



Universidad
Zaragoza

Trabajo Fin de Grado

Título del trabajo final de grado

Framework de despliegue de simulaciones
distribuidas en Cloud

Framework for the implementation of distributed
simulations in the cloud

Autor

Ángel Espinosa Gonzalo

Director

Unai Arronategui Arribalzaga

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2023



DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./D^a. ÁNGEL ESPINOSA GONZALO,

con nº de DNI 73478768R en aplicación de lo dispuesto en el art.

14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster)
Grado _____, (Título del Trabajo)
Framework de despliegue de simulaciones distribuidas en Cloud

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 31 de agosto de 2023

Fdo: _____

Angel

AGRADECIMIENTOS

Querida familia, amigos, profesores y tutor,

Hoy quiero expresar mi agradecimiento por todo el apoyo y la ayuda que he recibido durante la realización de mi trabajo de fin de grado. Sin vosotros, este logro habría sido imposible.

A mi familia, quiero agradecerles por estar siempre a mi lado, por su amor incondicional y por creer siempre en mí, en mis altas y mis bajas. Su constante apoyo moral y emocional fue fundamental para mantenerme motivado durante todo el proceso.

A mis amigos, les agradezco por su compañía y ayudarme a desconectar cuando lo necesitaba. Gracias por entender las veces que no he podido estar por estar ocupado con estos temas. Sois fundamentales.

A mis profesores, les agradezco por su guía y conocimientos impartidos. Gracias por su paciencia, por responder a todas mis dudas y por brindarme las herramientas necesarias para finalizar este grado. Sobre todo gracias a mi tutor Unai Arronategui por su ayuda, paciencia y dedicación, me ha enseñado valiosas lecciones durante la realización de este largo proyecto las cuales realmente han calado en mí.

También quiero agradecer a mis compañeros de trabajo con los cuales tengo una buena relación y me han apoyado a lo largo de la realización de mi TFG, en especial a Sergio y Alejandro, con los cuales hago un gran equipo.

A todos ustedes, quiero expresar mi gratitud por estar siempre presentes, por creer en mis capacidades y por ser un apoyo incondicional.

Con gratitud y cariño, Ángel

Framework de despliegue de simulaciones distribuidas en Cloud

RESUMEN

En el marco de la investigación científica, las simulaciones distribuidas juegan un papel crucial para el avance del conocimiento en diversas disciplinas. Este Trabajo de Fin de Grado se centra en el desafío del aprovisionamiento eficiente de instancias en la nube, con el objetivo de optimizar los recursos y minimizar los costes asociados a la ejecución de estas simulaciones.

La Simulación Distribuida, que implica simular sistemas complejos interconectados, está estrechamente relacionada con la posibilidad de su despliegue en Cloud. La nube ofrece escalabilidad y recursos bajo demanda para afrontar la necesidad de recursos computacionales intensivos. El paralelismo y la distribución ligados a la simulación distribuida se alinean con la capacidad de la nube para ejecutar tareas concurrentes. Esto reduce el tiempo y los costos de ejecución. La nube también facilita la gestión de recursos y la optimización de la asignación de los mismos. Además, elimina barreras tecnológicas y proporciona disponibilidad global.

La computación en la nube ha revolucionado la forma en que se accede y se utilizan los recursos informáticos, brindando una escalabilidad y flexibilidad sin precedentes. Sin embargo, el proceso de aprovisionamiento de instancias en la nube puede ser complejo y costoso si no se planifica adecuadamente.

En este trabajo de fin de grado, se investigan y comparan diferentes estrategias y herramientas para el aprovisionamiento de instancias en la nube, teniendo en cuenta factores como el rendimiento, la disponibilidad y, especialmente, los costes. Se analizan las soluciones ofrecidas por los proveedores de servicios en la nube, empleando finalmente Google Cloud, de tal forma que se evaluarán sus características y prestaciones en términos de costes.

Además, se desarrolla una herramienta personalizada que automatiza el proceso de aprovisionamiento de instancias en Google Cloud. Esta herramienta tiene en cuenta una configuración inicial proporcionada por el usuario y utiliza diversos procedimientos para identificar la zona de Google Cloud y los métodos necesarios que supondrán el menor coste económico y temporal para el despliegue de las máquinas virtuales que

constituirán la base de la ejecución de las simulaciones distribuidas.

Se lleva a cabo un estudio experimental para evaluar el rendimiento y la eficacia de la herramienta desarrollada en comparación con enfoques tradicionales de aprovisionamiento en la nube. Los resultados obtenidos demuestran que la herramienta es capaz de seleccionar la zona de Google Cloud que minimiza los costes de forma significativa, garantizando al mismo tiempo el rendimiento y la disponibilidad requeridos para las simulaciones distribuidas.

En conclusión, este Trabajo de Fin de Grado presenta una solución integral para el aprovisionamiento eficiente de instancias en la nube, con un enfoque especial en Google Cloud, diseñada específicamente para el equipo de investigación universitario. La herramienta desarrollada demuestra su capacidad para identificar la zona de menor coste y emplear el menor número de llamadas a la API para el despliegue de las máquinas virtuales, lo que resulta en una optimización de recursos y una reducción significativa de los gastos asociados al aprovisionamiento en la nube. Estos resultados ofrecen una base sólida para futuras investigaciones en este campo, promoviendo la eficacia y la productividad en el ámbito científico.

Framework for the implementation of distributed simulations in the cloud

RESUME

In the context of scientific research, distributed simulations play a crucial role in advancing knowledge across various disciplines. This Bachelor's Thesis focuses on the challenge of efficiently provisioning cloud instances with the goal of optimizing resources and minimizing costs associated with running these simulations.

Distributed Simulation, which involves simulating interconnected complex systems, is closely tied to the possibility of deployment in the cloud. The cloud offers scalability and on-demand resources to address the need for intensive computational resources. The parallelism and distribution inherent in distributed simulation align with the cloud's ability to execute concurrent tasks, reducing time and execution costs. The cloud also facilitates resource management and optimization of their allocation, while eliminating technological barriers and providing global availability.

Cloud computing has revolutionized how computing resources are accessed and utilized, providing unprecedented scalability and flexibility. However, the process of provisioning cloud instances can be complex and costly if not adequately planned.

In this Bachelor's Thesis, different strategies and tools for provisioning cloud instances are investigated and compared, taking into consideration factors such as performance, availability, and especially costs. Solutions offered by cloud service providers are analyzed, ultimately using Google Cloud to evaluate its features and capabilities in terms of costs.

Furthermore, a custom tool is developed that automates the process of provisioning instances on Google Cloud. This tool considers initial configuration provided by the user and employs various procedures to identify the optimal Google Cloud zone and methods that will result in the least economic and time cost for deploying the virtual machines that will form the basis for the execution of distributed simulations.

An experimental study is conducted to assess the performance and effectiveness of the developed tool in comparison to traditional cloud provisioning approaches. The results obtained demonstrate that the tool is capable of selecting the Google Cloud zone that significantly minimizes costs while ensuring the required performance and

availability for distributed simulations.

In conclusion, this Bachelor's Thesis presents a comprehensive solution for efficiently provisioning cloud instances, with a specific focus on Google Cloud, designed specifically for the university research team. The developed tool showcases its ability to identify the lowest-cost zone and use the fewest API calls for deploying virtual machines, resulting in resource optimization and a significant reduction in cloud provisioning expenses. These results provide a solid foundation for future research in this field, promoting efficiency and productivity in the scientific domain.

Índice

1. Introducción y objetivos	1
1.1. Contexto	1
1.2. Objetivos	1
1.3. Metodología	2
2. Conceptos y Tecnologías	3
2.1. Conceptos básicos de Cloud	3
2.2. Técnicas de despliegue en Cloud	3
2.3. Proveedores de cloud público	5
2.4. Modelo económico Cloud	5
2.5. Automatización de despliegues	6
2.6. Herramientas	7
2.6.1. Golang	7
2.6.2. Terraform	7
2.7. Skypilot	8
3. Análisis y diseño del sistema	10
3.1. Análisis del problema	10
3.2. Análisis de requisitos	11
3.2.1. Requisitos funcionales	12
3.2.2. Requisitos no funcionales	12
3.2.3. Requisitos de rendimiento	12
3.2.4. Requisitos económicos	13
3.3. Diseño y arquitectura del sistema	13
3.4. Configuración inicial	16
3.5. Despliegue	17
3.6. Ejecución de la simulación distribuida	17
3.7. Destrucción de las instancias y limpieza del entorno	19
3.8. Análisis de precios	19
3.9. Disponibilidad de cuotas	19

3.10. Depuración de la simulación	20
3.11. Diseño de la base de datos	21
4. Implementación	23
4.1. Configuración inicial	24
4.2. Despliegue	24
4.3. Ejecución de la simulación distribuida	24
4.4. Destrucción de las instancias y limpieza del entorno	25
4.5. Análisis de precios	26
4.6. Disponibilidad de cuotas	26
4.7. Depuración de la simulación	27
4.8. Uso de la base de datos	27
5. Validación y pruebas	29
5.1. Prueba de Simulación Distribuida	29
5.2. Prueba de volumen	30
5.3. Prueba de velocidad de borrado	30
5.4. Prueba de latencia	31
6. Conclusiones	32
7. Bibliografía	34
Lista de Figuras	35
Lista de Tablas	36
Anexos	37
A. Diagrama de Gantt	39
B. Horas de dedicación	40
C. Aspectos relevantes de los comandos del framework	41
C.1. Deploy	41
C.2. Exec	42
C.3. Destroy	42
C.4. Price	43
D. Ejecución del framework	44

E. Base de datos	49
F. Líneas de código	54

Acrónimos

GCP Google CLOUD Platform

DIIS Departamento de Informática e Ingeniería de Sistemas

COSMOS Computer Science for Complex System Modeling

SED Sistemas de Eventos Discretos

TFG Trabajo FIn de Grado

Go Golang

IaaS Infraestructure as Service

PaaS Platform as Service

AaaS Software as Service

AWS Amazon Web Service

IaC Infraestructure as Code

CI Continuous Integration

CD Continuous Delivery

VPN Virtual Private Network

DNS Domain Name Server

CPU Central Processing Unit

GPU Graphics Processing Unit

TPU Tensor Processing Unit

API Application Programming Interface

TCP Transfer Control Protocol

VPC Virtual Private Cloud

ORM Object Relational Mapping

SSH Secure SHell

Bucket Bucket de GCP

Glosario

Framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Cloud Paradigma que permite ofrecer servicios de computación a través de una red

Goroutine Hilo de implementación simple en el Golang que se ejecuta simultáneamente con el resto del programa.

