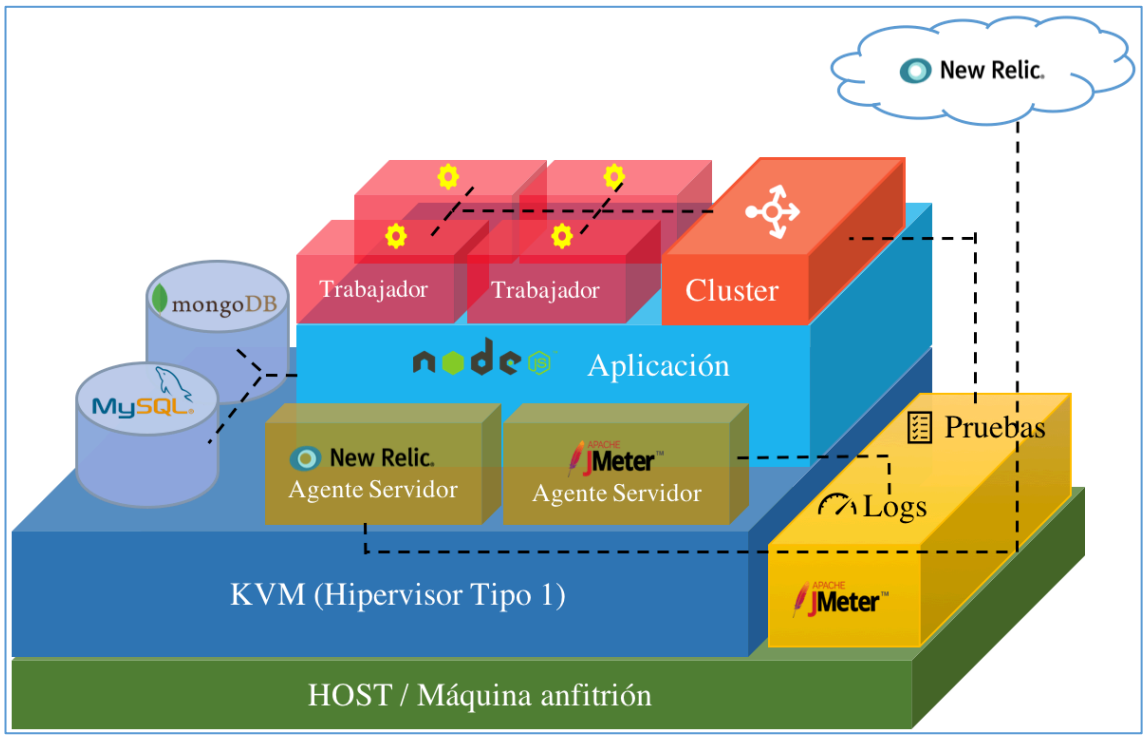


Gráfico 29: Estructura de la Aplicación Monolítica con KVM.
Elaborado por: Investigador.

Las Bases de Datos – BD, tanto MySQL como MongoDB, inician sus servicios al momento que se enciende la KVM. Estas Bases de Datos responden a las solicitudes que se envían desde los trabajadores o Workers[30].

Para aprovechar al máximo los recursos configurados en la KVM, se usa un Cluster[31] manejado por una librería de NodeJS, con esto se puede generar un Worker por cada núcleo existente. Un Worker, es un proceso que se ejecuta sobre un núcleo lógico y brinda el servicio de conexión para interactuar con la Aplicación. En este experimento se cuenta con 4 núcleos lógicos, por lo tanto, se tienen 4 Workers que no solamente procesan las solicitudes de conexión sino también dan estabilidad al sistema de la Aplicación en general garantizando disponibilidad en caso de que un Worker falle por algún error en el procesamiento de solicitudes. Cada Worker tiene establecida una conexión con las Bases de Datos, además de los Endpoints que permiten especificar el servicio al cual se envía la solicitud desde el cliente.

[30] Proceso en ejecución sobre un núcleo lógico manejado por un cluster encargado del balanceo de carga.
[31] Conjunto de recursos definidos o computadores interconectados para escalar servicios.

El Cluster funciona como un balanceador de carga y asigna el procesamiento de solicitudes a los Workers para finalmente enviar una respuesta al cliente. El puerto 3000 que es por defecto para servidores de NodeJS, es el mismo que se usa para enviar solicitudes desde el Host, en este caso usando JMeter.

El servicio JMeter Server Agent, localizado dentro de la KVM expone el puerto 4444, y ejecuta un demonio que recepta instrucciones que se envían desde JMeter (en el Host) al momento de ejecutar las pruebas de estrés para iniciar la recolección de datos relacionados al rendimiento de los recursos ejecutando dichas pruebas.

Por otro lado, esta NewRelic Server Agent, que de forma similar al JMeter Server Agent recolecta datos de rendimiento con la diferencia de que estos datos son más generales, es decir, recolecta datos desde que se inicia como servicio al arrancar el Sistema Operativo. Los datos recolectados se almacenan en archivos logs en el Host y se sincronizan en tiempo real con su servidor en la Nube (NewRelic Server) donde analiza, interpreta y muestra los resultados con graficas interactivas, muy útil para un rápido análisis y entendimiento. Este servicio permite ver el comportamiento del servidor en general y efectivamente ver el rendimiento de los recursos al momento de ejecutar las pruebas de estrés. El fin de usar este servicio en coordinación con el de JMeter es para corroborar los resultados y que el análisis de estos demuestre la factibilidad del presente trabajo.

La KVM que se ejecuta sobre el Host tiene las siguientes características:

Característica	Detalle
CPU	4 núcleos lógicos 2.4 GHz
RAM	16 GB
Disco	30GB HD IDE
Red	Red Virtual NAT
Sistema Operativo	Ubuntu Server 16.04 – x86_64

Tabla 17: Características KVM.
Elaborado por: Investigador.

Con las características detalladas en la Tabla 17, lo que se trata es de usar al máximo los recursos del Host, con relación a las características de este se muestran en la Tabla 14, en la KVM se usan todos los núcleos de CPU disponibles y 16 de los 16GB de RAM. Esto no afecta el rendimiento del Host pues en las pruebas de estrés no hay momento en que se usen esos recursos al 100%.

En el lado del Host esta JMeter, aplicación basada en tecnología Java que permite la edición de los Scripts para crear las pruebas de estrés, así como recolectar los resultados cuando se ejecutan, además de recolectar los logs con datos de los procesos ejecutados y el rendimiento de los recursos de Hardware. En la edición de los scripts se especifican el Host y Puerto (de la KVM) donde se ejecutarán las pruebas de estrés.

5.1.5.2 Escenario 2: Aplicación con Arquitectura de Microservicios y Contenedores.

En este escenario, básicamente lo que se hace es poner en práctica una solución al problema planteado en el Capítulo I. Se propone transformar una Aplicación basada en una Arquitectura Monolítica funcional sobre una KVM en una Aplicación con una Arquitectura de Microservicios funcional sobre Contenedores. Para el manejo de dichos Contenedores se ha optado por la tecnología en tendencia, Docker. Al igual que en el Escenario 1, a continuación, en el Gráfico 30 se representa la estructura del enfoque aplicado para el experimento.

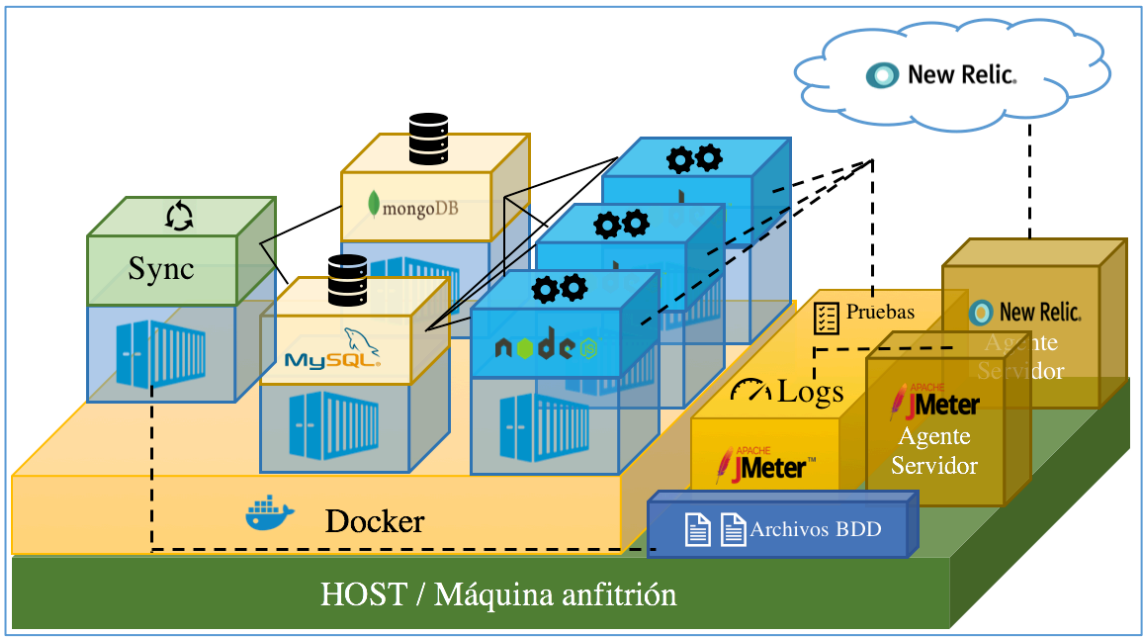


Gráfico 30: Estructura de la Aplicación con Microservicios y Contenedores.
Elaborado por: Investigador.

Docker Engine o Motor de Docker en español, hace lo que comúnmente se conoce como Orquestación de Contenedores[32], que no es más que la administración de estos, además les provee de los recursos necesarios para que cada Contenedor cumpla con las condiciones requeridas y siga los lineamientos que esta tecnología exige.

A diferencia del primer escenario, ya no se usa el Cluster ni los Workers, vale recordar que ese enfoque era para aprovechar al máximo los recursos del sistema, lo cual no es necesario en este escenario ya que Docker se encarga de manejar ese tema siendo más eficiente y aprovechando mejor los recursos. Continuando, para este caso se exponen puertos por cada nuevo Microservicio. Las Bases de Datos también se ejecutan dentro de Contenedores y exponen cada una un puerto a través del cual se puede acceder desde el Host. Agregado a eso, está un Contenedor extra, Sync Volumes, cuyo fin es el de mantener sincronizados los datos y archivos que generen las Bases de Datos en el Contenedor con un directorio en el Host anfitrión, esto es para no perder la información generada una vez que el Contenedor termine su ejecución.

[32] Es la administración de los contenedores de Docker, responsable de la configuración, ejecución y adecuación del ambiente. Generalmente este cargo recae sobre un DevOps experimentado.

Para la interconexión e intercomunicación entre Contenedores se la define en el archivo docker-compose.yml, y tal como se muestra en el Gráfico 30, hay Contenedores que se conectan entre sí por una red privada que la genera Docker al momento de iniciar los servicios. En dicha red, se exponen puertos por los cuales no se puede acceder desde un Host externo, puesto que Docker expone puertos por cada Contenedor para que se pueda acceder desde el Host.

Ahora, existen 3 aplicaciones de NodeJS, cada una representa un servicio que es un segmento del todo, tal como se puede apreciar en el Gráfico 29 (Ver sección 5.1.5.1) cuando el enfoque era una Aplicación Monolítica. Cada una es independiente y cuenta con sus propias librerías y puertos de conexión interna y externa.

El servicio JMeter Server Agent, se localiza en el Host, pues Docker en este caso forma parte del sistema principal. Expone su puerto 4444 para recolectar la información de rendimiento de los recursos cuando se ejecuten las pruebas de estrés. Cabe señalar que esto no afecta los resultados que se obtengan en las pruebas ya que JMeter se centra en el comportamiento de los recursos respecto al script de prueba que se ejecute bajo su supervisión.

El servicio de NewRelic Server Agent, también se encuentra del lado del Host, en este caso la información enviada al Servidor de NewRelic se ve afectada ya que contiene información global que incluye procesos fuera del experimento que se ejecutan y generan consumo de recursos. Para este caso se usa un filtro en el lado del servidor de NewRelic para capturar el comportamiento de los recursos en el momento exacto cuando se ejecutaron las pruebas de estrés, esto nos da una mejor idea y permite hacer una comparativa respecto a los resultados arrojados por en el primer escenario.

Las características del Host sobre el cual se ejecuta Docker se muestran en la Tabla 14 (Ver sección 5.1.2). En cuanto a recursos, se trata de mantener las mismas características en ambos escenarios para que los datos generados en las pruebas sean lo más precisas posible.

5.1.6 Pruebas y Resultados

Las pruebas de estrés son configuradas de igual forma para ambos escenarios. Estas son pruebas generadas en JMeter usando la interfaz gráfica.

Básicamente se han configurado dos escenarios:

El primer escenario, hace solicitudes GET vía HTTP a los endpoints de cada servicio para generar las Bases de Datos y los Datos. El número de datos es variable y se lo puede editar en el código fuente.

El segundo escenario, hace solicitudes GET vía HTTP a los endpoints de cada servicio para seleccionar la información, en este caso selecciona todos los datos generados que se encuentran almacenados en las Bases de Datos (MySQL y MongoDB).

Cabe señalar que estas pruebas están dirigidas únicamente para recolectar información del rendimiento de los recursos en el lado del servidor, es decir, una vez que se obtiene una respuesta desde el servidor se termina el proceso de prueba sin tomar en cuenta el tiempo que toma procesar la información en el lado cliente o más comúnmente conocido como navegador.

La razón de recolectar datos del comportamiento en el lado Servidor es, en el Escenario 1 se usa Ubuntu Server que por defecto no cuenta con interfaz gráfica, en el Escenario 2 tenemos a Docker que de igual forma no cuenta con una Interfaz gráfica. Para ambos escenarios se ejecutan comandos únicamente a través de un CLI incluyendo el comando que ejecuta las pruebas de estrés, en ningún caso se usa una GUI que procese los datos obtenidos en las respuestas desde el servidor, esto con el fin de que el consumo de recursos sea mínimo en la máquina Host. Una vez que las pruebas terminan se procede a usar aplicaciones con interfaz gráfica para procesar, analizar, interpretar y generar reportes.

5.1.6.1 Caso 1

En este caso, se ejecuta la prueba de estrés con un número de datos regular a generar, tal como se muestra en la Tabla 18, con el fin de poder recolectar información suficiente para hacer una primera interpretación y análisis. Este experimento se lo realiza debido a que, en experimentos afines, el rendimiento de algunos recursos cambia dependiendo la cantidad de datos procesados y eso es posible verificar comparando con los resultados del Caso 2.

DATOS					
Escenario JMeter	**Nro. Datos**	**Nro. Repetición**	**Nro. Hilos**	**Nro. Terminales**	**Total Solicitudes**
Generar	100000	1	1	3	3
Seleccionar		30	3	3	270
				Total	273

Tabla 18: Configuración de la prueba de estrés – Ambos escenarios.
Elaborado por: Investigador.

A continuación, en la Tabla 19 se muestran los resultados obtenidos en logs sobre las solicitudes ejecutadas desde JMeter hacia la Aplicación. Estos resultados se los obtiene mediante el uso de la GUI de JMeter, el cual automáticamente procesa los datos almacenados en el archivo log y presenta al usuario para su interpretación.

RESULTADOS		
	Escenario 1	**Escenario 2**
Total solicitudes OK: Error:	 273 0	 273 0
Tiempo duración:	00:01:50	00:01:29
Solicitudes/s. (promedio):	2.5/s	3.1/s

Duración por solicitud		
Min:	7 ms	4 ms
Max:	3677 ms	2793 ms
Promedio:	1150 ms	936 ms
Mediana:	695 ms	312 ms
Desviación estándar:	1094.66 ms	1082.4 ms
Total datos		
Recibidos:	85014.43 KB/s	105193.29
Enviados:	0 KB/s	KB/s
		0 KB/s

Tabla 19: Resultados de la prueba de estrés – Ambos escenarios.
Elaborado por: Investigador.

En la Tabla 20 se muestran los resultados del rendimiento de recursos obtenidos durante la ejecución de las pruebas de estrés para la Aplicación Monolítica.

	CPU (%)	Disco lectura (bytes/s)	Disco escritura (bytes/s)	Memoria (%)	Memoria (MB)	Red recibidos (bytes/s)	Red enviados (bytes/s)
Min	31.313	0	0	3.6	867.207	0	0
Max	79.648	1084	22495232	23.348	3707.085	260251236	280736440
Promedio	53.7275	18.2018	927837.9450	18.0823	2955.7629	89644424.3761	175794816.4587
Mediana	53.263	0	327680	18.846	3074.171	85744688	181136008
Desv. Est.	12.1825	114.7164	3007978.0514	3.9715	568.4580	73383955.1696	49939865.9056

Tabla 20: Rendimiento de recursos – Aplicación Monolítica.
Elaborado por: Investigador.

Para revisar la tabla completa y los gráficos de la prueba de estrés ver el Anexo 3.

A esta información se suman los datos recopilados con la herramienta NewRelic enfocándose al rendimiento de la Red, tal como se muestra en la Tabla 21, en la sección de Análisis se describirá a detalle la razón de este caso especial.

	Red recibidos (Kilobytes/s)	Red enviados (Kilobytes/s)
Promedio	238.6	46899.2

Tabla 21:Rendimiento de red – Logs NewRelic – Aplicación Monolítica.
Elaborado por: Investigador.

En la Tabla 22 se muestran los resultados del rendimiento de recursos obtenidos durante la ejecución de las pruebas de estrés para la Aplicación con Microservicios.

	CPU (%)	Disco lectura (bytes/s)	Disco escritura (bytes/s)	Memoria (%)	Memoria (MB)	Red recibidos (bytes/s)	Red enviados (bytes/s)
Min	31.578	0.000	0.000	13.961	3810.584	0.000	0.000
Max	75.757	3024.00	119980032.000	34.939	5492.270	703874728.000	300732765.000
Promedio	65.955	41.182	4968261.818	29.626	5063.181	430807543.545	217503213.818
Mediana	67.250	0.000	2654208.000	30.313	5120.795	439659753.000	225701617.000
Desv. Est.	8.089	324.981	13852888.717	4.239	341.069	119897485.065	54237027.081

Tabla 22: Rendimiento de recursos – Aplicación de Microservicios.
Elaborado por: Investigador.

Para revisar la tabla completa y los gráficos de la prueba de estrés ver el Anexo 4

Al igual que para el Escenario 1, aquí también se suman los datos recopilados con la herramienta NewRelic enfocándose al rendimiento de la Red, tal como se muestra en la Tabla 23, en la sección de análisis se describirá a detalle la razón de este caso especial.

	Red recibidos (Kilobytes/s)	Red enviados (Kilobytes/s)
Promedio	32.4	40960

Tabla 23:Rendimiento de red – Logs NewRelic – Aplicación con Microservicios.
Elaborado por: Investigador.

5.1.6.2 Caso 2

La cantidad de datos a generar se duplica y se modifica la configuración de ejecución del Script con las pruebas de estrés. En la Tabla 24 se muestran los detalles de las pruebas de estrés que se ejecutarán.

DATOS					
Escenario JMeter	**Nro. Datos**	**Nro. Repetición**	**Nro. Hilos**	**Nro. Terminales**	**Total Solicitudes**
Generar	200000	1	1	3	3
Seleccionar		70	5	3	1050
				Total	1053

Tabla 24: Configuración de la prueba de estrés – Ambos escenarios.
Elaborado por: Investigador.

Las configuraciones de este caso están dadas en un límite máximo (especificadas por el investigador), esto debido a que las características del Hardware y Software utilizados se prestan para ello, modificar con valores mayores genera errores en de diferentes tipos, como: sobrecarga de memoria (JMeter o NodeJS), error de procesamiento de solicitudes, apagado inesperado del servidor por sobrecarga de procesos, entre otros.

A continuación, en la Tabla 25 se muestran los resultados obtenidos en logs sobre las solicitudes ejecutadas desde JMeter hacia la Aplicación. Estos resultados se los obtiene mediante el uso de la GUI de JMeter, el cual automáticamente procesa los datos almacenados en el archivo log y presenta al usuario para su interpretación.

RESULTADOS		
	Escenario 1	**Escenario 2**
Total solicitudes	1053	1053
OK:	1051	1053
Error:	2	0

Tiempo duración:	00:14:17	00:12:08
Solicitudes/s. (promedio):	1.2/s	1.4/s
Duración por solicitud		
Min:	22 ms	8 ms
Max:	35784 ms	10832 ms
Promedio:	3934 ms	3411 ms
Mediana:	2548 ms	715 ms
Desviación estándar:	38.05 ms	4220.21 ms
Total datos		
Recibidos:	86342.29 KB/s	102433.12 KB/s
Enviados:	0 KB/s	0 KB/s

Tabla 25: Resultados de la prueba de estrés – Ambos escenarios.
Elaborado por: Investigador.

En la Tabla 26 se muestran los resultados del rendimiento de recursos obtenidos durante la ejecución de las pruebas de estrés para la Aplicación Monolítica.

	CPU (%)	Disco lectura (bytes/s)	Disco escritura (bytes/s)	Memoria (%)	Memoria (MB)	Red recibe (bytes/s)	Red envía (bytes/s)
Min	24.55	0.00	0.00	3.74	813.12	0.00	0.00
Max	94.24	1096.00	31145984.00	37.21	5706.84	375661432.00	375661537.00
Promedio	56.98	2.74	688016.11	27.45	4301.83	90811265.87	179366474.93
Mediana	56.74	0.00	409600.00	27.94	4368.93	78459774.00	179846010.50
Desv. Est.	14.47	43.00	1832276.46	4.88	691.42	79231955.44	62141811.66

Tabla 26: Rendimiento de recursos – Aplicación Monolítica.
Elaborado por: Investigador.

Para revisar la tabla completa y los gráficos de la prueba de estrés ver el Anexo 5.

Datos recopilados con la herramienta NewRelic enfocándose al rendimiento de la Red, tal como se muestra en la Tabla 27, en la sección de Análisis se describirá a detalle la razón de este caso especial.

	Red recibidos (Kilobytes/s)	Red enviados (Kilobytes/s)
Promedio	385.4	82124.8

Tabla 27:Rendimiento de red – Logs NewRelic – Aplicación Monolítica.
Elaborado por: Investigador.

En la Tabla 28 se muestran los resultados del rendimiento de recursos obtenidos durante la ejecución de las pruebas de estrés para la Aplicación con Microservicios.

	CPU (%)	Disco lectura (bytes/s)	Disco escritura (bytes/s)	Memoria (%)	Memoria (MB)	Red recibe (bytes/s)	Red envía (bytes/s)
Min	32.58	0.00	0.00	10.88	4501.38	0.00	0.00
Max	89.34	12864.00	231849984.00	38.59	9013.43	1289436817.00	491857961.00
Promedio	69.51	56.60	2918823.14	33.63	8247.73	420819801.83	211516570.63
Mediana	70.45	0.00	2138112.00	33.71	8262.72	384435911.50	217117518.50
Desv. Est.	8.32	726.03	9524478.20	2.75	453.14	256367313.85	100257530.04

Tabla 28: Rendimiento de recursos – Aplicación de Microservicios.
Elaborado por: Investigador.

Para revisar la tabla completa y los gráficos de la prueba de estrés, ver el Anexo 6.

Datos recopilados con la herramienta NewRelic enfocándose al rendimiento de la Red, tal como se muestra en la Tabla 29 en la sección de Análisis se describirá a detalle la razón de este caso especial.

	Red recibidos (Kilobytes/s)	Red enviados (Kilobytes/s)
Promedio	55.7	72499.2

Tabla 29: Rendimiento de red – Logs NewRelic – Aplicación Monolítica.
Elaborado por: Investigador.

5.1.7 Análisis Comparativo

Para realizar el análisis comparativo, se presentan en esta sección los resultados que se muestran en las tablas de cada caso y seguido la interpretación escrita por el investigador.

En el Capítulo II, sección Rendimiento, se especifica el parámetro de medición para cada recurso tomado en cuenta en este experimento.

Es importante tener en cuenta que los resultados sobre los cuales se hace el análisis son los generados con la herramienta JMeter, a esto se suman las gráficas generadas con la herramienta NewRelic con el fin de corroborar los resultados y el análisis. Existe un trato especial con el análisis del rendimiento de la red, con la debida justificación se trae al análisis los valores generados con la herramienta NewRelic.

5.1.7.1 Rendimiento de la Aplicación.

El rendimiento de la Aplicación consiste en analizar los resultados de la prueba de estrés, enfocándose en el número de solicitudes generadas, el tiempo que toma procesarlas y si terminaron correctamente con respuesta desde el servidor.

Se toma como referencia los resultados de la Tabla 19 y la Tabla 25 (ver sección 5.1.6.1 y 5.1.6.2 respectivamente), se obtiene que:

- En el Caso 1 el total (273) de las solicitudes generadas han sido procesadas exitosamente en ambos escenarios. En el Caso 2 (1053 solicitudes), la Aplicación con Microservicios ha terminado con éxito todas las solicitudes, mientras que en el Escenario 1 con la Aplicación Monolítica 2 solicitudes terminaron con error.
- El tiempo, desde que inicia la ejecución de las pruebas hasta que termina de procesarse la última solicitud, muestra en ambos escenarios una diferencia considerable:

	App. Monolítica	App. Microservicios	Diferencia tiempo	Diferencia Porcentaje
Caso 1:	00:01:50	00:01:29	00:00:21	19.09 %
Caso 2:	00:14:17	00:12:08	00:02:09	15.05 %

Tabla 30: Comparación tiempo de duración de ejecución de las pruebas.
Elaborado por: Investigador.

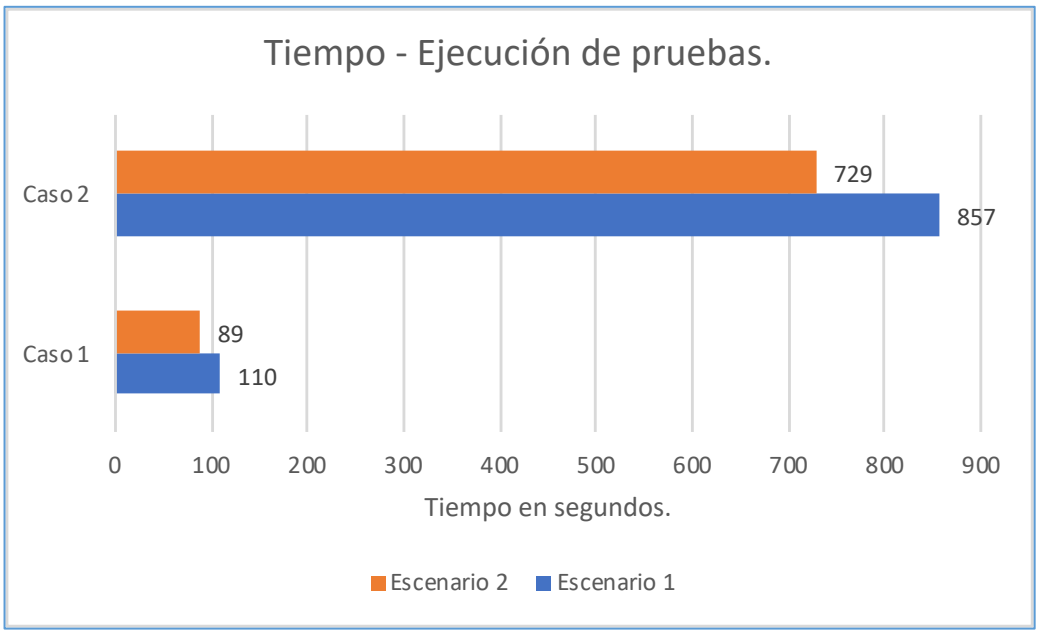


Gráfico 31: Tiempo que dura la ejecución de las pruebas.
Elaborado por: Investigador.

De la Tabla 30, se puede notar que con la Aplicación de Microservicios el tiempo que se demora en la ejecución de las pruebas se reduce, lo cual permite deducir que la eficiencia de los recursos ha mejorado, sin embargo, eso depende también del análisis de cada recurso.

- De igual forma, el número de solicitudes procesadas por segundo mejora usando la Aplicación de Microservicios.

	App. Monolítica	App. Microservicios	Diferencia	Diferencia Porcentaje
Caso 1:	2.5/s	3.1/s	0.6/s	24 %
Caso 2:	1.2/s	1.4/s	0.2/s	16.67 %

Tabla 31: Comparación número de solicitudes procesadas por segundo.
Elaborado por: Investigador.

De la Tabla 31, con el uso de la Aplicación con Microservicios podemos afirmar que se procesan una mayor cantidad de solicitudes por segundo, sin embargo, en este punto se desconoce aún si el consumo de recursos aumentó o disminuyó.

- En promedio, se reduce en un 17.07% el tiempo total de ejecución de las pruebas en el Escenario 1.

5.1.7.2 Rendimiento de CPU

El parámetro de medición es el Rendimiento. En los archivos logs se registra el porcentaje de uso de CPU en un momento exacto cuando un proceso se está ejecutando durante las pruebas de estrés.

Para realizar el análisis de este recurso, se hace referencia:

Caso 1: Tabla 20 y Tabla 22

Caso 2: Tabla 26 y Tabla 28

De cada tabla se analizan los valores de la columna CPU (%), el valor promedio.

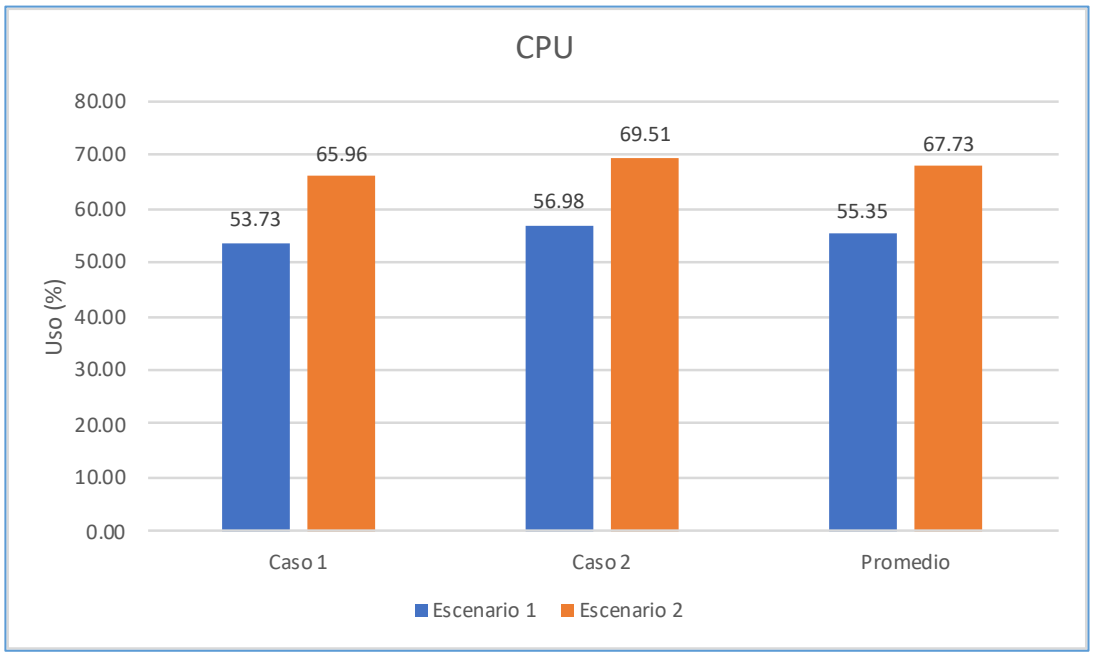


Gráfico 32: Rendimiento CPU - Análisis.
Elaborado por: Investigador.

Si bien en el rendimiento de la Aplicación se obtuvo ventaja con el Escenario 2, en este caso, según el Gráfico 32, hay un mayor consumo del recurso CPU en dicho escenario que en promedio consume el 67.73% contra el 55.35% del Escenario 1, dando una diferencia del 12.38% a favor de este último escenario, el cual genera menor consumo de este recurso.

5.1.7.3 Rendimiento de Memoria

El parámetro de medición es el Ancho de Banda. En el archivo logs, se registra el porcentaje y la cantidad en Megabytes de memoria utilizada en un momento justo durante la ejecución de las pruebas.

Para realizar el análisis de este recurso, se hace referencia del
Caso 1: Tabla 20 y Tabla 22
Caso 2: Tabla 26 y Tabla 28
De cada tabla se analizan los valores de la columna Memoria (%), el valor promedio.

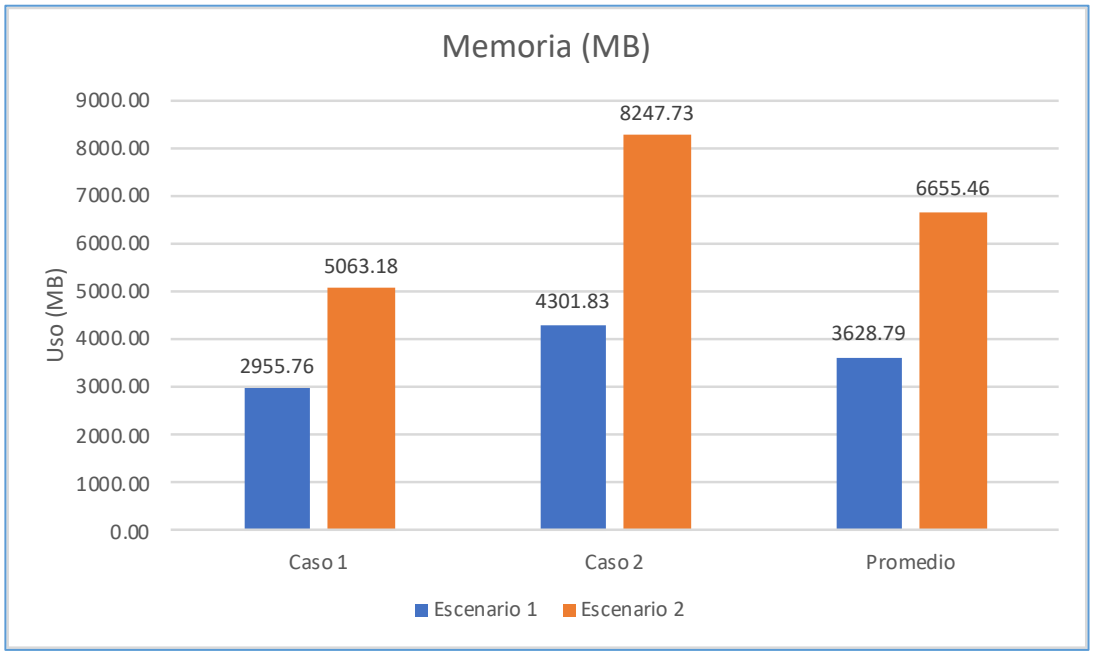


Gráfico 33: Rendimiento de Memoria, uso en Megabytes – Análisis.
Elaborado por: Investigador.

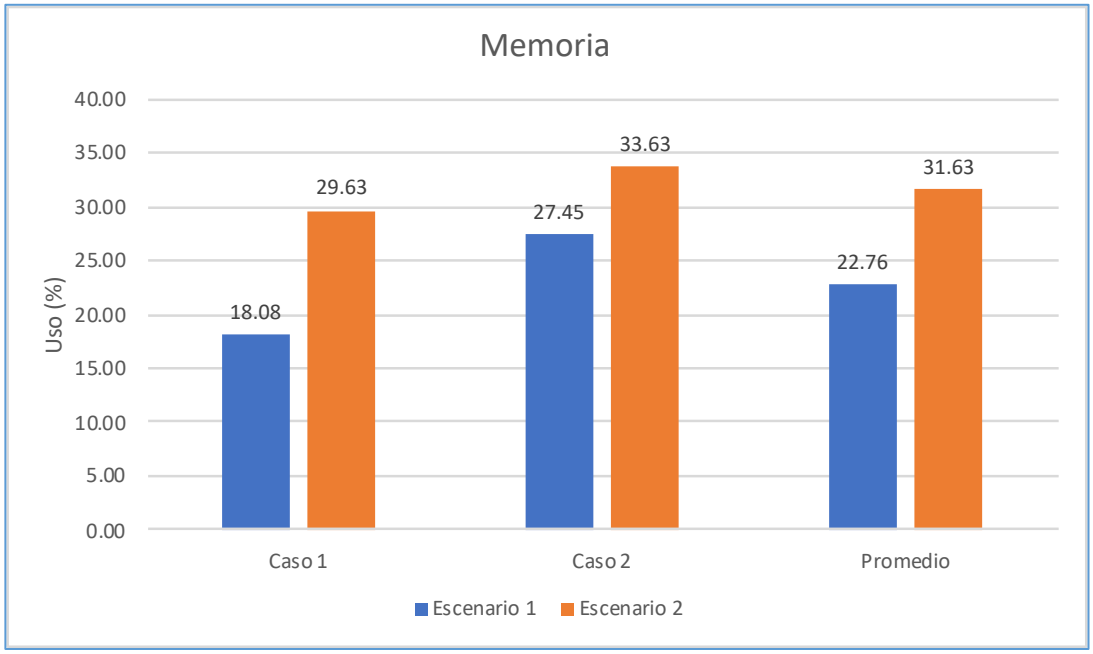


Gráfico 34: Rendimiento Memoria, uso en porcentaje – Análisis.
Elaborado por: Investigador.

Al igual que en el rendimiento del recurso CPU, en este caso con el recurso Memoria se puede ver que en promedio hay un mayor consumo, está el 31.63% del Escenario 2 frente al 22.76% del Escenario 1, con una diferencia del 8.87% a favor de este último escenario.

5.1.7.4 Rendimiento de Disco

El parámetro de medición es el Rendimiento. Para lograr interpretar la información en logs, se hace referencia a la velocidad en promedio que usa para realizar la lectura/escritura en disco medido en Bytes por Segundo.

Para realizar el análisis de este recurso, se hace referencia:

Caso 1: Tabla 20 y Tabla 22

Caso 2: Tabla 26 y Tabla 28

De cada tabla se analizan los valores de la columna disco lectura (bytes/s) y disco escritura (bytes/s), el valor promedio.

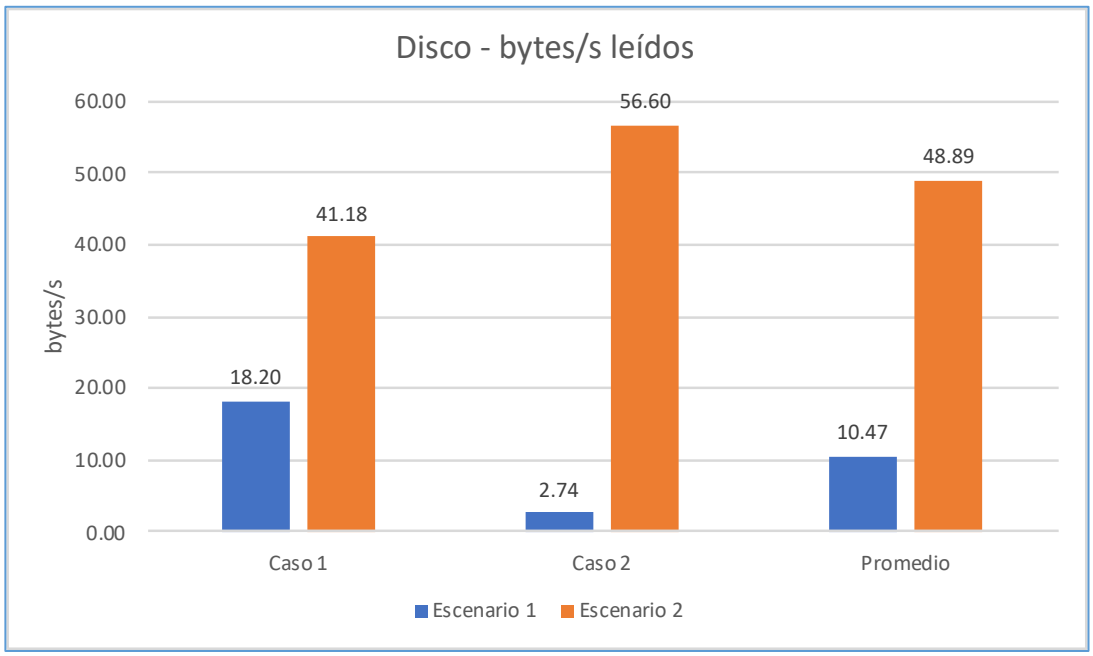


Gráfico 35: Rendimiento de lectura de disco – Análisis.
Elaborado por: Investigador.

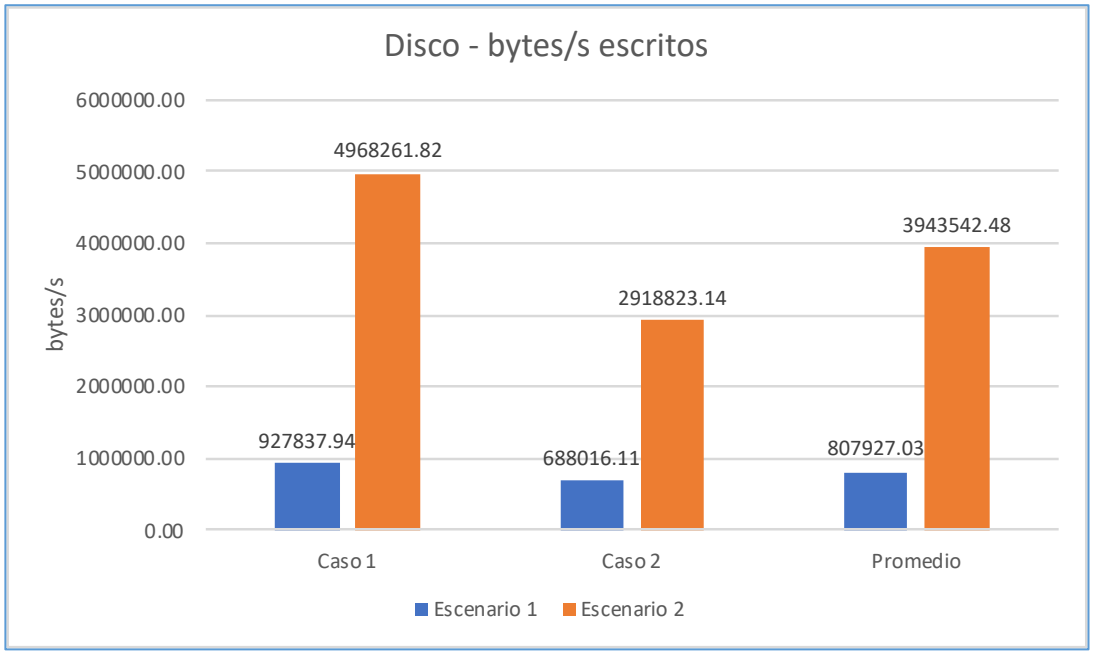


Gráfico 36: Rendimiento de escritura de disco – Análisis.
Elaborado por: Investigador.

Para el Gráfico 35 y Gráfico 36, en ambos casos, el Ancho de Banda es mayor en el Escenario 2, lo cual implica mayor eficiencia. Para este caso hay una diferencia de 78.5% en lectura de disco y 79.2% en escritura de disco. En conjunto y para fines de la investigación da un resultado final de 78.85% a favor del Escenario 2.

Para tener una mejor idea en otra unidad de medida, en promedio el Escenario 1 hace escritura de disco a una velocidad de 788.11 Kilobytes por segundo y en el Escenario 2 la cantidad de datos es de 3.76 Megabytes por segundo.

5.1.7.5 Rendimiento de Red

El parámetro de medición es el Ancho de banda. En el archivo de registro de logs generado por JMeter, se almacena la cantidad de Bytes transmitidos por segundo en un momento durante la ejecución de las pruebas de estrés, el promedio de esos valores son los mostrados en las siguientes gráficas.

Para realizar el análisis de este recurso, se hace referencia:

Caso 1: Tabla 20 y Tabla 22

Caso 2: Tabla 26 y Tabla 28

De cada tabla se analizan los valores de la columna Red recibidos (bytes/s) y Red enviados (bytes/s), el valor promedio.

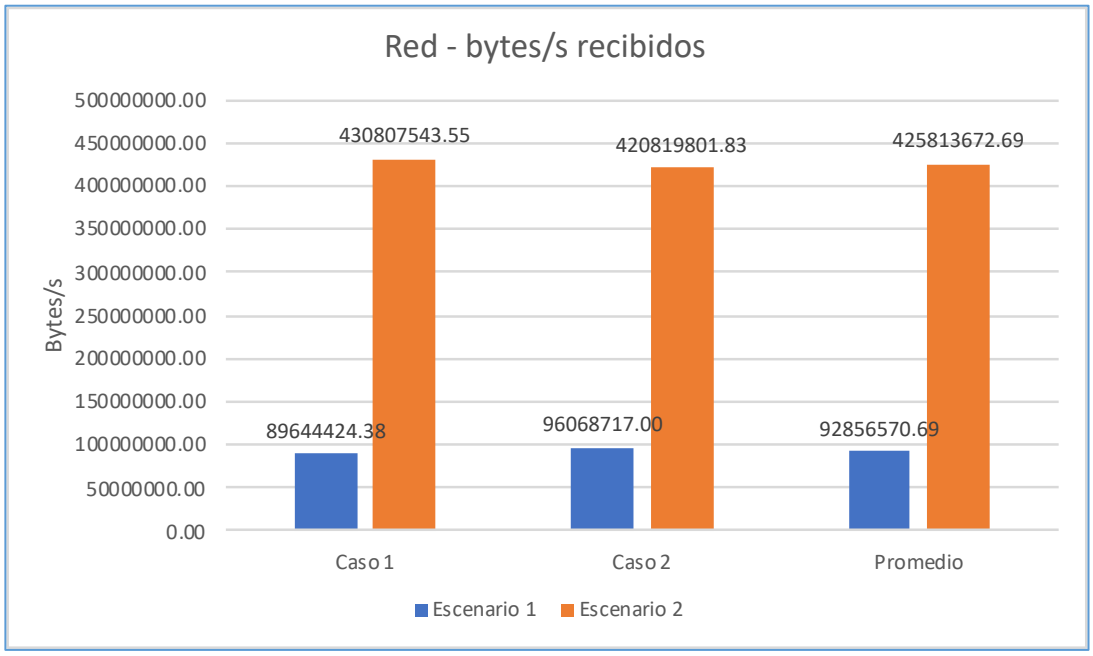


Gráfico 37: Rendimiento de Red bytes/s recibidos – Análisis.
Elaborado por: Investigador.

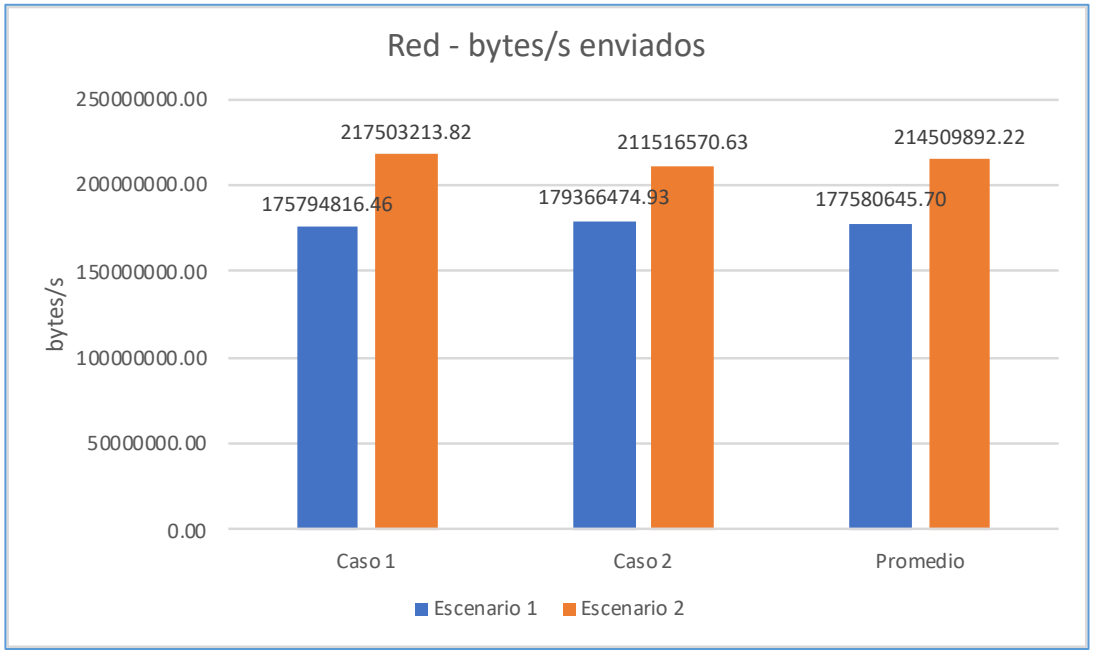


Gráfico 38: Rendimiento de Red bytes/s enviados – Análisis.
Elaborado por: Investigador.

Para este caso en particular, existe una interpretación errónea en las gráficas anteriores (Gráfico 37 y Gráfico 38). Esto se debe a que en el Escenario 1, los Servicios Agentes que recolectan la información del rendimiento de los recursos, están instalados en la Máquina Virtual (KVM) y la información de red envío/recepción es interpretada correctamente, pues el agente captura la información que entra y sale por la red de ese Host virtual tal como se muestra en el Gráfico 29 (ver sección 5.1.5.1).

Por otro lado, en el Gráfico 30 (ver sección 5.1.5.2) se observa que los Agentes, tanto JMeter y NewRelic, se ejecutan en el Host real y no dentro de Docker como se podría esperar. Ahora, volviendo a la documentación (ver sección 2.7.2), a Docker y sus Contenedores no se los considera estrictamente como Máquinas Virtuales y funcionan como un proceso que es parte del Host real y consume los mismos recursos. Además, no hay forma (aún) de instalar dichos Agentes como parte de Docker o instalar los agentes dentro de cada Contenedor pues sería una mala práctica. Por esta razón los agentes van a nivel del Host principal.

Entonces, cuando JMeter inicia la ejecución de las pruebas de estrés, la interpretación que estos Agentes le dan al tráfico de red es totalmente contraria (aunque sigue siendo válida para el análisis), es decir, lo que entra a la red privada generada por Docker se lo

interpreta como salida de datos en la red del Host real, y viceversa con lo que sale de la red de Docker hacia el Host real. El driver que maneja el tráfico de la red privada de Docker aparece como uno más en la lista de drivers del Host real, por lo que JMeter registra un mismo log como entrada y salida alterando de esta forma los registros en logs y por ende afecta la interpretación de los resultados.

Por lo tanto, la información de red registrada en los archivos logs relacionadas a cada uno de los escenarios, no es lo suficientemente clara y debido a la gran cantidad de datos se opta por traer al análisis los valores registrados en el servidor de NewRelic. Cabe señalar que, NewRelic cuenta con soporte (configuración por defecto para Contenedores) para recolectar información de rendimiento de recursos de Docker. Una limitante es que presenta valores ya analizados e interpretados por este mismo servicio, lo cual no permite tener el mismo detalle de información como los obtenidos con los registros de JMeter.

Ahora bien, tomando los datos de las tablas: Tabla 21, Tabla 23, Tabla 27 y Tabla 29, se obtienen las siguientes gráficas.

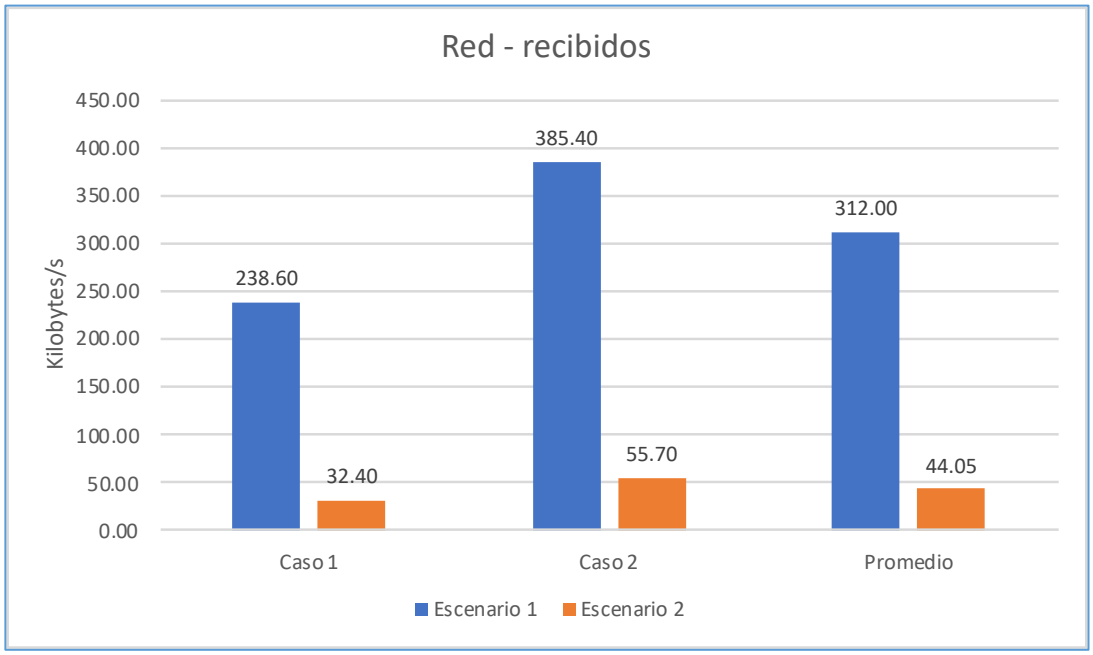


Gráfico 39: Rendimiento de red, kilobytes/s recibidos – Análisis.
Elaborado por: Investigador.

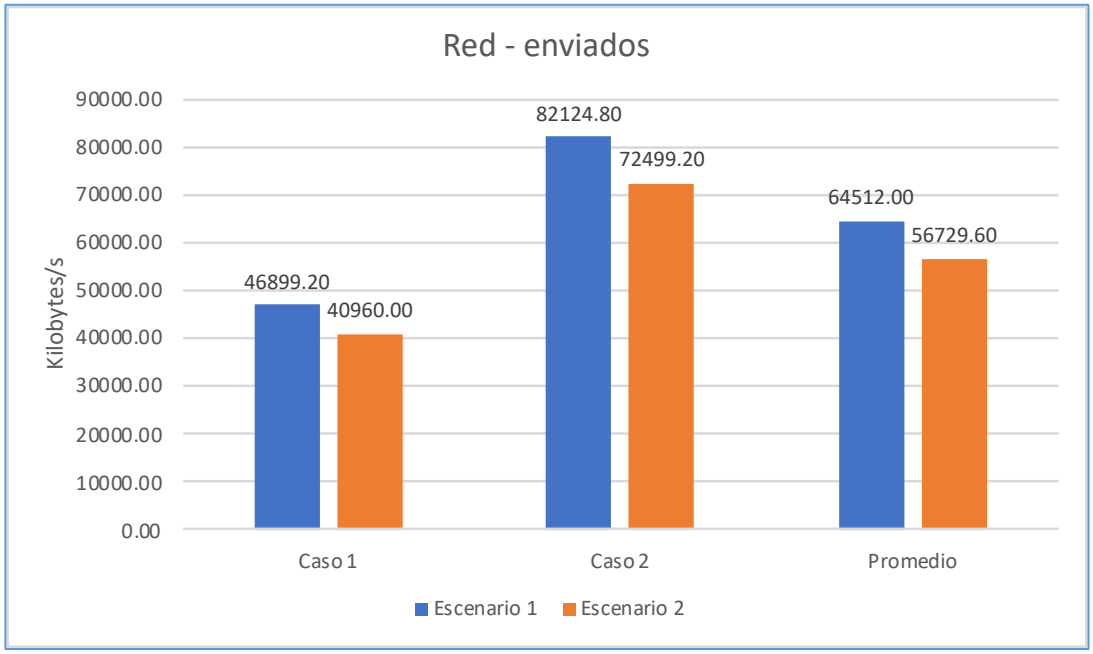


Gráfico 40: Rendimiento de red, kilobytes/s enviados – Análisis.
Elaborado por: Investigador.

En el caso del Escenario 2 la cantidad de Kilobytes por Segundo es reducida en comparación al Escenario 1. Esto es debido a que Docker genera una red privada para la ejecución de sus Contenedores el mismo que le sirve de medio de comunicación entre ellos, eso reduce la capacidad de Ancho de Banda tal como se refleja en las gráficas.

En el Gráfico 39, se puede interpretar que en promedio el Ancho de Banda de una KVM supera a los Contenedores en un 85%, y en el Gráfico 40, los resultados mejoran teniendo una diferencia del 12.06%. Para fines del experimento, se obtiene el promedio entre estos últimos valores, dando como resultado un 48.5% de diferencia, donde el Escenario 1 supera en rendimiento al Escenario 2.

5.1.7.6 Análisis General

El análisis de la investigación se basa en los resultados obtenidos para cada recurso de Hardware y adicional el rendimiento de la Aplicación.

En resumen, los resultados muestran la siguiente tendencia:

Recurso	A favor de	Diferencia
Aplicación	Escenario 2	17.07%
Disco	Escenario 2	78.85%
	Suma:	95.92%
	Promedio:	47.96%
CPU	Escenario 1	12.38%
Memoria	Escenario 1	8.87%
Red	Escenario 1	48.5%
	Suma:	69.75%
	Promedio:	23.25%

Tabla 32: Resumen de los resultados.
Elaborado por: Investigador.

El análisis del rendimiento de la Aplicación, coincide con el resultado de otro experimento científico publicado en la IEEE, donde sus autores (Villamizar et al., 2015) dan el siguiente análisis, los Microservicios permiten crear instancias más granulares - refiriéndose a los servicios – lo cual ayuda a la reducción de tiempo y costo en infraestructura (Hardware), esto es considerando que los recursos están disponibles a usarse al 100%. Al igual que los resultados arrojados en este experimento, coinciden en que el uso de Microservicios mejora tiempos de respuesta en un 17% y por consiguiente se da mayor validez al experimento realizado en esta investigación.

Revisando la Tabla 32, en relación con el tiempo de respuesta y el consumo de recursos se presentan 2 nuevos escenarios válidos, para cada caso, un experto en TI o un DevOps es quien toma la decisión de cual podría ser conveniente, pues dependen de los requerimientos y recursos disponibles, estos son:

a) **Escenario A: a menor tiempo de respuesta es mayor el consumo de recursos**, Esta situación en particular se da en ambientes donde no hay mucha compartición de recursos. Es decir, donde la Aplicación puede hacer uso completo de los mismos, lo cual optimiza tiempos de respuesta y procesamiento. Es conveniente

que suceda este caso cuando el objetivo del aplicativo sea procesar el mayor número de peticiones en el menor tiempo posible. Los Microservicios y Contenedores apuntan a sistemas de gran escala donde la asignación de recursos muchas veces es ilimitada y se pretende que los tiempos de respuesta sean lo más cortos posible, por lo tanto, este caso se acopla perfectamente al Escenario 2 del experimento.

b) **Escenario B: a mayor tiempo de respuesta es menor el consumo de recursos,** Casos como este se dan cuando el aplicativo no requiere muchos recursos de Hardware o simplemente el número de peticiones a procesar es reducido. Los sistemas con estas características no necesariamente exigen de tecnología moderna por lo que el Escenario 1 con Arquitectura Monolítica se acopla perfectamente.

A criterio del investigador, en la actualidad la mayoría de los sistemas informáticos tienden a mejorar la velocidad de procesamiento de información con el objetivo de mejorar tiempos de respuesta al punto de que todo parezca ser en tiempo real. Dicho eso, se hace énfasis a que el presente experimento usa tecnología moderna y por consiguiente se enfoca en sistemas informáticos que tiendan a ser de gran escala, y finalmente se coincide con el primer escenario de este análisis (Escenario A), donde se da mayor importancia a la mejora de tiempos de respuesta asumiendo que los recursos son suficientes o ilimitados. El rendimiento no solamente se enfoca en consumir en menor porcentaje un recurso, sino también, en optimizar tiempos aprovechando las bondades del Hardware haciendo que funcione en su mejor nivel.

Para el rendimiento del disco, en el Caso 1, el Escenario 2 presentan pocas lecturas según JMeter, esto se debe a que las Bases de Datos procesan las datos que se almacenan optimizándolos para que ocupen menos espacio en disco, sin embargo, en el Caso 2 se puede notar una mayor lectura de disco, lo cual es un justificativo del por qué la CPU y Memoria tienden a consumirse en mayor porcentaje, es decir, en este caso la lectura de disco presenta mejor desempeño permitiendo hacer lectura de un mayor número de Bytes por segundo lo cual requiere mayor uso de CPU y Memoria y por ende aumenta la exigencia de estos últimos recursos.

Lo anterior coincide con lo mencionado en el artículo publicado por IBM, donde sus autores (Felter et al., 2015) mencionan, que el hecho de que la cantidad de Bytes/s manejados en la lectura/escritura de disco sea menor en el Escenario 1 se debe a que cada operación debe pasar a través de QEMU que es el software especializado para la virtualización de la KVM. Por otro lado, la cantidad de Bytes/s en lectura/escritura de disco aumenta en el Escenario 2, esto se debe a que los contenedores se comunican más directamente con los recursos del Host, el disco en este caso.

Finalmente, la investigación presenta una propuesta de hacer uso de una tecnología y arquitectura modernas. Esto va de la mano con el enfoque que tiene hacia proyectos de mediana y gran escala donde la optimización tanto de Hardware y Software juegan un papel importante. A criterio del investigador, en consecuencia, de la aparición de los dos escenarios en el análisis y basado en la hipótesis planteada en esta investigación, resuelve lo siguiente:

1. Basándose en el Escenario A del análisis, lo propuesto en la hipótesis de mejorar el rendimiento en un 15% se cumple e incluso sobrepasa con un 2% más de lo esperado. Si bien, se detalla que se consumen más recursos en menos tiempo, eso también significa mejor aprovechamiento de las características de los recursos.

Tomando el Escenario B del análisis, e interpretando los valores de la

2. Tabla 32, vemos que en el Escenario 1 del experimento, hay menos consumo de recursos, pero el tiempo de respuesta es mayor, sin embargo, el recurso Disco sobresale en el Escenario 2 lo cual significa ventaja para este último escenario del experimento. Es decir, que haciendo un comparativa de los recursos que favorecen al Escenario 1 con los que favorecen al Escenario 2 se obtiene que, se cumple con la hipótesis, y en este caso sobrepasa en un 9.71% más de lo esperado.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

- El análisis comparativo de rendimiento entre la Arquitectura de Microservicios basada en Contenedores y la Arquitectura Monolítica, cumple con su objetivo y da validez a la hipótesis planteada incluso con creses.
- Una Aplicación con Arquitectura de Microservicios basada en Contenedores, en general, mejora el rendimiento de los recursos en comparación con una Aplicación con Arquitectura Monolítica que se ejecuta en una KVM.
- Se logró cumplir con lo propuesto en los objetivos específicos, esto en cuanto a recopilación de bibliografías, experimentación y presentación de los resultados.
- Los resultados y el análisis obtenidos en esta investigación fueron corroborados con investigaciones científicas ligadas al tema, esto trasciende al punto de que tanto el desarrollo del experimento y los resultados obtenidos en esta investigación tengan mayor credibilidad y se la considere una fuente para futuras investigaciones.

Recomendaciones:

- Esta investigación da lugar a nuevos temas de investigación, considerando que las tecnologías propuestas han mostrado mejoras durante el último año, es decir, las mejoras en las funcionalidades deben ser consideradas como una oportunidad para investigar y proponer soluciones que ayuden a optimizar aún más los sistemas informáticos.
- Los experimentos realizados en esta investigación pueden llevarse a cabo en servidores con características de alto nivel, más específicamente en servidores en la nube, de donde se puedan captar nuevos resultados y presentar propuestas que validen esta investigación.
- Los Contenedores y Microservicios, no solamente se enfocan a la producción de Software, sino, también a temas de nivel avanzado como: Inteligencia Artificial o Inteligencia de Negocios, donde aún existen temas abiertos a la investigación.

REFERENCIAS BIBILIOGRÁFICAS

Literatura Citada

1471-2000 IEEE. (2000). *1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE / Institute of Electrical and Electronics Engineers Incorporated.

Adufu, T., Choi, J., & Kim, Y. (2015). Is container-based technology a winner for high performance scientific applications? En *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 507-510). https://doi.org/10.1109/APNOMS.2015.7275379

Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., & Steinder, M. (2015). Performance Evaluation of Microservices Architectures using Containers. *arXiv:1511.02043 [cs]*, 27-34. https://doi.org/10.1109/NCA.2015.49

Apache. (2018, marzo 31). Apache JMeter - Apache JMeter™. Recuperado 31 de marzo de 2018, de https://jmeter.apache.org/

Asale, R. (2014). Rendimiento. Recuperado 8 de marzo de 2017, de http://dle.rae.es/?id=VwxnN6O

Bartholomew, D. (2006). Qemu: a multihost, multitarget emulator. *Linux Journal*, *2006*(145), 3.

Basar Bener, A., Morisio, M., Miranskyy, A., & Akinii Kocak, S. (2014, octubre). TI Verde y Software Verde - IEEECS. Recuperado 14 de febrero de 2017, de https://www.computer.org/web/computingnow/archive/october2014-spanish

Bashari Rad, B., Bhatti, H., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, *173*, 8.

Bernal Torres, C. A. (2010). *Metodología de la investigación: administración, economía, humanidades y ciencias sociales*. México: Pearson Educación, Prentice Hall.

Berrangé, D. (2017). Virtual Machine Manager Home. Recuperado 11 de abril de 2018, de https://virt-manager.org/

Bligh, A. (2014, febrero 5). What Does a Hypervisor Do? |. Recuperado 5 de noviembre de 2017, de https://www.flexiant.com/2014/02/05/what-does-a-hypervisor-do/

Campbell, S., & Jeronimo, M. (2006). An introduction to virtualization. *Published in "Applied Virtualization", Intel*.

Dhara, K., Dharmala, M., & Sharma, C. (2015). A Survey Paper on Service Oriented Architecture Approach and Modern Web Services. *All Capstone Projects*. Recuperado de http://opus.govst.edu/capstones/157

Docker. (2017). Docker. Recuperado 19 de septiembre de 2017, de https://www.docker.com/

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2016). Microservices: yesterday, today, and tomorrow. *arXiv:1606.04036 [cs]*. Recuperado de http://arxiv.org/abs/1606.04036

Erl, T. F. (2015). *US9213526 B1*. Recuperado de http://www.google.com/patents/US9213526

Faura, O. (2014, septiembre 26). Integración continua (CI), Entrega continua (CD) y Despliegue Continuo (CD). Recuperado 20 de noviembre de 2017, de https://devopsti.wordpress.com/2014/09/26/integracion-continua-ci-entrega-continua-cd-y-despliegue-continuo-cd/

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015). An updated performance comparison of virtual machines and linux containers. En *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on* (pp. 171–172). IEEE. Recuperado de http://ieeexplore.ieee.org/abstract/document/7095802/

Flygare, R., & Holmqvist, A. (2017). *Performance characteristics between monolithic and microservice-based systems*. Recuperado de http://www.diva-portal.org/smash/record.jsf?pid=diva2:1119785

Foundation, N. js. (2018, 30). Node.js. Recuperado 31 de marzo de 2018, de https://nodejs.org/en/

Fowler, M. (2006, mayo 1). Continuous Integration. Recuperado 20 de noviembre de 2017, de https://martinfowler.com/articles/continuousIntegration.html

Fowler, M. (2014, marzo 6). bliki: CircuitBreaker. Recuperado 17 de noviembre de 2017, de https://martinfowler.com/bliki/CircuitBreaker.html

Fowler, M., & Lewis, J. (2014, marzo 25). Microservices. Recuperado 16 de septiembre de 2017, de https://martinfowler.com/articles/microservices.html

Fussell, M. (2017, junio 2). Why a microservices approach to building applications? [Articulo]. Recuperado 10 de noviembre de 2017, de https://docs.microsoft.com/es-es/azure/service-fabric/service-fabric-overview-microservices

Gartner. (2017). IT Glossary: Network Virtualization. Recuperado 3 de noviembre de 2017, de

GIT. (2017). Git. Recuperado 31 de marzo de 2018, de https://git-scm.com/

Gutiérrez M., A. (2006). *Curso de Métodos de Investigación*. Quito. Recuperado de http://biblioteca.unach.edu.ec/opac_css/index.php?lvl=notice_display&id=9430

IBM. (2017, noviembre 9). IBM : ¿Qué es cloud computing? - España. Recuperado 24 de noviembre de 2017, de https://www.ibm.com/cloud-computing/es-es/learn-more/what-is-cloud-computing/

IEEE 610.12-1990. (1990). *610.12-1990 IEEE standard glossary of software engineering terminology*. New York: Institute of Electrical and Electronics Engineers.

Intel. (2014). Linux* Containers Streamline, Complement Hypervisor-Based VMs.

Java. (2017). Conozca más sobre la tecnología Java. Recuperado 31 de marzo de 2018, de https://java.com/es/about/

JMeter. (2013, julio 12). Documentation :: JMeter-Plugins.org. Recuperado 31 de marzo de 2018, de https://jmeter-plugins.org/wiki/PerfMonAgent/

Jurenka, V. (2015). *Virtualization using Docker Platform* (Thesis). Masaryk University, Brno, Czech Republic. Recuperado de http://is.muni.cz/th/430604/fi_m/thesis.pdf

Kerrisk, M. (2013, enero 4). Namespaces in operation, part 1: namespaces overview. Recuperado 28 de noviembre de 2017, de https://lwn.net/Articles/531114/

Khazaei, H., Barna, C., Beigi-Mohammadi, N., & Litoiu, M. (2016). Efficiency Analysis of Provisioning Microservices. En *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 261-268). https://doi.org/10.1109/CloudCom.2016.0051

Kratzke, N. (2015). About Microservices, Containers and their Underestimated Impact on Network Performance. En *ResearchGate*. Lubeck, Germany. http://dx.doi.org/10.13140/RG.2.1.2039.3046

Lee, K.-H., Cram, R., & Mukundan, A. (2002). *US6336137 B1*. Recuperado de http://www.google.com/patents/US6336137

Leiva Zea, F. (2010). *Nociones de metodología de investigación cientifica*. Recuperado de http://biblioteca.unach.edu.ec/opac_css/index.php?lvl=notice_display&id=5353

Meier, S., Virun, B., Blumert, J., & Jones, M. T. (2008). *IBM Systems Virtualization: Servers, Storage, and Software*. IBM Corp. April. Recuperado de http://www.redbooks.ibm.com/redpapers/pdfs/redp4396.pdf

MongoDB. (2018, marzo 31). What Is MongoDB? Recuperado 31 de marzo de 2018, de https://www.mongodb.com/what-is-mongodb

Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Inc.

NewRelic. (2018, marzo 31). New Relic: Digital Intelligence SaaS. Recuperado 31 de marzo de 2018, de https://newrelic.com/about

Nginx. (2017). What Is Load Balancing? How Load Balancers Work [Resource]. Recuperado 17 de noviembre de 2017, de https://www.nginx.com/resources/glossary/load-balancing/

Nielsen, C. D. (2015). *Investigate availability and maintainability within a microservice architecture*. Aarhus University, Nordre Ringgade 1, 8000 Aarhus C, Denmark. Recuperado de http://cs.au.dk/fileadmin/site_files/cs/AA_pdf/ClausDNielsen_rapport.pdf

Oracle. (2018). MySQL. Recuperado 31 de marzo de 2018, de https://www.mysql.com/

Peláez, J. (2017). Arquitectura basada en Componentes [Blog]. Recuperado 10 de noviembre de 2017, de https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/

Portnoy, M. (2012). *Virtualization Essentials* (1st ed.). Alameda, CA, USA: SYBEX Inc.

Prashant, D. (2016). A Survey of Performance Comparison between Virtual Machines and Containers, *4*(7). Recuperado de http://www.academia.edu/27776697/A_Survey_of_Performance_Comparison_between_Virtual_Machines_and_Containers

QEMU. (2017). QEMU. Recuperado 21 de noviembre de 2017, de https://www.qemu.org/

Richardson, C., & Smith, F. (2016). *Microservices From Design to Deployment*. Nginx. Recuperado de https://www.nginx.com/resources/library/designing-deploying-microservices/

Salah, T., Zemerly, M. J., Yeun, C. Y., Al-Qutayri, M., & Al-Hammadi, Y. (2016). The evolution of distributed systems towards microservices architecture. En *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)* (pp. 318-325). https://doi.org/10.1109/ICITST.2016.7856721

Salesforce. (2017). ¿Qué es Cloud Computing? Recuperado 24 de noviembre de 2017, de https://www.salesforce.com/mx/cloud-computing/

Sayed, M., & Khan, Z. (2017). *From monolith to containers: How Verizon containerized legacy applications on OpenShift*. Presentado en Red Hat Summit, Estados Unidos. Recuperado de https://www.redhat.com/files/summit/session-assets/2017/S101869-Monolith-to-Containers-Verizon.pdf

Scott, J. (2017). *A practical Guide to Microservices and Containers: Mastering the Cloud, Data and Digital Transformation*. Recuperado de https://dzone.com/storage/assets/7208502-book-microservices-and-containers.pdf

Sharma, S. (2014). *DevOps for dummies*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Siddiqui, B., & Mahdy, A. (2015). Storage Virtualization: Towards an Efficient and Scalable Framework, *7*(1), 12.

Sublime. (2018). Sublime Text - A sophisticated text editor for code, markup and prose. Recuperado 30 de marzo de 2018, de https://www.sublimetext.com/

Sundman, Y. (2013, febrero 5). Crisp's Blog » Continuous Delivery vs Continuous Deployment. Recuperado 20 de noviembre de 2017, de

http://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment

Turnbull, J. (2016). *The docker book*. Lulu. com. Recuperado de https://books.google.com.ec/books?hl=en&lr=&id=CtMEBwAAQBAJ&oi=fnd&pg=PP2&dq=the+docker+book&ots=Pptp4UUkwV&sig=byEj2-tCmOLItFgBn-2a7hMjvb0

Ubuntu. (2018, marzo 30). The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu. Recuperado 30 de marzo de 2018, de https://www.ubuntu.com/

UDSEnterprise. (2017). Application virtualization - Virtual Desktops. Recuperado 4 de noviembre de 2017, de https://www.udsenterprise.com/en/solutions/application-virtualization/

Vaduva, A. (2015). Virtualization.

Vaucher, S. (2015). Comparing Virtual Machines and Linux Containers. Recuperado de https://zenodo.org/record/47644/files/Comparing_Virtual_Machines_and_Linux_Containers_-_Final.pdf

Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. En *2015 10th Computing Colombian Conference (10CCC)* (pp. 583-590). https://doi.org/10.1109/ColumbianCC.2015.7333476

VMWare. (2017). Virtualización de VMware. Recuperado 7 de marzo de 2017, de http://www.vmware.com/latam/solutions/virtualization.html

Xalambrí, D. (2017). Deploy y orquestación de microservicios con now.sh. Recuperado 17 de noviembre de 2017, de https://platzi.com/blog/deploy-microservicios/

ANEXOS

Anexo 1: Script – Prueba de estrés generado y ejecutado con JMeter.

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="3.2" jmeter="3.3 r1808647">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Plan test monolith"
enabled="true">
      <stringProp name="TestPlan.comments"></stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">true</boolProp>
      <elementProp name="TestPlan.user_defined_variables" elementType="Arguments"
guiclass="ArgumentsPanel" testclass="Arguments" testname="User Defined Variables"
enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Scenario -
Generate Data" enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller" elementType="LoopController"
guiclass="LoopControlPanel" testclass="LoopController" testname="Loop Controller"
enabled="true">
          <boolProp name="LoopController.continue_forever">false</boolProp>
          <stringProp name="LoopController.loops">1</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">1</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <longProp name="ThreadGroup.start_time">1516056498000</longProp>
        <longProp name="ThreadGroup.end_time">1516056498000</longProp>
        <boolProp name="ThreadGroup.scheduler">false</boolProp>
        <stringProp name="ThreadGroup.duration"></stringProp>
        <stringProp name="ThreadGroup.delay"></stringProp>
      </ThreadGroup>
      <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="HTTP Request - Users" enabled="true">
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="Variables pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/generate/users</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy"
testname="HTTP Request - Threads" enabled="true">
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="Variables pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
```

```
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/generate/threads</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
        <HTTPSamplerProxy      guiclass="HttpTestSampleGui"      testclass="HTTPSamplerProxy"
testname="HTTP Request - Posts" enabled="true">
          <elementProp          name="HTTPsampler.Arguments"          elementType="Arguments"
guiclass="HTTPArgumentsPanel"    testclass="Arguments"    testname="Variables    pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/generate/posts</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
      </hashTree>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Scenario - Get
Data" enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp      name="ThreadGroup.main_controller"      elementType="LoopController"
guiclass="LoopControlPanel"     testclass="LoopController"     testname="Loop     Controller"
enabled="true">
          <boolProp name="LoopController.continue_forever">false</boolProp>
          <stringProp name="LoopController.loops">30</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">3</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <longProp name="ThreadGroup.start_time">1373789594000</longProp>
        <longProp name="ThreadGroup.end_time">1373789594000</longProp>
        <boolProp name="ThreadGroup.scheduler">false</boolProp>
        <stringProp name="ThreadGroup.duration">120</stringProp>
        <stringProp name="ThreadGroup.delay">1</stringProp>
        <stringProp name="TestPlan.comments">Virtual Users Running Scenario 1.
Make test last 1 minute (see Scheduler)</stringProp>
      </ThreadGroup>
      <hashTree>
        <HTTPSamplerProxy      guiclass="HttpTestSampleGui"      testclass="HTTPSamplerProxy"
testname="HTTP Request - Users" enabled="true">
          <elementProp          name="HTTPsampler.Arguments"          elementType="Arguments"
guiclass="HTTPArgumentsPanel"    testclass="Arguments"    testname="Variables    pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/users</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
```

```
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
        <HTTPSamplerProxy      guiclass="HttpTestSampleGui"      testclass="HTTPSamplerProxy"
testname="HTTP Request - Threads" enabled="true">
          <elementProp          name="HTTPsampler.Arguments"          elementType="Arguments"
guiclass="HTTPArgumentsPanel"    testclass="Arguments"    testname="Variables    pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/threads</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
        <HTTPSamplerProxy      guiclass="HttpTestSampleGui"      testclass="HTTPSamplerProxy"
testname="HTTP Request - Posts" enabled="true">
          <elementProp          name="HTTPsampler.Arguments"          elementType="Arguments"
guiclass="HTTPArgumentsPanel"    testclass="Arguments"    testname="Variables    pré-définies"
enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp name="HTTPSampler.domain">192.168.122.159</stringProp>
          <stringProp name="HTTPSampler.port">3000</stringProp>
          <stringProp name="HTTPSampler.protocol"></stringProp>
          <stringProp name="HTTPSampler.contentEncoding"></stringProp>
          <stringProp name="HTTPSampler.path">api/posts</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree/>
      </hashTree>
      <ResultCollector    guiclass="ViewResultsFullVisualizer"    testclass="ResultCollector"
testname="View Results Tree" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>false</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
```

```
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="TestPlan.comments">For scripting only</stringProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector  guiclass="TableVisualizer"  testclass="ResultCollector"  testname="View
Results in Table" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector      guiclass="RespTimeGraphVisualizer"      testclass="ResultCollector"
testname="Response Time Graph" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
```

```
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
        <stringProp name="RespTimeGraph.interval">1000</stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector guiclass="GraphVisualizer" testclass="ResultCollector" testname="Graph
Results" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Summary
Report" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
```

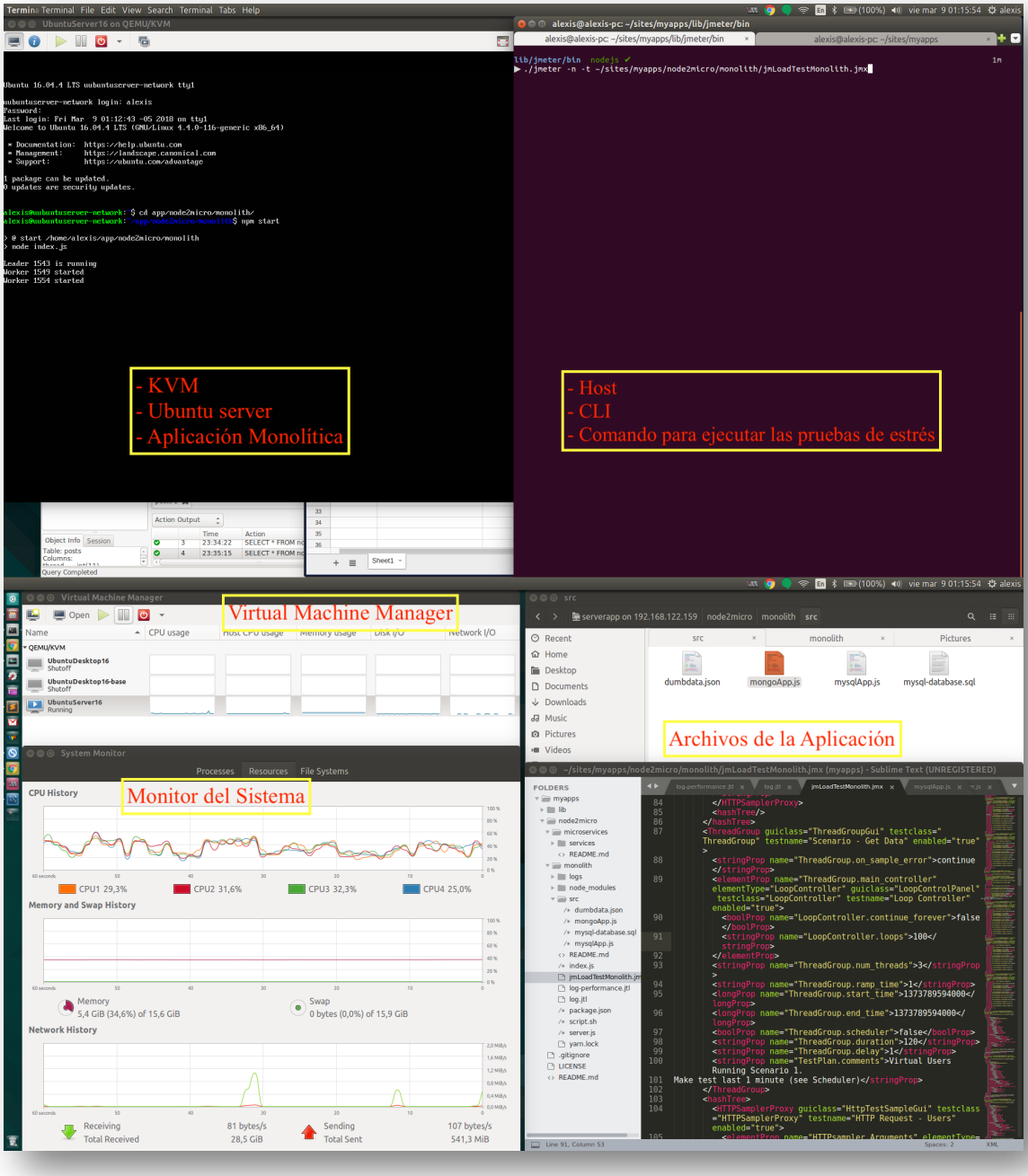
```
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
        <boolProp name="useGroupName">true</boolProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector          guiclass="StatVisualizer"          testclass="ResultCollector"
testname="Aggregate Report" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
      </ResultCollector>
      <hashTree/>
      <ResultCollector         guiclass="SimpleDataWriter"         testclass="ResultCollector"
testname="Simple Data Writer" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
```