Machine Learning Disciplina del campo de la Inteligencia Artificial que, a través de algoritmos, dota a los ordenadores de la capacidad de identificar patrones en datos masivos y elaborar predicciones

Instancia Máquina virtual que representa un recurso de servidor que brinda servicios en la nube de terceros.

Red de Petri Representación matemática o gráfica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente

Latencia de Red Retraso en la comunicación de la red. Muestra el tiempo que tardan los datos en transferirse a través de la red.

Ancho de Banda Cantidad máxima de datos que pueden ser transmitidos a través de una conexión en un tiempo determinado.

Proxy Intermediario que se ubica entre las solicitudes que realiza un cliente hacia otro servidor.

Zona de GCP Área de implementación dentro de una región.

Logs Grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular.

Capítulo 1

Introducción y objetivos

1.1. Contexto

Este proyecto forma parte de las actividades de investigación llevadas a cabo por el equipo de investigación del del Departamento de Informática e Ingeniería de Sistemas (DIIS) y del grupo Computer Science for Complex System Modeling (COSMOS). Su labor se centra en el desarrollo de sistemas distribuidos complejos. El objetivo principal de este proyecto es desarrollar soluciones para el aprovisionamiento y ejecución eficiente de simulaciones distribuidas en infraestructuras en la nube.

El enfoque de investigación se centra en la simulación de escenarios y modelos para evaluar, diseñar, tomar decisiones o reconfigurar sistemas complejos. Se exploran arquitecturas de ejecución basadas en la nube para respaldar estas simulaciones.

En este contexto, el trabajo se enfocará en la gestión de infraestructuras distribuidas y el despliegue de servicios específicos para soportar la simulación de Sistemas de Eventos Discretos (SED) de gran escala.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es diseñar y desarrollar un framework especializado para el despliegue de simulaciones distribuidas en la nube. Este framework de despliegue de simulaciones distribuidas, se centrará en aspectos clave como la velocidad de despliegue de un gran número de máquinas virtuales y el coste asociado al despliegue en la nube..

El framework se encargará de gestionar de manera eficiente y automatizada el despliegue de las simulaciones distribuidas, teniendo en cuenta la escalabilidad requerida y optimizando los recursos disponibles.

Además, el framework también contemplará la gestión básica de datos, comunicación y almacenamiento necesaria para el despliegue de código de simulación

y modelos a simular. Esto incluirá la capacidad de gestionar la transferencia de datos entre las diferentes instancias, así como el almacenamiento y recuperación de resultados de las simulaciones.

Cabe destacar que el framework va a desplegar los recursos en GCP puesto que la Universidad cuenta con recursos económicos en dicho proveedor.

1.3. Metodología

La metodología seguida en este Trabajo de Fin de Grado se divide en varias etapas que han permitido el desarrollo y validación del framework propuesto para el despliegue de simulaciones distribuidas en la nube.

En primer lugar, se ha llevado a cabo un análisis de los requisitos necesarios para el despliegue de simulaciones distribuidas en la nube. A continuación, se procedió al diseño de una solución adaptada a los requisitos identificados. Se definieron las arquitecturas y componentes necesarios para implementar el framework de despliegue de simulaciones distribuidas.

Una vez completado el diseño, se implementó un prototipo del framework utilizando el lenguaje de programación Go y entornos de sistemas Linux. Se realizaron pruebas y ajustes para garantizar su funcionamiento adecuado y su compatibilidad con el entorno de cloud público seleccionado, en este caso, Google Cloud.

El prototipo se utilizó para llevar a cabo una serie de experimentos que permitieron validar su funcionamiento y evaluar su rendimiento en términos de velocidad de despliegue, eficiencia de costes y escalabilidad. Finalmente, se completó el trabajo con la redacción de la memoria del TFG. Se han añadido un diagrama de Gantt y una distribución de las horas en el Anexo A y Anexo B respectivamente.

Capítulo 2

Conceptos y Tecnologías

2.1. Conceptos básicos de Cloud

La computación Cloud se basa en varios conceptos fundamentales:

- Nube: Infraestructura virtualizada y distribuida que proporciona acceso bajo demanda a recursos informáticos a través de Internet. Servicio en la nube: Funcionalidad ofrecida por proveedores de servicios en la nube, como almacenamiento, cómputo, bases de datos, redes, etc.

- Modelos de servicio en la nube: IaaS (infraestructura como servicio), PaaS (plataforma como servicio) y SaaS (software como servicio), que definen cómo se ofrecen los servicios en la nube.

- Escalabilidad: Capacidad de aumentar o disminuir rápidamente los recursos en la nube según las necesidades, tanto horizontal como verticalmente.

- Virtualización: Tecnología que permite la creación de instancias virtuales de recursos físicos, mejorando la eficiencia y facilitando la gestión de aplicaciones en la nube.

- Alta disponibilidad: Capacidad de un sistema en la nube para funcionar continuamente sin interrupciones o tiempo de inactividad significativo, mediante redundancia y recuperación ante fallas.

Estos conceptos son fundamentales para comprender y utilizar eficientemente los servicios y recursos en la nube.

2.2. Técnicas de despliegue en Cloud

Las técnicas de despliegue de recursos en Cloud son fundamentales para lograr una implementación eficiente y escalable de infraestructuras y aplicaciones. Estas técnicas permiten aprovechar al máximo los servicios y la flexibilidad que ofrece la nube, garantizando una gestión óptima de los recursos. A continuación, se presentan algunas

técnicas comunes de despliegue en la nube:

- Orquestación de recursos: La orquestación es un enfoque clave para automatizar la implementación y configuración de recursos en la nube. Permite describir la infraestructura necesaria y sus interdependencias utilizando lenguajes de definición de plantillas, como AWS CloudFormation o Azure Resource Manager. La orquestación facilita la implementación y administración de entornos complejos, garantizando la coherencia y la replicabilidad de las configuraciones.

- Contenedores: Los contenedores ofrecen una forma eficiente de encapsular y desplegar aplicaciones en la nube. Con herramientas como Docker, se puede crear una imagen de contenedor que incluya todas las dependencias y configuraciones necesarias. Los contenedores son altamente portátiles y se pueden desplegar rápidamente en entornos de nube, brindando beneficios en términos de aislamiento, escalabilidad y eficiencia en la implementación.

- Escalado automático: La capacidad de escalar automáticamente los recursos en función de la demanda es esencial en la nube. Al utilizar métricas de rendimiento, como la carga de trabajo o el tráfico, es posible ajustar dinámicamente la capacidad de los recursos. El escalado automático garantiza un rendimiento óptimo y una utilización eficiente de los recursos al agregar o eliminar instancias automáticamente según las necesidades cambiantes.

- Infraestructura como código (IaC): La IaC se basa en describir y aprovisionar la infraestructura utilizando código en lugar de configuraciones manuales. Mediante herramientas como Terraform o Ansible, es posible definir la infraestructura de manera programática. Esta técnica proporciona una gestión más eficiente, escalable y reproducible de los recursos en la nube, permitiendo una implementación coherente y fácilmente replicable.

- Prácticas de DevOps y entrega continua: Las prácticas de DevOps se centran en la colaboración entre equipos de desarrollo y operaciones para lograr una implementación más ágil y confiable. La entrega continua (CI/CD) se basa en la automatización de los procesos de compilación, prueba y despliegue para ofrecer actualizaciones frecuentes y confiables de las aplicaciones. Estas prácticas mejoran la eficiencia y calidad del despliegue en la nube, acelerando los tiempos de implementación y garantizando la estabilidad del software.

Las técnicas de despliegue en la nube son fundamentales para lograr una implementación eficiente y escalable de recursos en la nube. Estas técnicas son relevantes para optimizar el despliegue de infraestructuras y aplicaciones en el contexto del TFG, mejorando la gestión de recursos.

2.3. Proveedores de cloud público

En el ámbito de este TFG, se exploran y evalúan los proveedores de servicios en la nube disponibles para el despliegue de instancias que permitan la ejecución de las simulaciones distribuidas. Entre los proveedores más reconocidos se encuentran AWS (Amazon), GCP (Google) y Azure (Microsoft). Cada proveedor ofrece características distintivas que los diferencian entre sí, y la elección de uno u otro dependerá del proyecto en cuestión, la familiaridad con el proveedor y el presupuesto disponible, entre otros factores.

En el caso específico de GCP, el cual es el que se ha empleado, se destacan diversas capacidades entre las que se incluyen la creación de máquinas virtuales, bases de datos y recursos de red, como firewalls, servidores VPN y servidores DNS, entre otros. Estas herramientas ofrecen una amplia gama de posibilidades para apoyar el despliegue con un costo económico, generalmente basado en un modelo de pago por uso o bajo demanda.[1]

2.4. Modelo económico Cloud

Los proveedores Cloud disponen de un modelo de precios y costos asociados con el uso de sus servicios. Estos proveedores, como Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform (GCP), ofrecen una amplia gama de servicios y recursos informáticos a través de modelos de precios flexibles y variables.

Algunos conceptos clave relacionados con la economía en los proveedores de servicios en la nube son los siguientes:

1. Pago por uso (Pay-as-you-go): Los proveedores de servicios en la nube generalmente adoptan un modelo de pago por uso, lo que significa que los clientes solo pagan por los recursos y servicios que utilizan. Esto permite una mayor flexibilidad, ya que los usuarios pueden escalar o reducir sus recursos según sea necesario.
2. Tarifas por hora o por segundo: Los servicios en la nube suelen ser facturados por hora o incluso por segundo de uso. Esto suele resultar en una mayor precisión y control de los costos.
3. Modelos de precios variables: Los proveedores de servicios en la nube ofrecen una amplia gama de servicios con diferentes modelos de precios. Por ejemplo, pueden cobrar por el almacenamiento de datos, el ancho de banda utilizado, el tiempo de cálculo o el número de solicitudes realizadas. Los precios pueden variar según la región geográfica, el tipo de servicio y otros factores.
4. Descuentos y planes de suscripción: Los proveedores de servicios Cloud a menudo

ofrecen descuentos y planes de suscripción para clientes que utilizan recursos a largo plazo o que tienen necesidades específicas. Estos descuentos pueden incluir tarifas reducidas por uso continuo o compromisos de uso a largo plazo.

5. Costos adicionales: Además de los servicios en sí, puede haber costos adicionales asociados con el uso de servicios cloud. Esto puede incluir costos de transferencia de datos entre regiones o fuera de la nube, costos de almacenamiento adicional, cargos por soporte técnico...

6. Herramientas de administración de costos: Los proveedores de servicios en la nube suelen proporcionar herramientas y paneles de control para ayudar a los usuarios a monitorizar y administrar sus costos. Estas herramientas permiten visualizar el gasto, establecer límites presupuestarios, recibir alertas y optimizar el uso de recursos para minimizar los costos.

Es importante tener en cuenta estos aspectos económicos al utilizar servicios en la nube, ya que permiten una gestión eficiente de los costos y garantizan que se ajusten a las necesidades y presupuestos específicos de cada usuario.

2.5. Automatización de despliegues

La automatización de despliegues en la nube es un enfoque de gestión y configuración de infraestructura que utiliza herramientas y scripts para agilizar y estandarizar el proceso de implementación de aplicaciones y servicios en entornos de computación en la nube. Esta automatización permite desplegar recursos y aplicaciones de manera rápida, consistente y repetible, reduciendo errores y tiempos de inactividad.

Algunos conceptos clave relacionados con la automatización de despliegues en la nube son los siguientes:

1. Infraestructura como código (Infrastructure as Code - IaC): Es una práctica donde se define y gestiona la infraestructura de la nube mediante código. Los archivos de configuración, generalmente escritos en lenguajes como YAML o JSON, describen los recursos necesarios para la aplicación y sus relaciones. Estos archivos son tratados como código fuente.

2. Herramientas de automatización: Existen diversas herramientas populares para la automatización de despliegues en la nube, como Ansible, Terraform, Pulumi, Skypilot... Estas herramientas permiten definir y gestionar recursos, configuraciones y políticas de seguridad de forma programática.[2]

3. Orquestación de servicios: Además de configurar recursos, la automatización en la nube también incluye la orquestación de servicios y aplicaciones. Esto implica definir cómo interactúan los diferentes componentes y servicios de una aplicación distribuida,

asegurando su comunicación y escalabilidad.

4. Integración continua y entrega continua (CI/CD): La automatización de despliegues en la nube es una parte clave de las prácticas de CI/CD, donde los cambios en el código se prueban e integran de manera confiable.

5. Plantillas: Las plantillas son elementos esenciales en la automatización. Las plantillas proporcionan una descripción declarativa de los recursos que se desplegarán en la nube.

La automatización de despliegues en la nube es un enfoque esencial para la gestión eficiente de recursos y aplicaciones en entornos Cloud, impulsando la eficiencia operativa y mejorando la calidad y velocidad de las implementaciones.

2.6. Herramientas

2.6.1. Golang

Golang, es un lenguaje de programación de código abierto desarrollado por Google. Fue diseñado con el objetivo de ser eficiente, rápido y fácil de usar, con un enfoque en la simplicidad y la concurrencia.[3]

Una de las características más destacadas de Go es su capacidad para manejar la concurrencia de forma nativa. Proporciona goroutines, que son unidades de ejecución livianas y concurrentes, y canales, que son estructuras de datos para la comunicación y sincronización entre goroutines. Esta combinación de goroutines y canales facilita la implementación de programas concurrentes y paralelos de manera eficiente y segura. Este aspecto ha sido relevante a la hora del diseño de la herramienta.

Go también ofrece una amplia biblioteca estándar que cubre una amplia gama de funcionalidades, lo que facilita el desarrollo de aplicaciones completas sin tener que depender de bibliotecas externas.

2.6.2. Terraform

Dentro del ámbito del despliegue de arquitecturas en la nube, existen herramientas como Terraform que simplifican y automatizan el proceso de implementación y gestión de infraestructuras. Terraform es una herramienta de infraestructura como código (IaC) que permite definir y configurar la infraestructura de manera declarativa, utilizando un lenguaje de alto nivel.[4]

Una de las principales ventajas de Terraform es su capacidad para gestionar proveedores de nube múltiples, lo que significa que se puede utilizar con diferentes proveedores de servicios en la nube, como AWS, GCP, Azure, entre otros. Esto

proporciona flexibilidad y portabilidad a la hora de desplegar arquitecturas en diferentes entornos.

Terraform permite definir la infraestructura mediante archivos de configuración, conocidos como 'archivos de configuración de Terraform' o 'tf files', donde se especifican los recursos, configuraciones y dependencias necesarias para el despliegue. Estos archivos son versionables y pueden ser gestionados con control de versiones, lo que facilita el seguimiento y la colaboración en proyectos de infraestructura.

Se han utilizado herramientas como Terraform como referencia para identificar la idea principal en la que se basan que es el diseño de un plan dividido en varias etapas, donde se parte de un plan inicial en el que se identifican las necesidades del despliegue, un plan intermedio donde se toma en consideración las necesidades del despliegue y se adapta a la configuración de los recursos que van a ser creados y una etapa final del plan en la que se ejecuta el despliegue de los recursos.

A pesar de esto, Terraform cuenta con una serie de desventajas entre las cuales se encuentran:

- Rendimiento Escalable: Terraform puede volverse lento cuando se despliega una gran cantidad de recursos, como VMs, en una sola operación. Esto puede aumentar el tiempo de ejecución y consumir recursos.
- Planificación Compleja: La fase de planificación previa a la ejecución puede ser lenta y exigir recursos significativos, especialmente con muchos recursos.
- Lentitud de Provisión: La creación de recursos en la nube, como VMs, puede ser lenta debido a factores como la latencia y la configuración requerida.
- Dependencia de API y Servidores Remotos: La velocidad de Terraform puede depender de la velocidad de las API y la capacidad de servidores remotos, lo que puede ralentizar el proceso.
- Optimización de Recursos: Terraform no siempre optimiza la secuencia de creación de recursos, lo que podría ralentizar el despliegue.
- Necesidad de Estrategias: Para un despliegue rápido de muchas VMs, se deben considerar estrategias como desplegar en lotes más pequeños o paralelizar el proceso.

2.7. Skypilot

Skypilot es un framework de código abierto que permite a los usuarios implementar y administrar fácilmente infraestructuras en la nube usando una variedad de proveedores de nube populares. La herramienta está diseñada para ser fácil de usar, pero también escalable para manejar cargas de trabajo complejas y críticas. Skypilot proporciona una interfaz de usuario intuitiva y también se puede administrar a través de la línea

de comandos. El objetivo de Skypilot es simplificar el proceso de implementación de infraestructura en la nube y hacer que los desarrolladores y administradores de sistemas sean más eficientes.[?]

Se ha tomado cómo referencia Skypilot ya que está dedicada al ámbito de machine learning por lo que es una herramienta de despliegue especializada en un tipo de aplicaciones en concreto. En este trabajo de fin de grado se plantea la concepción e implementación de una herramienta también especializada en otro tipo de aplicaciones; las simulaciones distribuidas. Al igual que Terraform sigue la idea de plan inicial, plan intermedio y plan final, en la que se ha basado este TFG. Skypilot hace inca pie en los costes de despliegue puesto que para machine learning se utilizan GPU's y TPU's las cuales acarrear un gran coste si el tiempo de uso es elevado. Para este framework en particular no se va a hacer uso de recursos tan costosos.