```
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp
name="filename">/home/alexis/sites/myapps/node2micro/monolith/log.jtl</stringProp>
      </ResultCollector>
      <hashTree/>
      <kg.apc.jmeter.perfmon.PerfMonCollector  guiclass="kg.apc.jmeter.vizualizers.PerfMonGui"
testclass="kg.apc.jmeter.perfmon.PerfMonCollector" testname="Performance - Metrics Collector"
enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>
            <saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp         name="filename">/home/alexis/sites/myapps/node2micro/monolith/log-
performance.jtl</stringProp>
        <longProp name="interval_grouping">1000</longProp>
        <boolProp name="graph_aggregated">false</boolProp>
        <stringProp name="include_sample_labels"></stringProp>
        <stringProp name="exclude_sample_labels"></stringProp>
        <stringProp name="start_offset"></stringProp>
        <stringProp name="end_offset"></stringProp>
        <boolProp name="include_checkbox_state">false</boolProp>
        <boolProp name="exclude_checkbox_state">false</boolProp>
        <collectionProp name="metricConnections">
          <collectionProp name="672821944">
            <stringProp name="838760698">192.168.122.159</stringProp>
            <stringProp name="1600768">4444</stringProp>
            <stringProp name="66952">CPU</stringProp>
            <stringProp name="-1259908780">label=CPU (%):combined</stringProp>
```

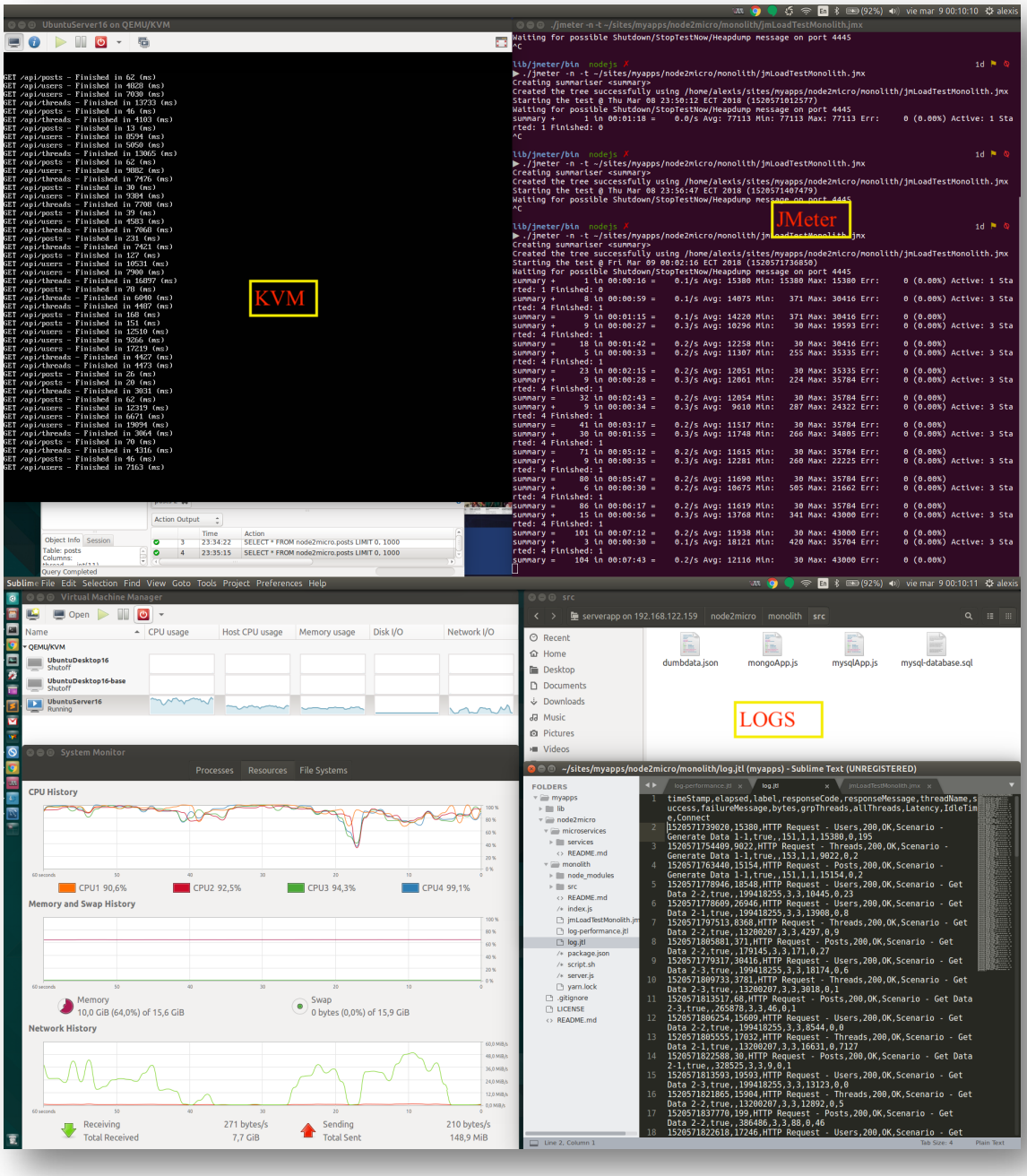
```
          </collectionProp>
          <collectionProp name="-251720848">
            <stringProp name="838760698">192.168.122.159</stringProp>
            <stringProp name="1600768">4444</stringProp>
            <stringProp name="-1993889503">Memory</stringProp>
            <stringProp name="-1030586553">label=Memory (%):usedperc</stringProp>
          </collectionProp>
          <collectionProp name="-408109218">
            <stringProp name="838760698">192.168.122.159</stringProp>
            <stringProp name="1600768">4444</stringProp>
            <stringProp name="-1993889503">Memory</stringProp>
            <stringProp name="1720442209">label=Memory (MB):unit=mb:used</stringProp>
          </collectionProp>
          <collectionProp name="569071277">
            <stringProp reference="../../collectionProp[3]/stringProp"/>
            <stringProp reference="../../collectionProp[3]/stringProp[2]"/>
            <stringProp name="2112896831">Disks I/O</stringProp>
            <stringProp            name="-1405501122">label=Disc            -            read
(bytes/s):unit=kb:readbytes</stringProp>
          </collectionProp>
          <collectionProp name="-1596493160">
            <stringProp reference="../../collectionProp[2]/stringProp"/>
            <stringProp reference="../../collectionProp[2]/stringProp[2]"/>
            <stringProp name="2112896831">Disks I/O</stringProp>
            <stringProp name="610484570">label=Disc - write (bytes/s):writebytes</stringProp>
          </collectionProp>
          <collectionProp name="-867629371">
            <stringProp reference="../../collectionProp[2]/stringProp"/>
            <stringProp reference="../../collectionProp[2]/stringProp[2]"/>
            <stringProp name="-274342153">Network I/O</stringProp>
            <stringProp          name="-1123791154">label=Network          -          receive
(bytes/s):bytesrecv</stringProp>
          </collectionProp>
          <collectionProp name="-282230401">
            <stringProp reference="../../collectionProp[2]/stringProp"/>
            <stringProp reference="../../collectionProp[2]/stringProp[2]"/>
            <stringProp name="-274342153">Network I/O</stringProp>
            <stringProp name="326926929">label=Network - sent (bytes/s):bytessent</stringProp>
          </collectionProp>
        </collectionProp>
      </kg.apc.jmeter.perfmon.PerfMonCollector>
      <hashTree/>
    </hashTree>
    <WorkBench     guiclass="WorkBenchGui"     testclass="WorkBench"     testname="WorkBench"
enabled="true">
      <boolProp name="WorkBench.save">true</boolProp>
    </WorkBench>
    <hashTree/>
  </hashTree>
</jmeterTestPlan>
```

Anexo 2: Capturas de pantalla del ambiente de experimento.

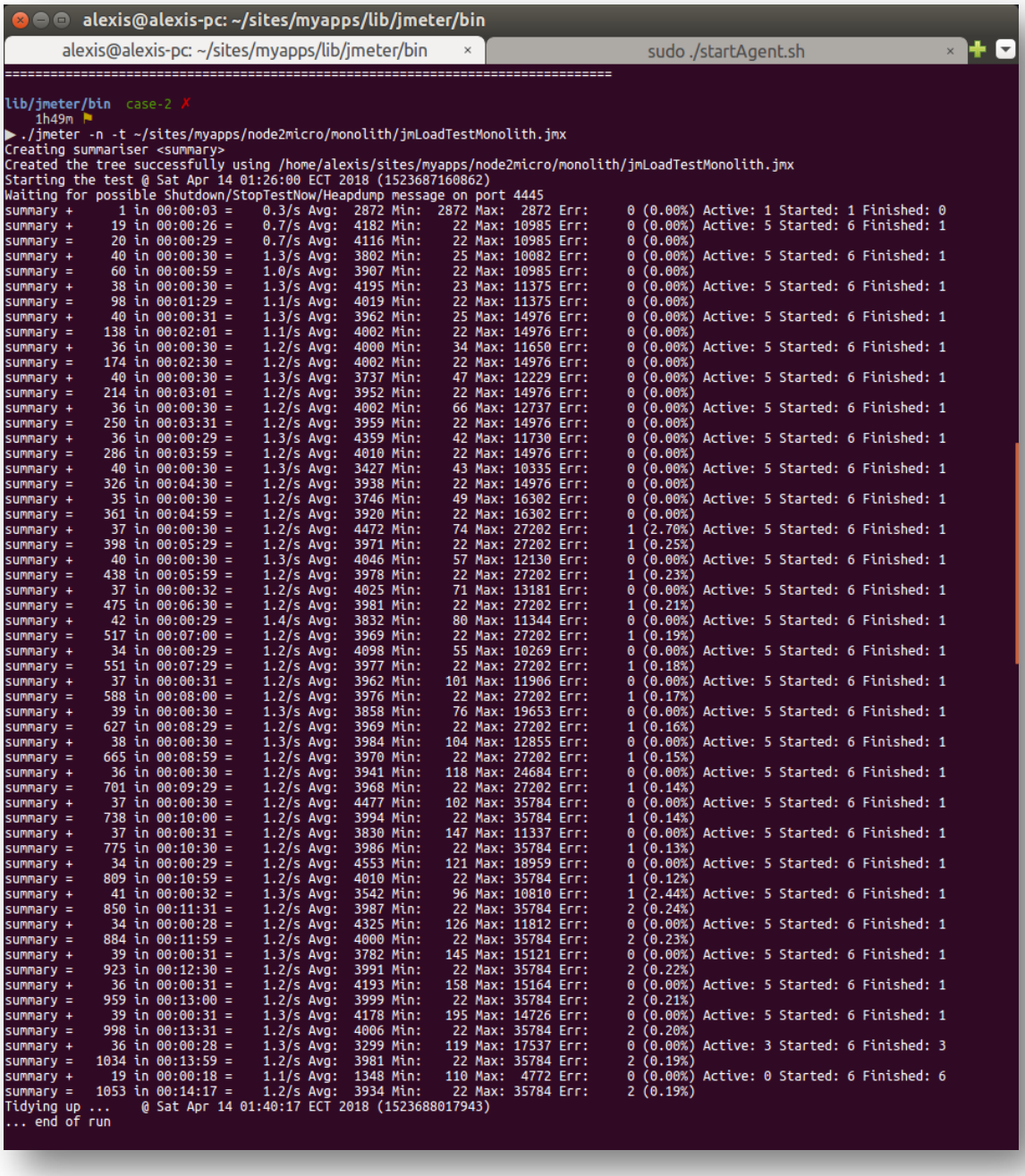
Ambiente para ejecutar la prueba de estrés. Antes de iniciar las pruebas.



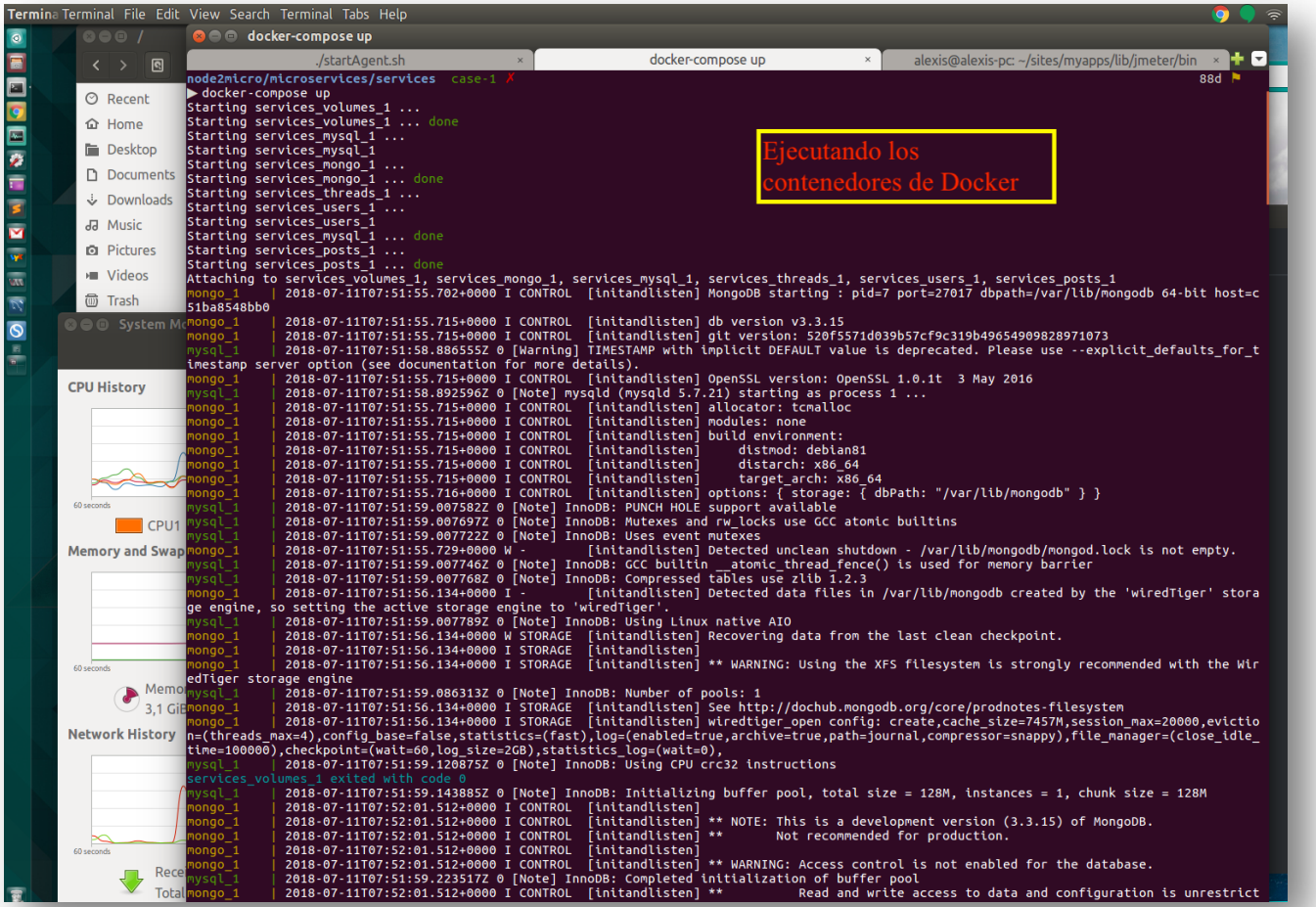
Ambiente para ejecutar la prueba de estrés. Después de terminar las pruebas.



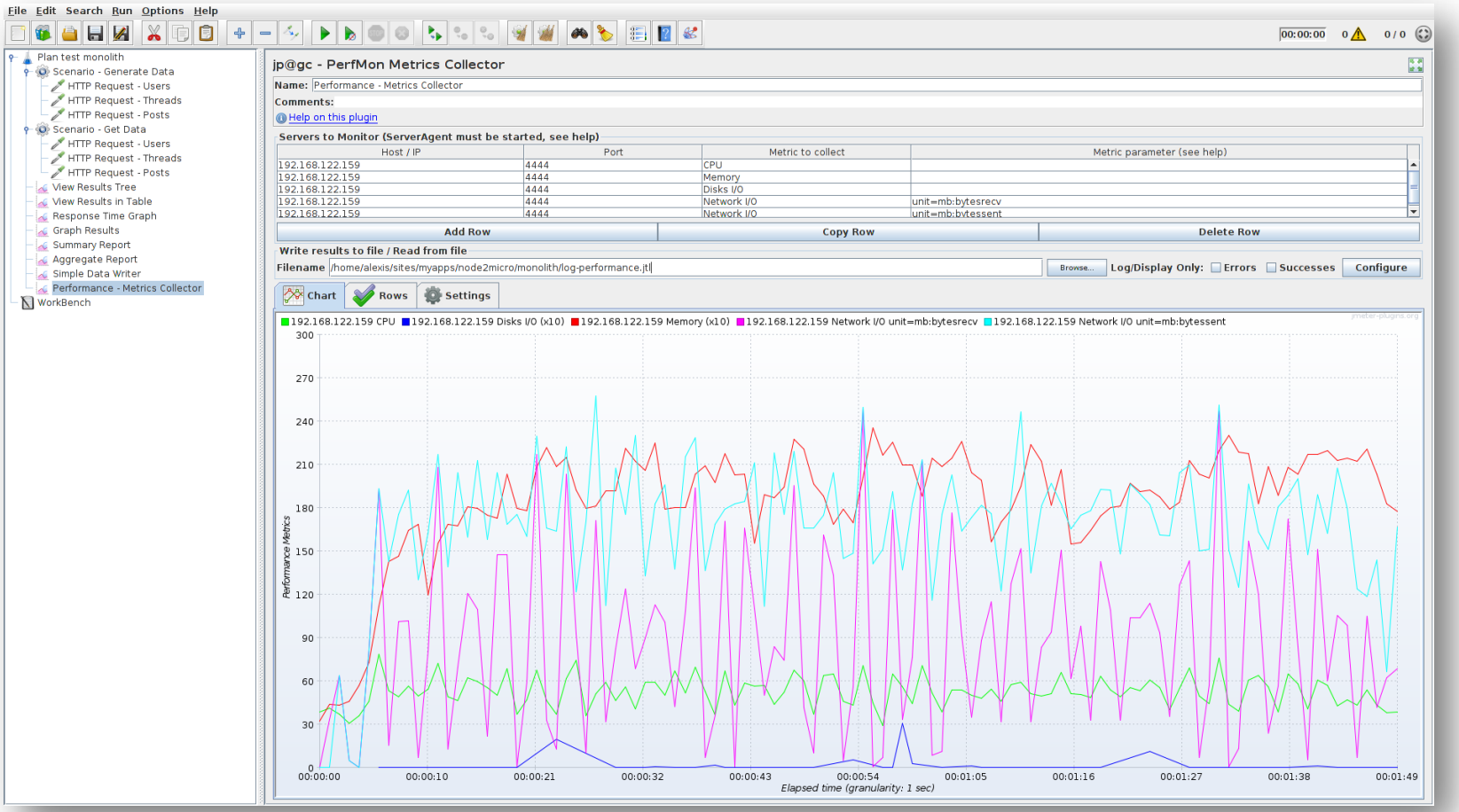
Resumen en el CLI de la ejecución de las pruebas de estrés



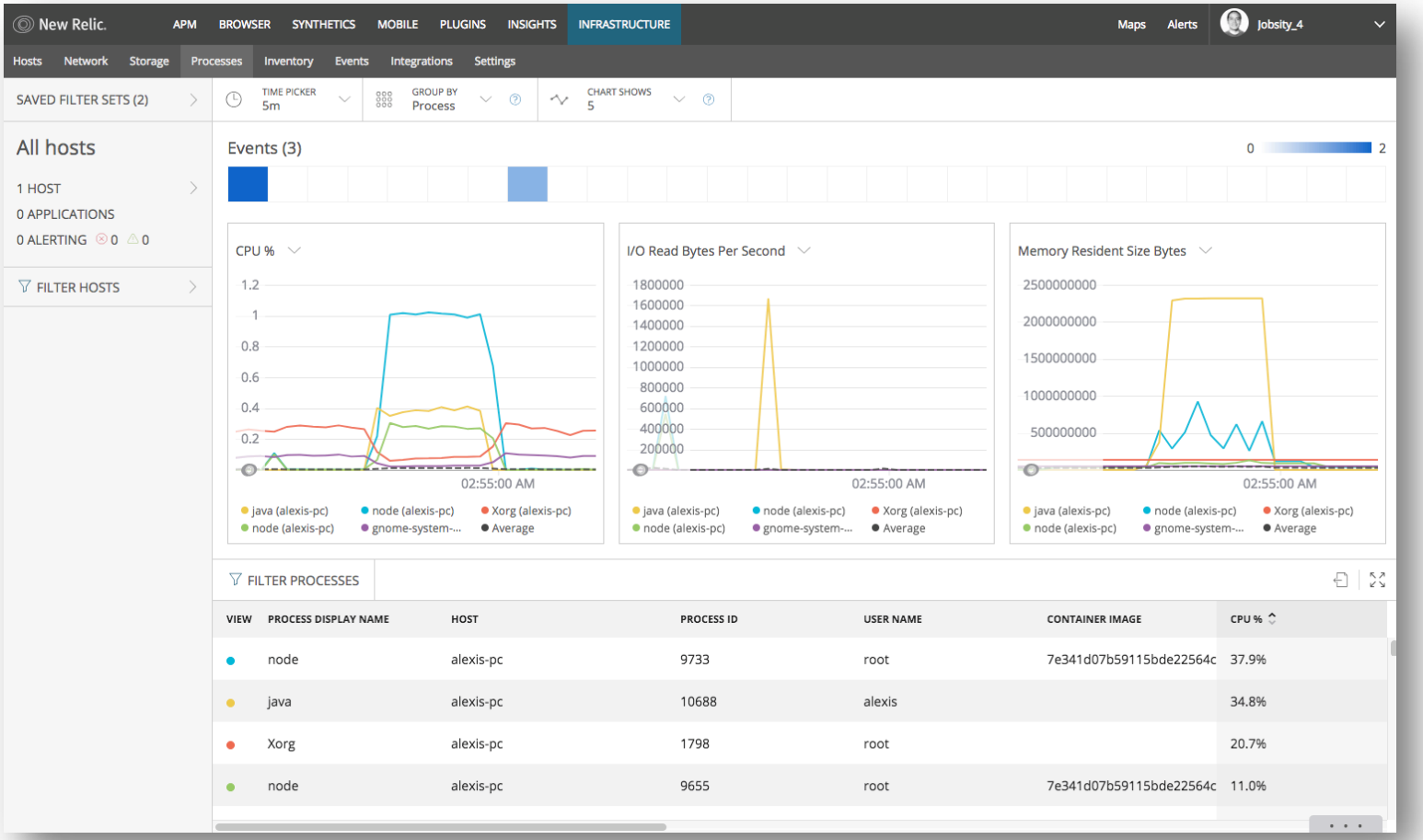
Logs en el CLI mostrando el levantamiento de los servicios en Contenedores de Docker.



Presentación de los resultados (logs) en el GUI de JMeter.



Presentación de los resultados en NewRelic.



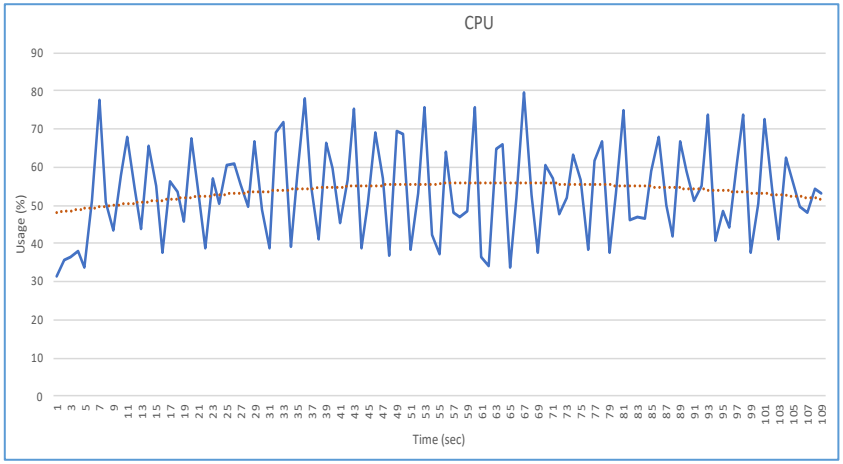
Anexo 3: Resultado obtenido con JMeter tras la ejecución de la prueba de estrés para la Aplicación Monolítica en el Caso 1.

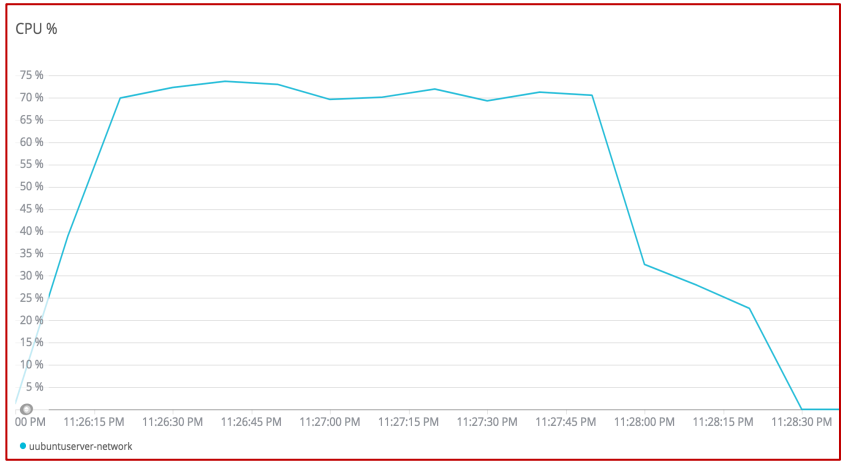
Elapsed time	CPU (%)	Disk - read bytes/s	Disk - write bytes/s	Memory (%)	Memory (MB)	Network - rec bytes/s	Network - sent bytes/s
26:10.0	31.313	0	0	3.6	867.207	0	0
26:11.0	35.82	0	0	3.6	867.207	0	0
26:12.0	36.616	1084	2.25E+07	4.572	1025.601	8.46E+07	8.46E+07
26:13.0	38	444	1.35E+07	4.65	1047.164	6591820	6593210
26:14.0	33.583	0	77824	5.572	1176.492	167	256
26:15.0	51.133	216	159744	7.075	1387.386	6.25E+07	6.25E+07
26:16.0	77.611	40	614400	10.885	1921.847	2.12E+08	2.12E+08
26:17.0	50.67	0	512000	14.198	2386.394	2.97E+07	1.50E+08
26:18.0	43.271	192	528384	14.059	2367.207	3.58E+07	1.88E+08
26:19.0	57.692	0	253952	10.887	1922.867	1.29E+08	1.73E+08
26:20.0	67.801	0	266240	14.018	2361.855	1.57E+08	2.17E+08
26:21.0	54.2	0	491520	16.726	2741.613	7708560	1.40E+08
26:22.0	43.927	0	397312	16.191	2666.871	1.11E+08	2.23E+08
26:23.0	65.482	0	462848	15.421	2559.121	1.56E+08	1.98E+08
26:24.0	54.987	0	16384	15.263	2537.042	5.71E+07	1.26E+08
26:25.0	37.467	0	471040	17.077	2791.488	634309	1.80E+08
26:26.0	56.41	0	454656	14.782	2470.011	1.96E+08	2.27E+08
26:27.0	53.482	0	471040	17.798	2892.957	2.00E+07	1.02E+08
26:28.0	45.905	0	614400	14.979	2498.031	8.17E+07	2.05E+08
26:29.0	67.609	0	417792	17.59	2864.23	2.32E+08	2.45E+08
26:30.0	54.712	0	487424	17.309	2825.132	5452184	1.37E+08
26:31.0	38.624	0	339968	16.084	2653.535	9802564	1.76E+08
26:32.0	57.07	0	327680	18.498	2992.085	1.22E+08	1.35E+08
26:33.0	50.387	0	303104	17.602	2866.589	1.07E+08	2.13E+08
26:34.0	60.465	0	196608	18.067	2931.847	1.10E+08	2.16E+08
26:35.0	60.902	0	147456	20.417	3261.285	1.11E+08	1.71E+08
26:36.0	55.324	0	110592	19.773	3171.089	8.57E+07	2.03E+08
26:37.0	49.612	0	397312	19.663	3155.933	8.47E+07	1.96E+08
26:38.0	66.834	0	540672	21.069	3353.175	1.53E+08	1.89E+08
26:39.0	49.071	0	241664	19.688	3159.777	8.00E+07	2.14E+08
26:40.0	38.753	0	245760	19.979	3200.593	8710208	1.79E+08
26:41.0	68.982	0	495616	20.748	3308.625	2.24E+08	2.38E+08
26:42.0	71.767	0	479232	22.916	3612.687	9.17E+07	1.52E+08
26:43.0	39.153	0	483328	21.458	3408.613	7317407	1.79E+08
26:44.0	57.974	0	2064384	21.814	3460.031	1.31E+08	2.10E+08
26:45.0	77.862	0	4517888	19.697	3182.343	1.87E+08	1.99E+08
26:46.0	55.118	8	1.73E+07	18.233	2977.972	3703538	1.36E+08
26:47.0	41.269	0	6737920	16.966	2806.714	4.53E+07	2.11E+08
26:48.0	66.406	0	573440	19.011	3093.617	1.64E+08	1.79E+08
26:49.0	59.844	0	212992	17.109	2827.07	1.16E+08	2.14E+08
26:50.0	45.431	0	495616	17.003	2812.347	3982226	1.66E+08
26:51.0	56.735	0	528384	17.098	2825.875	1.50E+08	2.08E+08
26:52.0	75.252	0	323584	21.342	3420.886	1.67E+08	1.97E+08
26:53.0	38.888	0	167936	20.127	3250.671	3632141	1.67E+08
26:54.0	50.259	0	294912	19.447	3155.429	8.21E+07	1.94E+08
26:55.0	69.041	0	339968	20.762	3339.824	2.13E+08	2.29E+08
26:56.0	56.842	0	249856	18.61	3038.332	3.00E+07	1.51E+08
26:57.0	36.87	0	364544	17.384	2866.535	1.11E+07	1.85E+08
26:58.0	69.554	0	380928	18.541	3028.906	2.60E+08	2.81E+08
26:59.0	68.607	0	131072	17.548	2889.773	5.04E+07	1.03E+08
27:00.0	38.251	0	282624	16.214	2702.91	1.04E+07	2.06E+08
27:01.0	53.233	0	315392	16.424	2732.375	1.58E+08	2.29E+08
27:02.0	75.5	0	114688	20.107	3248.667	1.57E+08	1.83E+08
27:03.0	42.133	0	294912	21.825	3489.687	602518	1.78E+08
27:04.0	37.046	0	491520	19.949	3226.945	3.28E+07	1.35E+08
27:05.0	64.16	0	471040	20.139	3253.742	1.91E+08	2.05E+08
27:06.0	48.205	0	360448	19.365	3145.484	1.00E+08	2.29E+08
27:07.0	46.882	0	69632	18.163	2977.07	1.48E+07	9.51E+07
27:08.0	48.37	0	593920	17.831	2930.652	1.17E+08	2.20E+08
27:09.0	75.75	0	499712	22.043	3521.421	1.94E+08	2.02E+08
27:10.0	36.363	0	401408	20.112	3250.937	647961	1.80E+08
27:11.0	33.947	0	409600	18.85	3074.171	2.11E+07	1.45E+08
27:12.0	64.987	0	286720	21.077	3386.496	2.05E+08	2.19E+08
27:13.0	65.989	0	176128	19.212	3125.144	9.96E+07	1.52E+08
27:14.0	33.676	0	520192	17.79	2925.992	597090	1.93E+08
27:15.0	51.804	0	372736	16.675	2769.832	8.24E+07	1.49E+08
27:16.0	79.648	0	159744	19.288	3136.273	2.39E+08	2.47E+08
27:17.0	52.835	0	143360	17.438	2876.898	5431208	1.33E+08
27:18.0	37.433	0	520192	16.179	2700.714	6607122	1.76E+08
27:19.0	60.362	0	434176	17.757	2922.054	2.07E+08	2.21E+08
27:20.0	57.069	0	372736	20.747	3341.308	1.02E+08	1.96E+08
27:21.0	47.708	0	45056	18.425	3015.98	2.52E+07	1.44E+08
27:22.0	51.842	0	180224	19.991	3235.542	1.24E+08	2.31E+08
27:23.0	63.197	0	253952	21.048	3383.746	1.47E+08	2.01E+08
27:24.0	56.692	0	126976	20.388	3291.281	4.63E+07	1.51E+08
27:25.0	38.219	0	229376	17.628	2904.562	2.41E+07	1.77E+08
27:26.0	61.675	0	200704	19.228	3129	2.03E+08	2.17E+08
27:27.0	66.666	0	143360	19.707	3196.14	9.37E+07	1.63E+08
27:28.0	37.468	0	548864	19.65	3188.335	7243628	1.77E+08
27:29.0	53.045	0	311296	19.01	3098.699	1.19E+08	1.92E+08
27:30.0	74.75	0	237568	21.933	3508.574	2.00E+08	2.12E+08
27:31.0	45.989	0	299008	23.348	3707.085	982837	1.69E+08
27:32.0	46.753	0	450560	22.853	3637.941	1.12E+08	1.57E+08
27:33.0	46.596	0	315392	21.521	3451.367	7242402	1.44E+08
27:34.0	58.974	0	237568	21.932	3509.113	1.49E+08	2.20E+08
27:35.0	67.857	0	159744	20.032	3242.851	1.68E+08	1.96E+08
27:36.0	50	0	217088	20.225	3269.929	882382	1.55E+08
27:37.0	42.036	0	491520	18.958	3092.57	4.05E+07	1.63E+08
27:38.0	66.834	0	536576	20.214	3268.917	1.95E+08	2.10E+08
27:39.0	58.868	0	286720	18.649	3049.617	9.08E+07	1.91E+08
27:40.0	51.288	0	233472	17.318	2863.238	1.21E+07	1.61E+08
27:41.0	55.242	0	303104	17.698	2916.617	1.53E+08	2.24E+08
27:42.0	73.869	0	90112	22.321	3564.718	1.62E+08	1.69E+08
27:43.0	40.659	0	286720	21.58	3460.886	758576	1.87E+08
27:44.0	48.556	0	327680	21.792	3490.789	1.13E+08	1.80E+08
27:45.0	44.155	0	401408	22.526	3593.812	1.04E+07	1.25E+08
27:46.0	60.858	0	344064	22.506	3591.101	1.95E+08	2.46E+08
27:47.0	73.536	0	245760	19.302	3142.062	1.18E+08	1.55E+08
27:48.0	37.692	0	262144	20.367	3291.46	604016	1.81E+08
27:49.0	50.26	0	303104	18.468	3025.343	8.15E+07	1.71E+08
27:50.0	72.405	0	344064	18.846	3078.652	1.92E+08	2.18E+08
27:51.0	54.155	0	180224	20.525	3313.945	5.73E+07	1.83E+08
27:52.0	41.099	0	397312	18.397	3015.89	1.97E+07	1.87E+08
27:53.0	62.531	0	471040	20.282	3280.324	1.72E+08	1.80E+08
27:54.0	55.497	0	249856	21.967	3516.714	8.85E+07	1.88E+08
27:55.0	49.489	0	1200128	17.126	2839.121	5.49E+07	1.61E+08
27:56.0	47.989	0	237568	18.769	3069.617	9.84E+07	2.02E+08
27:57.0	54.353	0	466944	18.787	3072.277	9.61E+07	1.68E+08
27:58.0	53.263	0	589824	17.568	2901.687	5.30E+07	1.42E+08

	CPU	Disk Read	Disk Write	Memory %	Memory MB	Network received	Network sent
Min	31.313	0	0	3.6	867.207	0	0
Max	79.648	1084	22495232	23.348	3707.085	260251236	280736440
AVG	53.7275	18.2018	927837.9450	18.0823	2955.7629	89644424.3761	175794816.4587
MED	53.263	0	327680	18.846	3074.171	85744688	181136008
DES	12.1825	114.7164	3007978.0514	3.9715	568.4580	73383955.1696	49939865.9056

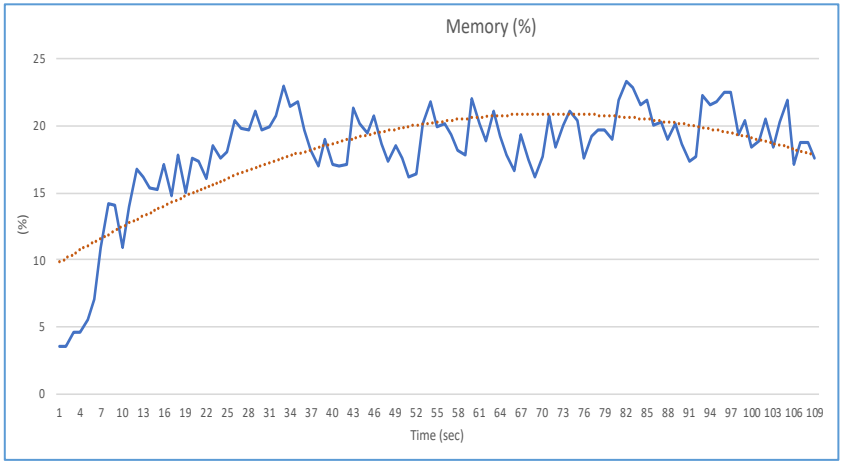
De las tablas anteriores, se presentan gráficas obtenidas de los logs de JMeter con borde azul y obtenidas con el servicio de NewRelic con borde rojo. En los gráficos de NewRelic, se debe considerar solamente la zona con mayor actividad para comparar con los gráficos correspondientes de JMeter.

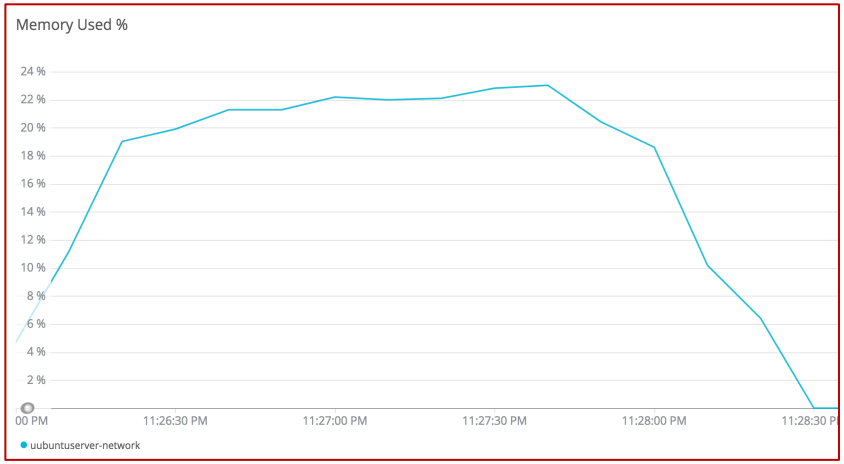
CPU

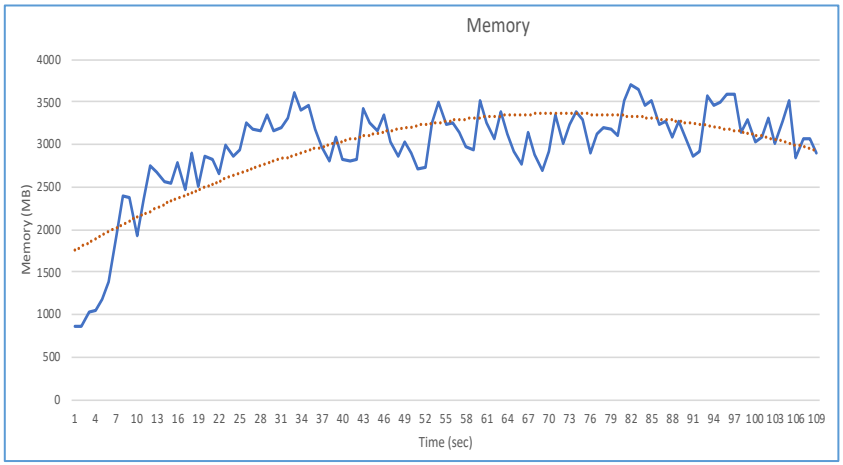


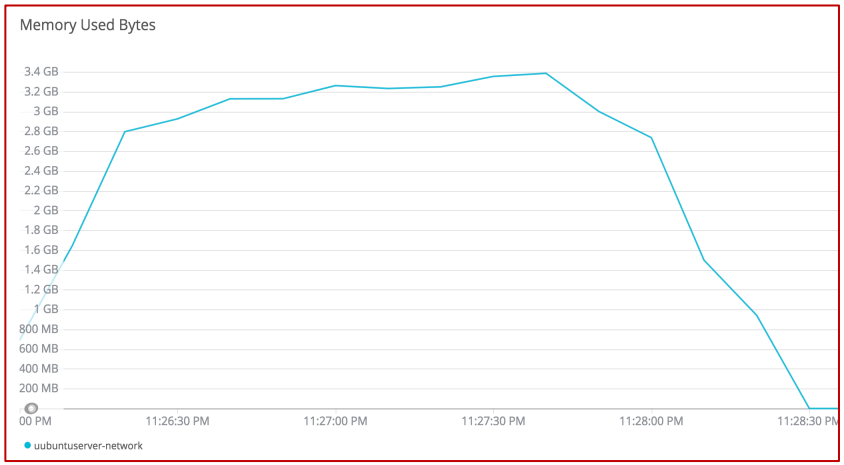


Memoria

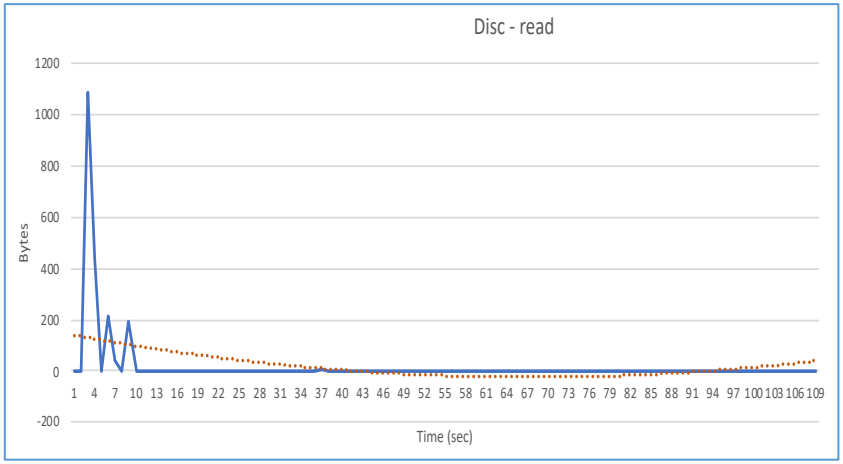

Memory (%)
25
20
15
10
5
0
(%)
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109
Time (sec)


Memory Used %
24 %
22 %
20 %
18 %
16 %
14 %
12 %
10 %
8 %
6 %
4 %
2 %
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
uubuntuserver-network

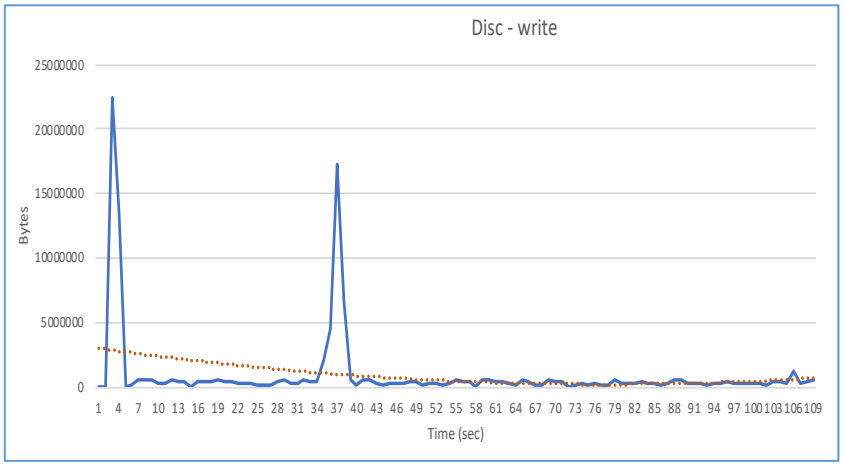

Memory
4000
3500
3000
2500
2000
1500
1000
500
0
Memory (MB)
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109
Time (sec)

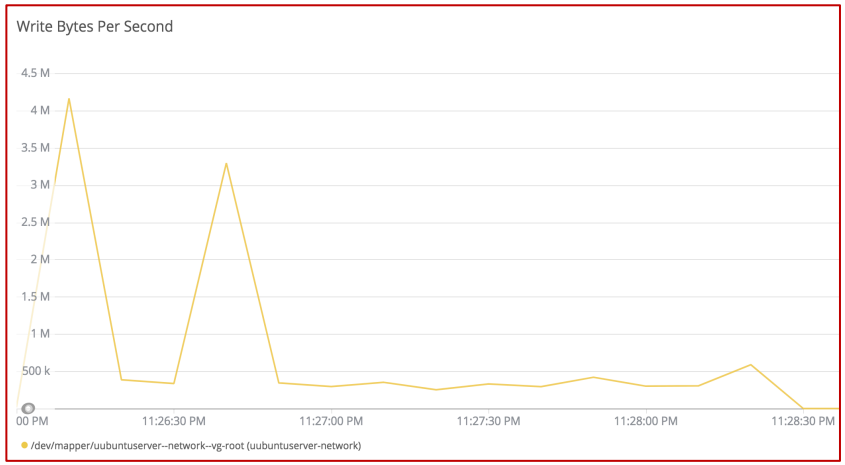

Memory Used Bytes
3.4 GB
3.2 GB
3 GB
2.8 GB
2.6 GB
2.4 GB
2.2 GB
2 GB
1.8 GB
1.6 GB
1.4 GB
1.2 GB
1 GB
800 MB
600 MB
400 MB
200 MB
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
uubuntuserver-network

Disco

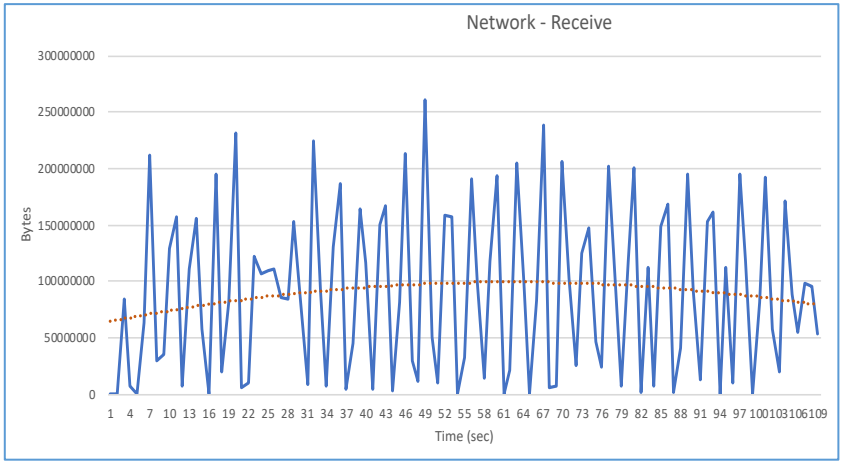

Disc - read
1200
1000
800
600
400
200
0
-200
Bytes
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109
Time (sec)

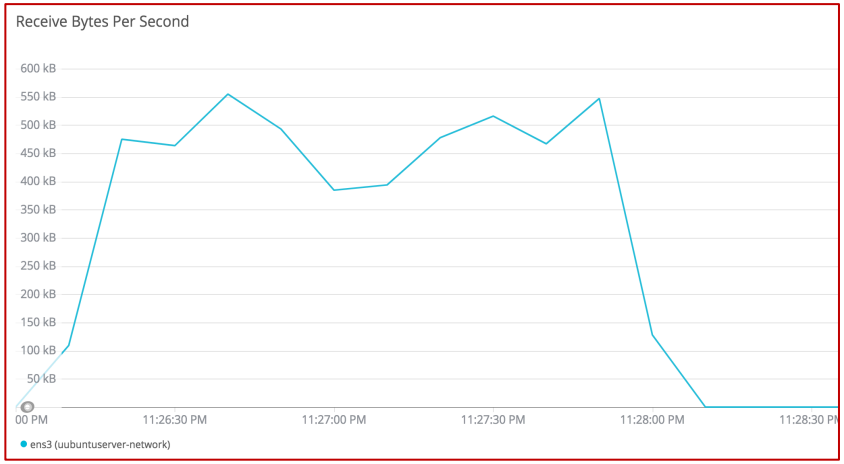

Read Bytes Per Second
220 k
200 k
180 k
160 k
140 k
120 k
100 k
80 k
60 k
40 k
20 k
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
11:28:30 PM
/dev/mapper/uubuntuserver--network--vg-root (uubuntuserver-network)

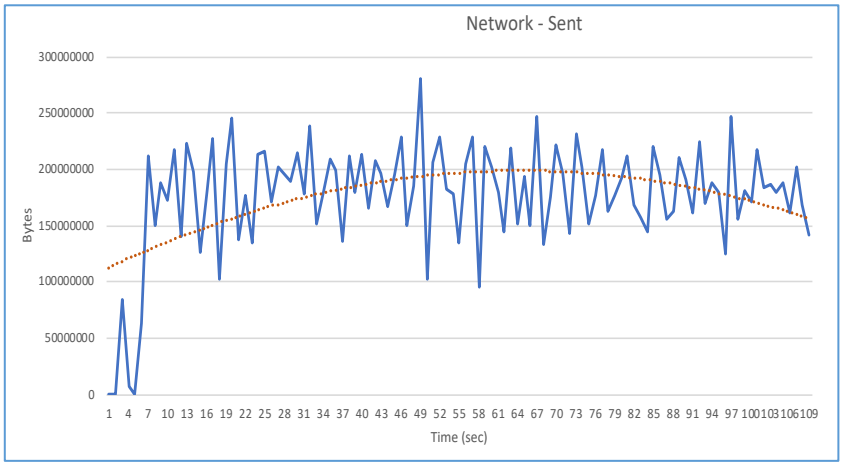

Disc - write
25000000
20000000
15000000
10000000
5000000
0
Bytes
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109
Time (sec)

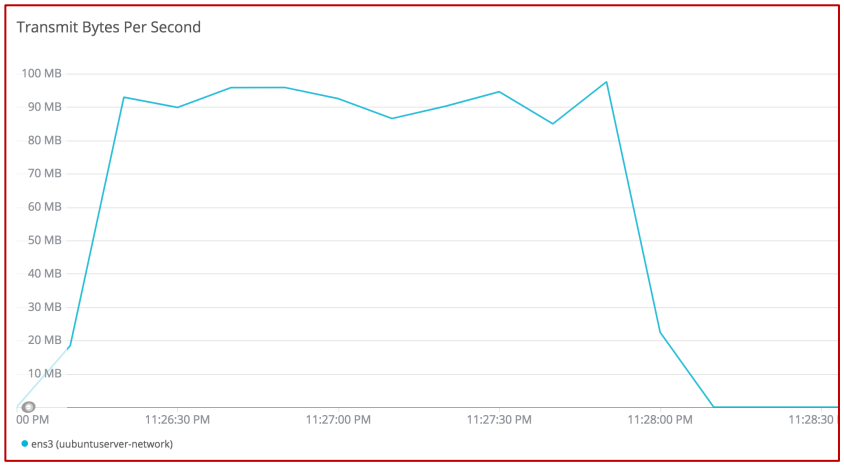

Write Bytes Per Second
4.5 M
4 M
3.5 M
3 M
2.5 M
2 M
1.5 M
1 M
500 k
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
11:28:30 PM
/dev/mapper/uubuntuserver--network--vg-root (uubuntuserver-network)

Red


Network - Receive
300000000
250000000
200000000
150000000
100000000
50000000
0
Bytes
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100103106109
Time (sec)


Receive Bytes Per Second
600 kB
550 kB
500 kB
450 kB
400 kB
350 kB
300 kB
250 kB
200 kB
150 kB
100 kB
50 kB
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
ens3 (uubuntuserver-network)


Network - Sent
300000000
250000000
200000000
150000000
100000000
50000000
0
Bytes
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100103106109
Time (sec)


Transmit Bytes Per Second
100 MB
90 MB
80 MB
70 MB
60 MB
50 MB
40 MB
30 MB
20 MB
10 MB
00 PM
11:26:30 PM
11:27:00 PM
11:27:30 PM
11:28:00 PM
ens3 (uubuntuserver-network)

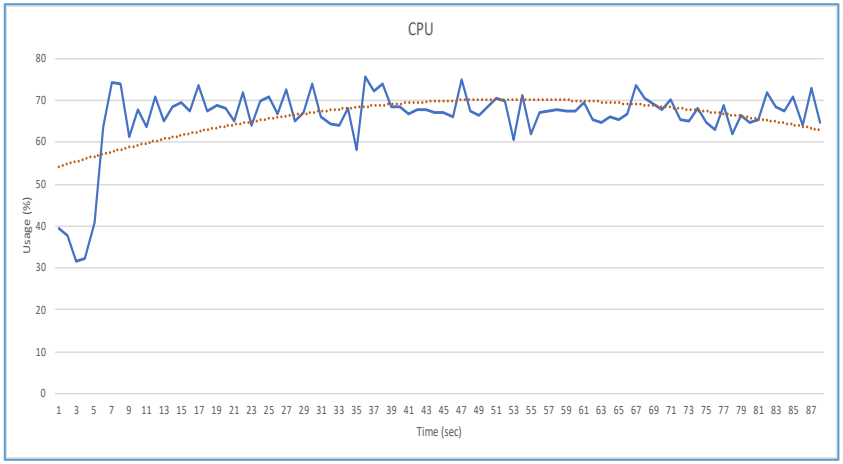
Anexo 4: Resultado obtenido con JMeter tras la ejecución de la prueba de estrés para la Aplicación de Microservicios en el Caso 1.

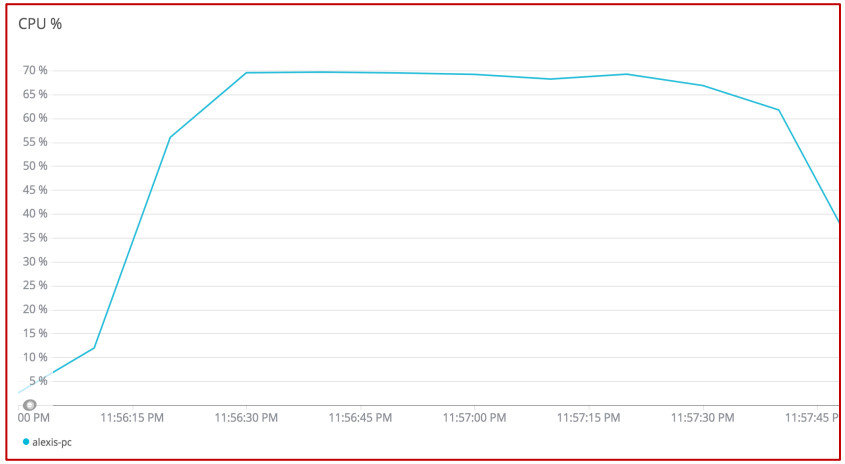
Elapsed time	CPU (%)	Disc - read bytes/s	Disc - write bytes/s	Memory (%)	Memory (MB)	Network - receive bytes/s	Network - sent bytes/s
56:17.0	39.593	0	0	13.961	3810.5835	0	0
56:18.0	37.81	0	0	13.961	3810.5835	0	0
56:19.0	31.578	48	5.06E+07	15.233	3906.8005	6.81E+07	6.81E+07
56:20.0	32.338	3024	1.09E+07	15.802	3947.291	72793	73509
56:21.0	40.722	24	1179648	16.505	4003.3335	6.45E+07	6.45E+07
56:22.0	63.805	432	2994176	20.533	4324.4195	4.23E+08	2.24E+08
56:23.0	74.193	0	2781184	22.611	4498.982	3.60E+08	1.87E+08
56:24.0	73.88	0	3366912	25.589	4735.91	4.98E+08	2.47E+08
56:25.0	61.369	0	4349952	24.634	4652.2085	4.30E+08	2.22E+08
56:26.0	67.91	0	2727936	24.711	4658.416	4.44E+08	2.29E+08
56:27.0	63.749	0	2949120	28.35	4948.451	4.50E+08	2.37E+08
56:28.0	70.812	0	2949120	32.821	5307.3475	4.68E+08	2.42E+08
56:29.0	65.185	0	2580480	29.574	5048.6265	4.22E+08	2.10E+08
56:30.0	68.55	0	2973696	29.724	5060.6695	4.30E+08	2.18E+08
56:31.0	69.478	0	2777088	29.901	5078.3435	4.25E+08	2.13E+08
56:32.0	67.241	0	2678784	30.08	5092.65	4.23E+08	2.10E+08
56:33.0	73.684	0	2727936	29.947	5078.5625	4.32E+08	2.19E+08
56:34.0	67.241	0	2850816	30.787	5145.539	4.23E+08	2.10E+08
56:35.0	68.75	48	4448256	31.348	5190.328	4.33E+08	2.18E+08
56:36.0	68	0	2580480	29.947	5078.7285	5.66E+08	2.57E+08
56:37.0	64.925	0	2334720	31.54	5205.656	4.15E+08	2.30E+08
56:38.0	71.82	0	2752512	32.657	5294.7925	6.18E+08	2.78E+08
56:39.0	63.909	0	2359296	29.587	5050.207	4.33E+08	2.20E+08
56:40.0	69.849	0	1.20E+08	29.266	5033.98	4.36E+08	2.23E+08
56:41.0	70.959	0	3170304	29.412	5045.5935	4.47E+08	2.34E+08
56:42.0	66.583	0	2383872	31.143	5183.6695	4.30E+08	2.17E+08
56:43.0	72.388	0	1941504	32.928	5326.021	4.31E+08	2.19E+08
56:44.0	65.162	0	1916928	29.553	5057.07	4.26E+08	2.13E+08
56:45.0	67.082	0	3366912	30.31	5117.441	4.36E+08	2.23E+08
56:46.0	73.75	0	4104192	29.94	5088.189	4.54E+08	2.28E+08
56:47.0	65.994	0	7864320	31.415	5206.277	4.34E+08	2.21E+08
56:48.0	64.5	0	2.18E+07	31.703	5227.7985	4.68E+08	2.55E+08
56:49.0	63.93	0	2.69E+07	34.234	5434.539	3.76E+08	1.62E+08
56:50.0	68.079	0	2654208	29.997	5096.871	5.08E+08	2.95E+08
56:51.0	58.25	0	3170304	33.242	5355.537	3.55E+08	1.46E+08
56:52.0	75.757	0	2826240	29.556	5061.8535	6.51E+08	2.75E+08
56:53.0	72.18	0	2777088	30.105	5108.3085	2.90E+08	2.35E+08
56:54.0	73.934	0	3956736	30.699	5160.32	3.71E+08	1.57E+08
56:55.0	68.407	0	3883008	33.127	5353.8515	4.40E+08	2.18E+08
56:56.0	68.5	0	3317760	33.484	5374.789	6.49E+08	2.94E+08
56:57.0	66.666	0	2777088	30.381	5124.5075	4.50E+08	2.24E+08
56:58.0	67.821	0	2740224	29.481	5052.832	5.03E+08	2.53E+08
56:59.0	67.919	0	2985984	29.449	5050.287	4.44E+08	2.31E+08
57:00.0	67.085	0	2457600	29.457	5050.943	4.49E+08	2.36E+08
57:01.0	66.917	0	2629632	29.641	5069.2635	4.48E+08	2.35E+08
57:02.0	65.914	0	2949120	31.545	5221.0915	4.34E+08	2.21E+08
57:03.0	74.937	0	2654208	33.578	5379.617	4.64E+08	2.36E+08
57:04.0	67.412	0	1769472	29.938	5089.5565	4.37E+08	2.23E+08
57:05.0	66.417	0	2850816	31.951	5250.0195	4.23E+08	2.09E+08
57:06.0	68.578	0	1941504	34.033	5415.986	4.32E+08	2.19E+08
57:07.0	70.351	0	1646592	30.513	5135.4625	4.32E+08	2.18E+08
57:08.0	69.849	0	1769472	31.752	5234.238	4.97E+08	2.27E+08
57:09.0	60.459	0	1376256	31.43	5208.658	3.73E+08	2.17E+08
57:10.0	71.287	0	1449984	33.378	5363.9195	7.04E+08	3.01E+08
57:11.0	61.928	0	1622016	30.503	5134.8375	3.89E+08	2.09E+08
57:12.0	66.915	0	1622016	29.31	5039.8105	5.07E+08	2.50E+08
57:13.0	67.424	0	1523712	30.569	5140.1445	4.39E+08	2.26E+08
57:14.0	67.581	0	1474560	32.502	5294.3045	4.52E+08	2.38E+08
57:15.0	67.25	0	2703360	30.582	5141.2515	4.37E+08	2.23E+08
57:16.0	67.249	0	1548288	29.683	5069.658	4.51E+08	2.37E+08
57:17.0	69.402	0	3121152	29.767	5079.7635	4.58E+08	2.32E+08
57:18.0	65.25	0	3268608	30.119	5107.824	4.41E+08	2.26E+08
57:19.0	64.646	0	2162688	31.861	5246.6875	4.41E+08	2.28E+08
57:20.0	65.92	0	2719744	33.83	5403.656	4.07E+08	1.93E+08
57:21.0	65.422	0	2146304	30.316	5123.66	4.47E+08	2.33E+08
57:22.0	66.582	0	2334720	33.106	5346.076	4.15E+08	2.01E+08
57:23.0	73.604	0	2654208	34.939	5492.2695	4.33E+08	2.19E+08
57:24.0	70.426	0	2801664	31.363	5207.2595	4.67E+08	2.26E+08
57:25.0	69.132	24	3710976	30.594	5146.068	6.08E+08	2.75E+08
57:26.0	67.758	0	2334720	29.392	5050.3395	5.39E+08	2.72E+08
57:27.0	70.223	0	1744896	29.515	5060.1675	4.40E+08	2.26E+08
57:28.0	65.25	0	1818624	29.515	5060.2085	4.50E+08	2.36E+08
57:29.0	64.987	0	1499136	30.052	5103.07	4.53E+08	2.39E+08
57:30.0	68.17	24	1720320	29.781	5081.457	4.47E+08	2.33E+08
57:31.0	64.676	0	1499136	29.66	5075.0545	4.65E+08	2.38E+08
57:32.0	63.065	0	1695744	31.151	5193.945	4.24E+08	2.08E+08
57:33.0	68.671	0	1548288	33.163	5347.785	4.40E+08	2.26E+08
57:34.0	62	0	1474560	30.3	5119.703	4.21E+08	2.07E+08
57:35.0	66.25	0	2752512	30.352	5123.8395	4.53E+08	2.38E+08
57:36.0	64.735	0	2285568	33.374	5364.7575	5.36E+08	2.40E+08
57:37.0	65.316	0	2482176	30.986	5174.7695	3.38E+08	1.94E+08
57:38.0	71.969	0	3047424	30.732	5154.57	6.58E+08	2.91E+08
57:39.0	68.592	0	2015232	29.841	5083.6795	5.02E+08	2.52E+08
57:40.0	67.418	0	2433024	30.719	5153.4275	4.47E+08	2.32E+08
57:41.0	70.864	0	2580480	32.582	5301.945	4.51E+08	2.36E+08
57:42.0	64.07	0	1400832	30.322	5121.8865	4.50E+08	2.36E+08
57:43.0	73.019	0	1327104	29.628	5069.898	4.23E+08	2.09E+08
57:44.0	64.824	0	2334720	30.498	5139.24	4.64E+08	2.42E+08

	CPU	Disk Read	Disk Write	Memory %	Memory MB	Network received	Network sent
MIN	31.578	0.000	0.000	13.961	3810.584	0.000	0.000
MAX	75.757	3024.000	119980032.000	34.939	5493.270	703874728.000	300732765.000
AVG	65.955	41.182	4968261.818	29.626	5063.181	430807543.545	217503213.818
MED	67.250	0.000	2654208.000	30.313	5120.795	439659753.000	225701617.000
STDEV	8.089	324.981	13852888.717	4.239	341.069	119897485.065	54237027.081

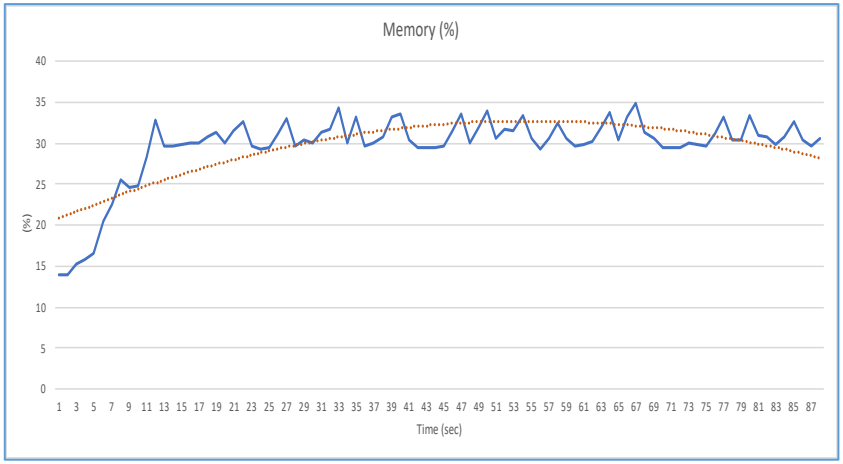
De las tablas anteriores, se presentan gráficas obtenidas de los logs de JMeter con borde azul y obtenidas con el servicio de NewRelic con borde rojo. En los gráficos de NewRelic, se debe considerar solamente la zona con mayor actividad para comparar con los gráficos correspondientes de JMeter.

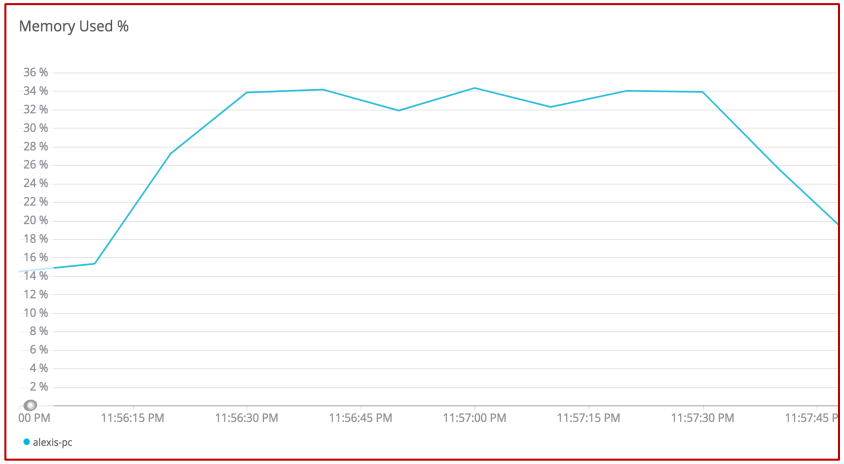
CPU

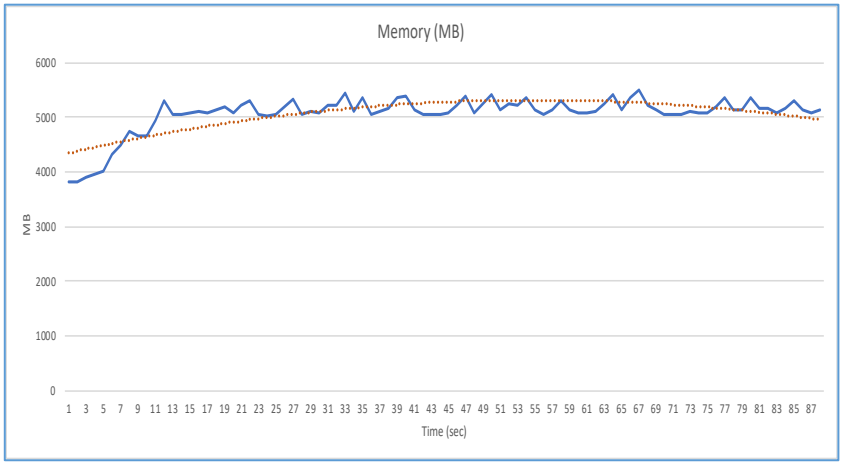


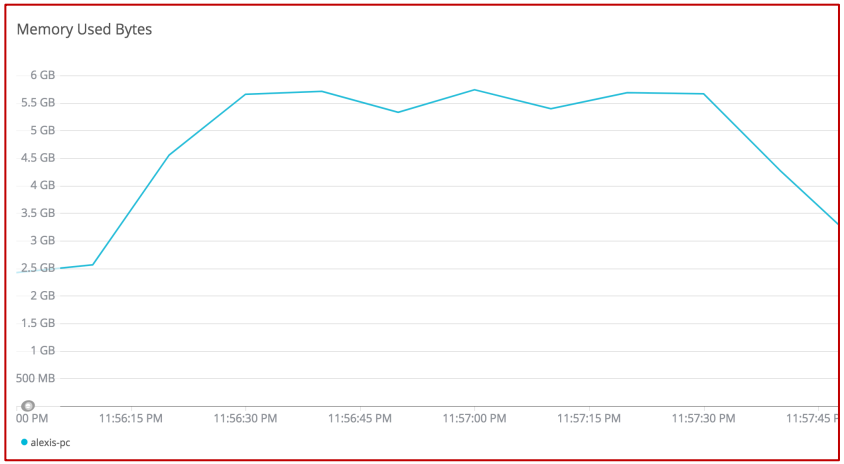


Memoria

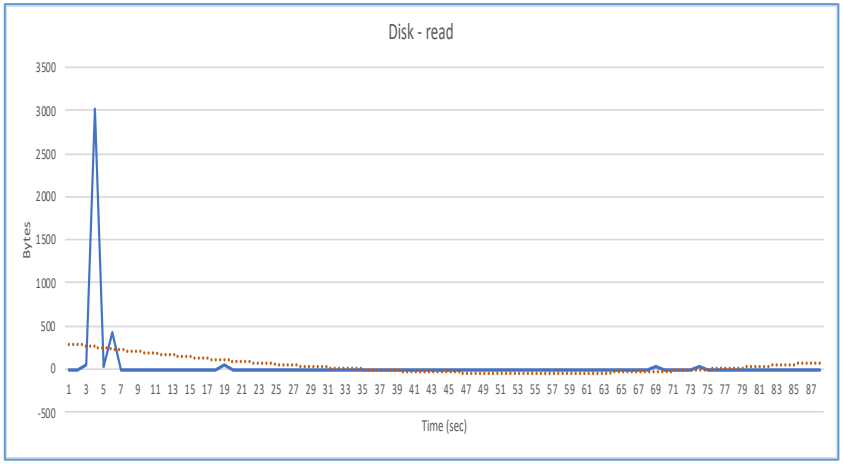

Memory (%)
(%)
Time (sec)

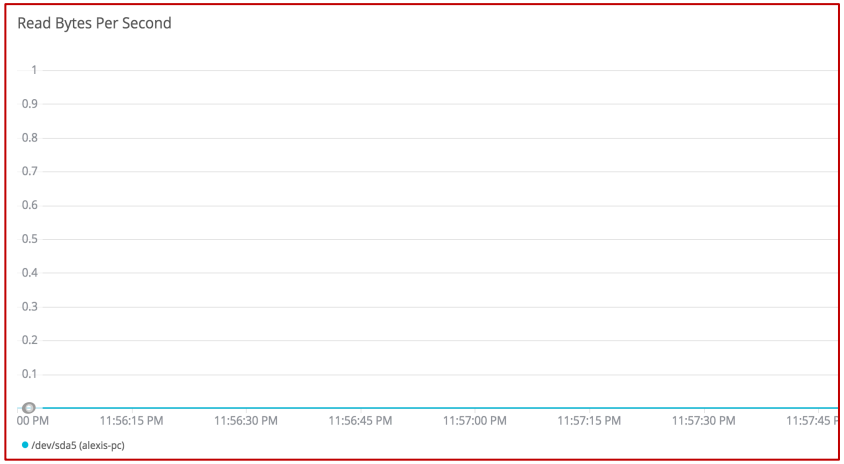

Memory Used %
alexis-pc

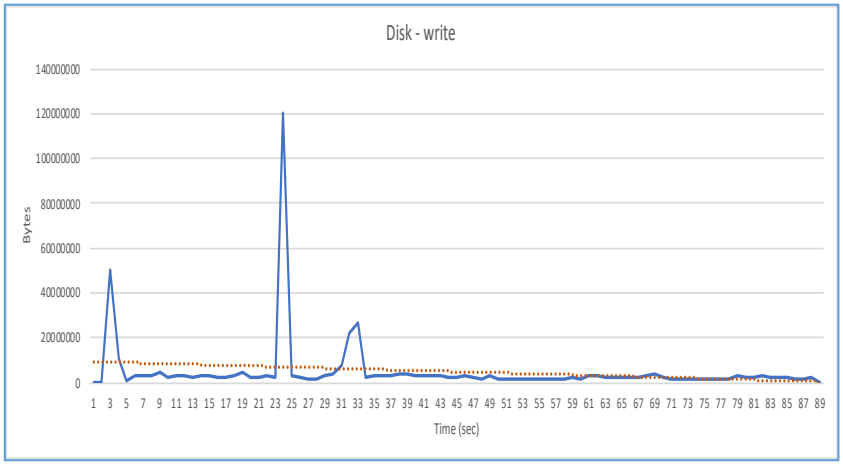

Memory (MB)
MB
Time (sec)

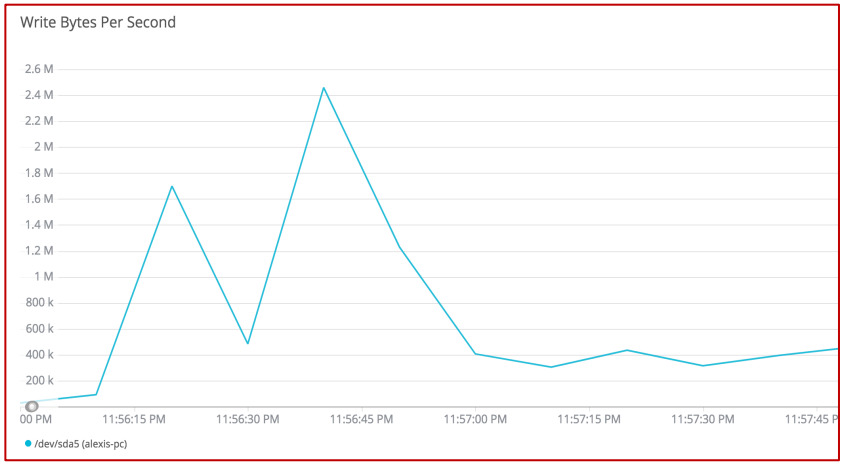

Memory Used Bytes
alexis-pc

Disco

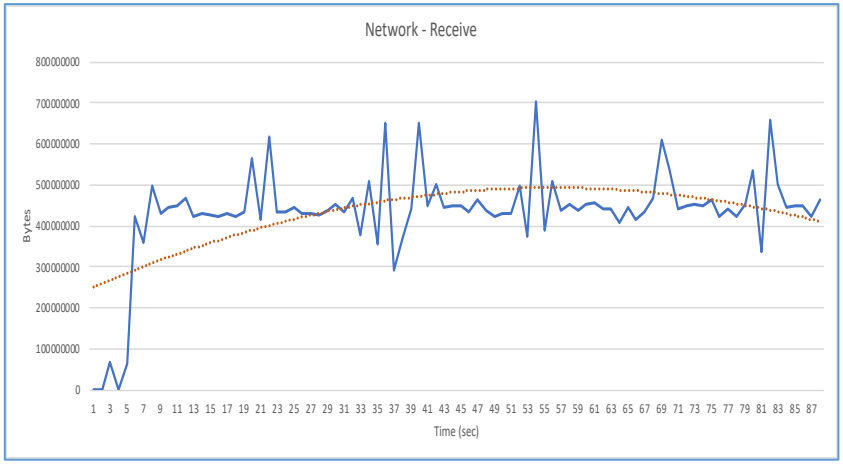

Disk - read
Bytes
Time (sec)

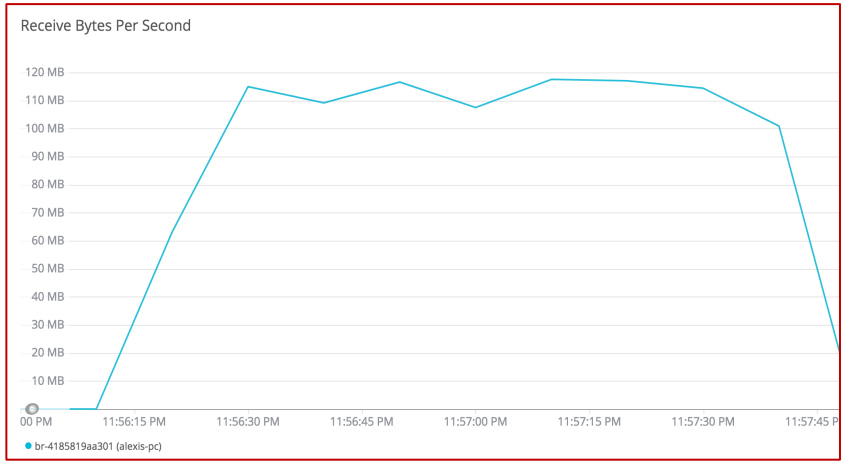

Read Bytes Per Second
/dev/sda5 (alexis-pc)

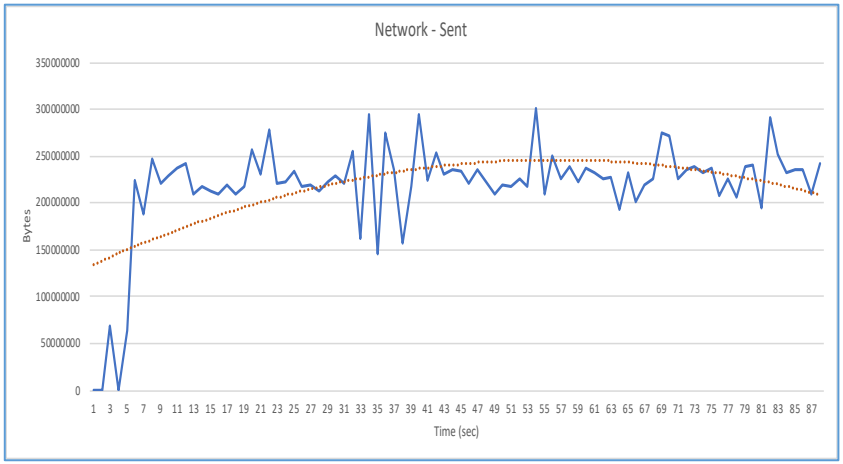

Disk - write
Bytes
Time (sec)

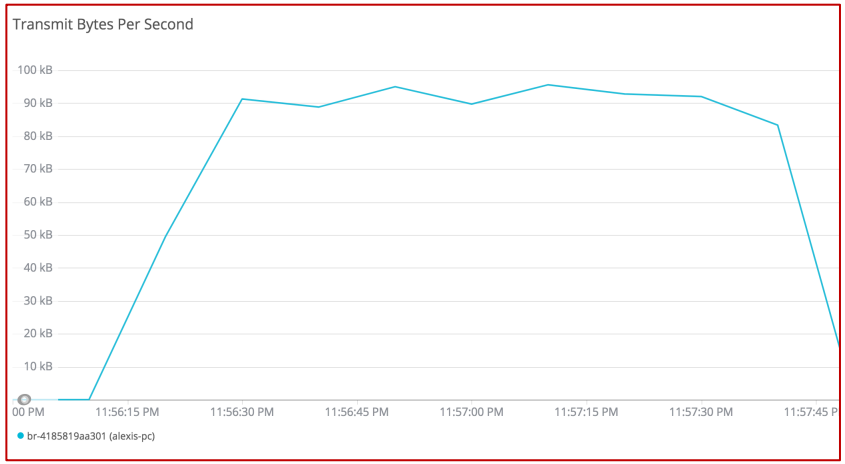

Write Bytes Per Second
/dev/sda5 (alexis-pc)

Red


Network - Receive
800000000
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
Bytes
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87
Time (sec)


Receive Bytes Per Second
120 MB
110 MB
100 MB
90 MB
80 MB
70 MB
60 MB
50 MB
40 MB
30 MB
20 MB
10 MB
00 PM
11:56:15 PM
11:56:30 PM
11:56:45 PM
11:57:00 PM
11:57:15 PM
11:57:30 PM
11:57:45 P
br-4185819aa301 (alexis-pc)


Network - Sent
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
Bytes
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87
Time (sec)


Transmit Bytes Per Second
100 kB
90 kB
80 kB
70 kB
60 kB
50 kB
40 kB
30 kB
20 kB
10 kB
00 PM
11:56:15 PM
11:56:30 PM
11:56:45 PM
11:57:00 PM
11:57:15 PM
11:57:30 PM
11:57:45 P
br-4185819aa301 (alexis-pc)

Anexo 5: Resultado obtenido con JMeter tras la ejecución de la prueba de estrés para la Aplicación Monolítica en el Caso 2.

Elapsed time	CPU (%)	Disk - read bytes/s	Disk - write bytes/s	Memory (%)	Memory (MB)	Network - receive bytes/s	Network - sent bytes/s
26:02.0	32.16	0	0	3.742	813.121	0	0
26:03.0	32.338	0	0	3.742	813.121	0	0
26:04.0	38.442	0	3.11E+07	5.36	1075.57	1.49E+08	1.49E+08
26:05.0	36.775	1096	1.90E+07	5.316	1081.789	4.43E+07	4.43E+07
26:06.0	32.169	4	1.24E+07	5.898	1178.23	8096401	8096508
26:07.0	30.303	444	1.40E+07	6.176	1226.894	5590265	5590329
26:08.0	33.668	0	90112	7.122	1359.746	418	1833
26:09.0	35.338	0	499712	8.163	1505.593	52	142
26:10.0	36.476	280	344064	9.479	1690.457	6117868	6117977
26:11.0	79.648	0	237568	10.99	1902.347	2.45E+08	2.45E+08
26:12.0	86.063	40	458752	13.598	2268.035	3.40E+08	3.40E+08
26:13.0	76.75	0	512000	15.976	2601.57	3.00E+08	3.00E+08
26:14.0	87.128	0	32768	20.519	3238.562	2.91E+07	2.91E+07
26:15.0	86.466	0	794624	29.813	4541.187	1141418	9564256
26:16.0	48.041	0	1097728	33.528	5062.042	8.61E+07	2.20E+08
26:17.0	56.451	0	413696	36.02	5411.566	534661	1.41E+08
26:18.0	27.792	0	479232	33.922	5117.73	687799	2.19E+08
26:19.0	24.552	0	487424	32.541	4924.398	386447	1.87E+08
26:20.0	28.967	0	548864	31.142	4728.585	2616554	1.89E+08
26:21.0	46.568	0	339968	28.764	4395.41	3.05E+07	1.51E+08
26:22.0	64.213	128	331776	29.747	4533.316	5.85E+07	9.81E+07
26:23.0	65.925	0	282624	31.16	4731.621	1.59E+08	1.85E+08
26:24.0	88.528	0	249856	28.857	4408.871	3.51E+08	3.51E+08
26:25.0	88.383	0	274432	31.297	4751.085	3.76E+08	3.76E+08
26:26.0	94.235	0	1945600	26.652	4100.007	3.10E+07	3.10E+07
26:27.0	77.192	0	2002944	33.927	5119.816	3.34E+07	8.05E+07
26:28.0	49.872	0	360448	35.333	5316.984	3.65E+07	1.87E+08
26:29.0	38.539	0	454656	36.073	5420.914	430984	1.76E+08
26:30.0	45.454	0	503808	34.8	5242.64	1.43E+07	1.82E+08
26:31.0	31.451	0	540672	32.068	4859.972	836183	1.87E+08
26:32.0	26.943	0	548864	30.684	4666.269	570116	2.19E+08
26:33.0	59.079	0	528384	29.514	4502.406	6.45E+07	1.27E+08
26:34.0	72.658	0	274432	22.022	3452.46	1.62E+08	1.88E+08
26:35.0	76.79	0	192512	23.826	3705.437	2.56E+08	2.69E+08
26:36.0	76.633	0	241664	26.21	4039.652	1.94E+08	2.07E+08
26:37.0	69.289	0	368640	30.365	4622.21	1.48E+08	2.08E+08
26:38.0	50.379	0	380928	26.63	4098.699	9.88E+07	2.22E+08
26:39.0	55.235	0	397312	28.45	4353.996	7.20E+07	1.76E+08
26:40.0	66.838	0	9736192	30.345	4656.847	7.60E+07	1.82E+08
26:41.0	80.867	4	2.96E+07	32.838	5006.277	2.26E+07	3.62E+07
26:42.0	32.08	0	622592	34.27	5207.14	2641974	2.04E+08