Capítulo 3

Análisis y diseño del sistema

3.1. Análisis del problema

El análisis del problema abordado en este TFG se enfoca en los desafíos asociados al aprovisionamiento de recursos para ejecutar simulaciones distribuidas. El objetivo principal del proyecto es lograr proporcionar los recursos necesarios para estas simulaciones, considerando que su ejecución distribuida implica un alto costo en términos de adquirir y mantener los recursos físicos requeridos. En este sentido, la utilización de un entorno de Cloud se presenta como la opción más adecuada.

Las simulaciones que se van a ejecutar en este contexto pueden involucrar un gran número de nodos y requerir un tiempo considerable para su finalización. Estos aspectos serán fundamentales para guiar la solución al problema planteado, ya que el costo de un despliegue en la nube puede variar significativamente en función de la configuración de las máquinas utilizadas en Google Cloud.

Los proveedores de servicios de Cloud público ofrecen una amplia gama de configuraciones, las cuales también están asociadas a una amplia variedad de precios. Por lo tanto, realizar un análisis exhaustivo de las opciones de configuración que ofrece Google Cloud Platform (GCP) y desplegar únicamente los recursos necesarios se convierte en la tarea prioritaria de esta herramienta.

En una etapa inicial del despliegue, se realiza una evaluación detallada de los recursos requeridos en base a la magnitud de la simulación. Esto implica identificar la cantidad de nodos necesarios, la capacidad de almacenamiento requerida, los requisitos de comunicación y otros aspectos relacionados. A partir de esta información, se realiza un análisis comparativo de las opciones de configuración ofrecidas por GCP, con el objetivo de determinar la configuración óptima en términos de rendimiento y costo.

El desarrollo de una herramienta especializada permitirá automatizar este proceso de análisis y aprovisionamiento de recursos en la nube. La herramienta, basada en la configuración inicial proporcionada, identificará la zona de Google Cloud que

implicará el menor costo para el despliegue, teniendo en cuenta los requisitos de la simulación distribuida. Esto permitirá optimizar tanto el costo como el rendimiento de las simulaciones, facilitando así la investigación del equipo y fomentando el escalado bajo demanda.

Como se ha mencionado antes, se han tomado como referencia herramientas ya existentes para comprender el enfoque principal utilizado en el diseño de un plan de despliegue dividido en varias etapas. Este enfoque se basa en la idea de establecer un proceso estructurado que abarque desde la identificación inicial de las necesidades del despliegue hasta la ejecución final del mismo.

La primera etapa del plan consiste en realizar un análisis exhaustivo de las necesidades específicas del despliegue. Esto implica evaluar los requisitos de recursos, como la cantidad de máquinas virtuales, capacidad de almacenamiento y configuraciones de red, entre otros.

Una vez identificadas las necesidades, se procede a la etapa intermedia del plan, en la cual se adaptan las especificaciones del despliegue a la configuración de los recursos que serán creados. En esta fase, se consideran diferentes aspectos, como la elección de la región o zona de implementación más adecuada en el proveedor de Cloud público, la selección de las instancias o máquinas virtuales óptimas según los requisitos de la simulación, y la configuración de los servicios adicionales necesarios, como almacenes de datos.

Finalmente, se llega a la etapa final del plan, donde se lleva a cabo la ejecución del despliegue de los recursos. Aquí, se implementan las configuraciones definidas previamente y se establecen las conexiones y comunicaciones necesarias entre los componentes del sistema distribuido. Durante esta fase, se verifica la correcta creación y funcionamiento de los recursos, y se realiza una monitorización continua para asegurar que todo esté en funcionamiento de manera adecuada.

La utilización de un plan dividido en etapas facilita el proceso de despliegue de manera estructurada y controlada. Permite una mayor eficiencia en la gestión de los recursos y una mayor flexibilidad para adaptarse a los cambios o ajustes necesarios durante el proceso. Además, al seguir este enfoque, se promueve la reutilización de código y configuraciones, lo que facilita la escalabilidad y el mantenimiento del sistema a largo plazo.

3.2. Análisis de requisitos

Análisis de requisitos del Framework de Despliegue de Simulaciones Distribuidas en el Cloud:

3.2.1. Requisitos funcionales

a. Despliegue de simulaciones distribuidas: El framework debe permitir el despliegue de simulaciones distribuidas en el Cloud, lo que implica la creación y gestión de múltiples máquinas virtuales.

b. Gestión básica de datos, comunicación y almacenamiento: El framework debe proporcionar funcionalidades para la gestión básica de datos, comunicación y almacenamiento necesarios durante las simulaciones. Esto puede incluir la transferencia de archivos, la configuración de redes y la interacción con sistemas de almacenamiento en la nube.

c. Recuperación de resultados: El framework debe permitir la recuperación de los resultados de las simulaciones, ya sea almacenándolos localmente o transfiriéndolos a un sistema de almacenamiento en la nube.

3.2.2. Requisitos no funcionales

a. Escalabilidad: El framework debe ser capaz de escalar de manera eficiente para soportar el despliegue de un gran número de máquinas virtuales. Debe poder gestionar tanto pequeñas simulaciones como simulaciones de gran escala que requieren cientos o miles de máquinas virtuales.

b. Latencia: El framework debe minimizar la latencia en el despliegue de las máquinas virtuales y en las comunicaciones entre ellas. Una baja latencia es crucial para garantizar un rendimiento óptimo de las simulaciones distribuidas.

c. Eficiencia de costes: El framework debe considerar el coste del despliegue en el Cloud y buscar optimizaciones para reducir los gastos. Esto implica tomar decisiones inteligentes sobre el uso de recursos y buscar opciones económicas en el proveedor de servicios Cloud.

3.2.3. Requisitos de rendimiento

a. Velocidad de despliegue: El framework debe ser capaz de desplegar un gran número de máquinas virtuales de forma rápida y eficiente. La velocidad de despliegue es esencial para reducir el tiempo de espera antes de que las simulaciones puedan comenzar lo que implica un ahorro en coste.

b. Rendimiento de las simulaciones: El framework debe proporcionar un entorno de ejecución eficiente que permita a las simulaciones distribuidas ejecutarse sin problemas y con el menor tiempo de ejecución posible. Esto implica minimizar la sobrecarga de comunicación entre las máquinas virtuales y optimizar los recursos utilizados.

3.2.4. Requisitos económicos

a. Evaluación de costes: El framework debe ofrecer funcionalidades para evaluar los costes asociados con el despliegue de las simulaciones en el Cloud. Esto puede incluir la estimación de costes antes del despliegue y el seguimiento de los costes tras la ejecución de las simulaciones.

b. Optimización de costes: El framework debe buscar activamente formas de optimizar los costes del despliegue, como la selección de instancias de máquinas virtuales económicas, el uso eficiente de recursos y la identificación de oportunidades para utilizar modelos de precios flexibles ofrecidos por los proveedores de servicios en la nube.

3.3. Diseño y arquitectura del sistema

Con respecto al diseño del sistema y la arquitectura para el aprovisionamiento de instancias en la nube y la ejecución de simulaciones distribuidas, se propone el despliegue de instancias de Google Cloud para cada una de las subredes que conforman la red de Petri de la simulación distribuida. Dado que la cantidad de instancias necesarias puede ser considerablemente alta, es fundamental optimizar los procesos de creación, ejecución y eliminación de las instancias.

Además, es importante tener en cuenta que la simulación distribuida implica la comunicación entre las instancias, por lo tanto, es necesario considerar aspectos de red para minimizar las latencias entre las máquinas.

Se plantea una arquitectura similar a la descrita en la figura 3.1 en la cual podemos observar el planteamiento general del despliegue. La arquitectura de la solución desarrollada se presenta en el diagrama como un conjunto de componentes interconectados que trabajan de manera conjunta para lograr la implementación eficiente de simulaciones distribuidas en la nube. El diseño se basa en la utilización de servicios y recursos proporcionados por la plataforma Google Cloud (GCP). A continuación, se describe la arquitectura en detalle:

- API de Google Cloud Platform (GCP): La arquitectura se inicia con la API de GCP, que sirve como punto de entrada para la interacción entre los diferentes componentes de la solución. Esta API proporciona la interfaz para la gestión y el control de los recursos en la nube, permitiendo la creación, configuración y administración de instancias y redes

- Proxy: Una instancia virtual en la nube actúa como un proxy. Esta instancia se sitúa en una zona específica de GCP y actúa como intermediario entre los componentes externos y las instancias de trabajo. El proxy juega un papel clave para optimizar la

utilización de los recursos cómo se explicará posteriormente. Dependiendo del tipo de ejecución el proxy puede no existir.

- Instancias de Trabajo (Workers): En la misma zona de GCP, se despliegan varias instancias virtuales. Estas instancias trabajan de manera distribuida y paralela para ejecutar las simulaciones requeridas. Cada worker recibe instrucciones y datos de la simulación y luego contribuye al procesamiento de la simulación de manera cooperativa.

- Base de Datos PostgreSQL: Para la persistencia de datos relevantes, se implementa una base de datos PostgreSQL. Esta base de datos almacena información crucial para las simulaciones, como resultados intermedios, configuraciones y otros datos relacionados.

- Bucket de Almacenamiento de GCP: Para el almacenamiento de datos y archivos relevantes, se utiliza un bucket de almacenamiento de GCP. Este bucket almacena archivos de entrada y salida, así como otros datos necesarios para las simulaciones.

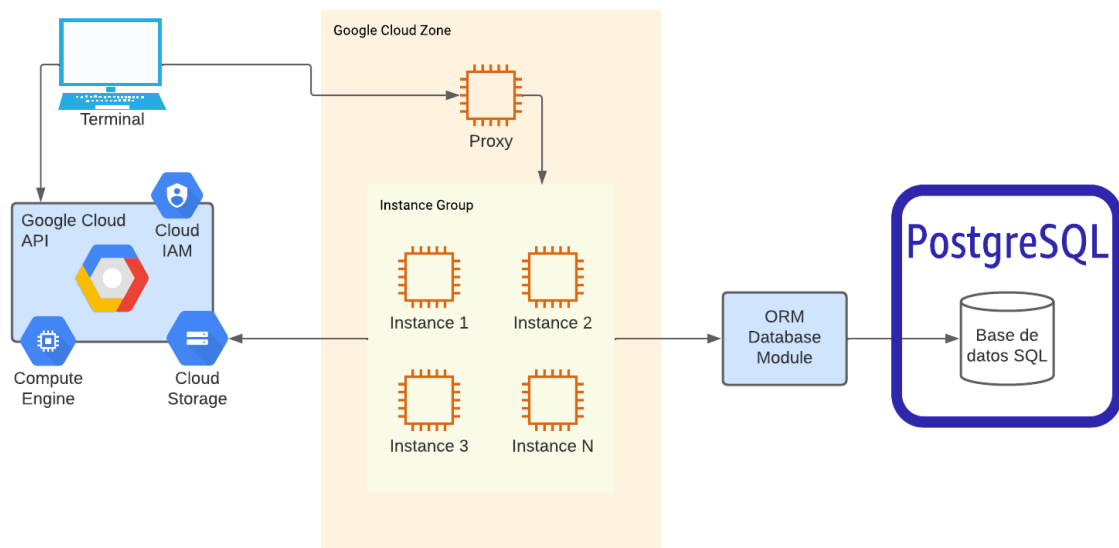


Figura 3.1: Diagrama de arquitectura del framework utilizando el proxy

Otro aspecto relevante es que la creación y eliminación de las máquinas se lleva a cabo mediante llamadas a la API de Google Cloud. Para realizar estas llamadas, se requiere tener una cuenta en Google Cloud y crear un proyecto. Es importante tener en cuenta que estas llamadas a la API tienen un costo asociado y existe un límite en la cantidad de llamadas que se pueden realizar. Por lo tanto, se debe reducir al máximo el número de llamadas a la API.

Para optimizar el costo, se han utilizado dos tipos de instancias que permiten reducir considerablemente los gastos: las instancias preemptibles y las instancias spot:

Instancias preemptibles: Estas instancias son máquinas virtuales de GCP que están

disponibles a un precio sustancialmente reducido en comparación con las instancias regulares. Sin embargo, hay una restricción de tiempo en su uso, ya que solo se pueden ejecutar durante un máximo de 24 horas continuas. Después de ese período, Google Cloud las interrumpe automáticamente. Estas instancias son ideales para cargas de trabajo no críticas y efímeras, como tareas de procesamiento por lotes, pruebas y simulaciones de corta duración.

Instancias spot: Las instancias spot de GCP también ofrecen un costo reducido en comparación con las instancias regulares. A diferencia de las instancias preemptibles, no tienen una restricción de tiempo y no se interrumpen automáticamente después de 24 horas. En su lugar, estas instancias se basan en la disponibilidad de capacidad no utilizada en los centros de datos de Google Cloud. Esto significa que el precio de las instancias spot puede variar según la oferta y la demanda en el mercado. Google Cloud puede interrumpir las instancias spot en caso de que se necesite la capacidad para cargas de trabajo prioritarias, pero proporciona una notificación anticipada antes de finalizarlas. Las instancias spot son adecuadas para cargas de trabajo flexibles y tolerantes a interrupciones, como tareas de procesamiento distribuido, análisis de datos y ciertas aplicaciones de backend.

De esta manera, se puede elegir entre el uso de una u otra opción dependiendo del tipo de simulación que se va a realizar. Si se prevé que la simulación durará menos de 24 horas, se pueden utilizar instancias preemptibles; en caso contrario, se utilizarán instancias spot.

Las instancias se crean a partir de una imagen almacenada en Google Cloud, a la cual se puede hacer referencia en la configuración de la instancia. Esta imagen contiene únicamente los componentes necesarios para la ejecución de la simulación distribuida. Además, se dispone de un bucket de Google que actúa como sistema de almacenamiento persistente, donde se almacenan los archivos necesarios para la ejecución de la simulación distribuida. Las instancias se comunicarán con este bucket para obtener los archivos necesarios para la ejecución.

Una vez creadas las instancias, se procederá a enviar una orden de ejecución para iniciar la simulación distribuida con los parámetros necesarios. Existen dos enfoques para realizar esta tarea: el primero consiste en enviar la orden directamente desde el ordenador que ejecuta la herramienta, y el segundo implica el uso de una instancia adicional interna que actúa como intermediario de comunicación con las demás instancias. La ventaja de utilizar la instancia adicional radica en que la comunicación a través de la red interna presenta numerosas ventajas en comparación con la comunicación a través de la red externa.

- Latencia reducida: La comunicación a través de la red interna entre las

instancias TCP generalmente ofrece una latencia mucho menor en comparación con la comunicación a través de la red externa. Las instancias que se encuentran en la misma red interna pueden comunicarse directamente sin necesidad de pasar por la infraestructura de red externa, lo que reduce el tiempo de transmisión de datos.

- Mayor ancho de banda: La red interna suele proporcionar un mayor ancho de banda disponible en comparación con la red externa. Esto permite una transferencia de datos más rápida y eficiente entre las instancias, especialmente en casos donde se realizan simulaciones distribuidas que implican un intercambio constante de información.

- Seguridad y aislamiento: La comunicación a través de la red interna ofrece un entorno más seguro y aislado en comparación con la comunicación a través de la red externa. Las instancias pueden configurarse para permitir una comunicación interna segura dentro de una red virtual privada (VPC), lo que reduce los riesgos de seguridad asociados con la exposición a la red externa.

Además, la instancia adicional creada se basa en una imagen que contiene todo lo necesario para llevar a cabo su tarea de ejecución. Esto asegura que la instancia esté aprovisionada con los componentes y configuraciones requeridos para el proceso de simulación distribuida.

Se hace uso además de una base de datos PostgreSQL destinada a respaldar el almacenamiento y gestión de datos de las simulaciones desarrolladas por los usuarios. La base de datos ha sido desplegada en un ordenador local y desempeña un papel fundamental en la organización eficiente, recuperación y análisis de los resultados de tus experimentos de simulación. En esta base se almacenan los datos relativos a la configuración de cada simulación y las instancias que las componen.

Se va a analizar también el diseño del framework teniendo en cuenta las decisiones tomadas para el desarrollo de las funcionalidades relevantes integradas en la herramienta.

3.4. Configuración inicial

La funcionalidad que se encarga del despliegue de las instancias en Google Cloud Platform (GCP) se inicia con la lectura de un archivo de configuración. Este archivo de configuración llamado `config`, incluye las funciones necesarias para extraer la configuración introducida por el usuario desde un archivo YAML. Dicho archivo de configuración contiene los siguientes campos:

- "projectID": Nombre del proyecto de GCP en el que se realizará el despliegue de las instancias.
- "zone": La zona en la que se ejecutará el despliegue, indicando la ubicación geográfica de los recursos.
- "machineType": El tipo de máquina que

se utilizará para el despliegue de las instancias, especificando sus características de CPU y almacenamiento. `instanceNamePrefix`: El prefijo que se utilizará como nombre para las máquinas desplegadas, permitiendo su identificación y distinción. `numInstances`: El número de instancias que se desplegarán para la simulación distribuida. `bucketName`: El nombre del bucket de GCP que se utilizará para el despliegue, donde se almacenarán algunos archivos necesarios para la simulación. `diskType`: El tipo de disco que utilizarán las instancias, determinando su rendimiento y características de almacenamiento. `diskSizeGb`: El tamaño del disco asignado a cada instancia, especificado en gigabytes (GB). `version`: La versión de la simulación utilizada para el despliegue. `time`: Una aproximación del tiempo en horas que se estima que durará la ejecución de la simulación distribuida.

3.5. Despliegue

Para la funcionalidad de despliegue, el diseño se puede apreciar en el diagrama de secuencia de la funcionalidad (figura: 3.2) y el diagrama de arquitectura explicado previamente.

3.6. Ejecución de la simulación distribuida

El framework desarrollado en este TFG ofrece dos formas diferentes de ejecución. La primera consiste en enviar la orden de ejecución desde un ordenador personal a través de SSH a todas las instancias desplegadas. La segunda forma utiliza un proxy, que implica la creación de una nueva instancia que actúa como intermediario. Desde el ordenador personal, se envía la orden al proxy, y este se encarga de comunicar a todas las instancias que deben iniciar la ejecución de su parte de la simulación distribuida.

Enviar la orden directamente a las instancias desde el ordenador personal permite prescindir del proxy y no depender de él para ejecutar la simulación. Esto puede ser beneficioso en situaciones donde se quiera evitar posibles fallos relacionados con el proxy que podrían originarse en GCP.

Por otro lado, la ejecución a través del proxy ofrece varias ventajas notables:

Latencia reducida: Al conectarse desde el interior de la red de GCP, las comunicaciones entre las instancias y otros servicios de GCP experimentan una latencia más baja debido a la menor distancia y al tráfico de red interno optimizado.