26:45.0	28.061	0	483328	34.326	5215.222	392694	1.68E+08
26:46.0	41.282	0	335872	33.096	5042.96	2.72E+07	2.07E+08
26:47.0	53.94	0	311296	31.063	4758.007	1.03E+08	1.77E+08
26:48.0	80.303	0	172032	32.56	4967.917	2.45E+08	2.66E+08
26:49.0	89.974	0	282624	30.04	4614.785	2.70E+08	2.70E+08
26:50.0	72.5	0	307200	31.991	4888.32	1.92E+08	2.20E+08
26:51.0	66.08	0	49152	36.688	5546.804	8.60E+07	1.52E+08
26:52.0	65.736	0	225280	35.327	5356.101	4.91E+07	1.37E+08
26:53.0	47.381	0	1839104	33.301	5073.535	8.09E+07	2.30E+08
26:54.0	51.804	0	7806976	35.473	5385.164	394692	1.53E+08
26:55.0	41.102	0	569344	27.478	4264.734	1.14E+07	1.58E+08
26:56.0	36.043	0	450560	21.989	3495.503	3568825	1.62E+08
26:57.0	30.919	0	413696	20.578	3297.757	754627	2.17E+08
26:58.0	56.171	0	471040	15.241	2549.894	9.98E+07	1.25E+08
26:59.0	66.582	0	389120	16.906	2783.468	2.55E+08	2.74E+08
27:00.0	78.908	0	253952	16.967	2792.167	1.91E+08	1.91E+08
27:01.0	77.272	0	372736	20.007	3218.351	1.02E+08	1.33E+08
27:02.0	61.246	0	1982464	24.351	3827.32	5164783	1.21E+08
27:03.0	50.128	0	2048000	25.951	4051.722	1.46E+07	1.00E+08
27:04.0	42.635	0	278528	26.192	4085.445	1.05E+07	1.64E+08
27:05.0	36.868	0	450560	25.155	3940.246	5.94E+07	2.24E+08
27:06.0	45.965	0	344064	24.663	3871.41	1.30E+08	2.12E+08
27:07.0	84.328	0	487424	26.677	4153.914	2.27E+08	2.32E+08
27:08.0	71.284	0	233472	25.183	3944.57	1.21E+08	1.48E+08
27:09.0	69.367	0	294912	29.563	4558.574	1.70E+08	2.09E+08
27:10.0	65.194	0	937984	34.166	5203.82	6.54E+07	1.65E+08
27:11.0	62.907	0	1089536	27.504	4269.996	1.09E+08	1.83E+08
27:12.0	49.222	0	421888	29.278	4518.867	1.44E+08	2.71E+08
27:13.0	57.512	0	106496	31.342	4808.261	1.68E+07	1.42E+08
27:14.0	39.313	0	557056	31.125	4777.906	7728411	1.45E+08
27:15.0	44.191	0	745472	29.068	4489.656	7093751	1.28E+08
27:16.0	27.979	0	376832	27.683	4295.57	424935	1.79E+08
27:17.0	40.979	0	487424	25.09	3932.242	1.82E+07	1.24E+08
27:18.0	63.775	0	430080	21.463	3424.101	2.20E+08	2.47E+08
27:19.0	86.318	0	450560	23.006	3640.441	3.01E+08	3.01E+08
27:20.0	81.704	0	159744	21.158	3381.605	8.73E+07	1.00E+08
27:21.0	66.161	0	143360	25.402	3976.5	2.75E+07	7.70E+07
27:22.0	49.226	0	274432	28.367	4392.097	8983836	1.49E+08
27:23.0	40.206	0	331776	29.29	4521.597	1.19E+08	2.51E+08
27:24.0	43.969	0	356352	28.919	4469.726	1.43E+08	2.60E+08
27:25.0	64.127	0	581632	28.815	4455.308	1.48E+08	1.97E+08
27:26.0	71.536	0	237568	27.662	4293.835	8.80E+07	1.16E+08
27:27.0	83.291	0	1454080	24.181	3805.894	1.69E+08	1.77E+08
27:37.0	52.487	0	1503232	18.302	2981.773	1.26E+08	2.43E+08
27:38.0	59.844	0	360448	17.589	2881.941	4.65E+07	1.43E+08
27:39.0	54.822	0	286720	22.515	3572.5	8417758	1.02E+08
27:40.0	38.888	0	471040	22.749	3605.488	5147587	1.60E+08
27:41.0	31.165	0	430080	22.749	3605.699	1476420	2.05E+08
27:42.0	44	0	516096	20.517	3293.031	7.26E+07	1.52E+08
27:43.0	67.164	0	483328	21.66	3453.437	1.21E+08	1.83E+08
27:44.0	49.36	0	425984	22.729	3603.453	1.46E+08	2.50E+08
27:45.0	80.25	0	610304	26.259	4098.371	2.45E+08	2.45E+08
27:46.0	66.066	0	237568	26.686	4158.359	6.38E+07	1.35E+08
27:47.0	57.215	0	3399680	29.66	4575.347	4.25E+07	1.31E+08
27:48.0	52.405	0	217088	33.623	5130.859	7.68E+07	1.86E+08
27:49.0	42.298	0	462848	33.22	5074.457	9.90E+07	2.28E+08
27:51.0	68.814	0	528384	36.709	5563.722	1.09E+08	1.37E+08
27:52.0	60.969	0	245760	28.597	4426.742	1.46E+08	2.33E+08
27:53.0	57.326	0	368640	30.205	4652.179	1.23E+08	2.32E+08
27:54.0	62.79	0	610304	33.136	5063.273	5.98E+07	1.21E+08
27:55.0	47.058	0	1150976	31.473	4830.195	8106406	1.10E+08
27:56.0	42.739	0	1290240	32.919	5032.828	1685684	1.54E+08
27:57.0	46.907	0	421888	33.583	5126.054	9.19E+07	2.51E+08
27:58.0	46.192	0	110592	31.453	4827.589	9.29E+07	1.46E+08
27:59.0	48.866	0	356352	30.605	4708.792	2.15E+08	3.26E+08
28:00.0	62.814	0	323584	30.141	4643.917	2.23E+08	2.82E+08
28:01.0	87.064	0	667648	28.184	4369.664	1.27E+07	1.32E+07
28:02.0	59.045	0	757760	34.507	5256.027	2179998	7.31E+07
28:03.0	43.112	0	339968	34.683	5280.734	7.49E+07	2.36E+08
28:04.0	41.481	0	479232	27.97	4339.906	1.10E+08	2.24E+08
28:05.0	48.628	0	405504	28.904	4470.886	4.76E+07	1.57E+08
28:06.0	72.235	0	466944	29.127	4502.25	1.31E+08	1.51E+08
28:07.0	60.957	0	258048	33.447	5107.992	1.84E+08	2.59E+08
28:08.0	60.256	0	282624	33.137	5064.554	1.44E+08	2.32E+08
28:09.0	52.392	0	368640	35.841	5443.652	7.17E+07	1.84E+08
28:10.0	56	0	114688	27.352	4253.828	2.30E+07	1.15E+08
28:11.0	57.031	0	319488	29.574	4565.371	8.06E+07	1.98E+08
28:12.0	34.065	0	1236992	30.944	4757.378	2460758	1.49E+08
28:13.0	56.122	0	1208320	32.662	4998.32	8.72E+07	1.90E+08
28:14.0	50.636	0	253952	28.376	4397.578	9.92E+07	2.07E+08
28:15.0	52.207	0	360448	22.473	3570.21	5.52E+07	1.51E+08
28:16.0	54.846	0	495616	18.733	3046.089	9715467	9.95E+07
28:17.0	63.5	0	331776	21.568	3443.539	2.18E+08	2.89E+08
28:18.0	61.786	0	409600	23.233	3677.183	2.14E+08	3.10E+08

28:19.0	67.661	0	126976	19.034	3088.601	1.42E+08	1.85E+08
28:20.0	77.53	0	311296	23.168	3668.082	6.17E+07	6.45E+07
28:21.0	58.247	0	253952	31.495	4835.316	9637122	8.00E+07
28:22.0	51.428	0	348160	25.68	4020.265	8.15E+07	2.17E+08
28:23.0	39.432	0	405504	25.084	3936.949	1.07E+08	2.79E+08
28:24.0	62.779	0	167936	22.858	3624.988	1.11E+08	1.13E+08
28:25.0	59.846	0	1585152	25.888	4049.765	1.11E+08	1.76E+08
28:26.0	47.286	0	1687552	28.08	4357.023	1.24E+08	2.32E+08
28:27.0	43.142	0	352256	28.733	4448.781	6.52E+07	1.60E+08
28:28.0	55.128	0	417792	30.748	4731.32	2.16E+07	1.49E+08
28:29.0	45.888	0	372736	27.975	4342.761	2821766	1.67E+08
28:30.0	25.831	0	421888	27.973	4342.585	472435	2.05E+08
28:31.0	45.843	0	544768	25.397	3981.664	2.75E+07	1.04E+08
28:32.0	59.701	0	491520	24.218	3816.636	1.15E+08	1.57E+08
28:33.0	71.065	0	258048	21.241	3399.546	1.99E+08	2.14E+08
28:34.0	75.621	0	507904	23.726	3747.992	3.15E+08	3.29E+08
28:35.0	71.078	0	417792	26.41	4124.375	2.71E+08	2.71E+08
28:36.0	77.556	0	1847296	26.525	4140.593	52	152
28:37.0	62.729	0	1880064	34.332	5234.917	5.39E+07	1.60E+08
28:38.0	60.937	0	167936	30.794	4739.035	3.49E+07	1.42E+08
28:39.0	49.494	0	335872	32.592	4991.207	6.49E+07	1.75E+08
28:40.0	42.337	0	495616	29.195	4514.996	5133819	1.21E+08
28:41.0	44.086	0	405504	27.601	4291.589	4581346	1.68E+08
28:42.0	51.733	0	417792	21.808	3479.734	2.98E+07	1.60E+08
28:43.0	44.191	0	233472	19.765	3193.558	9.59E+07	2.35E+08
28:44.0	45.979	0	393216	17.851	2925.308	1.02E+08	2.52E+08
28:45.0	62.162	0	1605632	18.007	2948.355	1.16E+08	1.31E+08
28:46.0	75.25	0	475136	22.368	3559.808	1.93E+08	2.07E+08
28:47.0	60.5	0	274432	18.656	3039.554	9.64E+07	1.83E+08
28:48.0	60.794	0	299008	21.009	3369.562	1.22E+08	1.99E+08
28:49.0	61.809	0	1499136	24.376	3841.562	7.16E+07	1.44E+08
28:50.0	64.136	0	1572864	28.086	4361.632	5.64E+07	1.69E+08
28:51.0	56.532	0	315392	30.687	4726.285	7.22E+07	1.70E+08
28:52.0	38.167	0	303104	31.577	4851.136	9.35E+07	2.03E+08
28:53.0	42.71	0	651264	32.984	5048.527	3.56E+07	2.03E+08
28:54.0	44.897	0	442368	31.009	4771.91	8419017	1.26E+08
28:55.0	52.755	0	548864	29.737	4593.73	2.01E+07	1.40E+08
28:56.0	48.507	0	589824	25.91	4057.46	1.18E+08	2.08E+08
28:57.0	71.212	0	212992	27.089	4222.941	2.74E+08	2.87E+08
28:58.0	81.637	0	413696	30.152	4652.332	2.16E+08	2.19E+08
28:59.0	66.579	0	266240	20.276	3268.167	3.97E+07	1.10E+08
29:00.0	58.397	0	274432	21.272	3407.742	1.07E+08	2.10E+08
29:01.0	51.732	0	1536000	21.638	3459.195	6.54E+07	1.51E+08
29:02.0	65.989	0	1581056	24.58	3871.601	9.47E+07	1.48E+08
29:03.0	59.897	0	143360	28.258	4387.171	7.87E+07	1.91E+08
29:04.0	51.02	0	258048	28.194	4378.273	6.57E+07	1.95E+08
29:05.0	40.151	0	385024	30.382	4684.941	4014397	9.61E+07
29:06.0	49.871	0	315392	30.49	4700.195	1.76E+07	1.68E+08
29:07.0	39.746	0	438272	29.515	4563.648	1.11E+07	1.64E+08
29:08.0	54.881	0	454656	28.924	4480.925	8.01E+07	1.86E+08
29:09.0	78.999	0	499712	26.705	4170.058	2.99E+08	3.11E+08
29:10.0	75.696	0	167936	28.943	4483.917	2.28E+08	2.44E+08
29:11.0	92.982	0	270336	23.372	3703.089	1.93E+08	1.93E+08
29:12.0	70.351	0	217088	26.651	4162.734	1.50E+08	2.08E+08
29:13.0	55.67	0	1970176	29.101	4506.269	461659	9.88E+07
29:14.0	48.724	0	237568	33.182	5078.351	1.25E+07	1.58E+08
29:15.0	37.717	0	294912	33.895	5178.316	8.14E+07	2.19E+08
29:16.0	49.481	0	167936	32.545	4989.156	1.03E+07	1.52E+08
29:17.0	60.206	0	425984	27.19	4238.566	5.30E+07	1.27E+08
29:18.0	54.102	0	401408	28.69	4448.816	1.09E+08	2.11E+08
29:19.0	41.396	0	421888	29.434	4553.304	7.27E+07	1.93E+08
29:20.0	67.493	0	192512	26.241	4105.871	1.23E+08	1.83E+08
29:21.0	50.379	0	471040	26.57	4151.992	8.34E+07	2.08E+08
29:22.0	56.01	0	487424	23.326	3697.523	1.38E+07	1.19E+08
29:23.0	57.908	0	507904	21.858	3491.851	2.93E+07	1.36E+08
29:24.0	54	0	389120	22.469	3577.816	1.34E+08	2.30E+08
29:25.0	42.13	0	454656	18.3	2993.476	1.45E+08	2.43E+08
29:26.0	72.019	0	540672	19.051	3098.914	2.82E+08	2.96E+08
29:27.0	85.467	0	528384	22.844	3630.847	2.15E+08	2.15E+08
29:28.0	83.541	0	249856	31.126	4791.867	3.78E+07	6.54E+07
29:29.0	57.286	0	1953792	26.475	4140.027	8337831	9.77E+07
29:30.0	32.558	0	2072576	28.471	4419.835	1411209	1.79E+08
29:31.0	27.157	0	540672	28.468	4419.457	478905	2.22E+08
29:32.0	25.954	0	380928	25.694	4030.972	385406	1.99E+08
29:33.0	39.603	0	331776	18.76	3059.046	3.54E+07	1.33E+08
29:34.0	73.618	0	335872	16.916	2800.71	1.16E+08	1.33E+08
29:35.0	64.95	0	389120	12.134	2130.554	1.55E+08	1.83E+08
29:36.0	77.722	0	438272	14.97	2528.144	2.80E+08	2.80E+08
29:37.0	74.193	0	393216	20.714	3333.394	1.80E+08	1.95E+08
29:38.0	58.585	0	159744	24.079	3805.304	3.48E+07	1.37E+08
29:39.0	48.533	0	270336	24.306	3837.183	9.67E+07	1.94E+08
29:40.0	58.734	0	3072000	28.171	4378.902	9.62E+07	1.67E+08
29:41.0	53.499	0	339968	29.506	4566.117	9.39E+07	2.01E+08
29:42.0	64.194	0	94208	26.071	4084.679	7993045	5.68E+07
29:43.0	55.968	0	221184	28.077	4365.929	6807035	1.11E+08
29:44.0	31.168	0	475136	26.714	4174.937	607860	2.08E+08
29:45.0	28.461	0	528384	26.732	4177.66	462841	1.66E+08
29:46.0	40.656	0	589824	24.072	3804.984	4.24E+07	1.63E+08
29:47.0	66.582	0	520192	21.247	3409.171	1.66E+08	1.94E+08
29:48.0	85.101	0	401408	23.495	3724.46	2.80E+08	2.82E+08
29:49.0	76.07	0	360448	26.586	4157.851	1.69E+08	1.86E+08
29:50.0	71.465	0	409600	27.689	4312.585	9.44E+07	1.57E+08
29:51.0	53.132	0	430080	28.5	4426.417	6.78E+07	1.87E+08
29:52.0	48.362	0	434176	28.981	4494.011	1.55E+08	3.00E+08
29:53.0	47.692	0	327680	28.151	4377.875	7.88E+07	1.98E+08
29:54.0	67.183	0	282624	29.447	4559.574	1.20E+07	8.61E+07
29:55.0	50.773	0	389120	27.27	4254.589	9652845	1.22E+08
29:56.0	30.521	0	376832	24.64	3886.011	369491	1.76E+08
29:57.0	39.331	0	413696	24.981	3934	6.93E+07	1.79E+08
29:58.0	50.253	0	385024	20.67	3329.843	8.32E+07	1.80E+08
29:59.0	68.461	0	540672	17.679	2910.792	1.95E+08	2.21E+08
30:00.0	79.753	0	155648	20.772	3344.566	1.60E+08	1.60E+08
30:01.0	59.844	0	282624	24.252	3832.437	9.31E+07	1.87E+08
30:02.0	61.167	0	253952	27.566	4297.023	5.92E+07	1.65E+08
30:03.0	44.416	0	73728	27.691	4314.656	8691891	1.22E+08
30:04.0	46.373	0	253952	29.61	4583.687	5.68E+07	1.71E+08
30:05.0	52.01	0	339968	26.293	4118.765	1.03E+08	1.82E+08
30:06.0	51.288	0	339968	27.566	4297.238	1.57E+08	2.66E+08
30:07.0	64.963	0	331776	23.526	3731.042	1.01E+08	1.40E+08
30:08.0	70.229	0	303104	23.888	3781.867	1.45E+08	1.84E+08
30:09.0	62.5	0	1814528	25.752	4043.218	1.29E+08	2.32E+08
30:10.0	60.15	0	1880064	26.691	4174.847	5.03E+07	1.49E+08
30:11.0	59.199	0	311296	23.044	3663.667	1.10E+08	2.18E+08
30:12.0	50.864	0	323584	22.848	3636.394	2.02E+07	1.28E+08
30:13.0	40.871	0	495616	24.465	3863.14	2537134	1.64E+08
30:14.0	50.773	0	450560	24.754	3903.777	2.77E+07	1.63E+08
30:15.0	43.718	0	532480	19.249	3132.371	6.15E+07	1.64E+08
30:16.0	75.862	0	258048	16.501	2747.25	2.36E+08	2.40E+08
30:17.0	77.551	0	1523712	15.257	2573.855	2.74E+08	2.88E+08
30:18.0	78.75	0	557056	18.851	3077.882	2.24E+08	2.24E+08
30:19.0	75.324	0	417792	23.232	3692.183	5.00E+07	1.06E+08
30:20.0	59.137	0	491520	29.333	4547.496	7.65E+07	1.87E+08
30:21.0	53.076	0	466944	29.771	4609.117	1.02E+08	2.15E+08
30:22.0	60.957	0	405504	32.345	4969.988	2.09E+07	1.13E+08
30:23.0	48.979	0	376832	32.133	4940.375	1.04E+07	1.50E+08
30:24.0	39.432	0	487424	32.452	4985.292	3.87E+07	1.91E+08
30:25.0	39.795	0	548864	31.854	4901.648	1.05E+08	2.41E+08
30:26.0	54.936	0	512000	31.52	4854.98	1.46E+08	2.05E+08
30:27.0	65.482	0	360448	33.325	5108.156	5.03E+07	1.20E+08
30:28.0	50.645	0	425984	32.191	4949.289	8.89E+07	2.18E+08
30:29.0	51.57	0	462848	32.734	5025.609	3.33E+07	1.66E+08
30:30.0	44.25	0	520192	30.099	4656.382	6101630	1.17E+08
30:31.0	41.351	0	286720	29.282	4541.992	1.55E+07	1.85E+08
30:32.0	75.765	0	331776	29.543	4578.707	2.45E+08	2.62E+08
30:33.0	77.722	0	274432	29.504	4573.335	3.02E+08	3.18E+08
30:34.0	85.607	0	360448	31.762	4889.91	2.65E+08	2.65E+08
30:35.0	79.852	0	397312	27.039	4228.031	1.42E+08	1.42E+08
30:36.0	64.175	0	159744	27.724	4324.132	2.17E+07	9.43E+07
30:37.0	40.759	0	1961984	31.668	4876.906	353757	1.38E+08
30:38.0	45.758	0	1970176	32.687	5019.808	1.60E+07	1.57E+08
30:39.0	45.679	0	221184	28.131	4381.234	5.09E+07	1.81E+08
30:40.0	57.662	0	421888	25.892	4067.453	9.23E+07	1.98E+08

30:41.0	74.626	0	81920	28.124	4380.378	1.08E+08	1.59E+08
30:42.0	54.404	0	352256	29.348	4551.847	5.03E+07	1.92E+08
30:43.0	51	0	188416	28.064	4372.011	4.21E+07	1.12E+08
30:44.0	41.396	0	462848	27.256	4258.933	9.72E+07	2.43E+08
30:45.0	40.954	0	512000	28.018	4365.859	6.23E+07	2.10E+08
30:46.0	55.357	0	790528	30.816	4758.175	1.89E+07	5.92E+07
30:47.0	54.773	0	864256	23.492	3731.496	1.55E+08	2.09E+08
30:48.0	48.329	0	376832	22.729	3624.703	1.59E+08	2.86E+08
30:49.0	56.923	0	405504	24.316	3847.187	8.15E+07	1.80E+08
30:50.0	71.611	0	532480	22.884	3646.664	4.83E+07	6.80E+07
30:51.0	53.763	0	471040	26.712	4183.437	8.91E+07	2.19E+08
30:52.0	51.322	0	446464	26.19	4110.441	1.03E+08	2.39E+08
30:53.0	56.521	0	413696	28.015	4366.414	1.89E+07	9.38E+07
30:54.0	48.302	0	532480	26.041	4089.792	3.91E+07	1.61E+08
30:55.0	51.052	0	512000	26.663	4177.289	7.82E+07	2.00E+08
30:56.0	58.247	0	638976	26.459	4148.765	1.76E+08	2.35E+08
30:57.0	75.373	0	483328	28.67	4458.832	2.11E+08	2.27E+08
30:58.0	81.173	0	483328	33.967	5201.48	2.41E+08	2.48E+08
30:59.0	73.737	0	98304	26.634	4173.886	2.79E+07	9.04E+07
31:00.0	58.145	0	389120	30.17	4669.527	7.19E+07	1.86E+08
31:01.0	45.698	0	1617920	32.696	5023.683	7783945	1.49E+08
31:02.0	52.04	0	1732608	32.895	5051.574	7376864	1.47E+08
31:03.0	33.798	0	532480	31.521	4859.117	733216	2.02E+08
31:04.0	46.7	0	602112	27.828	4341.738	8.07E+07	1.16E+08
31:05.0	66.169	0	540672	26.89	4210.324	2.02E+08	2.51E+08
31:06.0	56.675	0	335872	26.532	4160.347	1.28E+08	2.20E+08
31:07.0	74.811	0	385024	25.38	3998.941	7.43E+07	9.22E+07
31:08.0	64.781	0	417792	24.921	3934.714	1.00E+08	1.80E+08
31:09.0	48.395	0	307200	25.566	4025.273	1.55E+08	2.89E+08
31:10.0	58.269	0	335872	25.986	4084.253	9.23E+07	2.17E+08
31:11.0	56.632	0	385024	30.099	4660.855	3.73E+07	1.34E+08
31:12.0	53.141	0	495616	33.466	5132.921	5637233	1.48E+08
31:13.0	53.67	0	458752	31.248	4822.207	9.64E+07	1.89E+08
31:14.0	54.75	0	389120	27.802	4339.257	1.56E+08	2.51E+08
31:15.0	58.711	0	544768	27.169	4250.667	1.75E+08	2.36E+08
31:16.0	84.079	0	430080	30.201	4675.82	9.46E+07	1.08E+08
31:17.0	74.874	0	409600	32.644	5018.414	1.45E+08	1.78E+08
31:18.0	54.936	0	172032	28.954	4501.371	1.07E+08	2.01E+08
31:19.0	50.502	0	516096	24.578	3888.027	1.17E+08	2.38E+08
31:20.0	47.029	0	294912	25.448	4009.984	1.27E+08	2.38E+08
31:21.0	59.007	0	147456	25.028	3951.242	5.05E+07	1.39E+08
31:22.0	68.324	0	307200	31.122	4805.378	5735004	9.80E+07
31:23.0	48.101	0	376832	30.384	4701.96	1.68E+07	1.44E+08
31:24.0	31.818	0	487424	30.519	4721.039	1355966	1.79E+08
31:25.0	52.319	0	659456	27.782	4337.574	1.28E+08	1.65E+08
31:26.0	60	0	475136	27.899	4354.152	2.53E+08	2.86E+08
31:27.0	83.037	0	221184	25.67	4041.882	1.77E+08	2.08E+08
31:28.0	66.412	0	364544	29.974	4645.23	7.21E+07	1.59E+08
31:29.0	58.115	0	192512	28.286	4408.789	1.27E+07	1.11E+08
31:30.0	49.875	0	204800	32.235	4962.371	1.13E+07	1.35E+08
31:31.0	34.177	0	339968	32.196	4956.863	3205976	1.55E+08
31:32.0	42.705	0	425984	29.724	4610.585	1.95E+07	1.50E+08
31:33.0	41.025	0	458752	25.12	3965.332	1.52E+07	1.56E+08
31:34.0	62.814	0	630784	24.703	3907.082	1.66E+08	2.10E+08
31:35.0	68.844	0	442368	21.997	3527.976	2.65E+08	2.84E+08
31:36.0	87.25	0	495616	24.947	3941.617	3.54E+08	3.54E+08
31:37.0	91.708	0	167936	19.965	3243.394	1.53E+08	1.53E+08
31:38.0	82.663	0	131072	26.582	4171.003	1.42E+07	4.99E+07
31:39.0	61.734	0	147456	34.204	5239.453	5.68E+07	1.46E+08
31:40.0	44.25	0	192512	35.204	5379.628	2.34E+07	1.66E+08
31:41.0	47.969	0	360448	34.885	5335.042	8897509	1.44E+08
31:42.0	35.839	0	1499136	33.715	5172.222	5638352	2.00E+08
31:43.0	31.233	0	454656	32.337	4979.226	526221	1.80E+08
31:44.0	35.786	0	499712	31.013	4793.89	7709958	1.91E+08
31:45.0	51.507	0	634880	30.15	4673.082	4.39E+07	1.05E+08
31:46.0	71.536	0	401408	31.478	4859.429	1.47E+08	1.80E+08
31:47.0	69.249	0	393216	33.226	5104.57	2.67E+08	2.90E+08
31:48.0	80.299	0	159744	27.404	4288.707	2.06E+08	2.06E+08
31:49.0	69.773	0	278528	27.186	4258.187	2.02E+08	2.56E+08
31:50.0	64.885	0	1978368	26.087	4104.218	9.15E+07	1.76E+08
31:51.0	69.329	0	2220032	30.145	4673.027	6.39E+07	1.29E+08

31:52.0	61.518	0	405504	30.797	4764.488	6.53E+07	1.42E+08
31:53.0	54.975	0	270336	34.978	5350.667	3802759	9.44E+07
31:54.0	40.86	0	577536	34.528	5287.64	3.73E+07	2.00E+08
31:55.0	40.302	0	651264	34.556	5291.824	9.91E+07	2.65E+08
31:56.0	38.578	0	487424	34.547	5290.75	6.99E+07	2.07E+08
31:57.0	64.438	0	536576	35.511	5425.98	1.47E+07	1.17E+08
31:58.0	47.959	0	393216	30.837	4771.023	1.37E+07	1.24E+08
31:59.0	46.153	0	471040	27.1	4247.359	1.05E+08	2.41E+08
32:00.0	63.846	0	376832	26.817	4207.781	1.37E+08	2.08E+08
32:01.0	67.002	0	483328	24.107	3828.097	1.21E+08	1.36E+08
32:02.0	72.405	0	479232	28.111	4389.496	1.75E+08	2.21E+08
32:03.0	56.708	0	208896	24.892	3938.492	3.56E+07	1.28E+08
32:04.0	56.01	0	442368	27.189	4260.57	1.26E+08	2.26E+08
32:05.0	50.895	0	499712	28.382	4427.847	1.50E+08	2.66E+08
32:06.0	62.21	0	57344	25.34	4001.46	8.05E+07	1.45E+08
32:07.0	66.009	0	1896448	28.612	4460.093	7.95E+07	9.67E+07
32:08.0	57.731	0	1994752	31.36	4845.425	5.97E+07	1.77E+08
32:09.0	48.378	0	131072	33.52	5148.234	3.53E+07	1.78E+08
32:10.0	58.684	0	385024	30.336	4701.96	7.94E+07	1.87E+08
32:11.0	64.05	0	352256	28.491	4443.335	1.57E+08	2.05E+08
32:12.0	52.658	0	196608	30.08	4666.203	1.09E+08	2.04E+08
32:13.0	60.15	0	262144	33.509	5146.878	8.88E+07	2.04E+08
32:14.0	64.267	0	147456	34.297	5257.507	5.78E+07	1.32E+08
32:15.0	57.393	0	557056	27.734	4337.57	5.45E+07	1.39E+08
32:16.0	51.851	0	507904	27.921	4363.91	1.45E+08	2.44E+08
32:17.0	54.285	0	450560	22.733	3636.859	7.19E+07	1.93E+08
32:18.0	63.01	0	389120	22.556	3612.148	1.14E+08	1.82E+08
32:19.0	55.725	0	458752	18.604	3058.445	1.41E+08	2.40E+08
32:20.0	61.5	0	356352	19.226	3145.691	7.23E+07	1.71E+08
32:21.0	59.595	0	1306624	23.191	3701.589	5.91E+07	1.21E+08
32:22.0	58.629	0	1359872	23.659	3767.14	1.14E+08	2.15E+08
32:23.0	65.994	0	421888	22.987	3673.046	1.80E+08	2.62E+08
32:24.0	56.115	0	249856	24.614	3901.3	8.81E+07	1.63E+08
32:25.0	58.646	0	393216	28.779	4485.171	403133	7.24E+07
32:26.0	50.529	0	491520	30.868	4778.039	2.79E+07	1.59E+08
32:27.0	45.063	0	409600	31.762	4903.425	9.81E+07	2.50E+08
32:28.0	49.357	0	364544	31.159	4819.125	9.20E+07	2.22E+08
32:29.0	49.234	0	446464	31.737	4900.207	6.21E+07	1.23E+08
32:30.0	70.822	0	245760	32.886	5061.417	1.97E+08	2.42E+08
32:31.0	62.436	0	454656	29.915	4645.003	1.93E+08	2.91E+08
32:32.0	68.316	0	339968	25.394	4011.492	1.17E+08	1.89E+08
32:33.0	75.247	0	294912	30.011	4658.796	5.96E+07	8.66E+07
32:34.0	60.102	0	196608	29.362	4567.89	6.10E+07	1.63E+08
32:35.0	45.792	0	1839104	33.261	5114.414	7896071	1.24E+08
32:36.0	47.619	0	1724416	32.164	4960.746	3.55E+07	1.56E+08
32:37.0	54.521	0	348160	31.585	4879.703	1.41E+08	2.46E+08
32:38.0	50.389	0	348160	32.202	4966.277	1.72E+08	2.93E+08
32:39.0	60.957	0	278528	31.398	4853.671	7.51E+07	1.09E+08
32:40.0	60.88	0	471040	32.768	5045.738	1.18E+08	1.59E+08
32:41.0	57.853	0	634880	29.81	4631.16	6.44E+07	1.83E+08
32:42.0	50.507	0	311296	26.138	4116.441	9630417	1.09E+08
32:43.0	43.734	0	487424	27.893	4362.531	1.91E+07	1.48E+08
32:44.0	38.363	0	495616	26.646	4188.011	7672085	1.86E+08
32:45.0	51.57	0	573440	24.846	3935.8	1.06E+08	2.05E+08
32:46.0	74.32	0	446464	18.204	3004.953	1.93E+08	2.09E+08
32:47.0	74.559	0	462848	15.33	2602.156	2.45E+08	2.67E+08
32:48.0	88.972	0	507904	20.461	3321.574	2.22E+08	2.22E+08
32:49.0	68.527	0	278528	24.524	3891.21	1.33E+08	1.99E+08
32:50.0	61.125	0	1409024	24.418	3876.41	4.50E+07	1.37E+08
32:51.0	58.717	0	1241088	28.363	4429.468	5.92E+07	1.48E+08
32:52.0	53.723	0	380928	29.303	4561.257	5.68E+07	1.98E+08
32:53.0	46.173	0	458752	30.229	4691.253	6269500	1.48E+08
32:54.0	31.111	0	610304	31.617	4885.89	2099861	1.85E+08
32:56.0	36.34	0	479232	27.79	4349.742	1.24E+07	1.56E+08
32:57.0	63.613	0	360448	22.593	3621.347	9.32E+07	1.32E+08
32:58.0	53.689	0	143360	19.31	3161.207	1.03E+08	2.14E+08
32:59.0	59.749	0	438272	15.531	2631.578	1.74E+08	2.28E+08
33:00.0	85.606	0	409600	18.516	3050.167	2.34E+08	2.37E+08
33:01.0	58.481	0	253952	20.47	3324.23	5.46E+07	1.43E+08
33:02.0	59.838	0	389120	25.158	3981.39	4741287	1.19E+08
33:03.0	49.476	0	376832	26.535	4174.445	7.93E+07	2.33E+08

33:04.0	43.147	0	286720	27.778	4348.89	1.65E+08	3.16E+08
33:05.0	45.965	0	475136	27.823	4355.328	1.29E+08	1.98E+08
33:06.0	77.99	0	413696	31.283	4840.367	6.12E+07	6.12E+07
33:07.0	52.295	0	372736	25.752	4065.117	4.80E+07	1.50E+08
33:08.0	46.428	0	454656	24.576	3900.382	9.41E+07	2.38E+08
33:09.0	38.422	0	495616	24.246	3854.257	6.88E+07	2.31E+08
33:10.0	64.948	0	532480	26.341	4148.195	5.29E+07	1.16E+08
33:11.0	56.377	0	380928	27.324	4286.164	1.20E+08	2.10E+08
33:12.0	50.252	0	1531904	28.454	4445.523	1.14E+08	2.27E+08
33:13.0	64.556	0	458752	28.859	4502.519	1.66E+08	2.10E+08
33:14.0	72.319	0	385024	30.683	4758.363	1.64E+08	2.04E+08
33:15.0	56.777	0	458752	30.265	4699.972	1.37E+08	2.24E+08
33:16.0	56.079	0	389120	34.025	5227.07	1.40E+08	2.11E+08
33:17.0	66.499	0	155648	30.015	4665.148	9.00E+07	1.76E+08
33:18.0	66.169	0	217088	31.986	4941.46	1.04E+08	2.02E+08
33:19.0	56.171	0	1781760	33.743	5187.808	8542192	1.09E+08
33:20.0	51.908	0	1376256	35.157	5385.988	4281266	1.12E+08
33:21.0	37.313	0	409600	34.105	5238.57	1.62E+07	1.63E+08
33:22.0	50.663	0	483328	34.29	5264.617	3.88E+07	1.92E+08
33:23.0	49.238	0	405504	27.206	4271.921	1.00E+08	2.18E+08
33:24.0	71.105	0	389120	23.765	3789.707	1.95E+08	2.49E+08
33:25.0	69.135	0	561152	25.295	4004.308	1.31E+08	1.56E+08
33:26.0	70.37	0	577536	29.666	4617.261	1.33E+08	1.43E+08
33:27.0	57.068	0	360448	33.115	5100.757	1.13E+08	2.17E+08
33:28.0	65.92	0	487424	31.025	4808.105	1.50E+08	2.48E+08
33:29.0	61.298	0	442368	29.821	4639.417	1.34E+08	2.31E+08
33:30.0	54.95	0	442368	27.908	4371.414	8.02E+07	1.87E+08
33:31.0	65.482	0	352256	29.506	4595.531	4758784	9.24E+07
33:32.0	53.708	0	274432	28.899	4510.539	8278069	1.36E+08
33:33.0	42.25	0	323584	29.182	4550.402	1.12E+07	2.02E+08
33:34.0	37.433	0	266240	26.888	4228.949	6.75E+07	2.20E+08
33:35.0	64.871	0	262144	21.036	3408.66	7.53E+07	1.36E+08
33:36.0	61.691	0	372736	21.788	3514.269	1.30E+08	1.67E+08
33:37.0	79	0	122880	24.141	3844.031	2.95E+08	2.98E+08
33:38.0	86.034	0	2080768	29.314	4569.269	2.69E+08	2.73E+08
33:39.0	81.565	0	1675264	32.944	5078.027	2.68E+07	5.83E+07
33:40.0	64.321	0	282624	32.502	5016.222	2355495	6.44E+07
33:41.0	28.86	0	602112	34.652	5317.816	444415	1.93E+08
33:42.0	35.602	0	598016	33.373	5138.824	1.13E+07	1.92E+08
33:43.0	33.593	0	479232	30.909	4793.562	1.25E+07	1.73E+08
33:44.0	58.461	0	413696	31.554	4884.156	1.70E+08	2.35E+08
33:45.0	59.605	0	331776	32.34	4994.332	1.09E+08	1.96E+08
33:46.0	50.248	0	241664	26.611	4191.273	1.21E+08	1.99E+08
33:47.0	76.485	0	290816	26.273	4143.972	1.55E+08	1.55E+08
33:48.0	59.398	0	1277952	27.815	4360.113	1.28E+08	2.04E+08
33:49.0	59.848	0	1064960	29.43	4586.625	8.17E+07	1.63E+08
33:50.0	50.487	0	225280	33.194	5114.355	8.37E+07	1.99E+08
33:51.0	47.236	0	385024	33.3	5129.281	1.80E+07	1.43E+08
33:52.0	44.02	0	335872	30.647	4757.417	6508525	1.57E+08
33:53.0	42.929	0	339968	27.805	4359.183	5.09E+07	1.74E+08
33:54.0	56.997	0	118784	28.506	4457.554	1.14E+08	2.23E+08
33:55.0	45.925	0	577536	27.989	4385.148	1.45E+08	2.32E+08
33:56.0	73.762	0	425984	29.344	4575.257	2.80E+08	2.84E+08
33:57.0	78.358	0	258048	28.705	4485.687	7.06E+07	8.82E+07
33:58.0	72.361	0	376832	29.134	4545.914	1.11E+08	1.40E+08
33:59.0	58.928	0	1060864	30.388	4721.75	1.27E+08	2.37E+08
34:00.0	63.065	0	1138688	29.774	4635.695	1.14E+08	1.76E+08
34:01.0	62.857	0	303104	33.839	5205.648	6.84E+07	1.62E+08
34:02.0	51.385	0	352256	34.926	5358.078	8797292	1.49E+08
34:03.0	33.333	0	356352	34.267	5265.89	696051	1.70E+08
34:04.0	30.053	0	561152	32.888	5072.773	644368	1.96E+08
34:05.0	31.331	0	638976	31.508	4879.546	579654	1.82E+08