Mejora de la seguridad: Al mantener las conexiones dentro de la red interna de GCP, se puede aprovechar las políticas de seguridad y los controles de red de GCP, como cortafuegos y configuraciones de acceso restringido. Esto proporciona una capa

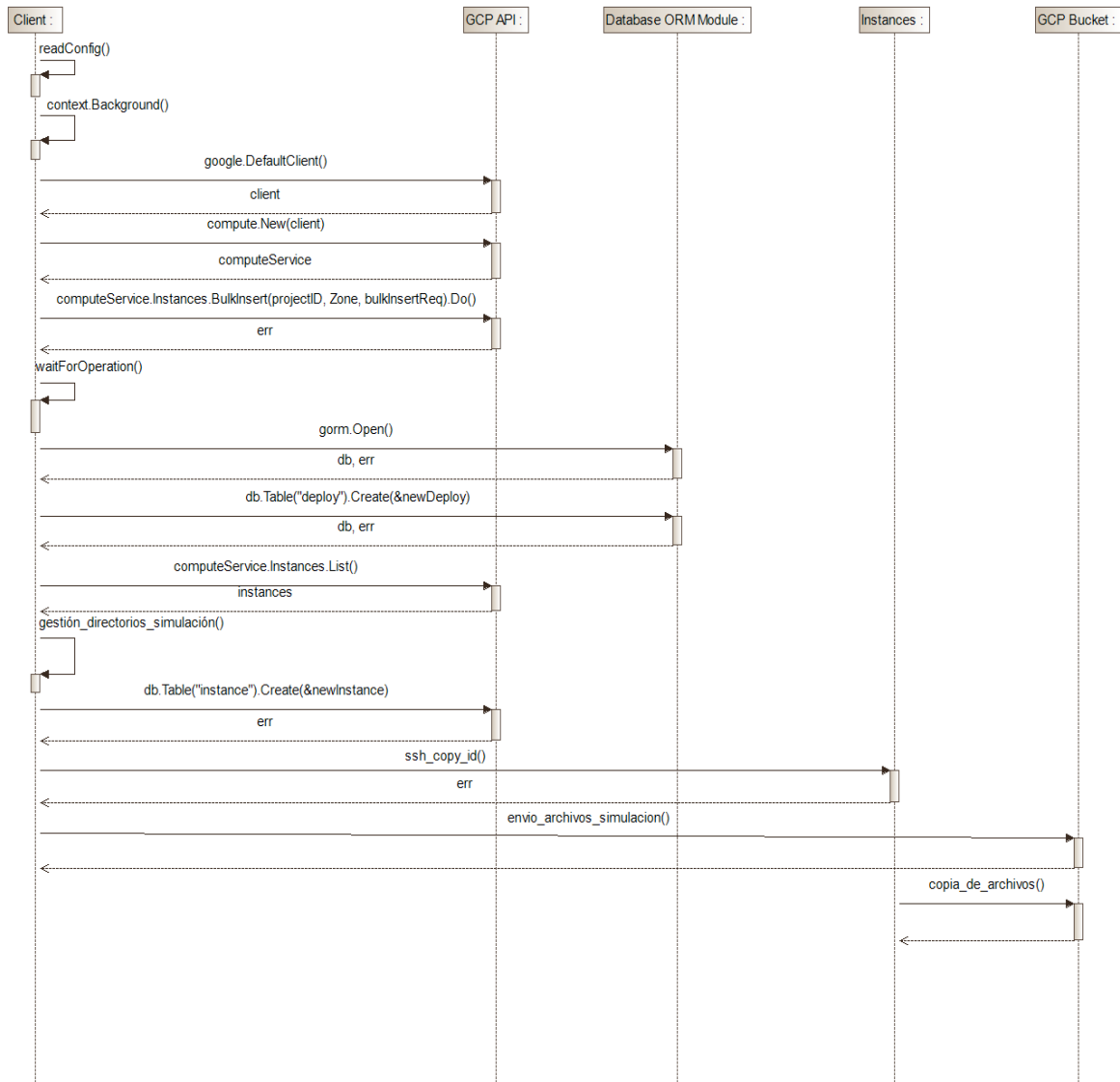


Figura 3.2: Diagrama de secuencia de la funcionalidad de despliegue.

adicional de seguridad durante la ejecución de la simulación distribuida. Transferencia de datos más rápida: Si se requiere transferir grandes volúmenes de datos entre las instancias y otros servicios de GCP, hacerlo desde el interior de la red de GCP puede ser más rápido y eficiente en comparación con realizarlo a través de conexiones externas.

Estas ventajas de ejecución a través del proxy hacen que sea una opción atractiva en casos donde la latencia reducida, la seguridad mejorada y la transferencia rápida de datos son aspectos críticos para el éxito y rendimiento de la simulación distribuida.

Cabe destacar también que dado que el número de instancias puede ser muy elevado, el hecho de enviar órdenes a través de ssh de forma iterativa puede suponer un gran coste en tiempo, por lo que el envío de órdenes se ha paralelizado mediante el uso de varios hilos de ejecución concurrentes.

Se plantean los diagramas de secuencia (figura: 3.3 y figura: 3.4)

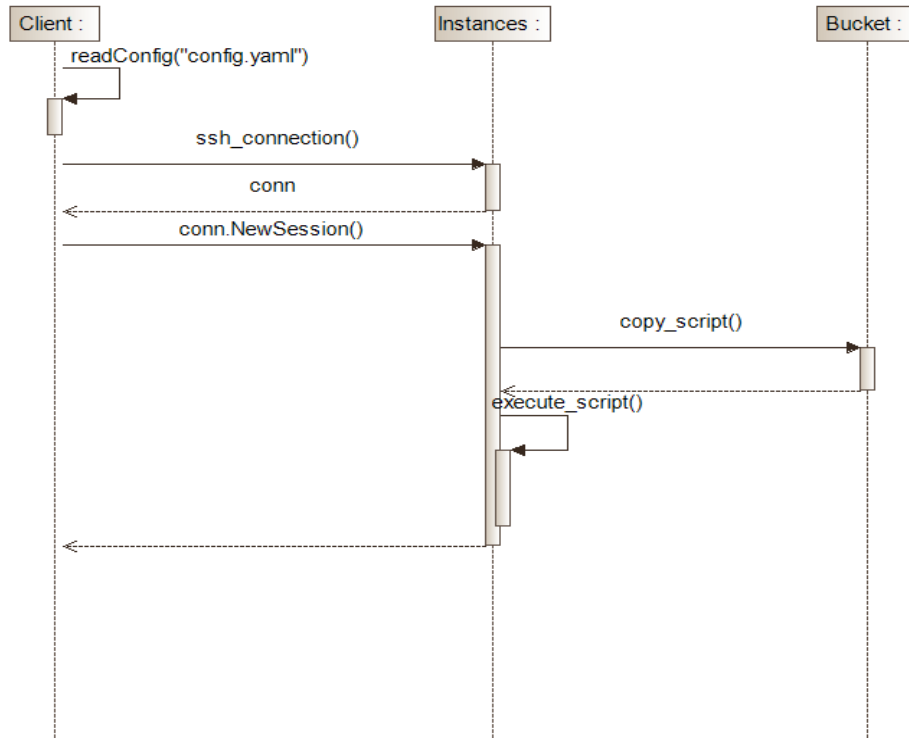


Figura 3.3: Diagrama de secuencia de la funcionalidad de ejecución de la simulación distribuida.

3.7. Destrucción de las instancias y limpieza del entorno

Para la funcionalidad de eliminación de los recursos, el diseño se puede apreciar en los diagramas de secuencia de la funcionalidad (figura: 3.5 y figura: 3.6). Como hemos explicado previamente, tenemos las dos opciones de diseño; la destrucción sin el uso de una instancia que actúe como proxy y la destrucción mediante el proxy. Se utilizan diferentes hilos de ejecución paralelos para aumentar la velocidad del borrado.

3.8. Análisis de precios

El análisis de los precios de despliegue de las instancias ha optado por realizarse utilizando la web de GCP que proporciona estos datos. Se ha creado un comando que toma los valores de la web en tiempo real y los dispone en la terminal.

3.9. Disponibilidad de cuotas

La presentación de la disponibilidad de cuotas es una funcionalidad relevante ya que limita la cantidad de recursos que tiene permitido el usuario desplegar en GCP.

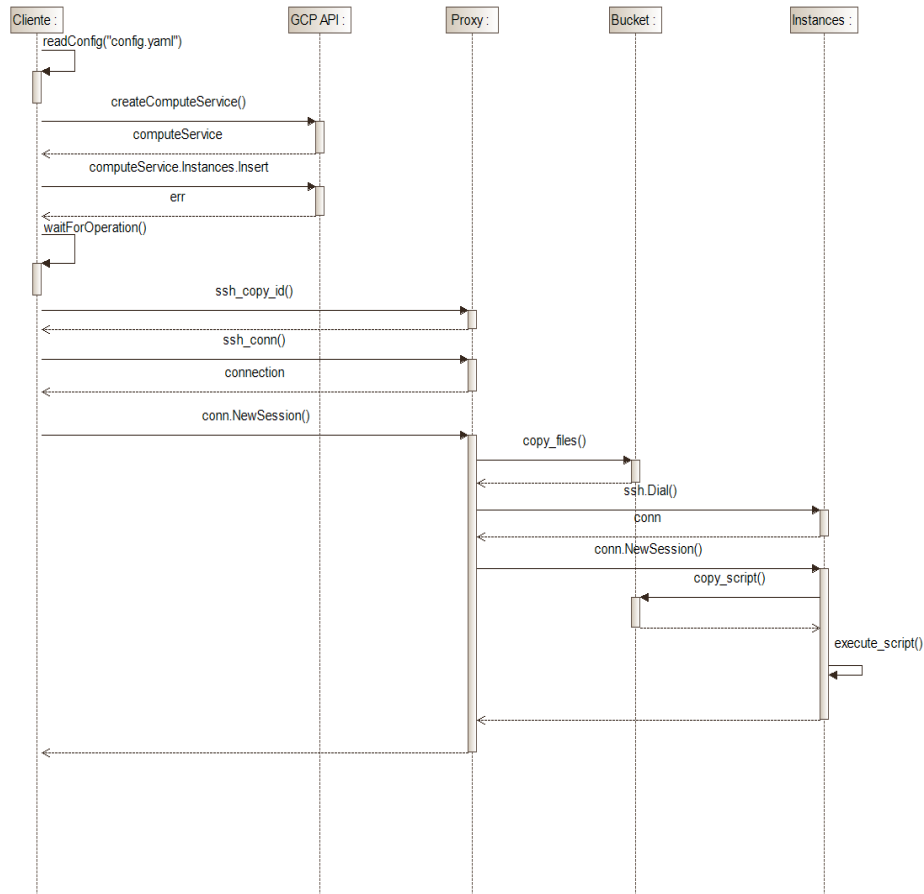


Figura 3.4: Diagrama de secuencia de la funcionalidad de ejecución de la simulación distribuida utilizando el proxy.

Se presentan las cuotas utilizadas para el despliegue de instancias las cuales abarcan desde el número de instancias permitido hasta el máximo de direcciones asignables a esas instancias.

3.10. Depuración de la simulación

Las simulaciones distribuidas ejecutadas pueden durar mucho tiempo e incluso, llegado un punto, puede ya no ser necesario continuar con la simulación, por lo que se debe poder controlar en todo momento el estado de la ejecución de la simulación en cada una de las instancias. Se ha implementado una funcionalidad que permite mandar una orden a cada una de las instancias, ya sea desde el proxy o desde el ordenador personal, a cada una de las instancias para que estas envíen el estado de su simulación al bucket de GCP para así poder evaluarlo.

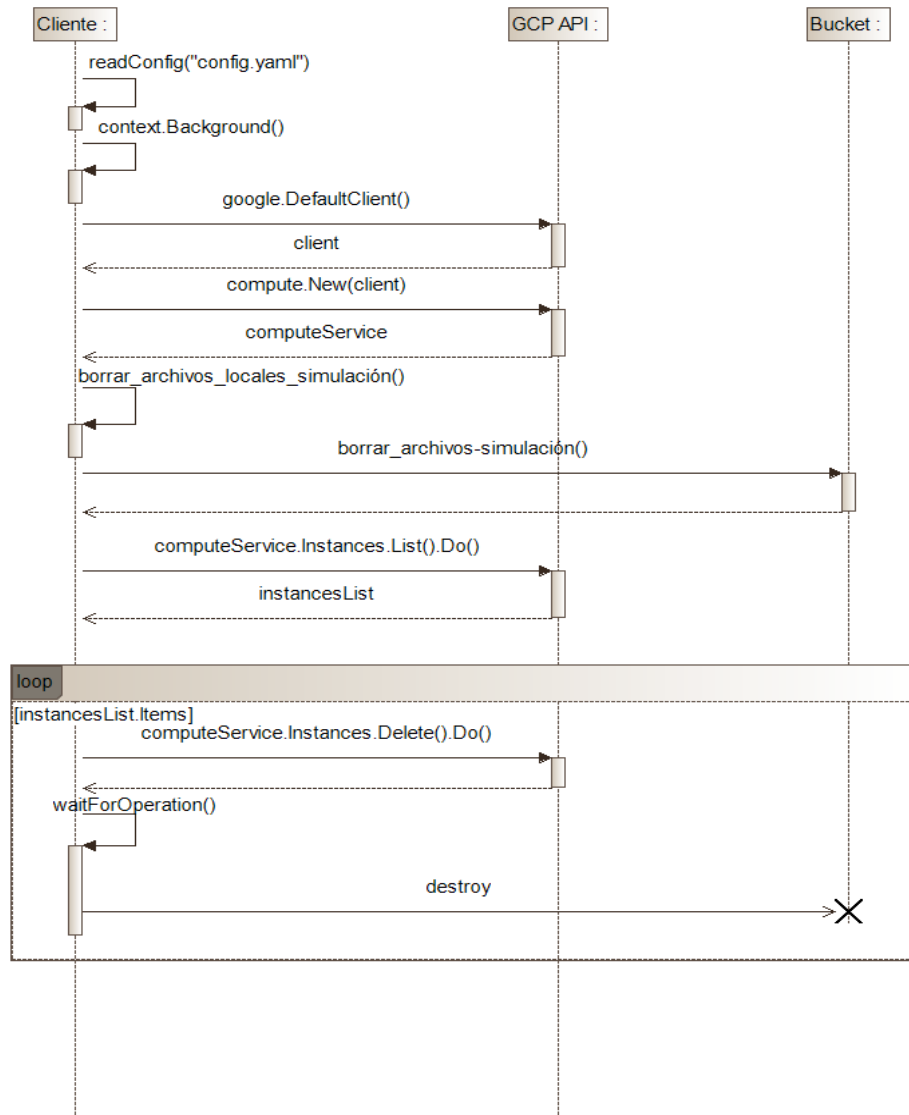


Figura 3.5: Diagrama de secuencia de la funcionalidad de destrucción de las instancias y limpieza del entorno.

3.11. Diseño de la base de datos

La base de datos que se utilizará para gestionar los registros de despliegues y las instancias desplegadas en el contexto de este TFG. La base de datos desempeña un papel fundamental en el seguimiento y la gestión de los datos generados durante el proceso de simulación distribuida.

Para garantizar la eficiencia y la coherencia en el almacenamiento de información, se diseñará una estructura de base de datos que contemple las necesidades específicas de registro y seguimiento de los despliegues. Esta estructura incorporará las tablas `deploy` e `instances`, detalladas en la sección de implementación, para capturar de manera organizada y accesible los datos esenciales relacionados con los despliegues y las instancias creadas.

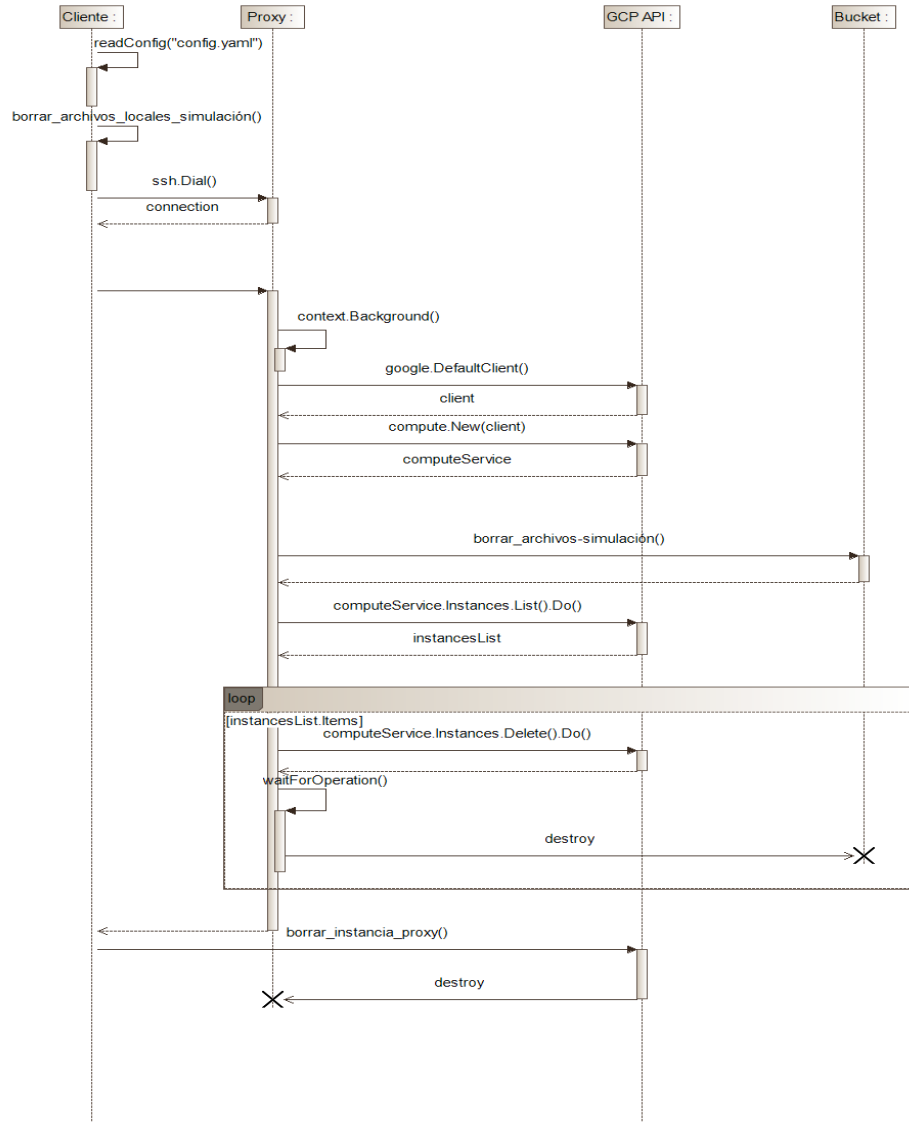


Figura 3.6: Diagrama de secuencia de la funcionalidad de destrucción de las instancias y limpieza del entorno mediante el uso del proxy.