34:06.0	54.135	0	569344	30.712	4768.011	4.34E+07	8.11E+07
34:07.0	64.735	0	401408	32.297	4990.406	1.99E+08	2.44E+08
34:08.0	89.303	0	372736	34.255	5265.019	3.41E+08	3.44E+08
34:09.0	86.352	0	335872	32.18	4974.3	2.76E+08	2.76E+08
34:10.0	88.585	0	94208	26.828	4224.097	1.88E+07	1.88E+07
34:11.0	74.43	0	172032	32.475	5015.585	1447484	4.21E+07
34:12.0	33.165	0	475136	29.013	4530.476	434999	1.79E+08
34:13.0	26.835	0	512000	29.012	4530.593	446635	2.13E+08
34:14.0	27.363	0	360448	27.664	4341.707	5130683	2.20E+08

34:15.0	37	0	446464	24.901	3954.57	8547103	1.64E+08
34:16.0	60.148	0	413696	24.408	3885.664	1.18E+08	1.49E+08
34:17.0	76.884	0	319488	19.896	3253.324	2.43E+08	2.61E+08
34:18.0	72.795	0	393216	20.133	3286.57	2.60E+08	2.78E+08
34:19.0	76.485	0	532480	26.062	4117.925	6.33E+07	6.33E+07
34:20.0	62.532	0	430080	27.977	4386.488	5.28E+07	1.47E+08
34:21.0	40.102	0	565248	25.619	4056.148	1.00E+08	2.73E+08
34:22.0	44.999	0	434176	24.479	3896.535	2.15E+07	1.48E+08
34:23.0	50.391	0	544768	26.113	4125.726	9338337	1.63E+08
34:24.0	42.346	0	507904	24.996	3969.312	1.55E+07	1.46E+08
34:25.0	46.578	0	565248	22.796	3661.085	8.20E+07	2.34E+08
34:26.0	72.682	0	540672	21.55	3486.601	2.11E+08	2.13E+08
34:27.0	63.383	0	348160	18.707	3088.164	1.66E+08	2.02E+08
34:28.0	78.054	0	3129344	21.232	3442.144	1.71E+08	1.71E+08
34:29.0	69.696	0	503808	28.755	4496.718	7.93E+07	1.19E+08
34:30.0	53.435	0	290816	31.488	4879.835	8.65E+07	1.98E+08
34:31.0	43.609	0	253952	30.807	4784.539	7.23E+07	1.88E+08
34:32.0	52.727	0	438272	32.009	4953.093	491613	1.68E+08
34:33.0	30.05	0	585728	29.272	4569.609	379510	1.76E+08
34:34.0	56.887	0	409600	25.623	4058.265	1.06E+08	2.29E+08
34:35.0	58.476	0	278528	26.597	4194.957	9.06E+07	1.58E+08
34:36.0	61.042	0	389120	21.988	3548.921	9.44E+07	1.42E+08
34:37.0	71.393	0	245760	19.438	3191.566	1.18E+08	1.45E+08
34:38.0	71.246	0	249856	23.873	3813.292	2.29E+08	2.87E+08
34:39.0	68.367	0	163840	25.74	4075.054	1.52E+08	2.32E+08
34:40.0	64.705	0	1380352	22.997	3691.742	1.47E+08	2.19E+08
34:41.0	54.102	0	192512	25.484	4040.386	1.13E+08	2.01E+08
34:42.0	56.955	0	2625536	29.878	4656.261	648007	9.45E+07
34:43.0	58.181	0	434176	31.847	4932.464	5.49E+07	1.33E+08
34:44.0	58.947	0	483328	33.313	5138.05	1.89E+07	1.32E+08
34:45.0	50.67	0	565248	32.66	5046.734	2952521	1.46E+08
34:46.0	37.325	0	540672	31.338	4861.582	5668295	1.61E+08
34:47.0	44.743	0	442368	30.257	4710.195	5.60E+07	2.10E+08
34:48.0	71.212	0	356352	26.089	4126.128	2.13E+08	2.34E+08
34:49.0	73.449	0	208896	27.704	4352.539	2.08E+08	2.29E+08
34:50.0	75.369	0	241664	21.267	3450.359	1.77E+08	1.95E+08
34:51.0	71.929	0	151552	26.522	4186.945	1.68E+08	1.89E+08
34:52.0	65.306	0	151552	25.501	4043.921	8.38E+07	1.76E+08
34:53.0	62.82	0	253952	27.54	4329.796	6.92E+07	1.82E+08
34:54.0	56.926	0	233472	29.279	4573.527	7.73E+07	1.80E+08
34:55.0	63.565	0	278528	31.684	4910.804	3.09E+07	1.20E+08
34:56.0	55.699	0	405504	31.645	4905.433	4.92E+07	1.07E+08
34:57.0	44	0	397312	32.492	5024.378	1.08E+08	2.63E+08
34:58.0	43.734	0	196608	30.428	4735.046	4.20E+07	1.66E+08
34:59.0	47.727	0	131072	27.581	4336.078	5.21E+07	1.48E+08
35:00.0	56.476	0	3719168	30.994	4814.503	7.75E+07	1.92E+08
35:01.0	44.332	0	360448	26.062	4123.402	8.69E+07	2.04E+08
35:02.0	50.389	0	495616	26.996	4254.511	4.59E+07	1.77E+08
35:03.0	63.037	0	536576	27.002	4255.5	9.55E+07	1.49E+08
35:04.0	57.692	0	548864	29.334	4582.535	1.27E+08	2.55E+08
35:05.0	60.814	0	663552	25.734	4078.218	9119903	7.99E+07
35:06.0	52.487	0	602112	29.179	4561.125	8.06E+07	1.62E+08
35:07.0	54.617	0	495616	28.73	4498.511	1.28E+08	2.63E+08
35:08.0	60.406	0	188416	20.068	3284.253	6.86E+07	9.96E+07
35:09.0	69.674	0	3338240	21.72	3515.949	1.69E+08	1.87E+08
35:10.0	60.814	0	266240	25.66	4068.3	1.34E+08	2.08E+08
35:11.0	58.247	0	323584	28.864	4517.433	1.14E+08	2.32E+08
35:12.0	55.958	0	458752	23.756	3801.664	8.00E+07	1.81E+08
35:13.0	51.421	0	536576	25.942	4108.273	8.53E+07	2.13E+08
35:14.0	45.108	0	561152	27.34	4304.332	4573811	1.47E+08
35:15.0	60.629	0	581632	28.806	4510.113	1.92E+07	1.34E+08
35:16.0	40.414	0	495616	27.859	4377.414	1.49E+07	1.50E+08
35:17.0	62.094	0	339968	28.634	4486.195	1.02E+08	1.26E+08
35:18.0	67.669	0	331776	21.507	3487.332	1.96E+08	2.34E+08
35:19.0	85.213	0	274432	24.257	3872.812	3.58E+08	3.58E+08
35:20.0	90.523	0	270336	27.57	4337.367	2.33E+08	2.33E+08
35:21.0	89.473	0	352256	29.325	4583.449	8.99E+07	9.59E+07
35:22.0	58.62	0	155648	32.482	5025.933	604345	1.17E+08
35:23.0	43.157	0	253952	32.534	5033.359	3.30E+07	1.51E+08
35:24.0	41.353	0	278528	30.788	4788.64	5.41E+07	1.64E+08
35:25.0	43.08	0	3260416	27.049	4264.64	622540	1.27E+08

35:26.0	32.697	0	327680	29.883	4661.8	6239005	1.73E+08
35:27.0	44.791	0	413696	29.207	4567.335	1.05E+08	2.40E+08
35:28.0	51.87	0	475136	28.719	4499.042	1.11E+08	1.99E+08
35:29.0	59.24	0	331776	24.991	3976.558	2.53E+07	1.11E+08
35:30.0	55.468	0	536576	19.509	3208.363	7.51E+07	1.92E+08
35:31.0	55.867	0	442368	19.06	3145.593	1.62E+08	2.68E+08
35:32.0	59.287	0	483328	19.268	3174.824	1.54E+08	2.29E+08
35:33.0	69.478	0	458752	23.012	3699.878	4.09E+07	5.28E+07
35:34.0	45.169	0	450560	27.859	4379.324	4.41E+07	1.66E+08
35:35.0	46.5	0	491520	29.017	4541.812	1.53E+08	2.94E+08
35:36.0	46.401	0	438272	28.867	4520.812	1.59E+08	2.58E+08
35:37.0	58.897	0	311296	29.711	4639.285	7.15E+07	1.30E+08
35:38.0	68.367	0	3395584	26.238	4152.496	3.62E+07	8.54E+07
35:39.0	46.134	0	217088	29.695	4637.085	1.01E+08	2.55E+08
35:40.0	42.631	0	499712	29.382	4593.468	7.40E+07	2.24E+08
35:41.0	62.269	0	585728	31.322	4865.503	2.28E+07	8.58E+07
35:42.0	41.145	0	471040	32.016	4963.039	5490923	1.55E+08
35:43.0	48.021	0	421888	30.936	4811.832	2.81E+07	1.07E+08
35:44.0	65.914	0	405504	27.375	4312.785	1.76E+08	2.14E+08
35:45.0	71.215	0	548864	28.533	4475.277	3.33E+08	3.41E+08
35:46.0	76.178	0	290816	27.928	4390.585	1.95E+08	1.95E+08
35:47.0	72.588	0	2506752	28.352	4449.953	1.08E+08	1.37E+08
35:48.0	57.065	0	438272	32.491	5030.253	774377	1.34E+08
35:49.0	49.487	0	552960	33.826	5217.527	1.84E+07	1.64E+08
35:50.0	60.199	0	372736	33.565	5181.078	1.19E+07	1.31E+08
35:51.0	30.989	0	495616	33.726	5203.785	2449194	1.91E+08
35:52.0	38.917	0	487424	27.403	4317.726	5.41E+07	2.08E+08
35:53.0	65.051	0	483328	27.239	4294.8	1.45E+08	2.09E+08
35:54.0	73.697	0	344064	24.238	3874.242	1.17E+08	1.36E+08
35:55.0	71.321	0	569344	26.079	4132.597	1.93E+08	2.06E+08
35:56.0	56.25	0	421888	29.737	4645.406	1.42E+08	2.01E+08
35:57.0	60.574	0	2691072	32.763	5069.636	6.13E+07	1.54E+08
35:58.0	66.752	0	135168	31.061	4831.062	5.45E+07	1.39E+08
35:59.0	52.864	0	323584	33.131	5121.32	1.09E+08	2.17E+08
36:00.0	53.316	0	315392	33.089	5115.539	2.58E+07	1.47E+08
36:01.0	53.589	0	495616	33.614	5189.242	1.67E+07	1.40E+08
36:02.0	42.968	0	495616	32.486	5031.355	5.04E+07	1.94E+08
36:03.0	49.21	0	458752	31.37	4875.011	9.66E+07	2.07E+08
36:04.0	70.812	0	1560576	32.576	5045.292	2.60E+08	2.93E+08
36:05.0	76.119	0	360448	24.997	3982.894	1.21E+08	1.40E+08
36:06.0	73.869	0	479232	26.557	4201.714	1.93E+08	2.07E+08
36:07.0	62.113	0	192512	23.135	3722.144	1.01E+08	1.93E+08
36:08.0	58.375	0	1949696	24.266	3880.714	6.62E+07	1.69E+08
36:09.0	54.75	0	135168	24.257	3879.398	7.36E+07	1.50E+08
36:10.0	60.957	0	200704	25.798	4095.585	9.51E+07	2.10E+08
36:11.0	57.812	0	192512	27.751	4369.328	3.29E+07	1.39E+08
36:12.0	58.51	0	307200	25.601	4068.046	6358228	1.09E+08
36:13.0	40.173	0	720896	26.954	4257.773	834706	1.71E+08
36:14.0	56.994	0	393216	25.688	4080.593	1.16E+08	1.68E+08
36:15.0	47.75	0	466944	22.58	3644.96	1.52E+08	2.41E+08
36:16.0	51.653	0	364544	23.676	3798.726	1.28E+08	2.39E+08
36:17.0	68.905	0	270336	22.75	3668.933	8.13E+07	1.12E+08
36:18.0	78.14	0	1470464	26.493	4193.589	1.85E+08	2.06E+08
36:19.0	58.524	0	106496	28.928	4535.046	1.28E+08	2.24E+08
36:20.0	64.122	0	286720	25.758	4090.73	8.42E+07	1.71E+08
36:21.0	54.591	0	303104	27.329	4311.097	8349609	9.96E+07
36:22.0	51.794	0	389120	29.149	4566.23	1.89E+07	1.26E+08
36:23.0	40.75	0	548864	27.099	4279.035	1.13E+08	2.69E+08
36:24.0	45.526	0	569344	28.184	4431.382	8.30E+07	2.44E+08
36:25.0	58.145	0	675840	30.911	4813.882	2.00E+07	3.64E+07
36:26.0	38.118	0	598016	31.034	4831.273	3.51E+07	1.84E+08
36:27.0	54.06	0	389120	29.197	4574.007	1.35E+08	2.51E+08
36:28.0	60.309	0	315392	18.88	3127.964	1.49E+08	2.27E+08
36:29.0	78.14	0	598016	19.79	3255.589	2.85E+08	2.85E+08
36:30.0	84.039	0	204800	24.862	3966.718	4.35E+07	4.35E+07
36:31.0	61.953	0	282624	28.535	4481.585	4938048	1.00E+08
36:32.0	38.046	0	425984	28.738	4510.136	7229000	1.60E+08
36:33.0	37.234	0	585728	27.724	4368.195	5.62E+07	2.22E+08
36:34.0	49.74	0	503808	22.917	3694.566	1.12E+08	2.50E+08
36:35.0	51.025	0	425984	22.406	3623.062	6.11E+07	1.40E+08
36:36.0	69	0	413696	24.077	3857.429	9.27E+07	1.17E+08

36:37.0	60.256	0	311296	27.691	4364.042	1.36E+08	1.92E+08
36:38.0	64.102	0	233472	25.241	4020.621	1.78E+08	2.59E+08
36:39.0	57.644	0	266240	25.088	3999.343	1.60E+08	2.60E+08
36:40.0	86.75	0	3010560	29.986	4685.914	1.14E+08	1.14E+08
36:41.0	72.681	0	217088	24.054	3854.476	3.29E+07	6.39E+07
36:42.0	52.185	0	397312	25.746	4091.804	1.16E+08	2.41E+08
36:43.0	48.267	0	421888	26.981	4265.027	1.64E+08	3.19E+08
36:44.0	43.17	0	397312	25.794	4098.82	8.39E+07	1.90E+08
36:45.0	70.646	0	532480	30.557	4766.574	1.30E+07	4.66E+07
36:46.0	39.528	0	454656	30.883	4812.425	8083587	1.69E+08
36:47.0	44.01	0	303104	30.307	4731.707	1.08E+08	2.63E+08
36:48.0	54.911	0	299008	29.713	4648.519	9.67E+07	1.72E+08
36:49.0	67.91	0	212992	30.801	4801.195	1.27E+08	1.72E+08
36:50.0	65.648	0	1490944	31.777	4937.957	1.71E+08	2.30E+08
36:51.0	61.636	0	1363968	31.782	4938.683	8.04E+07	1.69E+08
36:52.0	48.786	0	233472	30.763	4795.984	8.28E+07	2.05E+08
36:53.0	42.929	0	344064	32.125	4987.011	452338	1.80E+08
36:54.0	32.323	0	417792	32.025	4973.089	397220	1.83E+08
36:55.0	42.245	0	626688	32.032	4974.32	4.91E+07	1.97E+08
36:56.0	61.67	0	552960	26.311	4172.464	1.26E+08	1.48E+08
36:57.0	62.249	0	376832	28.335	4456.468	1.42E+08	1.81E+08
36:58.0	73.869	0	372736	27.73	4371.761	2.19E+08	2.19E+08
36:59.0	64.857	0	217088	30.519	4762.667	7.09E+07	1.54E+08
37:00.0	60.714	0	249856	30.052	4697.378	5.12E+07	1.45E+08
37:01.0	49.081	0	2772992	31.687	4926.507	7351623	1.22E+08
37:02.0	46.616	0	143360	32.514	5042.496	8868703	1.47E+08
37:03.0	36.723	0	487424	29.751	4655.472	800003	1.80E+08
37:04.0	49.872	0	573440	30.436	4751.664	6.31E+07	1.02E+08
37:05.0	66.585	0	466944	22.742	3673.21	2.33E+08	2.59E+08
37:06.0	80.845	0	614400	24.674	3944.195	3.56E+08	3.56E+08
37:07.0	75.87	0	344064	27.623	4357.683	1.56E+08	1.56E+08
37:08.0	90.523	0	270336	24.934	3980.792	4.94E+07	5.19E+07
37:09.0	56.171	0	135168	30.049	4697.937	1.00E+07	1.10E+08
37:10.0	53.108	0	192512	32.547	5048.125	6451635	1.47E+08
37:11.0	25.628	0	434176	27.69	4367.367	391428	1.89E+08
37:12.0	50.659	0	413696	26.64	4220.355	2.38E+07	1.55E+08
37:13.0	39.686	0	430080	25.547	4067.378	4985180	1.60E+08
37:14.0	38.753	0	372736	24.404	3907.32	3.39E+07	1.69E+08
37:15.0	68.261	0	339968	21.843	3548.441	2.46E+08	2.62E+08
37:16.0	78.553	0	335872	23.991	3849.648	2.88E+08	2.94E+08
37:17.0	71.712	0	344064	27.504	4342.207	1.70E+08	1.83E+08
37:18.0	69.367	0	57344	28.502	4482.085	6951229	4.61E+07
37:19.0	48.337	0	3629056	33.411	5170.183	6.48E+07	1.85E+08
37:20.0	47.258	0	540672	33.231	5145.277	5.88E+07	1.93E+08
37:21.0	51.371	0	290816	33.879	5236.101	1.41E+07	8.84E+07
37:22.0	28.795	0	352256	32.501	5043.105	620713	2.23E+08
37:23.0	37.179	0	397312	31.277	4871.667	1.41E+07	1.47E+08
37:24.0	43.947	0	524288	30.01	4694.246	5.70E+07	1.70E+08
37:25.0	70.854	0	462848	30.577	4773.847	1.45E+08	1.71E+08
37:26.0	70.398	0	524288	27.894	4397.839	1.59E+08	1.73E+08
37:27.0	75.81	0	516096	22.908	3699.171	1.98E+08	2.18E+08
37:28.0	60.567	0	528384	21.69	3528.683	1.64E+08	2.48E+08
37:29.0	60.362	0	348160	23.612	3798.222	8.63E+07	1.91E+08
37:30.0	68.286	0	290816	26.731	4235.46	1.17E+08	1.67E+08
37:31.0	62.239	0	335872	30.778	4802.792	1.29E+08	2.38E+08
37:32.0	53.35	0	159744	32.211	5003.773	3.95E+07	1.63E+08
37:33.0	64.383	0	3297280	34.914	5382.695	596286	9.86E+07
37:34.0	60.582	0	233472	32.183	4999.894	2.46E+07	1.35E+08
37:35.0	37.037	0	512000	32.47	5040.289	9699224	1.41E+08
37:36.0	28.03	0	1748992	31.134	4854.109	7266184	1.99E+08
37:37.0	49.222	0	569344	30.455	4759.16	8.34E+07	1.35E+08
37:38.0	75.826	0	380928	22.168	3597.777	1.85E+08	2.10E+08
37:39.0	75.447	0	434176	19.036	3158.816	2.90E+08	3.12E+08
37:40.0	81.592	0	200704	22.305	3617.164	3.00E+08	3.00E+08
37:41.0	90	0	1683456	23.39	3769.292	1.32E+08	1.34E+08
37:42.0	78.304	0	126976	30.211	4725.367	8145566	4.05E+07
37:43.0	48.461	0	360448	34.519	5329.437	6.58E+07	2.05E+08
37:44.0	48.854	0	389120	37.211	5706.835	376089	1.74E+08
37:45.0	32.33	0	495616	35.117	5413.468	241745	1.51E+08
37:46.0	42.587	0	417792	32.503	5047.183	1.29E+07	1.70E+08
37:47.0	33.424	0	360448	32.664	5069.851	2302536	1.89E+08

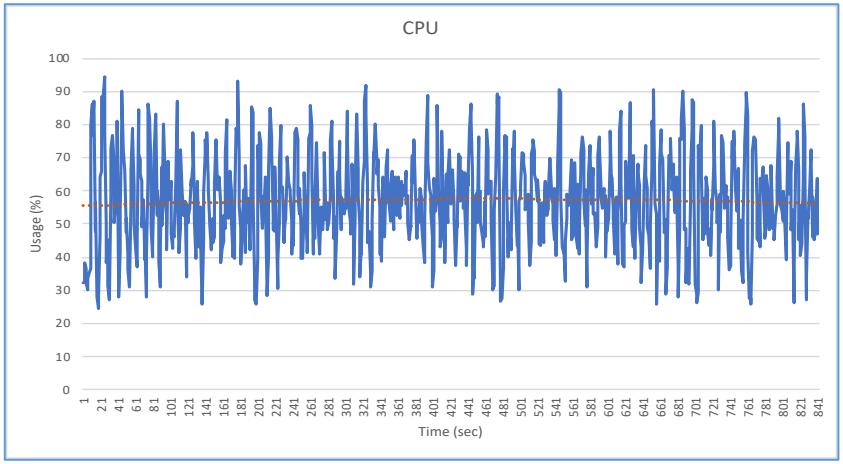
37:48.0	31.794	0	380928	29.98	4693.656	9350230	1.63E+08
37:49.0	56.666	0	352256	31.022	4839.914	1.08E+08	1.49E+08
37:50.0	66.165	0	335872	32.592	5060.058	1.94E+08	2.27E+08
37:51.0	78.68	0	262144	29.569	4636.273	2.83E+08	2.90E+08
37:52.0	87.562	0	462848	32.15	4998.246	2.55E+08	2.55E+08
37:53.0	86.868	0	192512	30.642	4786.871	3.83E+07	5.39E+07
37:54.0	65.903	0	208896	32.659	5069.761	590101	7.77E+07
37:55.0	32.08	0	499712	33.417	5176.066	2314119	1.52E+08
37:56.0	28.753	0	561152	32.039	4983.007	447045	2.08E+08
37:57.0	26.142	0	540672	32.039	4983.308	368667	1.93E+08
37:58.0	29.23	0	520192	27.881	4400.535	5884667	1.73E+08
37:59.0	73.934	0	491520	27.393	4332.457	1.32E+08	1.49E+08
38:00.0	66.917	0	360448	21.836	3553.605	2.00E+08	2.42E+08
38:01.0	71.782	0	315392	24.061	3865.589	1.77E+08	1.85E+08
38:02.0	79.948	0	413696	23.335	3763.855	1.79E+08	1.92E+08
38:03.0	64.781	0	385024	25.876	4120.23	1.31E+08	2.12E+08
38:04.0	45.238	0	196608	29.69	4654.855	6611592	1.57E+08
38:05.0	57.575	0	1933312	24.541	3933.187	6.54E+07	1.50E+08
38:07.0	47.146	0	1507328	27.945	4410.371	9.54E+07	1.91E+08
38:08.0	60.981	0	233472	25.45	4060.734	1.13E+08	1.89E+08
38:09.0	64.102	0	126976	28.218	4448.796	7.62E+07	1.32E+08
38:10.0	60.416	0	126976	33.206	5147.953	1.15E+08	2.05E+08
38:11.0	47.3	0	225280	34.215	5289.441	4.02E+07	1.45E+08
38:12.0	43.511	0	339968	31.125	4856.5	6098573	1.41E+08
38:13.0	48.806	0	520192	31.399	4895.046	2.78E+07	1.62E+08
38:14.0	40.512	0	475136	30.751	4804.339	5.27E+07	1.60E+08
38:15.0	57.552	0	401408	31.932	4969.957	1.54E+08	2.46E+08
38:16.0	54.177	0	331776	33.241	5153.644	1.84E+08	2.46E+08
38:17.0	67.839	0	450560	29.691	4656.027	1.78E+08	2.03E+08
38:18.0	81.188	0	188416	30.44	4761.078	1.39E+08	1.46E+08
38:19.0	60.869	0	172032	26.067	4148.167	6.41E+07	1.50E+08
38:20.0	56.708	0	393216	28.643	4509.296	7.13E+07	1.77E+08
38:21.0	50.37	0	356352	28.556	4497.273	2.57E+07	1.51E+08
38:22.0	51.269	0	524288	32.007	4981.23	9180591	1.59E+08
38:23.0	30.232	0	552960	29.258	4595.964	1.01E+07	1.56E+08
38:24.0	59.173	0	475136	25.855	4119.21	9.61E+07	1.83E+08
38:25.0	47.229	0	462848	25.331	4045.953	7.79E+07	2.08E+08
38:26.0	60.75	0	589824	22.362	3629.945	1.91E+08	2.30E+08
38:27.0	76.6	0	356352	24.322	3904.753	1.55E+08	1.68E+08
38:28.0	74.747	0	471040	21.609	3524.57	1.76E+08	1.96E+08
38:29.0	58.005	0	155648	26.265	4177.339	8692396	1.13E+08
38:30.0	54.911	0	204800	30.4	4757.007	1.31E+08	2.37E+08
38:31.0	61.538	0	352256	30.236	4734.003	1.65E+08	2.30E+08
38:32.0	55.612	0	372736	31.95	4974.441	9.20E+07	2.01E+08
38:33.0	58.291	0	471040	35.249	5437.085	388272	1.07E+08
38:34.0	45.931	0	393216	36.128	5560.351	1.82E+07	1.48E+08
38:35.0	35.989	0	471040	34.936	5393.464	1.20E+07	1.66E+08
38:36.0	37.688	0	524288	35.347	5451.21	3.56E+07	1.73E+08
38:37.0	45.572	0	339968	33.385	5176.289	1.07E+08	2.21E+08
38:38.0	74.87	0	176128	33.919	5251.285	1.47E+08	1.84E+08
38:39.0	70.025	0	208896	29.908	4689.035	2.40E+08	2.65E+08
38:40.0	74.32	0	90112	28.152	4442.996	1.78E+08	1.82E+08
38:41.0	78.085	0	139264	25.636	4090.367	1.27E+08	1.41E+08
38:42.0	61.246	0	262144	27.457	4345.644	9.52E+07	2.12E+08
38:43.0	53.846	0	221184	28.869	4543.625	1.24E+08	2.48E+08
38:44.0	63.184	0	69632	27.515	4353.835	3.29E+07	6.80E+07
38:45.0	67.007	0	2240512	26.683	4237.281	7.13E+07	1.18E+08
38:46.0	42.643	0	1863680	26.717	4242.078	1.06E+08	2.62E+08
38:47.0	47.199	0	286720	26.566	4221.136	3.31E+07	1.73E+08
38:48.0	51.413	0	294912	29.819	4677.113	1.40E+07	1.31E+08
38:49.0	35.572	0	356352	29.748	4667.285	357035	1.97E+08
38:50.0	32.124	0	331776	29.744	4666.882	8067578	2.17E+08
38:51.0	51.629	0	458752	25.873	4124.371	4.18E+07	1.24E+08
38:52.0	71.464	0	237568	22.026	3585.203	7.55E+07	1.15E+08
38:53.0	73.827	0	450560	23.703	3820.359	3.21E+08	3.44E+08
38:54.0	85.642	0	475136	26.025	4146.152	3.45E+08	3.45E+08
38:55.0	89.826	0	155648	29.854	4682.765	8.63E+07	8.63E+07
38:56.0	82.75	0	2838528	29.144	4583.382	41747	4592834
38:57.0	41.191	0	278528	33.311	5167.414	1.23E+07	1.63E+08
38:58.0	27.735	0	315392	33.475	5190.57	2382189	1.88E+08
38:59.0	27.631	0	589824	32.112	4999.73	601958	2.11E+08

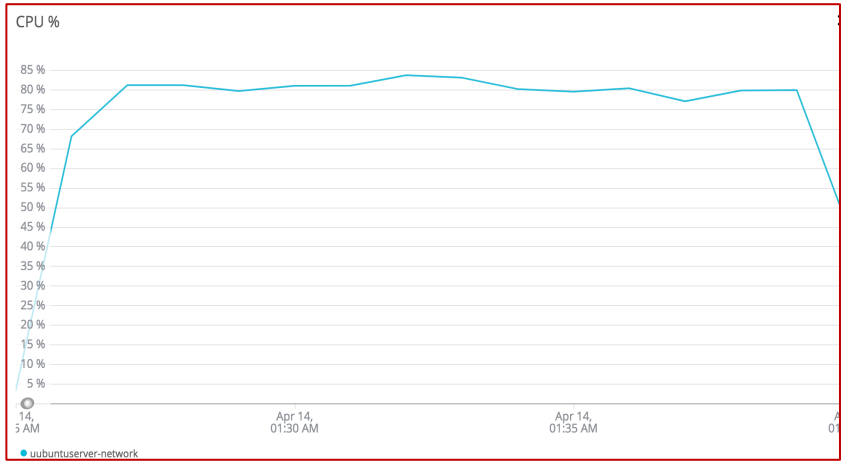
39:00.0	26	0	585728	30.779	4813.082	7911682	1.88E+08
39:01.0	47.15	0	536576	28.46	4488.281	5.42E+07	1.40E+08
39:02.0	72.051	0	212992	23.002	3723.312	1.35E+08	1.77E+08
39:03.0	76.441	0	516096	18.108	3037.414	2.65E+08	2.85E+08
39:04.0	76.13	0	270336	21.008	3444.117	2.64E+08	2.64E+08
39:05.0	75.561	0	2994176	27.082	4295.523	2.78E+07	2.79E+07
39:06.0	56.608	0	393216	30.429	4764.8	4.54E+07	1.17E+08
39:07.0	45.522	0	430080	29.388	4618.968	9.25E+07	2.39E+08
39:08.0	45.012	0	499712	30.921	4833.98	1.01E+07	1.49E+08
39:09.0	58.73	0	499712	28.149	4445.566	8.48E+07	1.79E+08
39:10.0	43.157	0	483328	27.733	4387.5	1.07E+08	2.35E+08
39:11.0	63.682	0	241664	20.639	3393.23	1.13E+08	1.77E+08
39:12.0	62.72	0	266240	21.017	3446.3	1.82E+08	2.14E+08
39:13.0	67.454	0	139264	25.436	4065.644	7.94E+07	1.62E+08
39:14.0	60.465	0	1323008	24.89	3990.203	6.81E+07	1.64E+08
39:15.0	48.717	0	253952	26.357	4195.886	3.81E+07	1.59E+08
39:16.0	45.876	0	372736	22.043	3591.261	4.19E+07	1.49E+08
39:17.0	49.481	0	425984	22.852	3704.828	1.10E+08	2.35E+08
39:18.0	64.646	0	401408	22.593	3668.683	8.32E+07	1.52E+08
39:19.0	60.759	0	327680	24.409	3923.484	5.57E+07	1.03E+08
39:20.0	57.397	0	413696	24.084	3878.019	1.71E+08	2.55E+08
39:21.0	48.32	0	311296	25.31	4049.976	1.59E+08	2.90E+08
39:22.0	54.659	0	184320	25.541	4082.382	1.03E+08	1.88E+08
39:23.0	67.624	0	3604480	28.663	4520.082	3.35E+07	1.27E+08
39:24.0	53.47	0	53248	31.79	4958.332	8.50E+07	1.85E+08
39:25.0	48.825	0	204800	32.854	5107.632	5.93E+07	1.98E+08
39:39.0	50.122	0	352256	31.31	4892.359	7.14E+07	1.85E+08
39:40.0	39.086	0	368640	27.36	4338.8	8.23E+07	2.12E+08
39:41.0	49.23	0	258048	21.961	3582.128	1.08E+08	2.46E+08
39:42.0	74.563	0	2502656	20.926	3436.949	1.78E+08	1.86E+08
39:43.0	66.584	0	159744	19.917	3295.55	1.63E+08	1.87E+08
39:44.0	58.974	0	393216	21.742	3551.437	1.31E+08	2.36E+08
39:45.0	69.191	0	397312	26.359	4198.664	6.41E+07	1.60E+08
39:46.0	48.484	0	454656	28.136	4447.769	6.51E+07	1.47E+08
39:47.0	53.787	0	516096	24.011	3869.628	1.24E+07	1.45E+08
39:48.0	54.427	0	536576	26.524	4221.89	2.18E+07	1.19E+08
39:49.0	26.356	0	520192	26.58	4229.792	437825	1.87E+08
39:50.0	36.708	0	528384	19.803	3279.89	2.23E+07	1.51E+08
39:51.0	67.179	0	323584	21.066	3456.996	1.33E+08	1.83E+08
39:52.0	76	0	372736	23.8	3840.265	3.42E+08	3.42E+08
39:53.0	78.25	0	233472	27.165	4312.085	1.50E+08	1.50E+08
39:54.0	66.057	0	106496	27.568	4368.523	9533573	9.12E+07
39:55.0	45.077	0	147456	30.56	4788.019	1538310	1.54E+08
39:56.0	55.639	252	253952	26.605	4233.757	2.26E+07	1.45E+08
39:57.0	40.648	56	258048	25.611	4094.644	4859967	1.28E+08
39:58.0	44.162	0	278528	24.517	3941.328	4.07E+07	1.85E+08
39:59.0	63.959	0	286720	21.946	3580.949	2.78E+08	3.16E+08
40:00.0	86.069	0	249856	24.591	3951.871	2.83E+08	2.83E+08
40:01.0	76.25	0	3600384	27.089	4301.996	5.44E+07	6.74E+07
40:02.0	63.224	0	114688	32.158	5012.48	8424525	6.69E+07
40:03.0	27.105	0	557056	32.173	5014.714	575801	2.17E+08
40:04.0	34.673	0	548864	29.423	4629.164	1.05E+07	1.40E+08
40:05.0	52.971	0	425984	25.737	4112.57	7.10E+07	1.30E+08
40:06.0	51.785	0	397312	21.93	3578.976	1.13E+08	2.19E+08
40:07.0	59.846	0	536576	21.905	3575.507	1.89E+08	2.67E+08
40:08.0	72.519	0	266240	19.837	3285.671	8.81E+07	1.01E+08
40:09.0	68.542	0	2633728	24.07	3879.042	9.60E+07	1.51E+08
40:10.0	46.052	0	360448	27.433	4350.503	1.06E+08	2.56E+08
40:11.0	58.224	0	446464	27.53	4364.25	7846114	1.28E+08
40:12.0	54.223	0	491520	24.697	3967.128	2.30E+07	1.30E+08
40:13.0	45.118	0	446464	25.207	4038.808	7.94E+07	2.22E+08
40:14.0	48.697	0	491520	25.124	4027.214	1.41E+08	2.15E+08
40:15.0	63.591	0	520192	23.182	3755.109	5.47E+07	7.63E+07
40:16.0	47.058	0	622592	25.902	4136.464	1.02E+08	2.21E+08
40:17.0	56.992	0	544768	25.983	4147.851	7.39E+07	1.54E+08

	CPU	Disk Read	Disk Write	Memory %	Memory MB	Network received	Network sent
MIN	31.578	0.000	0.000	13.961	7621.167	0.000	0.000
MAX	75.757	0.000	119980032.000	34.939	10984.539	703874728.000	300732765.000
AVG	65.955	0.000	4968261.818	29.626	10126.362	430807543.545	217503213.818
MED	67.250	0.000	2654208.000	30.313	10241.590	439659753.000	225701617.000
STDEV	8.089	0.000	13852888.717	4.239	682.138	119897485.065	54237027.081

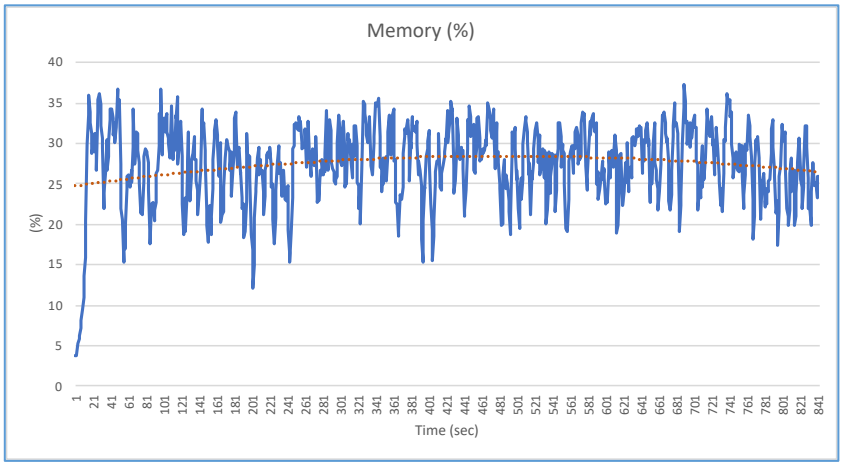
De las tablas anteriores, se presentan gráficas obtenidas de los logs de JMeter con borde azul y obtenidas con el servicio de NewRelic con borde rojo. En los gráficos de NewRelic, se debe considerar solamente la zona con mayor actividad para comparar con los gráficos correspondientes de JMeter.

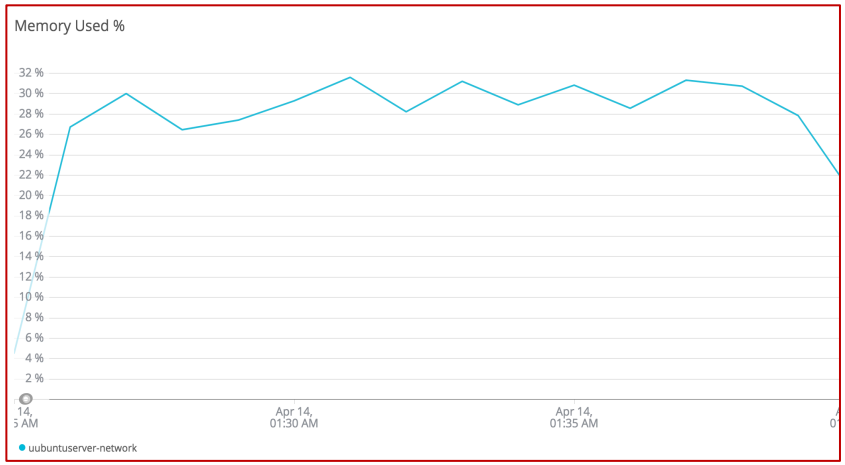
CPU

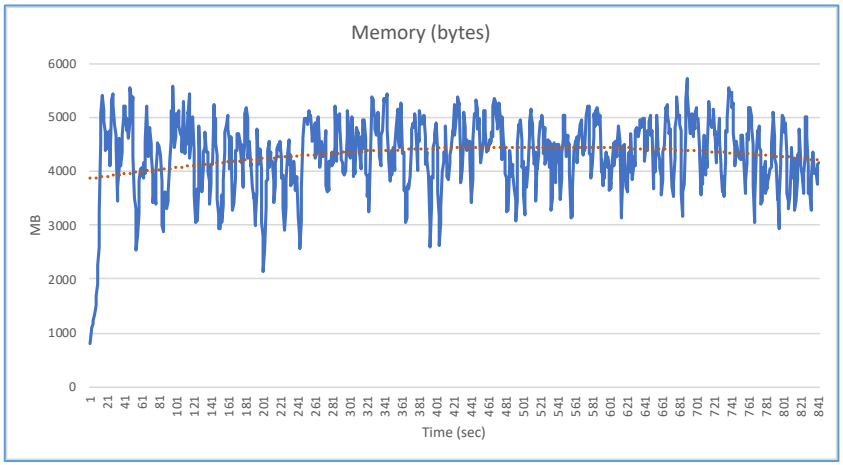


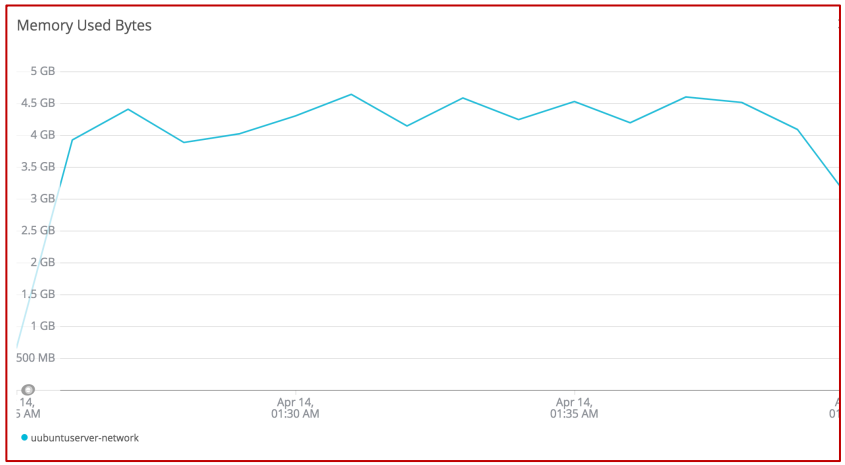


Memoria

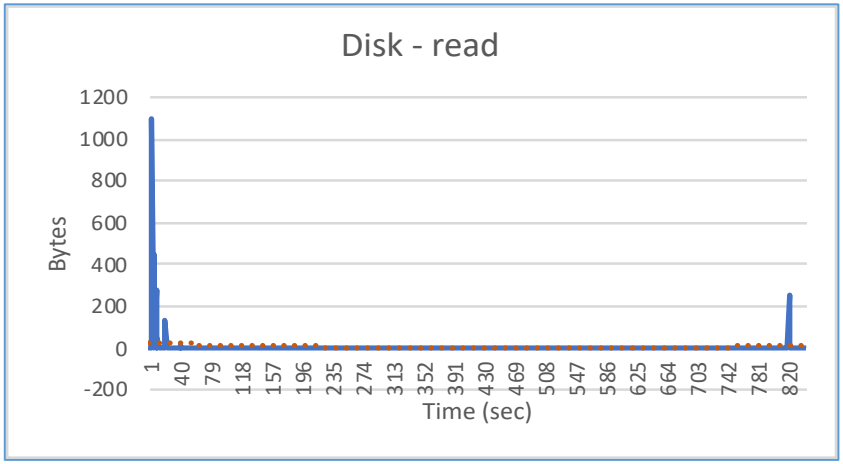

Memory (%)
(%)
Time (sec)

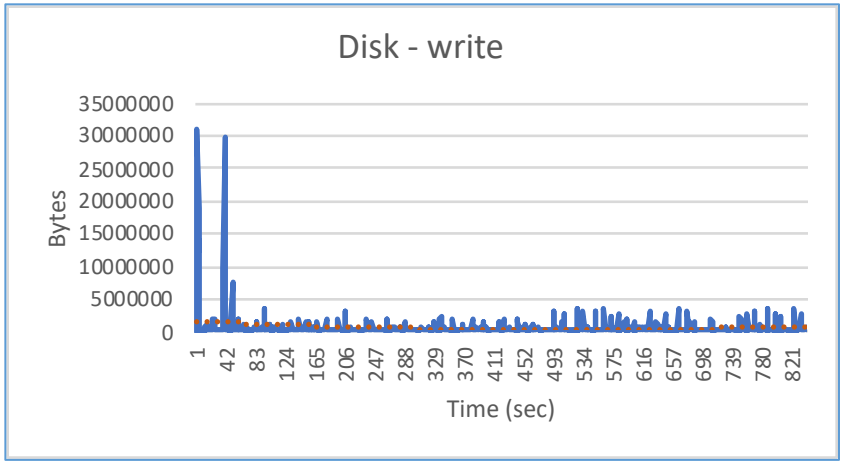

Memory Used %
Apr 14, 01:30 AM
Apr 14, 01:35 AM
uubuntuserver-network


Memory (bytes)
MB
Time (sec)

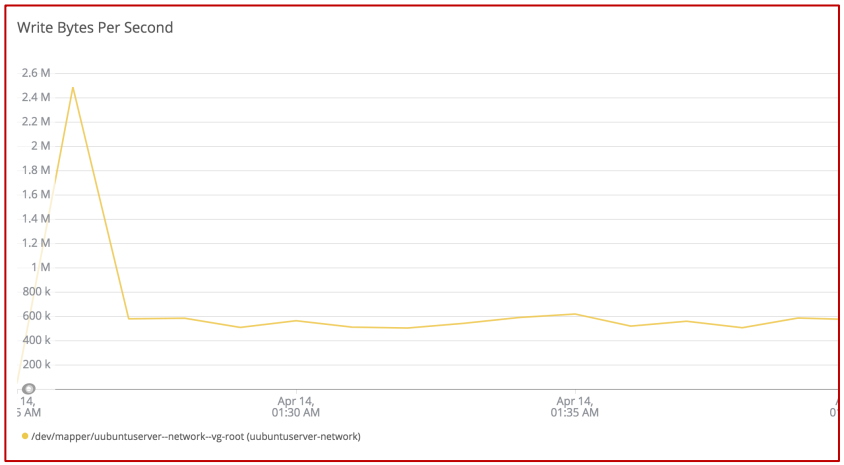

Memory Used Bytes
Apr 14, 01:30 AM
Apr 14, 01:35 AM
uubuntuserver-network

Disco

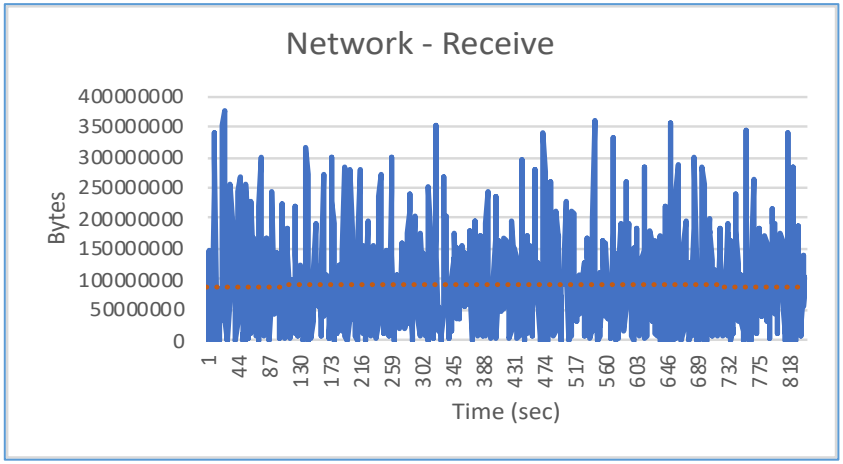

Disk - read
Bytes
1200
1000
800
600
400
200
0
-200
1
40
79
118
157
196
235
274
313
352
391
430
469
508
547
586
625
664
703
742
781
820
Time (sec)

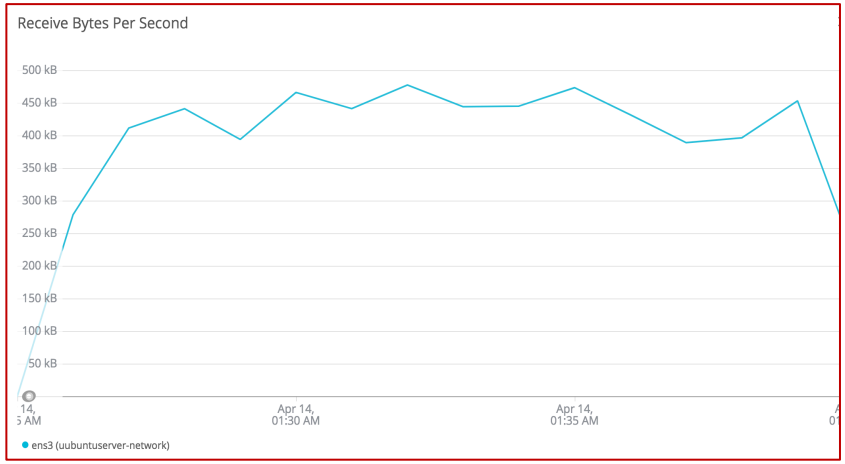

Disk - write
Bytes
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
Time (sec)

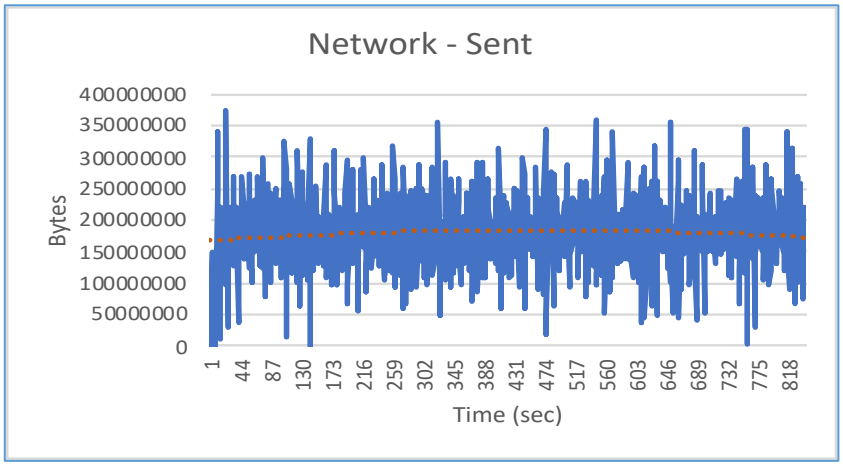

Read Bytes Per Second
900 k
850 k
800 k
750 k
700 k
650 k
600 k
550 k
500 k
450 k
400 k
350 k
300 k
250 k
200 k
150 k
100 k
50 k
Apr 14,
01:30 AM
Apr 14,
01:35 AM
/dev/mapper/uubuntuserver--network--vg-root (uubuntuserver-network)

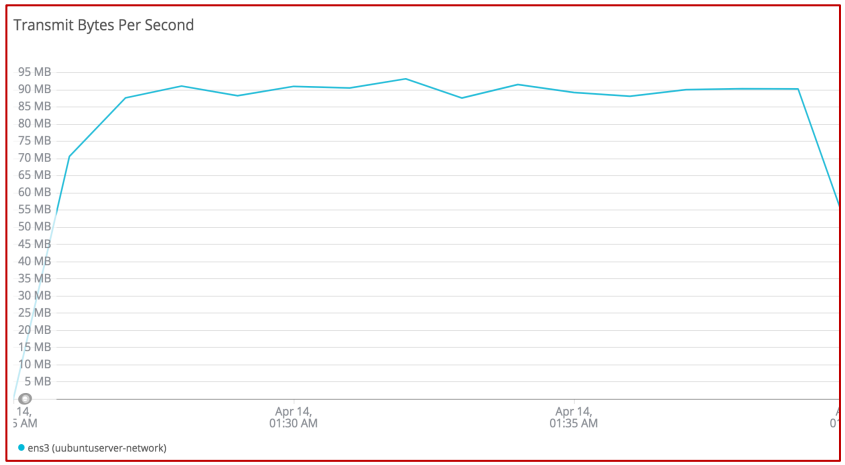

Write Bytes Per Second
2.6 M
2.4 M
2.2 M
2 M
1.8 M
1.6 M
1.4 M
1.2 M
1 M
800 k
600 k
400 k
200 k
Apr 14,
01:30 AM
Apr 14,
01:35 AM
/dev/mapper/uubuntuserver--network--vg-root (uubuntuserver-network)

Red


Network - Receive
Bytes
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
1 44 87 130 173 216 259 302 345 388 431 474 517 560 603 646 689 732 775 818
Time (sec)


Receive Bytes Per Second
500 kB
450 kB
400 kB
350 kB
300 kB
250 kB
200 kB
150 kB
100 kB
50 kB
Apr 14, 01:30 AM
Apr 14, 01:35 AM
ens3 (uubuntuserver-network)


Network - Sent
Bytes
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
1 44 87 130 173 216 259 302 345 388 431 474 517 560 603 646 689 732 775 818
Time (sec)


Transmit Bytes Per Second
95 MB
90 MB
85 MB
80 MB
75 MB
70 MB
65 MB
60 MB
55 MB
50 MB
45 MB
40 MB
35 MB
30 MB
25 MB
20 MB
15 MB
10 MB
5 MB
Apr 14, 01:30 AM
Apr 14, 01:35 AM
ens3 (uubuntuserver-network)

Anexo 6: Resultado obtenido con JMeter tras la ejecución de la prueba de estrés para la Aplicación de Microservicios en el Caso 2.

Elapsed time	CPU (%)	Disc - read (bytes/s)	Disc - write (bytes/s)	Memory (%)	Memory (MB)	Network - receive (bytes/s)	Network - sent (bytes/s)
05:12.0	42.786	0	0	10.884	4501.375	0	0
05:13.0	44.278	0	0	10.884	4501.375	0	0
05:14.0	41.37	24	9.63E+07	12.25	4727.73	1.86E+08	1.86E+08
05:15.0	37.25	6984	1.71E+07	12.84	4827	5449422	5449752
05:16.0	39.152	1296	4.20E+07	13.153	4883.507	1.33E+07	1.33E+07
05:17.0	35	864	3096576	13.581	4950.062	1317	1331
05:18.0	32.581	0	2949120	14.765	5138.875	1966	46674
05:19.0	39.249	432	2752512	15.821	5307.16	2.52E+07	2.52E+07
05:20.0	56.109	0	3244032	16.821	5467.214	2.14E+08	2.14E+08
05:21.0	54.773	0	2654208	18.412	5720.632	2.29E+08	2.29E+08
05:22.0	54.385	0	2727936	20.198	6005.699	1.71E+08	1.71E+08
05:23.0	68.686	0	2801664	25.347	6826.695	4.94E+08	2.01E+08
05:24.0	74.12	0	3907584	25.401	6835.734	2.71E+08	1.65E+08
05:25.0	71.683	0	2654208	26.198	6962.867	7.06E+08	2.81E+08
05:26.0	68.079	0	2752512	28.883	7391.074	1.32E+08	7.72E+07
05:27.0	74.936	0	2752512	29.787	7539.824	7.29E+08	3.58E+08
05:28.0	61.46	0	2826240	30.634	7671.152	1.63E+08	1.37E+08
05:29.0	54.659	0	4177920	32.115	7907.695	5517817	5175419
05:30.0	86.18	0	2801664	34.124	8228.515	1.29E+09	4.92E+08
05:31.0	82	0	2801664	32.548	7976.488	2.28E+08	2.13E+08
05:32.0	62.779	0	2531328	33.634	8180.804	1.23E+08	6.62E+07
05:33.0	85.856	24	2.32E+08	32.536	8008.449	7.49E+08	3.68E+08
05:34.0	64.16	0	4644864	33.417	8152.199	1.50E+08	7.46E+07
05:35.0	76.633	0	2629632	31.695	7877.851	6.66E+08	3.16E+08
05:36.0	66.749	0	2187264	33.339	8140.128	6.95E+08	2.68E+08
05:37.0	74.12	0	2359296	32.074	7939.105	2.50E+08	2.23E+08
05:38.0	54.636	0	2236416	32.792	8054.019	2.02E+08	2.02E+08
05:39.0	60.453	0	1744896	36.357	8620.781	2.30E+08	8.97E+07
05:40.0	59.045	0	2138112	38.041	8890.234	5.12E+08	1.77E+08
05:41.0	86.386	0	1327104	33.326	8135.925	6.27E+08	3.05E+08
05:42.0	73.469	0	1622016	33.669	8193.785	1.91E+08	1.37E+08
05:43.0	68.181	0	2310144	34.197	8278	4.89E+08	1.95E+08
05:44.0	72.01	0	3809280	35.465	8480.332	7.39E+08	2.89E+08
05:45.0	86.616	0	3022848	31.808	7926.847	2.74E+08	1.92E+08
05:46.0	63.358	0	1.11E+07	33.225	8144.976	9.50E+07	6.82E+07
05:47.0	73.75	0	3.38E+07	31.381	7845.371	7.49E+08	3.50E+08
05:48.0	65.743	0	1.60E+07	31.505	7859.05	2.66E+08	2.39E+08
05:49.0	53.383	0	2727936	34.235	8294.339	7.44E+07	7.44E+07
05:50.0	74.747	0	2826240	35.1	8429.492	7.25E+08	3.26E+08
05:51.0	63.93	0	2605056	36.253	8616.371	7.76E+07	5.01E+07
05:52.0	75.81	0	2359296	33.401	8158.949	7.40E+08	3.41E+08
05:53.0	69.095	0	2482176	35.853	8553.035	1.95E+08	8.75E+07
05:54.0	78.625	0	3981312	32.461	8012.437	5.49E+08	2.29E+08
05:55.0	80.05	0	2285568	33.919	8244.988	7.42E+08	3.15E+08
05:56.0	68.578	0	2875392	32.583	8032.589	2.49E+08	2.21E+08
05:57.0	56.965	0	2285568	33.778	8222.906	7.23E+07	5.79E+07
05:58.0	76.296	0	2383872	32.743	8055.113	7.61E+08	3.77E+08
05:59.0	60.101	0	2211840	34.041	8265.246	1.28E+08	1.00E+08
06:00.0	75.818	0	2875392	32.756	8056.703	7.17E+08	3.18E+08
06:01.0	73.182	0	1892352	33.157	8129.515	1.53E+08	1.26E+08
06:02.0	59.493	0	1548288	33.941	8249.554	2.00E+08	8.02E+07
06:03.0	81.979	0	1425408	35.2	8450.41	1.02E+09	3.95E+08
06:04.0	84	0	2924544	33.135	8121.382	2.44E+08	1.75E+08
06:05.0	73.067	0	1400832	35.592	8513.117	5.83E+08	2.38E+08
06:06.0	77.25	0	1302528	32.501	8021.042	3.82E+08	2.59E+08
06:07.0	53.634	0	1277952	33.905	8244.824	8.94E+07	8.94E+07
06:08.0	75.062	0	1376256	32.225	7973.691	7.29E+08	3.30E+08
06:09.0	64.179	0	1622016	35.231	8456.062	8.54E+07	5.73E+07
06:10.0	74.686	0	1253376	31.872	7920.675	7.73E+08	3.75E+08
06:11.0	63.749	0	2039808	32.593	8035.855	2.57E+08	2.29E+08
06:12.0	55.499	0	2433024	35.74	8537.64	2.81E+07	2.81E+07
06:13.0	60.301	0	2752512	37.877	8878.597	1.48E+08	5.23E+07
06:14.0	89.34	0	3907584	35.463	8490.925	1.16E+09	4.55E+08
06:15.0	81.453	0	2482176	35.072	8442.285	1.42E+08	8.71E+07
06:16.0	73.803	0	2826240	32.985	8109.828	7.24E+08	3.23E+08
06:17.0	63.909	0	2555904	33.152	8136.574	1.73E+08	1.45E+08
06:18.0	66.666	0	2875392	32.481	8029.359	6.84E+08	2.85E+08
06:19.0	70.719	0	2899968	32.674	8050.242	1.99E+08	1.71E+08
06:20.0	64.912	0	2334720	32.264	7984.894	6.67E+08	2.68E+08
06:21.0	75.318	0	2899968	32.27	7985.875	2.54E+08	2.25E+08
06:22.0	52.369	0	3219456	35.105	8438.757	7.34E+07	7.34E+07
06:23.0	68	0	2482176	36.165	8607.421	1.56E+08	6.56E+07
06:24.0	89.141	0	4079616	33.023	8103.667	1.15E+09	4.45E+08
06:25.0	79.898	0	2383872	33.496	8182.355	2.03E+08	1.46E+08
06:26.0	70.1	0	1769472	33.281	8158.156	7.15E+08	3.16E+08
06:27.0	73.164	0	1695744	33.397	8177.253	1.16E+08	8.73E+07
06:28.0	71.859	0	2138112	32.751	8074.343	7.34E+08	3.35E+08
06:29.0	68.407	0	2236416	32.535	8029.617	2.62E+08	2.34E+08
06:30.0	58.987	0	2211840	33.603	8199.972	2.71E+08	1.10E+08
06:31.0	71	0	2187264	32.59	8043.632	5.57E+08	3.12E+08
06:32.0	58.08	0	1818624	34.594	8358.035	1.16E+08	9.38E+07
06:33.0	64.974	0	1523712	34.417	8329.917	4.23E+08	1.67E+08
06:34.0	85.536	0	3416064	34.571	8354.703	8.91E+08	3.49E+08
06:35.0	82.575	0	3588096	33.678	8212.574	1.71E+08	1.35E+08
06:36.0	74.111	0	2899968	33.014	8107.273	7.21E+08	3.00E+08
06:37.0	65.326	0	3612672	33.088	8118.859	1.96E+08	1.67E+08
06:38.0	64.572	0	3121152	33	8104.988	6.70E+08	2.70E+08
06:39.0	76.616	0	3219456	33.112	8123.16	2.58E+08	2.30E+08
06:40.0	57.644	0	3194880	34.637	8366.308	2.97E+08	1.14E+08
06:41.0	78.054	0	3538944	33.732	8222.296	5.28E+08	3.04E+08
06:42.0	61.152	0	1941504	34.017	8267.785	1.48E+08	1.26E+08
06:43.0	69.306	0	2949120	34.317	8315.832	2.82E+08	1.20E+08
06:44.0	79.145	0	3932160	35.708	8537.652	9.18E+08	3.47E+08
06:45.0	83.709	0	2015232	33.599	8201.542	2.45E+08	1.52E+08
06:46.0	77.777	0	2088960	35.017	8428.21	7.23E+08	2.95E+08
06:47.0	67.91	0	1966080	33.904	8250.457	2.32E+08	2.03E+08
06:48.0	59.798	0	2703360	34.808	8394.648	4.78E+08	1.85E+08
06:49.0	76.368	0	8601600	33.128	8128.085	4.10E+08	2.74E+08
06:50.0	54.975	0	2580480	33.939	8257.515	1.57E+08	1.57E+08
06:51.0	64.5	0	2826240	35.952	8578.437	6.59E+08	2.60E+08
06:52.0	79.04	0	2826240	35.647	8529.875	1.72E+08	1.43E+08
06:53.0	65.508	0	3194880	36.05	8594.421	1.30E+08	5.51E+07
06:54.0	67.848	0	3588096	38.587	8999.097	9.66E+08	3.48E+08
06:55.0	87.437	0	2678784	34.169	8295.058	2.78E+08	1.74E+08
06:56.0	78.195	0	2752512	36.833	8720.218	6.86E+08	2.78E+08
06:57.0	74.875	0	2260992	33.845	8243.683	3.13E+08	2.34E+08
06:58.0	54.568	0	1597440	33.865	8246.937	1.68E+08	1.68E+08
06:59.0	64.925	0	1548288	33.977	8264.613	6.69E+08	2.70E+08
07:00.0	74.43	0	1794048	33.991	8267.238	2.21E+08	1.91E+08
07:01.0	61.712	0	1794048	35.77	8550.64	4.87E+08	1.97E+08
07:02.0	76	0	2211840	34.672	8380.843	3.31E+08	1.96E+08
07:04.0	60.957	0	1843200	35.773	8550.554	9815621	6409738
07:05.0	82.706	0	3293184	35.817	8558.75	1.28E+09	4.82E+08
07:06.0	85.536	0	1425408	34.265	8311.476	1.61E+08	1.61E+08
07:07.0	70.854	0	1499136	36.159	8613.769	7.46E+08	2.88E+08
07:08.0	77	0	1966080	33.896	8252.761	2.35E+08	2.05E+08
07:09.0	56.708	0	2285568	33.979	8266.117	2.37E+08	2.37E+08
07:10.0	55.334	0	2285568	36.417	8654.949	4.01E+07	4.01E+07
07:11.0	56.109	0	2211840	38.493	8985.953	1.64E+08	5.75E+07
07:12.0	88.118	0	2138112	35.144	8449.351	1.14E+09	4.49E+08
07:13.0	81.453	12864	1646592	35.086	8445.238	1.90E+08	1.45E+08
07:14.0	75.438	10176	1572864	35.615	8531.218	5.43E+08	2.26E+08
07:15.0	69.191	0	3563520	36.862	8730.222	5.12E+08	1.97E+08

07:16.0	88.413	7992	958464	32.47	8031.644	4.67E+08	2.59E+08
07:17.0	67.331	0	1523712	32.358	8014.242	1.39E+08	1.06E+08
07:18.0	73.697	432	1548288	31.889	7939.3	7.20E+08	3.22E+08
07:19.0	65.847	0	1966080	32.128	7977.386	2.63E+08	2.33E+08
07:20.0	53.67	0	2457600	34.077	8288.277	8.95E+07	8.95E+07
07:21.0	60.249	0	2678784	33.536	8202.312	4.96E+08	1.94E+08
07:22.0	79.648	0	2408448	33.757	8237.769	8.30E+08	3.34E+08
07:23.0	79.25	0	2605056	32.876	8097.484	2.54E+08	2.16E+08
07:24.0	68.765	0	2285568	33.697	8228.613	1.19E+08	6.17E+07
07:25.0	67.341	0	4644864	34.715	8391.042	4.50E+08	1.84E+08
07:26.0	84.771	0	2260992	32.255	7996.042	8.61E+08	3.64E+08
07:27.0	75.561	0	1966080	32.612	8056.449	1.98E+08	1.37E+08
07:28.0	65.829	0	2383872	32.438	8029.085	6.96E+08	2.97E+08
07:29.0	70.632	0	1990656	32.298	8006.468	2.64E+08	2.34E+08
07:30.0	54.094	0	2113536	33.109	8135.949	1.36E+08	1.36E+08
07:31.0	62.182	0	2138112	33.959	8271.449	5.57E+08	2.18E+08
07:32.0	68.939	0	1671168	35.291	8488.937	5.90E+08	2.34E+08
07:33.0	82.338	0	1966080	32.737	8082.109	3.71E+08	2.38E+08
07:34.0	71.393	0	1130496	33.961	8272.078	1.01E+08	7.02E+07
07:35.0	73.566	0	2727936	32.574	8051.14	6.84E+08	2.85E+08
07:36.0	78.553	0	2236416	35.192	8468.558	5.51E+08	2.29E+08
07:37.0	76.59	0	1228800	32.53	8044.753	3.94E+08	2.55E+08
07:38.0	54.385	0	1204224	32.77	8082.589	1.75E+08	1.76E+08
07:39.0	61.868	0	1277952	33.542	8205.792	5.49E+08	2.28E+08
07:40.0	72.25	0	1105920	32.744	8078.753	3.68E+08	2.59E+08
07:41.0	63.909	0	1622016	34.061	8288.66	2.47E+08	9.67E+07
07:42.0	78.163	0	2457600	33.726	8235.5	4.64E+08	2.06E+08
07:43.0	73.869	0	2629632	32.685	8069.921	6.97E+08	2.77E+08
07:44.0	79.396	0	2777088	35.417	8505.5	5.33E+08	2.12E+08
07:45.0	80.102	0	3883008	32.937	8110.371	3.89E+08	2.50E+08
07:46.0	57.071	0	3239936	33.148	8144.753	1.05E+08	1.05E+08
07:47.0	73.232	0	2633728	32.633	8062.339	7.30E+08	3.31E+08
07:48.0	66.084	0	2752512	32.905	8105.777	2.70E+08	2.39E+08
07:49.0	54.499	0	8798208	34.95	8433.07	5.53E+07	5.53E+07
07:50.0	74.559	0	2138112	34.571	8372.792	6.64E+08	2.65E+08
07:51.0	73.25	0	2039808	37.259	8801.41	6.20E+08	2.44E+08
07:52.0	78.481	0	2211840	34.162	8307.867	2.21E+08	1.40E+08
07:53.0	65.75	0	2138112	33.518	8205.343	6.80E+08	2.77E+08
07:54.0	77.078	0	2605056	33.751	8242.64	1.60E+08	1.28E+08
07:55.0	67.839	0	3735552	33.314	8173.14	6.84E+08	2.85E+08
07:56.0	71.608	0	2088960	33.654	8227.621	2.75E+08	2.44E+08
07:57.0	53.5	0	2236416	36.702	8713.988	6.03E+07	6.04E+07
07:58.0	71.75	0	1597440	32.765	8086.175	7.58E+08	3.59E+08
07:59.0	65.664	0	1671168	34.72	8397.878	3.06E+08	1.19E+08
08:00.0	73.924	0	1572864	33.392	8186.308	4.84E+08	2.42E+08
08:01.0	72.795	0	1474560	32.892	8106.683	7.18E+08	2.87E+08
08:02.0	70.719	0	1228800	33.122	8148.566	2.45E+08	2.14E+08
08:03.0	58.438	0	1376256	34.361	8346.078	4.54E+08	1.77E+08
08:04.0	80.856	0	2236416	33.142	8146.652	4.04E+08	2.55E+08
08:05.0	62.344	0	2875392	34.01	8285.316	2.51E+08	1.03E+08
08:06.0	79.648	0	1351680	32.97	8120.175	5.67E+08	3.13E+08
08:07.0	62.344	0	2260992	35.922	8590.761	8.65E+07	5.48E+07
08:08.0	78.195	0	2310144	32.99	8123.128	7.30E+08	3.31E+08
08:09.0	68.447	0	2138112	34.66	8389.664	4.47E+08	1.71E+08
08:10.0	80.049	0	2138112	33.019	8128.054	4.69E+08	2.87E+08
08:11.0	56.999	0	1794048	33.929	8273.117	1.77E+08	8.57E+07
08:12.0	77.611	0	2334720	32.852	8101.82	6.49E+08	3.36E+08
08:13.0	61.46	0	2088960	35.354	8500.699	1.05E+08	7.30E+07
08:14.0	77.419	0	2211840	32.567	8053.57	7.34E+08	3.35E+08
08:15.0	69.949	0	3710976	33.42	8195.863	1.59E+08	1.27E+08
08:16.0	69.095	0	2162688	32.896	8112.421	6.88E+08	2.89E+08
08:17.0	71.177	0	2260992	35.584	8541.445	2.75E+08	1.21E+08
08:18.0	75.373	0	2211840	33.294	8176.078	5.86E+08	3.09E+08
08:19.0	59	0	2285568	34.634	8389.71	1.31E+08	9.87E+07
08:20.0	72.864	0	2531328	33.036	8129.234	7.34E+08	3.35E+08
08:21.0	64.646	0	2334720	33.845	8261.308	1.35E+08	1.03E+08
08:22.0	73.934	0	2162688	33.122	8143.226	7.28E+08	3.29E+08
08:23.0	69.444	0	1941504	33.421	8194.128	2.23E+08	1.91E+08

08:24.0	74.009	0	2211840	35.078	8458.375	4.87E+08	1.94E+08
08:25.0	55.583	0	3538944	36.214	8639.496	3.24E+08	1.31E+08
08:26.0	87	0	1867776	33.902	8271.16	6.58E+08	3.19E+08
08:27.0	68.922	0	1597440	35.203	8479.152	1.20E+08	8.16E+07
08:28.0	71.5	0	1400832	33.229	8164.675	7.49E+08	3.50E+08
08:29.0	64.411	0	1941504	33.387	8189.535	1.40E+08	1.08E+08
08:30.0	70.959	0	1523712	33.23	8164.414	7.29E+08	3.30E+08
08:31.0	64.393	0	1597440	33.615	8226.019	1.60E+08	1.27E+08
08:32.0	69.191	0	1499136	32.996	8132.503	7.11E+08	3.12E+08
08:33.0	67.758	0	1499136	33.496	8212.218	2.68E+08	2.36E+08
08:34.0	63.065	0	1302528	35.825	8578.609	3.24E+08	1.23E+08
08:35.0	76.426	0	3268608	34.33	8340.339	4.37E+08	2.13E+08
08:36.0	72.249	0	2260992	33.969	8282.968	6.94E+08	2.89E+08
08:37.0	69.72	0	2039808	34.726	8404.167	1.27E+08	9.40E+07
08:38.0	73.299	0	2052096	33.35	8184.562	7.35E+08	3.36E+08
08:39.0	66.582	0	1585152	33.419	8195.718	1.53E+08	1.21E+08
08:40.0	69.975	0	1892352	32.06	7978.98	7.02E+08	3.03E+08
08:41.0	71.212	0	1204224	32.243	8008.32	2.69E+08	2.37E+08
08:42.0	50.5	0	1449984	35.71	8560.949	5.26E+07	5.24E+07
08:43.0	69.25	0	1425408	35.676	8555.648	7.03E+08	3.04E+08
08:44.0	71.356	0	958464	37.339	8820.902	3.01E+08	1.19E+08
08:45.0	80.701	0	3710976	33.656	8233.769	5.01E+08	2.44E+08
08:46.0	71.212	0	2310144	36.003	8608.203	6.81E+08	2.67E+08
08:47.0	73.75	0	2211840	33.115	8148.496	2.44E+08	2.02E+08
08:48.0	58.56	0	2211840	33.599	8225.363	3.16E+08	1.29E+08
08:49.0	78.195	0	8478720	32.741	8088.929	5.46E+08	3.08E+08
08:50.0	53.634	0	3268608	33.405	8195.718	1.02E+08	9.59E+07
08:51.0	72.11	0	2310144	32.805	8097.171	7.18E+08	3.19E+08
08:52.0	64.572	0	1794048	33.219	8166.242	1.44E+08	1.11E+08
08:53.0	71.105	0	1794048	32.967	8126.367	7.27E+08	3.28E+08
08:54.0	71.96	0	2015232	35.577	8542.492	8.24E+07	4.91E+07
08:55.0	77.099	0	3047424	32.423	8039.875	7.47E+08	3.48E+08
08:56.0	71.82	0	1449984	35.058	8460.429	5.56E+08	2.13E+08
08:57.0	74.307	0	2138112	32.278	8016.917	3.74E+08	2.52E+08
08:58.0	53.282	0	2187264	34.126	8311.648	7.84E+07	7.84E+07
08:59.0	71.464	0	2064384	32.133	7991.191	7.46E+08	3.47E+08
09:00.0	67.25	0	2801664	33.146	8155.71	1.58E+08	1.25E+08
09:01.0	67.336	0	2162688	31.869	7952.335	6.96E+08	2.97E+08
09:02.0	67.848	0	2138112	33.06	8150.425	1.62E+08	1.29E+08
09:03.0	67.91	0	1351680	32.009	7982.98	6.94E+08	2.95E+08
09:04.0	74.563	0	1449984	34.927	8443.085	2.98E+08	1.21E+08
09:05.0	73.047	0	2752512	32.903	8117.617	5.57E+08	2.99E+08
09:06.0	65.404	0	2408448	33.832	8266.246	1.15E+08	8.45E+07
09:07.0	72.932	0	2039808	32.237	8008.734	7.49E+08	3.50E+08
09:08.0	65.413	0	1499136	32.626	8073.8	1.82E+08	1.48E+08
09:09.0	67.669	0	2260992	32.096	7989.523	6.84E+08	2.85E+08
09:10.0	77.192	0	2383872	32.303	8022.621	2.07E+08	1.73E+08
09:11.0	63.909	0	2629632	31.963	7968.664	6.47E+08	2.48E+08
09:12.0	74.055	0	1720320	34.583	8386.375	1.22E+08	7.27E+07
09:13.0	77.25	0	2088960	32.578	8066.992	7.36E+08	3.53E+08
09:14.0	68.888	0	2187264	32.785	8100.125	2.26E+08	1.93E+08
09:15.0	65.909	0	3809280	33.897	8277.453	5.03E+08	2.02E+08
09:16.0	73.945	0	2408448	33.004	8135.8	3.42E+08	2.10E+08
09:17.0	62.656	0	2064384	32.79	8101.769	6.47E+08	2.48E+08
09:18.0	72.319	0	2211840	33.101	8151.085	2.56E+08	2.22E+08
09:19.0	57.323	0	2236416	34.147	8317.925	3.60E+08	1.42E+08
09:20.0	74.177	0	1449984	34.765	8416.535	3.62E+08	1.55E+08
09:21.0	80.15	0	1228800	32.792	8102.179	7.36E+08	3.30E+08
09:22.0	62.216	0	2015232	32.897	8119.039	2.02E+08	1.68E+08
09:23.0	64.661	0	2162688	32.832	8108.886	6.73E+08	2.74E+08
09:24.0	78.589	0	2236416	32.788	8102.007	2.59E+08	2.25E+08
09:25.0	53.482	0	3637248	36.006	8615.082	5.70E+07	5.65E+07
09:26.0	71.428	0	2629632	33.525	8216.714	7.51E+08	3.52E+08
09:27.0	66.334	0	2408448	34.565	8386.242	2.70E+08	1.10E+08
09:28.0	79.389	0	1597440	33.199	8168.066	5.17E+08	2.43E+08
09:29.0	71.859	0	2088960	32.716	8091.054	7.02E+08	2.68E+08
09:30.0	72.864	0	1449984	32.43	8045.75	2.11E+08	1.77E+08
09:31.0	63.909	0	2383872	33.386	8198.167	6.11E+08	2.45E+08

09:32.0	77.551	0	2506752	32.352	8038.617	2.87E+08	2.28E+08
09:33.0	57.035	0	1769472	35.193	8486.457	6.74E+07	5.94E+07
09:34.0	81.094	0	2211840	32.248	8014.265	7.49E+08	3.50E+08
09:35.0	68.765	0	2850816	33.263	8179.714	1.36E+08	1.01E+08
09:36.0	72.613	0	2138112	31.727	7934.859	7.12E+08	3.13E+08
09:37.0	65.816	0	1646592	34.68	8405.226	3.73E+08	1.52E+08
09:38.0	78.553	0	1351680	32.473	8053.515	5.34E+08	2.96E+08
09:39.0	57.25	0	1204224	33.723	8252.792	2.71E+08	1.12E+08
09:40.0	76.01	0	1183744	32.537	8063.679	5.70E+08	3.13E+08
09:41.0	56.03	0	1298432	33.201	8169.726	1.81E+08	1.55E+08
09:42.0	66.666	0	1744896	32.446	8049.613	6.75E+08	2.76E+08
09:43.0	72	0	958464	32.776	8102.277	1.77E+08	1.43E+08
09:44.0	70.78	0	1695744	32.142	8001.261	6.73E+08	2.74E+08
09:45.0	70.935	0	2801664	33.041	8145.062	1.34E+08	9.97E+07
09:46.0	69.597	0	1155072	32.656	8083.796	7.25E+08	3.26E+08
09:47.0	64.483	0	1277952	35.226	8493.117	1.07E+08	7.16E+07
09:48.0	74.875	0	1449984	32.911	8124.234	7.71E+08	3.73E+08
09:49.0	61.46	0	1400832	35.244	8496.121	1.13E+08	6.36E+07
09:50.0	72.864	0	7495680	32.618	8075.703	7.59E+08	3.74E+08
09:51.0	61.055	0	1769472	33.023	8143.39	2.77E+08	2.42E+08
09:52.0	55.667	0	1376256	35.993	8616.777	3.82E+07	3.82E+07
09:53.0	75.124	0	2285568	34.828	8428.281	7.38E+08	3.39E+08
09:54.0	74.055	0	1081344	36.456	8690.871	7.23E+07	3.72E+07
09:55.0	63.476	0	2850816	35.446	8529.878	4.97E+08	1.95E+08
09:56.0	81	0	1327104	33.127	8157.367	8.48E+08	3.53E+08
09:57.0	81.17	0	958464	33.205	8173.3	1.76E+08	1.06E+08
09:58.0	71.641	0	1523712	32.912	8126.339	7.18E+08	3.19E+08
09:59.0	65.508	0	2236416	33.146	8163.777	1.71E+08	1.36E+08
10:00.0	69.9	0	2260992	32.446	8052.316	7.01E+08	3.02E+08
10:01.0	68.341	0	2211840	32.654	8085.648	2.75E+08	2.39E+08
10:02.0	53.96	0	2064384	35.001	8459.785	8.15E+07	8.15E+07
10:03.0	72.842	0	2310144	33.668	8249.558	7.23E+08	3.23E+08
10:04.0	78.109	0	1597440	33.731	8262.687	1.37E+08	1.03E+08
10:05.0	56.359	0	3883008	35.229	8497.015	1.66E+08	6.49E+07
10:06.0	80.952	0	2285568	34.966	8451.816	1.14E+09	4.46E+08
10:07.0	82.706	0	2850816	33.751	8261.339	2.11E+08	1.73E+08
10:08.0	66.172	0	2138112	36.239	8658.011	6.29E+08	2.50E+08
10:09.0	73.869	0	1400832	33.356	8198.582	3.25E+08	2.37E+08
10:10.0	55	0	1400832	34.269	8344.269	1.54E+08	8.12E+07
10:11.0	72.405	0	1179648	32.847	8114.664	6.74E+08	3.48E+08
10:12.0	63.249	0	1843200	33.919	8288.542	1.00E+08	5.99E+07
10:13.0	73	0	1376256	32.698	8091.027	7.65E+08	3.71E+08
10:14.0	75.561	0	1695744	32.797	8109.937	2.32E+08	1.97E+08
10:15.0	58.481	0	2482176	33.91	8287.437	1.21E+08	5.50E+07
10:16.0	62.849	0	1449984	36.646	8723.531	7.00E+08	2.56E+08
10:17.0	84.328	0	1474560	33.159	8168.386	5.52E+08	2.63E+08
10:18.0	77.329	0	1597440	35.956	8614.417	5.85E+08	2.36E+08
10:19.0	77.135	0	1327104	33.378	8202.933	4.10E+08	2.54E+08
10:20.0	52.882	0	1376256	35.788	8587.242	7.66E+07	6.82E+07
10:21.0	71.534	0	1523712	33.432	8208.847	7.57E+08	3.66E+08
10:22.0	68.734	0	2162688	33.513	8224.871	2.68E+08	2.33E+08
10:23.0	53.787	0	2334720	37.166	8807.16	3.07E+07	3.03E+07
10:24.0	84.343	0	2482176	36.366	8677.027	7.47E+08	3.49E+08
10:25.0	67.326	0	3440640	37.255	8821.757	8.35E+07	4.77E+07
10:26.0	56.641	0	2162688	38.263	8983.156	2.77E+08	1.01E+08
10:27.0	88.5	0	2088960	35.829	8591.863	1.05E+09	4.28E+08
10:28.0	82.619	0	2482176	34.818	8433.839	1.56E+08	9.35E+07
10:29.0	77.171	0	2310144	34.951	8455.222	7.54E+08	3.46E+08
10:30.0	64.339	0	1449984	33.618	8242.82	2.74E+08	2.38E+08
10:31.0	56.218	0	1327104	34.864	8441.468	2.13E+08	9.27E+07
10:32.0	73.857	0	1376256	33.77	8272.328	6.00E+08	3.21E+08
10:33.0	61.904	0	1179648	34.572	8395.269	1.25E+08	8.99E+07
10:34.0	74.623	0	1351680	33.21	8175.195	7.12E+08	3.14E+08
10:35.0	73.684	0	2408448	33.954	8297.296	1.61E+08	1.25E+08
10:36.0	66.243	0	1007616	33.943	8295.621	6.84E+08	2.84E+08
10:37.0	67.005	0	1523712	36.064	8633.433	3.31E+08	1.40E+08
10:38.0	77.499	0	3072000	33.391	8207.375	5.68E+08	2.99E+08
10:39.0	58.706	0	2162688	35.869	8602.46	3.25E+08	1.37E+08

10:40.0	74.686	0	1892352	33.255	8185.945	5.44E+08	2.97E+08
10:41.0	58.793	0	1007616	33.403	8209.515	1.67E+08	1.57E+08
10:42.0	66.832	0	1695744	33.096	8160.851	6.66E+08	2.67E+08
10:43.0	71.246	0	2457600	33.58	8238.152	2.21E+08	1.85E+08
10:44.0	68.781	0	2777088	35.207	8497.664	6.42E+08	2.48E+08
10:45.0	76.395	0	2531328	36.162	8650.593	1.06E+08	5.92E+07
10:46.0	77.02	0	3268608	33.515	8228.73	7.46E+08	3.53E+08
10:47.0	59.85	0	2457600	33.716	8260.441	1.69E+08	1.33E+08
10:48.0	67.336	0	2555904	33.677	8254.488	7.03E+08	3.04E+08
10:49.0	66.919	0	1327104	33.55	8234.261	2.26E+08	1.89E+08
10:50.0	59.749	0	1302528	34.996	8464.816	5.09E+08	2.08E+08
10:51.0	76.942	0	1597440	33.639	8248.66	3.91E+08	2.56E+08
10:52.0	58.457	0	1204224	34.813	8435.933	3.90E+08	1.49E+08
10:53.0	75.438	0	1744896	33.646	8250.015	3.58E+08	1.64E+08
10:54.0	75.939	0	1425408	33.613	8244.855	7.13E+08	3.14E+08
10:55.0	67.75	0	9068544	33.607	8245.562	1.75E+08	1.38E+08
10:56.0	70.398	0	2383872	33.501	8228.792	6.90E+08	2.91E+08
10:57.0	76.142	0	1056768	33.268	8191.324	2.67E+08	2.30E+08
10:58.0	55.334	0	1818624	34.443	8378.511	1.73E+08	7.76E+07
10:59.0	72.335	0	1400832	33.238	8183.621	6.46E+08	3.42E+08
11:00.0	60.353	0	1155072	36.615	8724.875	9.36E+07	5.73E+07
11:01.0	74.875	0	1376256	34.141	8330.921	6.67E+08	2.69E+08
11:02.0	73.618	0	1204224	35.613	8570.539	7.46E+08	3.10E+08
11:03.0	67.331	0	1327104	34.008	8309.597	2.43E+08	2.06E+08
11:04.0	65.422	0	1843200	35.205	8500.605	4.74E+08	1.81E+08
11:05.0	75.888	0	2777088	33.939	8299.355	4.17E+08	2.73E+08
11:06.0	57.25	0	1769472	35.025	8472.542	2.06E+08	9.35E+07
11:07.0	79.648	0	1499136	33.691	8256.582	6.10E+08	3.24E+08
11:08.0	64.864	0	1007616	33.833	8282.246	1.94E+08	1.57E+08
11:09.0	54.114	0	1794048	35.301	8516.324	3602085	3155508
11:10.0	73.604	0	1818624	36.546	8714.972	1.26E+09	4.68E+08
11:11.0	78.802	0	2334720	34.2	8341.332	1.31E+08	1.26E+08
11:12.0	83.668	0	1597440	35.166	8495.328	7.73E+08	2.99E+08
11:13.0	73.791	0	2113536	33.607	8246.984	2.39E+08	2.02E+08
11:14.0	64.764	0	2359296	35.675	8576.722	3.37E+08	1.42E+08
11:15.0	77.664	0	3710976	33.658	8255.875	5.03E+08	2.83E+08
11:16.0	62.591	0	1720320	33.588	8244.757	2.64E+08	2.44E+08
11:17.0	57.25	0	2949120	35.176	8497.597	6.64E+07	6.64E+07
11:18.0	67.839	0	2039808	35.058	8475.949	6.98E+08	2.99E+08
11:19.0	66.666	0	2187264	35.812	8599.281	8.58E+07	4.11E+07
11:20.0	63.727	0	2039808	38.41	9013.433	6.81E+08	2.41E+08
11:21.0	85.964	0	1941504	34.078	8320.167	6.57E+08	3.07E+08
11:22.0	77	0	1327104	36.912	8774.96	4.80E+08	1.95E+08
11:23.0	76.543	0	1400832	34.329	8363.429	5.00E+08	2.85E+08
11:24.0	60.794	0	1327104	34.329	8363.492	2.29E+08	2.18E+08
11:25.0	59.445	0	2752512	35.992	8629.101	2.61E+08	1.10E+08
11:26.0	79.648	0	1056768	34.351	8367.613	5.98E+08	3.24E+08
11:27.0	63.567	0	983040	35.441	8540.855	8.75E+07	5.00E+07
11:28.0	77.02	0	1474560	34.172	8335.664	7.39E+08	3.66E+08
11:29.0	60.552	0	2138112	35.491	8548.972	1.80E+08	7.72E+07
11:30.0	74.111	0	1425408	34.078	8323.917	6.14E+08	2.80E+08
11:31.0	71.356	0	1449984	34.213	8345.648	7.29E+08	2.92E+08
11:32.0	74.559	0	983040	33.946	8308.101	1.91E+08	1.54E+08
11:33.0	68.686	0	1253376	33.731	8268.894	6.65E+08	2.66E+08
11:34.0	78.802	0	1843200	33.994	8311.527	2.60E+08	2.22E+08
11:35.0	54.156	0	3022848	35.811	8601.226	8.65E+07	8.65E+07
11:36.0	71.177	0	2310144	34.443	8379.894	7.31E+08	3.32E+08
11:37.0	66.75	0	2703360	34.824	8443.679	1.88E+08	1.50E+08
11:38.0	66.75	0	3391488	36.073	8642.996	4.92E+08	1.99E+08
11:39.0	67.91	0	2703360	37.434	8860.007	6.57E+08	2.49E+08
11:40.0	83.123	0	2678784	34.307	8361.75	3.10E+08	1.75E+08
11:41.0	74.371	0	1662976	35.177	8500.515	7.26E+08	2.89E+08
11:42.0	76.178	0	1753088	33.893	8296.054	2.41E+08	2.07E+08
11:43.0	52.631	0	2433024	35.887	8614.046	7.36E+07	7.02E+07
11:44.0	79.949	0	2531328	33.661	8256.566	7.42E+08	3.43E+08
11:45.0	70.25	0	3661824	33.976	8310.343	2.01E+08	1.64E+08
11:46.0	63.775	0	1867776	33.793	8281.289	6.53E+08	2.55E+08
11:47.0	72.568	0	1548288	34.238	8351.828	2.79E+08	2.41E+08

11:48.0	57.213	0	1523712	34.483	8390.894	1.27E+08	5.69E+07
11:49.0	60.902	0	1425408	36.972	8787.789	5.40E+08	1.95E+08
11:50.0	87.277	0	1572864	33.732	8268.531	7.15E+08	3.33E+08
11:51.0	80.15	0	1327104	35.196	8504.859	1.61E+08	8.41E+07
11:52.0	75.438	0	1400832	32.475	8071.265	7.54E+08	3.55E+08
11:53.0	60.913	0	1499136	32.617	8093.929	2.40E+08	2.02E+08
11:54.0	68.09	0	2678784	33.758	8276.16	5.09E+08	2.03E+08
11:55.0	72.681	0	3563520	32.616	8094.71	3.78E+08	2.46E+08
11:56.0	62.121	0	2138112	33.584	8249.097	4.70E+08	1.86E+08
11:57.0	72.139	0	2113536	32.382	8057.125	4.34E+08	2.81E+08
11:58.0	49.748	0	2187264	35.927	8622.41	5.83E+07	5.83E+07
11:59.0	64.588	0	8773632	33.907	8301.609	5.92E+08	2.37E+08
12:00.0	78.085	0	2211840	33.69	8267.07	7.30E+08	2.88E+08
12:01.0	81.565	0	1744896	32.942	8148.062	1.98E+08	1.50E+08
12:02.0	73.067	0	2187264	32.461	8076.632	7.12E+08	2.84E+08
12:03.0	70.324	0	2088960	32.734	8115.144	2.80E+08	2.42E+08
12:04.0	61.083	0	2088960	33.728	8274.218	1.94E+08	8.11E+07
12:05.0	77.586	0	2555904	32.616	8096.984	6.26E+08	3.34E+08
12:06.0	61.012	0	1548288	33.939	8307.566	3.07E+08	1.18E+08
12:07.0	74.242	0	1376256	32.712	8112.062	5.63E+08	3.21E+08
12:08.0	65.5	0	1179648	33.356	8214.734	1.95E+08	1.56E+08
12:09.0	66.831	0	1228800	32.568	8089.378	6.67E+08	2.68E+08
12:10.0	75.566	0	1818624	35.784	8601.914	3.07E+08	1.30E+08
12:11.0	73.827	0	909312	33.577	8250.359	5.60E+08	3.00E+08
12:12.0	60.25	0	1351680	35.793	8603.574	1.02E+08	6.23E+07
12:13.0	75.373	0	1720320	32.864	8134.019	7.76E+08	3.77E+08
12:14.0	65.336	0	1572864	35.544	8564.308	1.02E+08	6.29E+07
12:15.0	78.894	0	3145728	33.041	8162.312	7.62E+08	3.63E+08
12:16.0	65.404	0	1351680	33.519	8242.062	1.83E+08	1.44E+08
12:17.0	65.491	0	1425408	32.822	8130.558	6.88E+08	2.89E+08
12:18.0	69.23	0	1720320	34.205	8351.035	1.40E+08	1.01E+08
12:19.0	71.929	0	1277952	32.625	8099.308	7.39E+08	3.40E+08
12:20.0	69.191	0	1499136	33.271	8202.367	1.52E+08	1.13E+08
12:21.0	69.576	0	1376256	32.428	8068.097	7.02E+08	3.03E+08
12:22.0	67.676	0	1228800	33.469	8234.113	1.51E+08	1.12E+08
12:23.0	68	0	1277952	32.777	8123.945	7.28E+08	3.29E+08
12:24.0	71.929	0	2482176	33.092	8174.968	1.60E+08	1.21E+08
12:25.0	69.132	0	3809280	32.546	8088.027	6.94E+08	2.95E+08
12:26.0	68.905	0	2359296	33.259	8201.269	1.87E+08	1.48E+08
12:27.0	71.032	0	2285568	32.523	8084.105	6.84E+08	2.85E+08
12:28.0	74.874	0	1843200	32.925	8148.238	2.13E+08	1.74E+08
12:29.0	63.383	0	2064384	33.643	8262.753	6.38E+08	2.60E+08
12:30.0	74.365	0	2113536	33.325	8212.292	2.25E+08	1.65E+08
12:31.0	66.583	0	2490368	32.557	8089.953	6.75E+08	2.76E+08
12:32.0	72.613	0	2154496	32.822	8137.441	2.05E+08	1.66E+08
12:33.0	65.063	0	2138112	33.224	8196.664	6.67E+08	2.68E+08
12:34.0	78.749	0	1548288	33.482	8238.386	2.07E+08	1.68E+08
12:35.0	64.179	0	2777088	33.359	8218.804	6.61E+08	2.62E+08
12:36.0	73.299	0	1695744	33.398	8224.644	2.61E+08	2.21E+08
12:37.0	57.25	0	1277952	34.633	8421.636	4.06E+08	1.57E+08
12:38.0	76.19	0	1720320	33.345	8216.363	4.23E+08	2.47E+08
12:39.0	68.844	0	1744896	33.602	8257.578	4.92E+08	2.03E+08
12:40.0	78.195	0	2236416	32.481	8079.023	4.18E+08	2.55E+08
12:41.0	62.656	0	3293184	33.714	8274.746	4.56E+08	1.78E+08
12:42.0	76.865	0	2408448	32.484	8079.695	3.69E+08	2.22E+08
12:43.0	66.08	0	1671168	34.284	8366.664	5.11E+08	2.02E+08
12:44.0	82.5	0	1818624	33.086	8175.925	3.62E+08	2.43E+08
12:45.0	65.664	0	3883008	33.946	8313.058	3.46E+08	1.43E+08
12:46.0	79.75	0	2138112	32.881	8144.074	5.86E+08	3.25E+08
12:47.0	55.276	0	2113536	33.642	8265.445	2.32E+08	9.92E+07
12:48.0	76.275	0	2187264	32.55	8090.996	5.95E+08	3.19E+08
12:49.0	57.816	0	1425408	34.854	8458.367	1.35E+08	7.10E+07
12:50.0	74.754	0	1253376	33.546	8250.023	7.10E+08	3.46E+08
12:51.0	68.069	0	1646592	34.59	8416.539	3.88E+08	1.50E+08
12:52.0	76.426	0	1179648	33.244	8202.039	5.34E+08	3.07E+08
12:53.0	61.055	0	2138112	35.4	8545.777	9.57E+07	5.67E+07
12:54.0	82.338	0	2359296	32.77	8123.875	7.27E+08	3.53E+08
12:55.0	63.591	0	3317760	35.834	8615.363	1.15E+08	6.68E+07

12:56.0	71.898	0	2138112	32.459	8074.48	7.68E+08	3.77E+08
12:57.0	64.588	0	1744896	34.742	8442.109	1.27E+08	8.73E+07
12:58.0	72.335	0	2088960	33.315	8214.222	7.42E+08	3.43E+08
12:59.0	66	0	1449984	34.287	8369.253	1.17E+08	6.94E+07
13:00.0	74.494	0	1425408	33.066	8174.671	7.69E+08	3.78E+08
13:01.0	61.25	0	1966080	35.961	8636.343	1.18E+08	7.73E+07
13:02.0	76.942	0	1277952	33.369	8225.347	7.57E+08	3.58E+08
13:03.0	62.842	0	1646592	36.001	8647.906	1.09E+08	6.86E+07
13:04.0	81.612	0	7692288	32.944	8153.683	7.60E+08	3.61E+08
13:05.0	64.898	0	2506752	34.001	8325.199	1.52E+08	1.12E+08
13:06.0	67.83	0	1081344	32.816	8133.933	7.22E+08	3.23E+08
13:07.0	71.536	0	860160	33.443	8236.515	1.47E+08	1.07E+08
13:08.0	70.025	0	2408448	32.341	8061.003	7.15E+08	3.17E+08
13:09.0	67.167	0	2383872	33.592	8260.476	1.49E+08	1.09E+08
13:10.0	69.72	0	2482176	32.251	8047	7.21E+08	3.22E+08
13:11.0	75.921	0	3170304	33.41	8231.812	1.60E+08	1.19E+08
13:12.0	65.909	0	2285568	33.27	8209.746	6.97E+08	2.98E+08
13:13.0	71.032	0	2531328	33.464	8240.628	2.05E+08	1.65E+08
13:14.0	72.704	0	2678784	33.292	8213.488	6.68E+08	2.69E+08
13:15.0	77.556	0	3932160	33.283	8212.726	2.17E+08	1.76E+08
13:16.0	64.824	0	2088960	33.041	8174.253	6.54E+08	2.55E+08
13:17.0	71.5	0	2236416	33.327	8219.48	2.14E+08	1.73E+08
13:18.0	63.104	0	2064384	32.871	8146.972	6.76E+08	2.77E+08
13:19.0	73.282	0	2359296	33.457	8240.527	1.99E+08	1.58E+08
13:20.0	65.491	0	2064384	33.597	8263.085	6.71E+08	2.72E+08
13:21.0	73.934	0	1990656	33.271	8211.289	2.10E+08	1.84E+08
13:22.0	65.162	0	2408448	34.366	8385.941	5.90E+08	2.30E+08
13:23.0	74.177	0	2285568	33.293	8215.113	3.11E+08	2.16E+08
13:24.0	68.09	0	1892352	34.416	8394.136	5.39E+08	2.05E+08
13:25.0	76.262	0	3612672	33.171	8195.781	3.15E+08	2.24E+08
13:26.0	66.25	0	1302528	34.672	8435.707	4.93E+08	1.88E+08
13:27.0	73.5	0	1966080	33.467	8243.218	3.92E+08	2.42E+08
13:28.0	61.111	0	1327104	34.669	8434.894	5.44E+08	2.14E+08
13:29.0	73.566	0	1105920	33.537	8254.507	3.68E+08	2.59E+08
13:30.0	55.889	0	1425408	35.125	8507.812	2.62E+08	1.09E+08
13:31.0	76.691	0	1351680	33.975	8324.625	5.91E+08	3.19E+08
13:32.0	57.506	0	1646592	34.905	8477.972	3.05E+08	1.26E+08
13:33.0	73.214	0	1499136	33.701	8281.035	5.49E+08	3.00E+08
13:34.0	63.25	0	1720320	34.734	8445.914	3.17E+08	1.26E+08
13:35.0	79.493	0	2580480	33.523	8253.359	5.61E+08	3.12E+08
13:36.0	64.764	0	1204224	34.454	8401.949	2.84E+08	1.14E+08
13:37.0	74.494	0	1843200	33.382	8230.64	5.83E+08	3.19E+08
13:38.0	66.084	0	2187264	34.876	8468.867	3.13E+08	1.22E+08
13:39.0	75.443	0	1400832	33.752	8289.734	5.67E+08	3.18E+08
13:40.0	56.281	0	1449984	36.506	8728.753	1.05E+08	6.99E+07
13:41.0	74.811	0	1499136	33.474	8242.765	7.72E+08	3.73E+08
13:42.0	63.341	0	2383872	35.072	8500.519	2.21E+08	9.03E+07
13:43.0	76.574	0	2187264	33.834	8303.285	6.63E+08	3.47E+08
13:44.0	62.842	0	2629632	34.757	8450.582	1.22E+08	6.79E+07
13:45.0	77.157	0	3293184	33.507	8248.363	7.43E+08	3.64E+08
13:46.0	68.25	0	2334720	36.312	8699.128	9.85E+07	5.66E+07
13:47.0	77.611	0	2138112	32.794	8138.183	7.72E+08	3.74E+08
13:48.0	61.5	0	2113536	34.722	8445.558	1.43E+08	1.01E+08
13:49.0	71.212	0	2285568	32.735	8129.042	7.30E+08	3.32E+08
13:50.0	64.912	0	2260992	35.54	8576.285	1.28E+08	8.61E+07
13:51.0	72.388	0	2260992	32.761	8133.562	7.49E+08	3.50E+08
13:52.0	63.104	0	2187264	36.134	8671.253	1.23E+08	8.11E+07
13:53.0	73.25	0	2138112	33.431	8240.582	7.57E+08	3.58E+08
13:54.0	69.326	0	2383872	34.254	8371.933	1.41E+08	9.87E+07
13:55.0	68.159	0	3342336	33.307	8218.046	7.11E+08	3.13E+08
13:56.0	74.563	0	1646592	33.527	8256.851	2.01E+08	1.59E+08
13:57.0	64.646	0	2260992	34.016	8334.898	6.68E+08	2.69E+08
13:58.0	77.694	0	2531328	34.084	8345.429	1.97E+08	1.55E+08
13:59.0	67.512	0	2285568	33.979	8328.789	6.70E+08	2.71E+08
14:00.0	75.93	0	1769472	33.938	8322.417	1.78E+08	1.51E+08
14:01.0	68.5	0	2064384	33.793	8299.429	6.84E+08	2.69E+08
14:02.0	72.208	0	8380416	33.968	8333.601	2.19E+08	1.77E+08
14:03.0	64.393	0	1818624	33.484	8251.402	6.66E+08	2.67E+08

14:04.0	80.653	0	1351680	33.499	8253.894	2.63E+08	2.21E+08
14:05.0	60.099	0	2703360	35.045	8500.476	2.81E+08	1.16E+08
14:06.0	77	0	1400832	33.946	8325.351	5.19E+08	2.73E+08
14:07.0	66.246	0	1794048	35.041	8499.964	5.16E+08	2.00E+08
14:08.0	75.949	0	1400832	33.99	8332.542	4.19E+08	2.63E+08
14:09.0	61.442	0	1302528	35.119	8512.625	4.68E+08	1.74E+08
14:10.0	74.873	0	1228800	33.411	8240.582	3.91E+08	2.59E+08
14:11.0	64.691	0	1007616	34.044	8341.367	3.16E+08	1.30E+08
14:12.0	76.381	0	1155072	32.89	8157.652	5.72E+08	3.16E+08
14:13.0	56.06	0	1597440	34.514	8416.484	9.81E+07	5.89E+07
14:14.0	83.25	0	2064384	33.341	8226.691	7.34E+08	3.58E+08
14:15.0	70.707	0	3194880	36.141	8676.125	9.34E+07	5.05E+07
14:16.0	76.5	0	2039808	33.396	8238.656	7.74E+08	3.75E+08
14:17.0	62.311	0	2162688	36.152	8678.66	1.12E+08	6.21E+07
14:18.0	76.53	0	2605056	33.588	8266.664	7.67E+08	3.76E+08
14:19.0	62.311	0	2285568	36.173	8681.683	1.22E+08	7.92E+07
14:20.0	71.712	0	7643136	33.805	8301.539	7.63E+08	3.64E+08
14:21.0	62.972	0	2.12E+07	35.778	8619.148	1.26E+08	8.30E+07
14:22.0	74.064	0	3342336	33.473	8248.894	7.49E+08	3.50E+08
14:23.0	66.666	0	2359296	35.515	8577.472	1.35E+08	9.26E+07
14:24.0	77.75	0	2260992	33.655	8281.132	7.37E+08	3.38E+08
14:25.0	73.618	0	2899968	35.452	8567.792	1.22E+08	7.88E+07
14:26.0	72.749	0	2408448	33.54	8263.152	7.57E+08	3.58E+08
14:27.0	66.583	0	2138112	33.875	8317.222	1.69E+08	1.26E+08
14:28.0	70.471	0	3317760	33.202	8209.921	7.14E+08	3.15E+08
14:29.0	72.04	0	1941504	33.872	8316.41	1.62E+08	1.19E+08
14:30.0	70	0	2088960	32.625	8117.777	7.14E+08	3.15E+08
14:31.0	68.407	0	2113536	33.962	8331.136	1.72E+08	1.29E+08
14:32.0	66.75	0	2088960	34.006	8343.265	7.09E+08	3.10E+08
14:33.0	66.666	0	1671168	34.809	8471.484	1.51E+08	1.08E+08
14:34.0	76.07	0	1499136	33.674	8285.55	7.24E+08	3.25E+08
14:35.0	68.079	0	2924544	34.074	8349.488	1.93E+08	1.50E+08
14:36.0	66.417	0	1155072	34.217	8372.343	6.86E+08	2.88E+08
14:37.0	69.974	0	1499136	34.501	8418.207	2.12E+08	1.69E+08
14:38.0	65.822	0	1892352	34.342	8392.351	6.61E+08	2.62E+08
14:39.0	71.82	0	1474560	34.678	8446.117	2.05E+08	1.61E+08
14:40.0	65.648	0	1425408	34.583	8431.152	6.87E+08	2.87E+08
14:41.0	73.631	0	1032192	33.78	8303.203	1.89E+08	1.46E+08
14:42.0	68.939	0	1499136	33.625	8278.531	6.93E+08	2.94E+08
14:43.0	70.75	0	1548288	33.788	8304.738	2.23E+08	1.79E+08
14:44.0	65.326	0	2555904	34.984	8495.398	4.90E+08	1.97E+08
14:45.0	78.894	0	3022848	33.683	8288.125	3.94E+08	2.61E+08
14:46.0	59.75	0	2088960	35.122	8521.164	4.26E+08	1.62E+08
14:47.0	76.309	0	9879552	34.07	8350.71	4.06E+08	2.28E+08
14:48.0	68.86	0	2588672	35.266	8540.933	5.67E+08	2.33E+08
14:49.0	78.446	0	2940928	34.065	8349.589	3.52E+08	2.26E+08
14:50.0	61.265	0	2211840	35.244	8537.746	5.52E+08	2.18E+08
14:51.0	75.634	0	2310144	34.099	8355.363	3.30E+08	2.38E+08
14:52.0	62.907	0	2457600	34.253	8379.992	2.94E+08	1.16E+08
14:53.0	77.638	0	2285568	33.255	8221.105	5.46E+08	3.07E+08
14:54.0	65.664	0	2285568	35.251	8539.511	1.32E+08	6.30E+07
14:55.0	76.962	0	3956736	33.916	8323.789	7.32E+08	3.59E+08
14:56.0	60.814	0	2506752	36.732	8778.843	1.17E+08	7.30E+07
14:57.0	75.689	0	2236416	33.724	8296.972	7.15E+08	3.16E+08
14:58.0	72.681	0	2555904	36.498	8739.41	6.66E+08	2.67E+08
14:59.0	78	0	2850816	34.109	8358.007	3.05E+08	2.35E+08
15:00.0	53.634	0	2383872	36.669	8766.441	9.17E+07	7.42E+07
15:01.0	75.81	0	2408448	34.355	8394.894	7.56E+08	3.57E+08
15:02.0	63.316	0	2187264	35.052	8514.023	1.79E+08	1.35E+08
15:03.0	64.898	0	2457600	34.38	8402.089	6.97E+08	2.98E+08
15:04.0	79.04	0	1400832	34.477	8417.539	1.42E+08	9.82E+07
15:05.0	74.185	0	9461760	34.179	8371.171	7.18E+08	3.19E+08
15:06.0	67.326	0	1572864	36.671	8768.964	2.36E+08	1.09E+08
15:07.0	76.942	0	1671168	34.176	8370.945	6.47E+08	3.31E+08
15:08.0	61.152	0	1867776	34.724	8458.3	1.95E+08	1.51E+08
15:09.0	67.676	0	1744896	33.991	8341.535	6.83E+08	2.84E+08
15:10.0	71.969	0	2433024	34.409	8408.39	2.08E+08	1.64E+08
15:11.0	66.501	0	1564672	33.958	8336.546	6.67E+08	2.68E+08

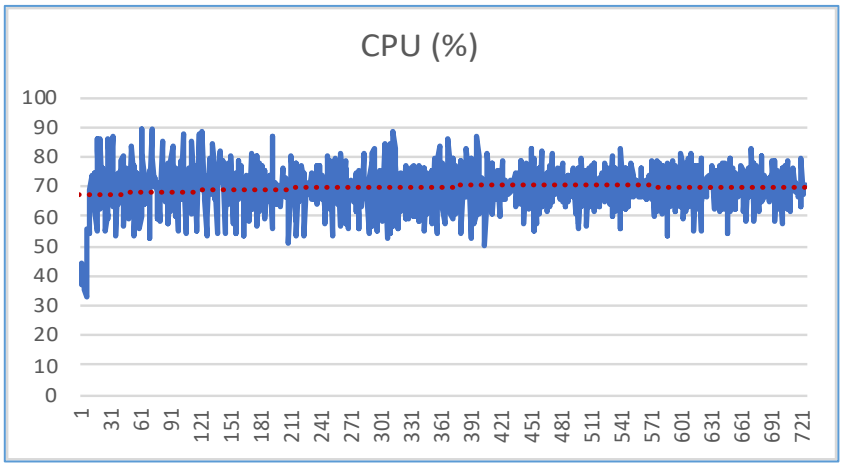
15:12.0	73.086	0	1581056	34.47	8418.214	2.16E+08	1.72E+08
15:13.0	63.065	0	1720320	34.446	8414.519	6.69E+08	2.70E+08
15:14.0	81.047	0	2088960	35.903	8646.839	1.06E+08	8.00E+07
15:15.0	77	0	3244032	33.905	8328.449	7.54E+08	3.38E+08
15:16.0	69.478	0	1769472	34.125	8364.042	2.54E+08	2.09E+08
15:17.0	59.057	0	2334720	35.098	8519.269	4.84E+08	1.91E+08
15:18.0	78.14	0	2498560	34.038	8350.074	3.85E+08	2.52E+08
15:19.0	61.633	0	2048000	35.576	8595.289	3.21E+08	1.30E+08
15:20.0	79.699	0	2310144	34.36	8401.406	5.33E+08	2.81E+08
15:21.0	62.871	0	2236416	34.403	8408.578	5.05E+08	1.98E+08
15:22.0	72.795	0	2039808	32.935	8174.582	3.61E+08	2.07E+08
15:23.0	65.586	0	2064384	32.838	8159.183	6.71E+08	2.72E+08
15:24.0	80.952	0	1941504	33.104	8201.82	2.48E+08	2.22E+08
15:25.0	59.798	0	3661824	34.084	8357.91	2.34E+08	9.51E+07
15:26.0	77.135	0	2359296	33.024	8189.164	6.10E+08	3.22E+08
15:27.0	55.276	0	2064384	36.042	8670.675	1.03E+08	6.79E+07
15:28.0	76.826	0	2162688	33.502	8262.472	7.65E+08	3.66E+08
15:29.0	62.311	0	2023424	35.969	8658.757	1.15E+08	6.54E+07
15:30.0	72.979	0	2424832	33.424	8253.121	7.35E+08	3.41E+08
15:31.0	74.129	0	2727936	34.695	8455.812	5.48E+08	2.13E+08
15:32.0	77.078	0	9486336	33.532	8275.566	4.04E+08	2.70E+08
15:33.0	54.887	0	9117696	36.288	8709.941	8.29E+07	6.39E+07
15:34.0	79.85	0	1327104	33.476	8258.695	7.51E+08	3.53E+08
15:35.0	65.228	0	2949120	34.901	8488.89	1.49E+08	1.04E+08
15:36.0	70.304	0	1351680	32.977	8179.265	7.32E+08	3.33E+08
15:37.0	70.025	0	1179648	33.648	8289.8	1.72E+08	1.27E+08
15:38.0	66.256	0	1155072	33.475	8261.769	6.97E+08	2.98E+08
15:39.0	70.646	0	3072000	36.201	8696.457	2.32E+08	1.06E+08
15:40.0	73.182	0	2506752	33.479	8262.675	6.51E+08	3.33E+08
15:41.0	73.924	0	2482176	33.844	8320.894	1.73E+08	1.27E+08
15:42.0	69.423	0	3366912	33.808	8315.417	6.93E+08	2.94E+08
15:43.0	71.212	0	2408448	33.903	8333.464	2.15E+08	1.69E+08
15:44.0	72.208	0	2875392	33.953	8341.527	6.62E+08	2.63E+08
15:45.0	76.942	0	4472832	34.069	8357.652	2.44E+08	1.99E+08
15:46.0	61.964	0	2334720	35.552	8593.992	4.84E+08	1.89E+08
15:47.0	75.75	0	2113536	34.354	8402.781	2.85E+08	1.55E+08
15:48.0	74.75	0	1.01E+07	33.746	8305.972	7.41E+08	3.23E+08
15:49.0	69.576	0	2260992	33.755	8307.503	2.47E+08	2.02E+08
15:50.0	63.888	0	2506752	34.86	8483.707	4.98E+08	2.03E+08
15:51.0	75.81	0	2260992	33.811	8316.597	4.08E+08	2.58E+08
15:52.0	61.5	0	2260992	35.274	8549.796	4.65E+08	1.76E+08
15:53.0	77.192	0	2285568	34.076	8359.082	4.04E+08	2.67E+08
15:54.0	62.216	0	2531328	35.27	8549.515	1.90E+08	8.17E+07
15:55.0	76.5	0	2678784	34.067	8357.824	5.78E+08	2.57E+08
15:56.0	78.054	0	1007616	33.404	8252.503	7.34E+08	3.00E+08
15:57.0	78.03	0	1548288	33.301	8235.699	2.67E+08	2.21E+08
15:58.0	62.06	0	9019392	34.32	8398.152	4.57E+08	1.78E+08
15:59.0	76.059	0	1474560	33.325	8239.457	4.16E+08	2.69E+08
16:00.0	54.385	0	1376256	34.334	8400.566	7.29E+07	4.52E+07
16:01.0	73.924	0	1769472	32.983	8182.335	7.68E+08	3.77E+08
16:02.0	61.557	0	1523712	36.311	8720.753	1.17E+08	7.03E+07
16:03.0	79.292	0	2359296	33.301	8233.019	7.58E+08	3.60E+08
16:04.0	73.299	0	2138112	34.336	8401.167	2.52E+08	1.03E+08
16:05.0	76.426	0	9609216	32.941	8180.316	6.26E+08	3.31E+08
16:06.0	62.562	0	2727936	34.135	8370.781	2.18E+08	9.06E+07
16:07.0	75	0	2236416	32.759	8148.109	6.65E+08	3.47E+08
16:08.0	62.468	0	5677056	34.314	8399.011	1.56E+08	1.10E+08
16:09.0	73.067	0	5824512	33.789	8315.328	7.17E+08	3.18E+08
16:10.0	71.536	0	1794048	33.913	8335.285	1.78E+08	1.32E+08
16:11.0	66.583	0	2039808	33.929	8337.871	6.89E+08	2.90E+08
16:12.0	69	0	2138112	33.878	8329.71	1.62E+08	1.16E+08
16:13.0	71.82	0	2211840	33.468	8264.433	7.21E+08	3.22E+08
16:14.0	77.386	0	3342336	33.513	8271.796	1.48E+08	1.01E+08
16:15.0	75.689	0	3391488	33.223	8225.601	7.11E+08	3.12E+08
16:16.0	68.527	0	2703360	33.656	8295.234	2.42E+08	1.95E+08
16:17.0	61.845	0	2088960	34.287	8395.843	5.54E+08	2.23E+08
16:18.0	74.12	0	2138112	33.533	8275.179	3.66E+08	2.53E+08
16:19.0	57.999	0	2334720	35.814	8638.902	3.70E+08	1.43E+08

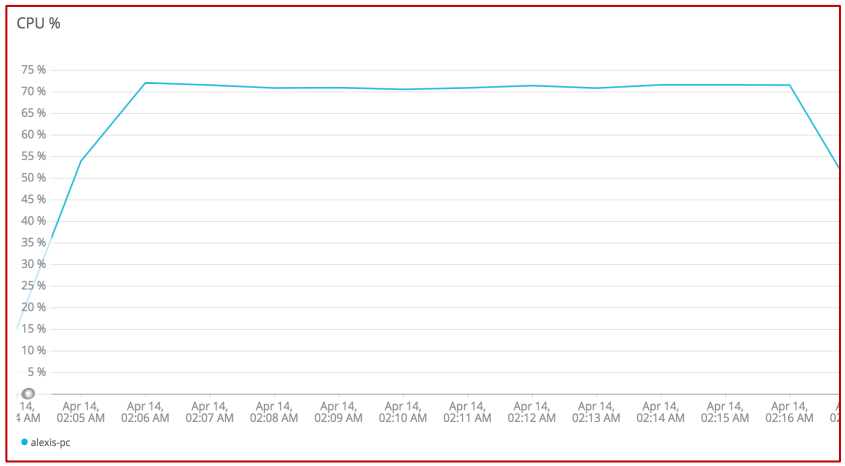
16:20.0	76.543	0	2187264	34.565	8439.871	4.35E+08	2.37E+08
16:21.0	66.501	0	1990656	35.348	8564.843	5.93E+08	2.44E+08
16:22.0	75.062	0	1155072	33.274	8234.14	2.85E+08	1.88E+08
16:23.0	69.576	0	1302528	33.154	8215.039	6.98E+08	2.79E+08
16:24.0	83.123	0	2457600	33.336	8244.085	2.38E+08	1.92E+08
16:25.0	62.907	0	2408448	34.274	8393.609	4.81E+08	1.88E+08
16:26.0	77.777	0	1474560	33.164	8216.839	4.25E+08	2.93E+08
16:27.0	58.291	0	2211840	34.422	8417.839	1.59E+08	7.28E+07
16:28.0	74.554	0	2359296	33.03	8195.632	6.59E+08	3.26E+08
16:29.0	67.085	0	2064384	34.387	8412.019	3.39E+08	1.36E+08
16:30.0	77.386	0	2433024	33.416	8260.253	5.48E+08	2.90E+08
16:31.0	61.809	0	2899968	34.411	8419.07	5.34E+08	2.12E+08
16:32.0	75.308	0	2629632	33.236	8233.976	3.84E+08	2.50E+08
16:33.0	65.835	0	1892352	34.507	8431.671	3.13E+08	1.26E+08
16:34.0	80.604	0	2801664	33.124	8211.21	5.28E+08	2.96E+08
16:35.0	64.572	0	3563520	34.504	8431.718	1.88E+08	1.41E+08
16:36.0	66.832	0	1695744	33.886	8333.152	6.91E+08	2.92E+08
16:37.0	68.058	0	1597440	35.954	8662.597	1.35E+08	8.77E+07
16:38.0	74.111	0	1695744	33.988	8349.078	7.56E+08	3.57E+08
16:39.0	68.922	0	1818624	34.957	8503.64	1.52E+08	1.05E+08
16:40.0	72.592	0	1646592	34.147	8374.601	7.18E+08	3.19E+08
16:41.0	64.572	0	1671168	36.04	8676.308	1.46E+08	9.86E+07
16:42.0	73.267	0	1597440	34.138	8373.144	7.37E+08	3.38E+08
16:43.0	71.144	0	1253376	34.679	8459.441	1.68E+08	1.21E+08
16:44.0	75.062	0	2113536	32.379	8092.937	6.87E+08	2.88E+08
16:45.0	78.643	0	3514368	32.324	8084.23	2.20E+08	1.74E+08
16:46.0	66.25	0	1916928	32.883	8173.414	5.11E+08	2.12E+08
16:47.0	78.894	0	1818624	32.353	8089.445	3.89E+08	2.42E+08
16:48.0	57.999	0	1597440	33.272	8235.457	4.76E+08	1.87E+08
16:49.0	75.628	0	1376256	32.981	8188.988	3.89E+08	2.32E+08
16:50.0	62.626	0	1449984	33.369	8251.046	6.15E+08	2.48E+08
16:51.0	75	0	2457600	32.6	8128.656	2.75E+08	1.96E+08
16:52.0	62.784	0	1474560	33.185	8221.902	6.60E+08	2.61E+08
16:53.0	72.222	0	1425408	32.678	8141.257	2.78E+08	2.31E+08
16:54.0	57.43	0	1818624	34.238	8389.851	2.23E+08	9.10E+07
16:55.0	75.939	0	2506752	32.947	8184.261	5.85E+08	3.18E+08
16:56.0	65.829	0	1351680	35.794	8638.042	1.05E+08	5.73E+07
16:57.0	76.942	0	2629632	32.263	8072.347	7.66E+08	3.68E+08
16:58.0	63.888	0	2555904	33.575	8284.964	3.12E+08	1.21E+08
16:59.0	78.75	0	2482176	32.516	8115.824	5.69E+08	3.07E+08
17:00.0	62.025	0	2433024	34.147	8375.855	2.46E+08	1.07E+08
17:01.0	73	0	2433024	32.948	8181.812	6.30E+08	3.29E+08
17:02.0	61.557	0	2555904	35.734	8634.136	1.25E+08	7.75E+07
17:03.0	74.559	0	2605056	33.396	8253.406	7.55E+08	3.57E+08
17:04.0	73.737	0	2629632	34.099	8368.601	1.73E+08	1.25E+08
17:05.0	69.154	0	3760128	33.546	8280.597	6.86E+08	2.87E+08
17:06.0	76.19	0	2015232	33.087	8208.007	1.77E+08	1.29E+08
17:07.0	68.421	0	2875392	32.843	8169.089	7.01E+08	3.02E+08
17:08.0	71.428	0	9019392	33.085	8208.246	1.84E+08	1.36E+08
17:09.0	69.873	0	2162688	32.719	8149.878	7.14E+08	3.15E+08
17:10.0	69.346	0	1990656	33.242	8233.507	1.98E+08	1.50E+08
17:11.0	66.835	0	1990656	33.016	8197.468	6.90E+08	2.91E+08
17:12.0	70.426	0	2039808	33.167	8221.746	2.34E+08	1.86E+08
17:13.0	63.25	0	1376256	32.891	8177.761	6.54E+08	2.56E+08
17:14.0	79.643	0	1843200	33.413	8260.972	2.15E+08	1.67E+08
17:15.0	67.171	0	2777088	33.302	8243.382	6.55E+08	2.56E+08
17:16.0	70.603	0	1253376	33.01	8197.382	1.82E+08	1.56E+08
17:17.0	70.379	0	2826240	32.69	8146.406	6.97E+08	2.77E+08
17:18.0	70.558	0	1105920	35.359	8571.492	2.96E+08	1.32E+08

	CPU (%)	Disk Read bytes/s	Disk Write bytes/s	Memory (%)	Memory (MB)	Network sent bytes/s	Network received bytes/sent
MIN	32.58	0.00	0.00	10.88	4501.38	0.00	0.00
MAX	89.34	12864.00	231849984.00	38.59	9013.43	1289436817.00	491857961.00
AVG	69.51	56.60	2918823.14	33.63	8247.73	420819801.83	211516570.63
MED	70.45	0.00	2138112.00	33.71	8262.72	384435911.50	217117518.50
STDEV	8.32	726.03	9524478.20	2.75	453.14	256367313.85	100257530.04

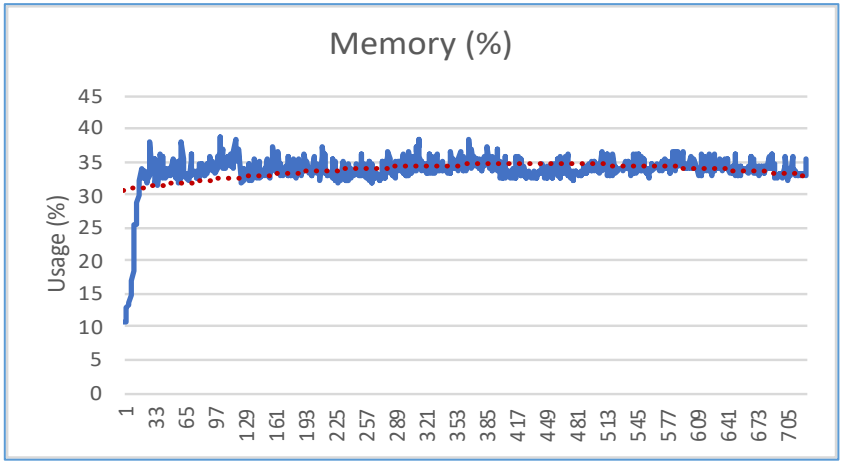
De las tablas anteriores, se presentan gráficas obtenidas de los logs de JMeter con borde azul y obtenidas con el servicio de NewRelic con borde rojo. En los gráficos de NewRelic, se debe considerar solamente la zona con mayor actividad para comparar con los gráficos correspondientes de JMeter.

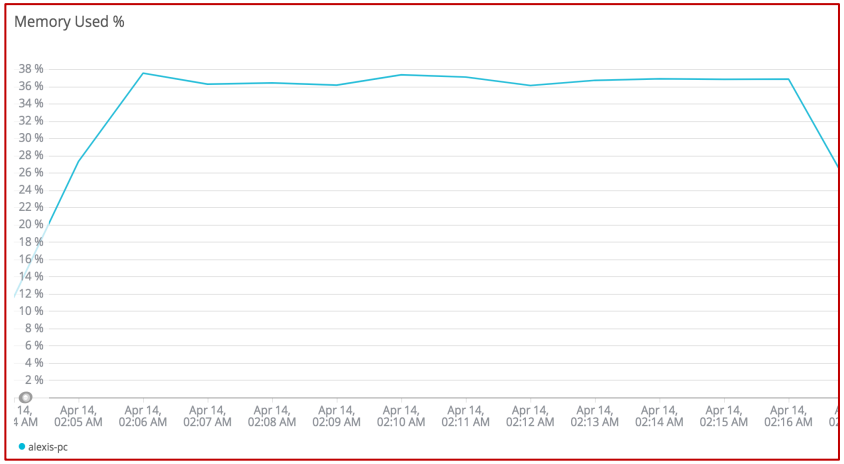
CPU

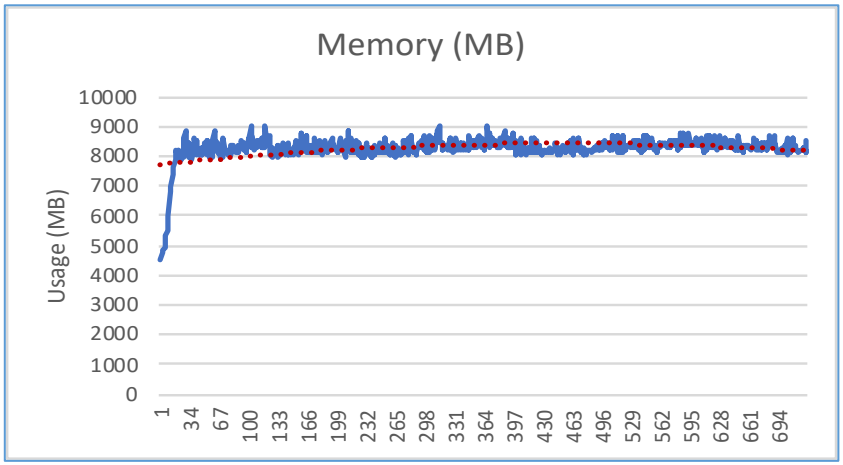


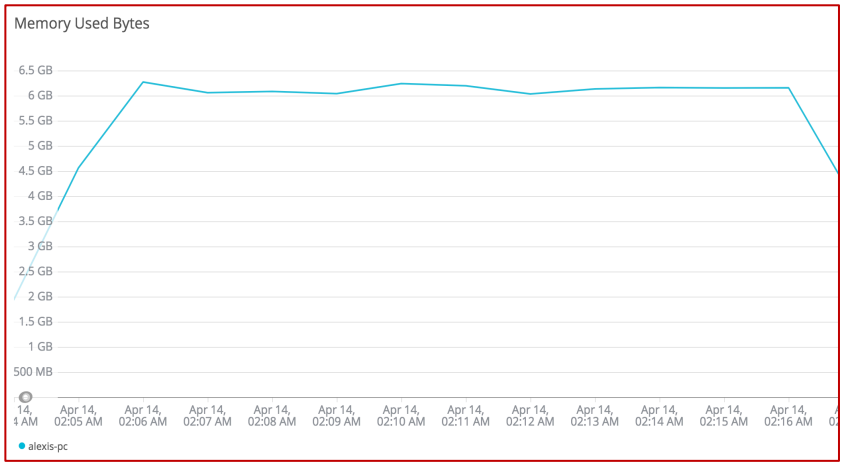


Memoria

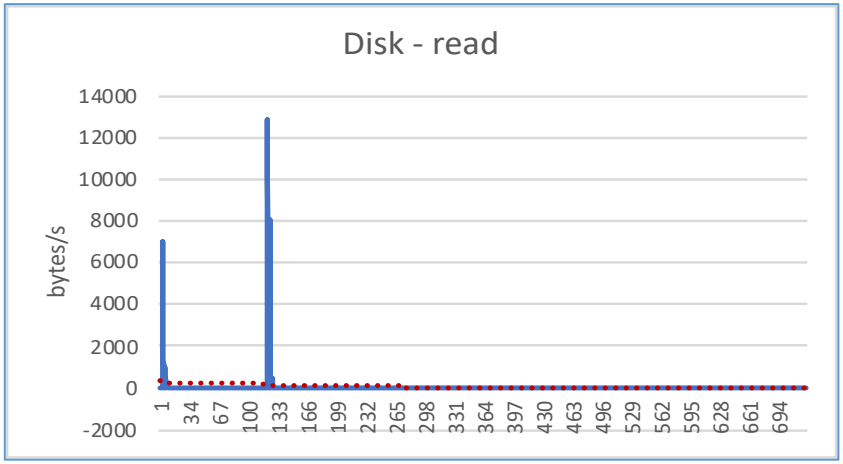

Memory (%)
Usage (%)
45
40
35
30
25
20
15
10
5
0
1
33
65
97
129
161
193
225
257
289
321
353
385
417
449
481
513
545
577
609
641
673
705

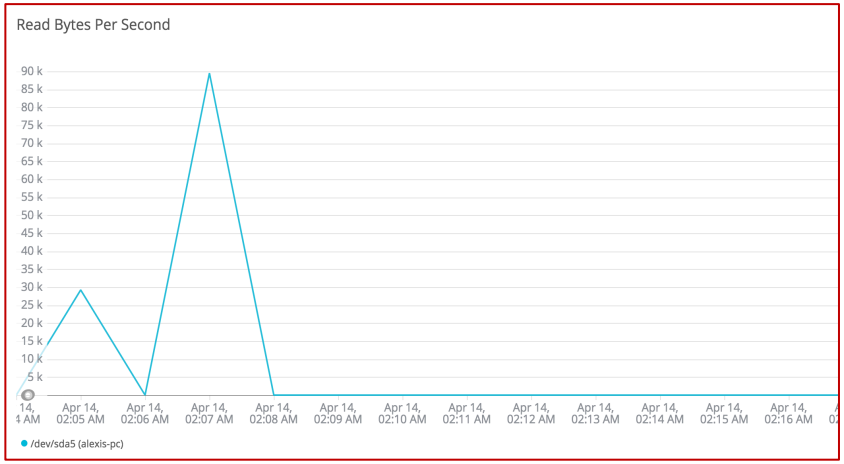

Memory Used %
38 %
36 %
34 %
32 %
30 %
28 %
26 %
24 %
22 %
20 %
18 %
16 %
14 %
12 %
10 %
8 %
6 %
4 %
2 %
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
alexis-pc

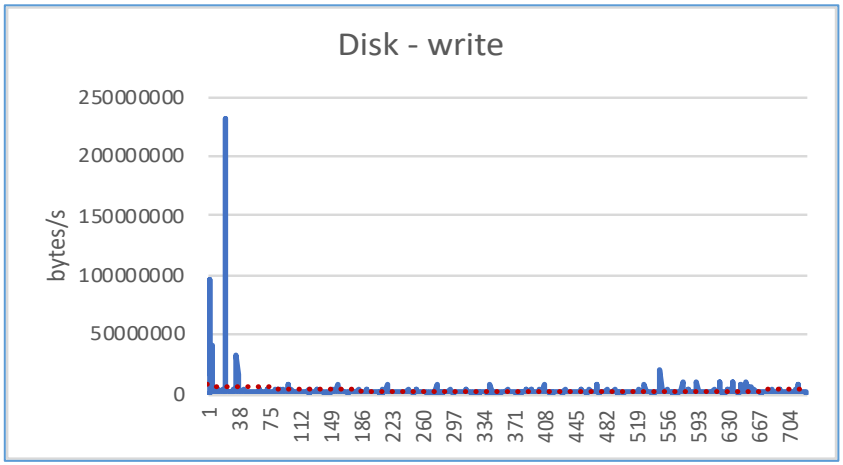

Memory (MB)
Usage (MB)
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694

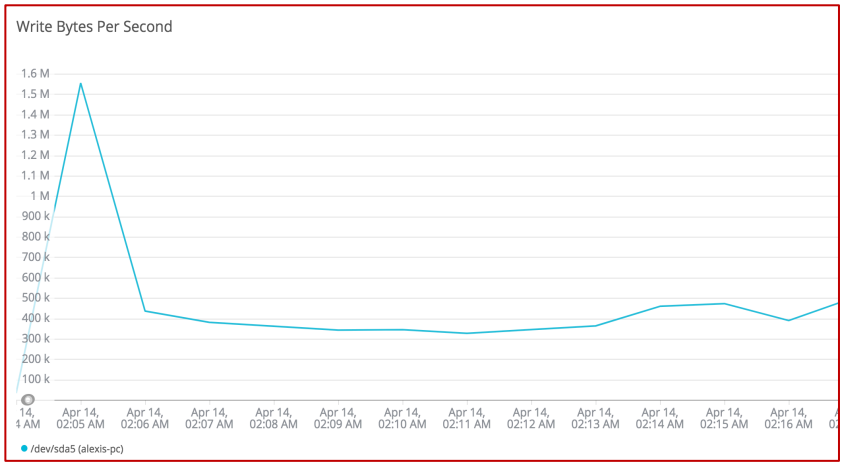

Memory Used Bytes
6.5 GB
6 GB
5.5 GB
5 GB
4.5 GB
4 GB
3.5 GB
3 GB
2.5 GB
2 GB
1.5 GB
1 GB
500 MB
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
alexis-pc

Disco

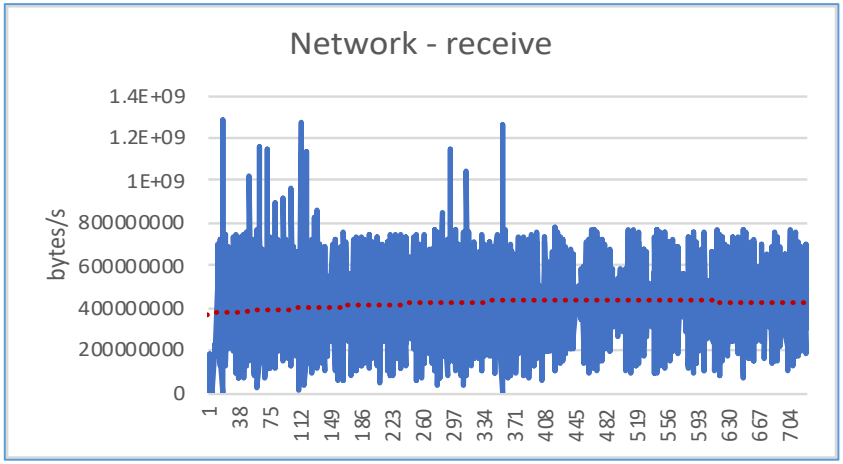

Disk - read
bytes/s
14000
12000
10000
8000
6000
4000
2000
0
-2000
1
34
67
100
133
166
199
232
265
298
331
364
397
430
463
496
529
562
595
628
661
694

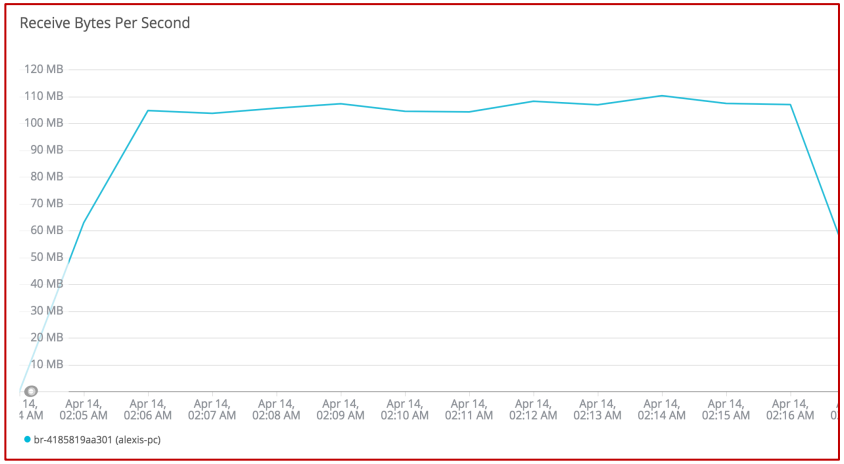

Read Bytes Per Second
90 k
85 k
80 k
75 k
70 k
65 k
60 k
55 k
50 k
45 k
40 k
35 k
30 k
25 k
20 k
15 k
10 k
5 k
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
/dev/sda5 (alexis-pc)

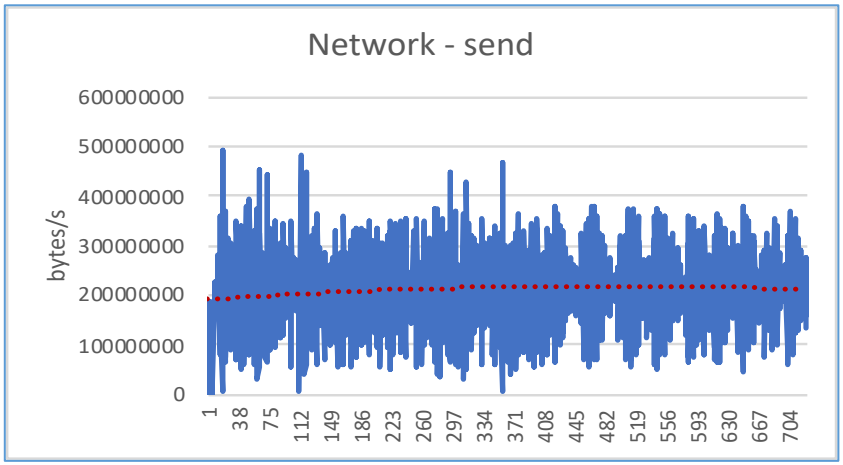

Disk - write
bytes/s
250000000
200000000
150000000
100000000
50000000
0
1
38
75
112
149
186
223
260
297
334
371
408
445
482
519
556
593
630
667
704

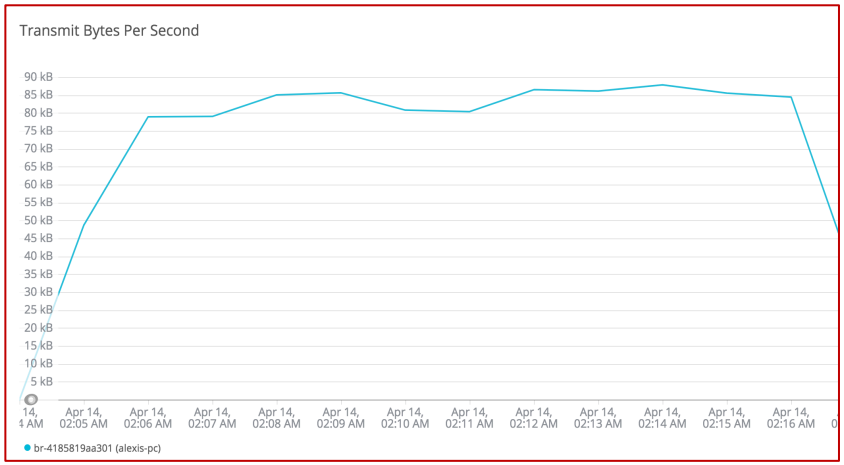

Write Bytes Per Second
1.6 M
1.5 M
1.4 M
1.3 M
1.2 M
1.1 M
1 M
900 k
800 k
700 k
600 k
500 k
400 k
300 k
200 k
100 k
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
/dev/sda5 (alexis-pc)

Red


Network - receive
bytes/s
1.4E+09
1.2E+09
1E+09
800000000
600000000
400000000
200000000
0
1
38
75
112
149
186
223
260
297
334
371
408
445
482
519
556
593
630
667
704


Receive Bytes Per Second
120 MB
110 MB
100 MB
90 MB
80 MB
70 MB
60 MB
50 MB
40 MB
30 MB
20 MB
10 MB
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
br-4185819aa301 (alexis-pc)


Network - send
bytes/s
600000000
500000000
400000000
300000000
200000000
100000000
0
1
38
75
112
149
186
223
260
297
334
371
408
445
482
519
556
593
630
667
704


Transmit Bytes Per Second
90 kB
85 kB
80 kB
75 kB
70 kB
65 kB
60 kB
55 kB
50 kB
45 kB
40 kB
35 kB
30 kB
25 kB
20 kB
15 kB
10 kB
5 kB
Apr 14, 02:05 AM
Apr 14, 02:06 AM
Apr 14, 02:07 AM
Apr 14, 02:08 AM
Apr 14, 02:09 AM
Apr 14, 02:10 AM
Apr 14, 02:11 AM
Apr 14, 02:12 AM
Apr 14, 02:13 AM
Apr 14, 02:14 AM
Apr 14, 02:15 AM
Apr 14, 02:16 AM
br-4185819aa301 (alexis-pc)

UNIVERSIDAD TÉCNICA DEL NORTE

INSTITUTO DE POSTGRADO

MAESTRÍA EN INGENIERÍA DE SOFTWARE

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

ANTEPROYECTO

Trabajo de Investigación previo a la obtención del Título de Magíster en Ingeniería de Software.

Director:
Ing. Freddy Mauricio Tapia León MSc.

Autor:
Ing. Alexis Saransig.

IBARRA - ECUADOR

2017

INDICE DE CONTENIDOS

Tabla de Contenidos

INDICE DE TABLAS

INDICE DE FIGURAS

CAPÍTULO I

EL PROBLEMA

1.10 Tema

"ANÁLISIS DE RENDIMIENTO ENTRE UNA ARQUITECTURA MONOLÍTICA Y UNA ARQUITECTURA DE MICROSERVICIOS – TECNOLOGÍA BASADA EN CONTENEDORES"

1.11 Contextualización del problema

La evolución tecnológica no da tregua, siendo esta una ventaja para la innovación e investigación, donde los límites llegan hasta donde el conocimiento así lo permite. Es así que en la actualidad, dichas innovaciones tienden a hacerse más eficientes y un verdadero soporte al desarrollo de las sociedades. Por mencionar un ejemplo, una de las mayores preocupaciones es el impacto que las nuevas tecnologías puedan causar al medio ambiente, ya en el año de 1992 aparece la TI verde[33], un término usado para hacer referencia al gran impacto ambiental que genera el hacer mal uso de las TI (a nivel social y empresarial). La TI Verde es aplicable a un rango de Dominios de Alta Tecnología, Centros de Datos, Computación Móvil, Sistemas Embebidos y Aplicaciones Web (Basar Bener, Morisio, Miranskyy, & Akinii Kocak, 2014).

Asi también, la producción de Software se ha incrementado de forma exponencial, lo que exige un desarrollo en la infraestructura físico-tangible para su debida operación. Sin embargo, si de optimizar recursos se trata, no se puede limitar a crear Software y/o Hardware a la par, sino más bien pensar en la forma de reutilizar recursos y reducir costos. En la década de los 60's, surgió el mismo planteamiento por parte de IBM[34], dando como resultado el desarrollo del primer método de Virtualización, que consistía en hacer particiones lógicas, donde cada partición trabaje independientemente y haga uso de los recursos compartidos provistos por una supercomputadora, esto implicaría una optimización y un mejor manejo de los recursos (Meier, Virun, Blumert, & Jones, 2008). Desde entonces, la virtualización ha sido parte de las innovaciones tecnológicas que hasta la actualidad conocemos y utilizamos masivamente.

Ahora bien, las Aplicaciones Web son tendencias por sus potentes funcionalidades, adaptabilidad y buena aceptación por parte de las empresas que desarrollan Software, siendo estas aplicaciones, en su mayoría diseñadas con una Arquitectura Monolítica - AM[35] (Ver Anexo 1) como base para su implementación e implantación. Comúnmente esta arquitectura se compone de tres capas: interfaz de usuario, lógica de la aplicación y

[33] TI Verde.- Tecnología de la Información Verde.
[34] IBM (International Business Machine).- Máquina de Negocio Internacional, es una empresa de tecnología y consultoría a nivel mundial.
[35] Arquitectura Monolítica.- Modelo tradicional unificado para diseñar Software.

el sistema gestor de datos, todo esto se ejecuta en un único servidor. Esta última, es la parte clave donde se limita la escalabilidad y existen altos riesgos de que un fallo tenga un impacto crítico a nivel global en el sistema. Ahora, conociendo las capacidades de un servidor, imaginemos que hay varias aplicaciones con tecnología diferente ejecutándose sobre este, y en un intento de optimizar recursos se utilizan Máquinas Virtuales – MV para emular el Hardware subyacente. Aquí se genera la interrogante ¿Se ha optimizado suficiente y adecuadamente los recursos?, más aún cuando todo profesional conoce las consecuencias de virtualizar una AM en una MV. Según Prashant, (2016), previo a cualquier proceso de virtualización, se deben considerar los siguientes recursos como factores importantes de análisis de desempeño:

- CPU
- La Memoria
- La Red
- Disco
- Aplicaciones o Servicios

Con estas consideraciones, podemos hacer mención de algunos puntos problemáticos que conlleva virtualizar una AM usando MV:

- **Gestión eficiente de Recursos (Hardware)**, Las MV requieren en su mayoría una instalación completa del Sistema Operativo, atribuyendo al consumo excesivo de recursos, pues el sistema en ejecución no diferencia una máquina física de una virtual, y dependerá mucho de la configuración para mejorar o limitar su rendimiento.

- **Portabilidad de servicio**, Algunos expertos en Desarrollo de Software optan por hacer una documentación detallada para poder reproducir las mismas condiciones en otra computadora o servidor, mientras otros expertos optan por exportar la MV preconfigurada con la AM, lo que resulta en generar un archivo de gran tamaño difícil de llevar en un disco o memoria USB.

- **Escalabilidad de Servicio**, Es limitado y aunque depende principalmente de los recursos disponibles, una AM no es lo suficientemente flexible como para escalar y su jerarquía vertical la vuelve incluso más complicada de ejecutar este proceso. Posiblemente se piense en hacer una clonación de la MV, sin embargo, esto significa mayor consumo de recursos y podría terminar ralentizando procesos y deteriorando el desempeño alcanzado.

- **Versión de la Aplicación o Servicio**, Las actualizaciones son en su mayoría un problema ya sea por temas de compatibilidad con las versiones anteriores y tiempo de ejecución. Además de que es difícil tener varias versiones de una aplicación en

el mismo sistema, donde el proceso de instalar y desinstalar Software hasta encontrar la versión correcta es algo tedioso y estresante.

- **Compatibilidad Multiplataforma**, Aunque en la actualidad la mayoría de aplicaciones presentan versiones multiplataforma, algunas requieren de software o librerías extras para poder funcionar, esto no asegura tener un ambiente íntegro o libre de errores a futuro.

- **Trabajo en equipo**, Significa que todos los involucrados y usuarios del servicio virtualizado deben de compartir un mismo ambiente, lo cual sería ideal, sin embargo, para ello se necesitaría contar con equipos similares en cuanto a Hardware se refiere. Se lo podría suplir usando MVs, pero con los inconvenientes mencionados sabemos que no es una solución óptima.

El Gráfico 1, muestra los orígenes del problema y sus consecuencias.

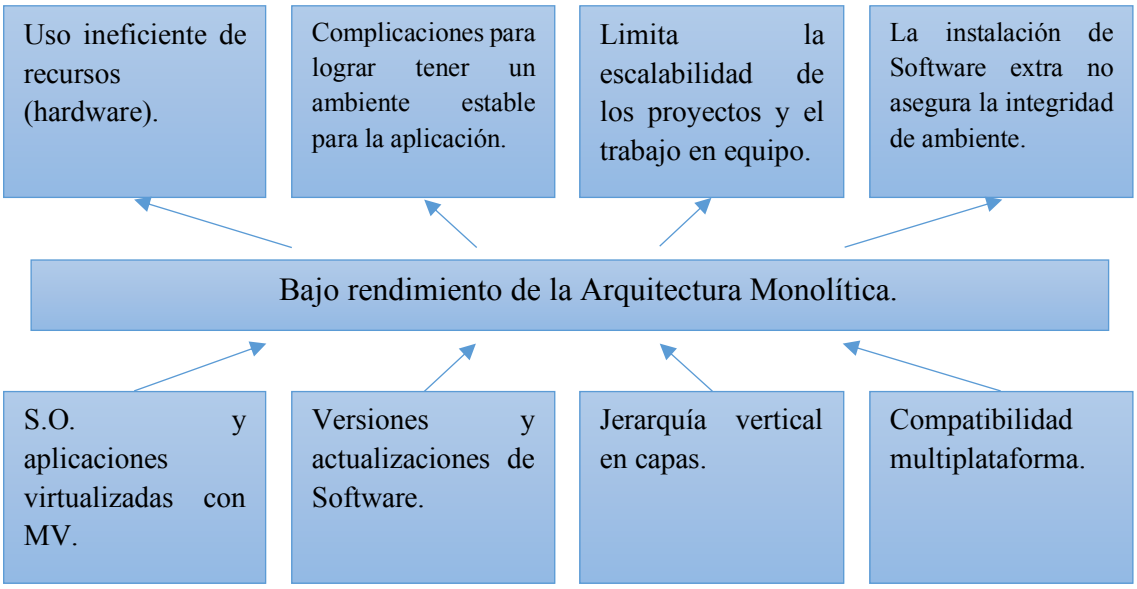


Gráfico 41: Árbol de Problemas.
Elaborado por: Investigador.

1.12 Planteamiento del problema

En el área informática se habla de una innovación constante, siendo el Software un elemento decisorio y clave al momento de implementar propuestas dinámicas, mismas que faciliten y optimicen la Toma de Desiciones. Donde, un modelo adecuado y estable de desarrollo permite una puesta en producción eficiente asegurando resultados y optimización de recursos.

Así también, para la virtualización de servicios, tanto el Hardware como el Software deben cumplir con cierto nivel de compatibilidad, pues el rendimiento dependerá mucho de las características que cada uno tenga, además de que en relación al tiempo no deberían tener mayor diferencia el uno del otro.

Como se ha evidenciado, el tema de rendimiento (Arquitectura Monolítica/Máquina Virtual) es la hipótesis a validar y objetivos de la presente investigación, mismos que influyen directamente en la producción y ejecución de Software, así como de los servicios a ofrecer e impacto que estas producen.

Dicho esto, es una necesidad investigar en esta área, donde el problema está identificado y hay propuestas innovadoras que motivan a involucrarse y aportar con un análisis adecuado muy útil incluso para futuras investigaciones.

1.13 Formulación del problema

Incidencia en el rendimiento de los recursos de hardware con una Arquitectura Monolitica o una Arquitectura de Microservicios basada en contenedores.

1.14 Justificación de la investigación

Los cambios tecnológicos en algunos escenarios permiten la masificación de servicios, asi como también su optimización y fácil acceso, donde la reducción de costos y mejora en los servicios son producto de una adecuada implementación y/o ejecución. Uno de estos servicios es la virtualización basada en contenedores, la cual toma a las Máquinas Virtuales y Arquitecturas Monolíticas, como objetos de estudio en términos de realizar comparativas que ayuden a coincidir y alcanzar los objetivos definidos en esta investigación.

En la presente investigación se propone analizar el uso de contenedores y la Arquitectura de Microservicios – AMS (ver Anexo 2), tendencias relativamente nuevas, las cuales desde su lanzamiento hasta la presente fecha, han sido ampliamente aceptadas no sólo por empresas que desarrollan Software, sino por gigantes tecnológicos como: Google, Microsoft y Amazon, empresas que han decidido migrar y hacer uso de estas tecnologías para ofrecer servicios más óptimos y acorde a las nuevas exigencias tecnológicas.

1.15 Objetivos de la investigación

1.15.1 Objetivo general

Hacer un análisis comparativo de rendimiento entre la Arquitectura de Microservicios basada en contenedores y la Arquitectura Monolítica.

1.15.2 Objetivos específicos

- Estudiar el contexto evolutivo de la Arquitectura de Software, enfatizando en la Arquitectura Monolítica y la de Microservicios, para obtener un Marco Teórico que sustente la presente investigación.
- Desarrollar una investigacion diagnóstica que permita conocer los factores que influyen en el rendimiento de los recursos de hardware.
- Realizar pruebas de desempeño y rendimiento referenciales.
- Analizar y exponer los resultados del análisis camparativo de rendimiento obtenidos.

1.16 Pregunta de Investigación

¿Cómo incide la Arquitetura de Microservicios basada en contenedores en el rendimiento de los recursos de hardware?

1.17 Hipótesis o preguntas directrices

La Arquitectura de Microservicios basada en contenedores mejora el rendimiento en un 15% en comparación a la Arquitectura Monolítica.

Preguntas directrices

- ¿Qué factores influyen en el rendimiento de las Arquitecturas de Software?
- ¿Cuáles son las áreas de influencia de las Arquitecturas de Software?
- ¿Existen investigaciones o trabajos relacionados con la Arquitectura de Software que den soporte a la presente experimentación?
- ¿Cuál es el nivel de conocimientos y aceptación de los microservicios basados en contenedores?
- ¿Es posible reemplazar o migrar las Máquinas Virtuales por Contenedores?

1.18 Variables e indicadores

Independiente: Arquitectura de Microservicios basada en contenedores.

Dependiente: Rendimiento.

Independiente: Arquitectura de Microservicios basada en contenedores.

Conceptualización	Dimensiones	Indicadores	Ítems básicos	Tecnica o instrumento
Está por delante de las virtualizaciones tradicionales en términos de usabilidad, pero esta también extiende la utilización de recursos y por lo tanto reduce el gasto en general de crear nuevos procesos de virtualización. **(Adufu, Choi, & Kim, 2015)**	Virtualización	Nivel de virtualizacion: Kernel, Sistema Operativo, red.	¿A qué nivel virtualizan los contenedores?	Análisis de archivos
	Contenedores	Adaptabilidad multiplataforma: si, no.	¿Los contenedores se adaptan a diferentes plataformas?	Análisis de archivos
	Recursos (Hardware)	Gestión eficiente de recursos: bajo, medio, alto.	¿Cuál es el nivel de eficiencia en el manejo de recursos ?	Análisis de archivos
	Costo - Beneficio	Costos de implantación: bajo, medio, alto.	¿El costo de implementar una AMS es retribuida una vez puesta en producción?	Análisis de archivos

Tabla 33: Indicadores de la variable Independiente.
Elaborado por: Investigador.

Dependiente: Rendimiento.

Conceptualización	Dimensiones	Indicadores	Ítems básicos	Técnicas e instrumentos
Es la proporción entre el producto o el resultado obtenido y los medios utilizados. (ASALE, 2014)	Hardware	Uso de recursos	¿Es posible optimizar el uso de los recursos implementando una AMS?	Análisis de archivos
	Software	Compatibilidad Actualizaciones Estabilidad Portabilidad	¿Con una AMS basada en contenedores, se puede obtener ambientes íntegros para producir Software?	Análisis de archivos

Tabla 34: Indicadores de la variable dependiente.
Elaborado por: Investigador.

CAPÍTULO II

MARCO REFERENCIAL

2.11. Marco Teórico

Antecedentes Investigativos

Como antecedente investigativo, el término de virtualización aparece en los años 60 por parte de IBM, donde la necesidad de optimizar y aprovechar los recursos existentes para esos años era imperioso. El objetivo de virtualizar, es crear una representación virtual, en lugar de realizar varias instalaciones físicas, dependiendo siempre de las características del equipo (VMWare, 2017). En la actualidad es posible virtualizar: servidores, aplicaciones, almacenamiento y redes. Según (Amaral et al., 2015), las Maquinas Virtuales – MV no gestionan eficientemente los recursos y esa ha sido una de las principales desventajas de esta tecnología. Una alternativa actual son los contenedores.

Con la propuesta de los contenedores viene inherente el uso de la Arquitectura de Microservicios. Para el análisis, (Felter, Ferreira, Rajamony, & Rubio, 2015) en su investigación, acuden al uso de trabajos de carga de estrés[36] para la CPU, memoria y red. Para hacer la comparativa por el lado de las Máquinas Virtuales lo más sobresaliente es la tecnología KVM[37], y en cuanto a los contenedores existe la tecnología llamada Docker[38]. Los mismos autores hacen incapie en el uso de los contenedores por ser prometedor y con mayor tendencia de uso a largo plazo. La Arquitectura de Microservicios, es una técnica relativamente nueva de virtualizar a nivel de Sistemas Operativos para ejecutar múltiples programas (contenedores) en aislamiento sobre una simple computadora (Vaucher, 2015).

En su publicación (Dragoni et al., 2016) nos habla de los grandes cambios que han tenido tecnológicamente la Arquitectura Monolitica – AM y la Arquitectura de Microservicios – AMS en el tiempo. Se consideran factores como: recursos, dependencias, despliegue, escalabilidad, entre otros, mismos que de ser un problema en la AM han hallado finalmente una solución en la AMS. El mismo autor, enfatiza la aceptación de la AMS en el mercado, donde grandes compañías han decidido formar parte de esta revolución tecnológica, también aclara que esta arquitectura es relativamente nueva y existe poca literatura relacionada, para lo cual deja temas abiertos en los que se puede contribuir con más investigación.

Los microservicios, según (Amaral et al., 2015), se los considera actualmente como un refinamiento y simplificación de la Arquitectura Orientada a Servicios – SOA, donde

[36] Carga de estrés, son pruebas que por lo general superan los límites determinados con el fin de analizar el respuestas y comportamiento de un sistema informático.
[37] KVM, Kernel Virtual Machine (Máquina Virtual de Núcleo).
[38] Docker, Proyecto de código abierto usado para automatizar el despliegue de aplicaciones usando contenedores.

la idea principal es reemplazar las Aplicaciones Monolíticas por servicios individuales (Microservicios) facilitando la habilidad para actualizar y escalar servicios de manera independiente. Así mismo, propone el uso de Microservicios como una opción efectiva, señalando que son: livianos, rápidos de iniciar y pueden agrupar dependencias y configuraciones dentro de sí mismas.

En las lecturas efectuadas, se señala un punto que marca la transición de una arquitectura a la otra, es decir, la aparición de una tecnología innovadora que merece ser analizada y utilizada. Si bien, los autores dan su punto de vista, es importante señalar que estas tecnologías han tenido una evolución drástica en los últimos años, lo cual motiva a continuar investigando.

Fundamentación Legal

El día jueves 10 de Abril del 2008, se emitió el decreto 1014 por parte de la presidencia del Ec. Rafael Correa Delgado, que promueve el uso de software libre en las instituciones públicas del Ecuador.

Decreto Ejecutivo 1014.
Art. 1.-: Establecer como política pública para las entidades de administración Pública central la utilización del Software Libre en sus sistemas y equipamientos informáticos. **Art. 2.-:** Se entiende por software libre, a los programas de computación que se pueden utilizar y distribuir sin restricción alguna, que permitan el acceso a los códigos fuentes y que sus aplicaciones puedan ser mejoradas. Estos programas de computación tienen las siguientes libertades: • Utilización de programa con cualquier propósito de uso común. • Distribución de copias sin restricción alguna • Estudio y modificación de programa (Requisito: código fuente disponible) • Publicación del programa mejorado (Requisito: código fuente disponible) **Art. 3.-:** Las entidades de la administración pública central previa a la instalación del software libre en sus equipos, deberán verificar la existencia de capacidad técnica que brinde el soporte necesario para este tipo de software. **Art. 4.-:** Se faculta la utilización de software propietario (no libre) únicamente cuando no exista una solución de software libre que supla las necesidades requeridas, o cuando esté en riesgo de seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retorno.

Tabla 35: Decreto Ejecutivo 1014
Realizado por: Investigador

Es necesario mencionar adicionalmente al Reglamento de Régimen Académico con fines de sustentación de la presente investigación.

Reglamento de Régimen Académico.
Artículo 3. **Literal d.** articular la formación académica y profesional, la investigación científica, tecnológica y social, y la vinculación con la colectividad en un marco de calidad, innovación y pertinencia. **Literal h.** Impulsar el conocimiento de carácter mutli, inter y trans disciplinarios en la formación de grado y postragrado, la investigación y la vinculación con la colectividad. **Literal j.** Desarrollar la educación superior bajo la perspectiva del bien público social, aportando a la democratización del conocimiento para la garantía de derechos y la reducción de inequidades.

Tabla 36: Reglamento de Régimen Académico
Realizado por: Investigador

En la Universidad Técnica del Norte - UTN se cuenta con el Plan de Desarrollo Informático, como fuente de apoyo y aseguramiento para que esta investigación sea parte contributiva en otras investigaciones relacionadas.

Plan de Desarrollo Informático UTN 2013 – 2017.
Justificación: Al contar con este instrumento estratégico, se reducirán muchos problemas en el uso de las TICs, evitar inversiones obsoletas, y dar buen uso de las TICs e innovaciones que ayudarán en la calidad de la información que biranda a la UTN el Sistema Integrado de Información de Gestión Universitaria y que descansa en tres elementos: • Exactitud: Información libre de errores, clara, fácil de entender. • Oportunidad: Información disponible cuando y donde se la necesite. • Relevancia: Brindar a cada usuario la información que realmente necesita su trabajo dentro de la Institución. **La misión de la Dirección de Informática** Dentro de lo más importante: • Asegurar la adecuada circulación interna y externa de la información en materia d computación; informando, capacitando y asesorando a funcionarios, docentes y estudiantes de la UTN.

• Proponer y desarrollar proyectos que involucren tecnologías computacionales y de información capaces de elevar la parre académica y de asegurar su competitividad tecnológica a nivel nacional e internacional. • Asesoría previa a la adquisición de Hardware y Software de los servicios de Ciencias de Informática, Computación y Comunicaciones, dando orientación y apoyo a la comunidad universitaria en la necesidad de adquirir equipos y componentes computacionales. Captar el avance tecnológico informático y aplicarlo adecuadamente a la Universidad en base a los requerimientos, así como sentar las bases para el desarrollo futuro.

Tabla 37: Plan de Desarrollo Informático UTN 2013 – 2017.
Realizado por: Investigador

2.12. Esquema del Marco Teórico de la Investigación

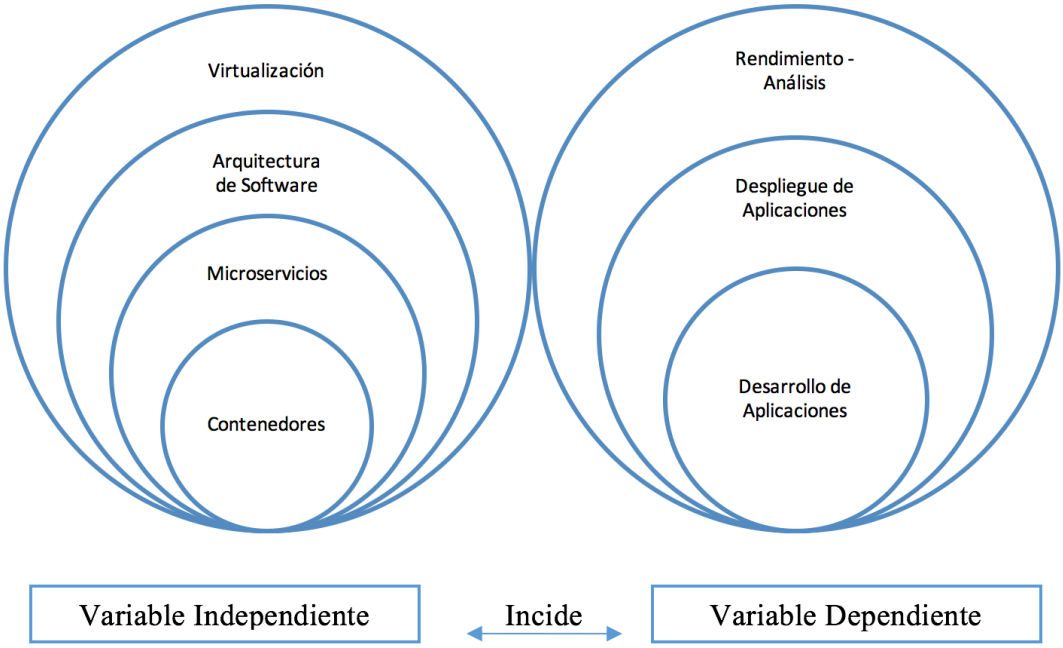


Gráfico 42: Categorías fundamentales.
Elaborado por: Investigador.

Los temas a desarrollar en el Marco Teórico del Informe Final serán:

- Virtualización
- Maquinas Virtuales y Contenedores Linux
- Arquitectura de Software
- Arquitecturas Monolítica y de Microservicios
- Desarrollo y despliegue de Aplicaciones
- Rendimiento del Software

CAPÍTULO III

MARCO METODOLÓGICO

3.1 Tipo de Investigación

Esta investigación es de tipo cuantitativa y cualitativa, porque se va a hacer un análisis de los procesos que intervienen en la experimentacion y se utilizarán parámetros para la medición del rendimiento en experimentos de tiempo real sobre ambientes virtuales.

3.2 Diseño de la Investigación

3.2.1 Modalidad de Investigación

Investigación Documental: Por el uso de fuentes como libros, documentos, artículos, revistas, entre otros, para la construcción del Marco Teórico de las variables dependiente e independiente. Permite además, realizar un proceso de abstracción científica, generalizando sobre los fundamentos que se plantean.

Investigación Bibliográfica: Apoya a la investigación evitando emprender investigaciones ya existentes o muy parecidas, determinando el conocimiento real sobre el tema de este proyecto y generar criterios de fundamentación científica, filosófica y legal. Además, permite la reutilización de experimentos ya realizados en otras investigaciones cuando sea necesario.

Investigación Experimental: El investigador realizará actividades intencionales en ambientes controlados (virtuales), con el fin de describir la causa-efecto que produce una situación en particular.

3.2.2 Tipos o niveles de Investigación

Investigación Correlacional: Para medir el grado de relación entre las variables que se manejan en esta investigación: Arquitectura de Microservicios basada en Contenedores y Rendimiento.

3.3 Métodos

Deductivo: "La deducción es un proceso que parte de un principio general ya conocido para inferir de él, consecuencias particulares" (Gutiérrez M., 2006). Nos permitirá partir de modelos generales (en el caso de las Arquitecturas de Software) para diseñar las comparativas que permitan el análisis respectivo.

Inductivo: “Este Método utiliza el razonamiento para obtener conclusiones que parten de hechos particulares aceptados como válidos, para llegar a conclusiones cuya aplicación sea de carácter general.” (Bernal Torres, 2010). Permitirá hacer un análisis profesional de los datos obtenidos, con el fin de determinar: estrategias, recursos, materiales y medios que puedan intervenir en la interpretación de dichos datos.

Analítico – Sintético: “El análisis consiste en descomponer en partes algo complejo, en desintegrar un hecho o una idea en sus partes, para mostrarlas, describirlas, numerarlas y para explicar las causas de los hechos o fenómenos que constituyen el todo” (Leiva Zea, 2010). Permitirá exponer conceptos fundamentales de una Arquitectura de Software y fases del ciclo de vida, esto con el fin de tener un panorama claro que permita correlacionar lo esperado con los resultados obtenidos.

3.4 Estrategias Técnicas

Se utilizará la técnica de Análisis de archivos, debido a que es una investigación experimental y los resultados de la experimentacion se registrarán en registros (archivos) de logs para su posterior análisis.

3.5 Instrumentos de investigación

Los instrumentos a emplearse son:

- Registros de logs.
- Dispositivos Móviles: Como equipo de comunicación, recolector de evidencia y multimedia.
- Computadora(s) Portátil(es): Para la realización de pruebas, análisis de datos y la documentación de la presente investigación.

CAPÍTULO IV

MARCO ADMINISTRATIVO

4.5 Vialidad

El desarrollo de la presente investigacion toma a consideración lo siguiente:

Factibilidad Operativa: Se cuenta con las herramientas necesarias para la investigación, incluyendo el conocimiento y el ambiente adecuado.

Factibilidad Tecnológica: Los recursos a utilizarse son considerados como Software Libre, y están disponibles para ser utilizados en todo momento.

Factibilidad Económica: El financiamiento de la investigación esta a cargo del investigador.

4.6 Valor Práctico

Beneficiario Directo, Son los expertos en Desarrollo de Software y empresas dedicadas a la producción de Software, que hagan uso de los resultados de este análisis para la toma de desiciones.

Beneficiario Indirecto, Son las empresas que proveen los servicios de Hosting, pues el mejorar el rendimiento incide en una mejor gestión de recursos y reducción de costos.

4.7 Presupuesto

PRESUPUESTO DETALLADO					
1	**EQUIPOS, SOFTWARE Y SERVICIOS**	**VALOR**	**2**	**RECURSOS HUMANOS, TRANSPORTE**	**VALOR**
	COMPUTADOR	$1,000.00		ASESORAMIENTO	$400.00
	SERVICIOS	$300.00		SALIDAS Y VISITAS	$300.00
				TRANSPORTE	$700.00
	SUBTOTAL 1	$1,300.00		**SUBTOTAL 2**	$1,400.00
3	**MATERIALES Y SUMINISTROS**	**VALOR**	**4**	**MATERIAL BIBLIOGRAFICO**	**VALOR**
	PAPEL RESMA	$30.00		THE DOCKER BOOK	$10.00
	FOTOCOPIAS	$20.00			
	IMPRESIONES	200			
	SUBTOTAL 3	$250.00		**SUBTOTAL 4**	$10.00
PRESUPUESTO GLOBAL					

1	EQUIPOS, SOFTWARE Y SERVICIOS	$1,300.00
2	RECURSOS HUMANOS, TRANSPORTE Y SALIDAS DE CAMPO	$1,400.00
3	MATERIALES Y SUMINISTROS	$250.00
4	MATERIAL BIBLIOGRÁFICO	$10.00
SUBTOTAL		$2,960.00
(+) 10% IMPREVISTOS		$296.00
(=) VALOR TOTAL		**$3,256.00**

Tabla 38: Presupuesto.
Elaborado por: Investigador.

4.8 Cronograma de actividades del Plan de Investigación

Las actividades señaladas en el cronograma se las ejecutarán durante el año 2017.

Actividades	Mes 1				Mes 2				Mes 3				Mes 4				Mes 5				Mes 6			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.Recopilación bibliográfica																								
2.Elaboración del Marco Teórico																								
3.Elaboración de instrumentos																								
4.Prueba de instrumentos																								
5.Recolección de datos																								
6.Procesamiento de datos																								
7.Análisis de los datos																								
8.Redacción del borrador																								
9.Revisión y corrección del borrador																								
10.Presentación del informe																								

Tabla 39: Cronograma de Actividades
Realizado por: Investigador.

CAPÍTULO V

5.1 Literatura Citada

Adufu, T., Choi, J., & Kim, Y. (2015). Is container-based technology a winner for high performance scientific applications? En *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (pp. 507-510). https://doi.org/10.1109/APNOMS.2015.7275379

Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., & Steinder, M. (2015). Performance Evaluation of Microservices Architectures using Containers. *arXiv:1511.02043 [cs]*, 27-34. https://doi.org/10.1109/NCA.2015.49

ASALE, R.-. (2014). rendimiento. Recuperado 8 de marzo de 2017, a partir de http://dle.rae.es/?id=VwxnN6O

Basar Bener, A., Morisio, M., Miranskyy, A., & Akinii Kocak, S. (2014, octubre). TI Verde y Software Verde - IEEECS. Recuperado 14 de febrero de 2017, a partir de https://www.computer.org/web/computingnow/archive/october2014-spanish

Bernal Torres, C. A. (2010). *Metodología de la investigación: administración, economía, humanidades y ciencias sociales*. México: Pearson Educación, Prentice Hall.

Bilorou, N. (2011). *Guia para la evaluación de impacto de la formación*. Montevideo, UR: OIT/Cintefor.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2016). Microservices: yesterday, today, and tomorrow. *arXiv:1606.04036 [cs]*. Recuperado a partir de http://arxiv.org/abs/1606.04036

Felter, W., Ferreira, A., Rajamony, R., & Rubio, J. (2015). An updated performance comparison of virtual machines and linux containers. En *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on* (pp. 171–172). IEEE. Recuperado a partir de http://ieeexplore.ieee.org/abstract/document/7095802/

Gutiérrez M., A. (2006). *Curso de Métodos de Investigación*. Quito. Recuperado a partir de http://biblioteca.unach.edu.ec/opac_css/index.php?lvl=notice_display&id=9430

Leiva Zea, F. (2010). *Nociones de metodología de investigación científica*. Recuperado a partir de http://biblioteca.unach.edu.ec/opac_css/index.php?lvl=notice_display&id=5353

Meier, S., Virun, B., Blumert, J., & Jones, M. T. (2008). *IBM Systems Virtualization: Servers, Storage, and Software*. IBM Corp. April. Recuperado a partir de http://www.redbooks.ibm.com/redpapers/pdfs/redp4396.pdf

Prashant, D. (2016). A Survey of Performance Comparison between Virtual Machines and Containers, *4*(7). Recuperado a partir de http://www.academia.edu/27776697/A_Survey_of_Performance_Comparison_between_Virtual_Machines_and_Containers

Vaucher, S. (2015). Comparing Virtual Machines and Linux Containers. Recuperado a partir de

https://zenodo.org/record/47644/files/Comparing_Virtual_Machines_and_Linux_Containers_-_Final.pdf

Villamizar, M., Garcés, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. En *2015 10th Computing Colombian Conference (10CCC)* (pp. 583-590). https://doi.org/10.1109/ColumbianCC.2015.7333476

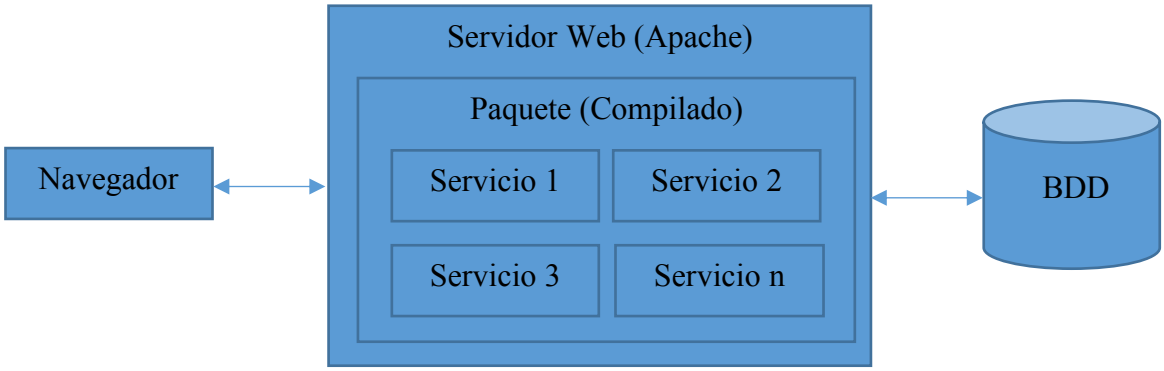
VMWare. (2017). Virtualización de VMware. Recuperado 7 de marzo de 2017, a partir de http://www.vmware.com/latam/solutions/virtualization.html

CAPÍTULO VI

ANEXOS

6.1 Anexo 1

Arquitectura Monolítica.



6.2 Anexo 2

Arquitectura de Microservicios.