Asimismo, en la fase de diseño, se han establecido las relaciones y conexiones entre estas tablas y las entidades ya existentes en la base de datos utilizada para la gestión de usuarios y simulaciones. Se han definidos las claves primarias y secundarias, así como las restricciones necesarias para mantener la integridad de los datos y permitir consultas eficientes.

Capítulo 4

Implementación

En la fase de implementación de este proyecto, se emplearon una serie de herramientas que desempeñaron un papel crucial en la materialización de los objetivos establecidos. Golang, un lenguaje de programación de alto rendimiento, fue la piedra angular de gran parte de la solución. Se ha utilizado para desarrollar los componentes que operan en la infraestructura distribuida, permitiendo la creación de sistemas eficientes y escalables que ejecutan las simulaciones distribuidas de manera eficaz.

Para la automatización de tareas en las instancias virtuales, se optó por utilizar scripts en el lenguaje de shell bash. Estos scripts desempeñaron un papel fundamental en la configuración y ejecución de operaciones en las instancias, contribuyendo a la fluidez y coherencia de las tareas distribuidas.

En cuanto a la persistencia y gestión de datos de las simulaciones, se implementó una base de datos PostgreSQL. La elección de PostgreSQL se basó en su capacidad probada para manejar datos estructurados, proporcionando una solución confiable para almacenar y acceder a resultados intermedios y configuraciones de simulación.

En las etapas iniciales del proyecto, Terraform desempeñó un papel importante al permitir pruebas preliminares y experimentos en un entorno controlado. A través de Terraform, se pudieron evaluar diferentes configuraciones de infraestructura y obtener información valiosa para afinar y optimizar la arquitectura de la solución antes de la implementación del framework.

Se presentan de forma más detallada algunos de estos aspectos de implementación en el ANEXO C.

A continuación se van a analizar los aspectos de implementación atendiendo a las decisiones tomadas en cada una de las funcionalidades del framework.

4.1. Configuración inicial

Las opciones de configuración mencionadas anteriormente en el apartado de diseño relacionadas con esta funcionalidad son utilizadas por las funcionalidades que componen el framework. La primera parte del framework se encarga de realizar el despliegue de las instancias y aprovisionar tanto el sistema de almacenamiento persistente de bajo coste como las propias instancias con los archivos necesarios para la ejecución de la simulación distribuida.

4.2. Despliegue

El despliegue se lleva a cabo mediante la creación masiva de instancias disponibles en Google Cloud Platform (GCP). Esta acción implica una llamada a la API de GCP que permite crear de forma paralela un número determinado de instancias. Esta llamada a la API se llama BulkInsert y recibe como parámetro una plantilla de configuración de las instancias que van a ser creadas la cual incluye el número de instancias a ser creadas entre otros campos. Dependiendo de las cuotas contratadas con Google Cloud, este número puede ser muy elevado. La creación masiva de instancias presenta varias ventajas significativas en comparación con la creación iterativa. El uso de esta llamada reduce considerablemente el tiempo de despliegue y el número de llamadas a la API, puesto que hay que tener en cuenta que existe un límite establecido.

Una vez desplegadas las instancias, se obtienen y almacenan las direcciones IP asignadas por Google tanto internamente como externamente a la VPC de GCP. A continuación, se realiza el aprovisionamiento del bucket de Google, que representa el sistema de almacenamiento persistente de bajo coste, con los archivos necesarios para la ejecución de la simulación distribuida. Estos archivos incluyen un script que se ejecutará dentro de las instancias y los archivos de la simulación distribuida.

4.3. Ejecución de la simulación distribuida

Una vez completado el despliegue de las instancias, la siguiente fase del framework se encarga de la ejecución de la simulación distribuida. Cada instancia desplegada adquiere los archivos necesarios para su ejecución desde el bucket de Google. El bucket se ha aprovisionado con cada una de las subredes que componen la ejecución distribuida, y cada instancia selecciona la subred correspondiente identificándola a través de su ID, que coincide con el ID de la instancia. Una vez que la instancia ha sido aprovisionada con los archivos necesarios, se ejecuta la simulación distribuida. La ejecución en cada

instancia desplegada se inicia mediante la ejecución del script obtenido del bucket. La orden para ejecutar este script viene dada a través de ssh.

Una vez finalizada la simulación distribuida, cada instancia envía un archivo con los registros (logs) obtenidos durante la ejecución de la simulación al bucket de Google. Esta información se almacena para su posterior análisis.

Como se ha mencionado en el apartado de diseño, el framework desarrollado en este TFG ofrece dos formas diferentes de ejecución. La primera consiste en enviar la orden de ejecución desde un ordenador personal a través de SSH a todas las instancias desplegadas. La segunda forma utiliza un proxy, que implica la creación de una nueva instancia que actúa como intermediario. Desde el ordenador personal, se envía la orden al proxy, y este se encarga de comunicar a todas las instancias que deben iniciar la ejecución de su parte de la simulación distribuida.

Ambas formas de ejecución tienen ventajas distintas, y su elección dependerá de las prioridades y requisitos específicos del usuario.

4.4. Destrucción de las instancias y limpieza del entorno

La última parte del despliegue consiste en la eliminación de los recursos. Para ello, se realiza una iteración sobre los recursos desplegados en la zona y se eliminan de forma iterativa. En esta fase también se hace uso de las goroutines de Golang para mejorar la eficiencia y la velocidad de eliminación de los recursos.

Inicialmente, se eliminan las instancias creadas durante el despliegue, así como el proxy en caso de que se haya creado.

Además, se eliminan los archivos relacionados con esa ejecución del bucket de Google, con la excepción de los archivos de log de las instancias. Estos archivos de depuración se conservan para que puedan ser analizados posteriormente y obtener información relevante sobre la ejecución de la simulación distribuida.

La utilización de goroutines en esta fase de eliminación permite una mayor eficiencia en el proceso de limpieza de los recursos, ya que las operaciones de eliminación pueden realizarse de forma concurrente. Esto reduce el tiempo total requerido para eliminar los recursos y mejora el rendimiento general del framework. Cabe destacar que, a diferencia del despliegue, Google Cloud Platform (GCP) no proporciona una forma nativa de realizar una eliminación masiva de recursos de manera eficiente. Esto puede deberse a consideraciones económicas, ya que un borrado masivo podría disminuir significativamente los costos para el cliente al garantizar la eliminación en poco tiempo.

4.5. Análisis de precios

A parte de las mencionadas, se han desarrollado funcionalidades adicionales que complementan el proceso de despliegue. La primera de ellas consiste en la selección de la zona más económica para el despliegue de las instancias. GCP ofrece diferentes precios para las diferentes zonas en el despliegue de instancias debido a varios factores y consideraciones: Costos de infraestructura: Las diferentes zonas de GCP están ubicadas en diferentes regiones geográficas y centros de datos. Estos centros de datos pueden tener costos de infraestructura variables en función de la ubicación, la disponibilidad de recursos, la demanda local y otros factores. Por lo tanto, los precios pueden variar para reflejar estos diferentes costos operativos. Disponibilidad y capacidad: Algunas zonas de GCP pueden tener una mayor disponibilidad de recursos o una mayor capacidad de computación en comparación con otras zonas. Esto puede deberse a la distribución desigual de la infraestructura o a la demanda regional de servicios en la nube. Los precios pueden reflejar esta variación en la disponibilidad y capacidad de recursos en cada zona. Por estos motivos, los precios pueden ser muy diferentes, por lo que seleccionar la zona más económica para el despliegue puede resultar en un gran ahorro en el coste. GCP ofrece información sobre el coste de las instancias para cada zona en su página web. El comando 'price' del framework extrae los valores de crear una instancia del tipo establecido en el fichero de configuración y los presenta organizándolos de forma geográfica mostrando la zona más económica para desplegar y las zonas más económicas para cada región. También realiza una aproximación del coste total que va a tener el despliegue teniendo en cuenta la aproximación de tiempo establecida en el fichero inicial de configuración y que el despliegue se va a realizar en la zona que menor coste tiene. De esta forma se puede realizar una predicción del coste de la ejecución de la simulación distribuida.

4.6. Disponibilidad de cuotas

Otra funcionalidad implementada consiste en la identificación de los límites de cuotas disponibles para la zona seleccionada en el despliegue de instancias de Google Cloud Platform (GCP). Estas cuotas representan restricciones impuestas por GCP y limitan la cantidad máxima de recursos que se pueden utilizar en una determinada zona.

Estas cuotas incluyen el número máximo de instancias que se pueden crear y el número máximo de direcciones IP que se pueden asignar a las instancias. Estas restricciones pueden variar según la región y la zona geográfica, y están diseñadas para

garantizar un uso equitativo de los recursos y evitar un consumo excesivo.

La funcionalidad implementada en el framework permite obtener información sobre estas cuotas y presentarla al usuario. Esto proporciona una visibilidad clara de las limitaciones existentes y permite tomar decisiones informadas al configurar la cantidad de instancias y direcciones IP que se desean desplegar en una zona determinada.

4.7. Depuración de la simulación

La siguiente funcionalidad aborda la depuración de la ejecución de la simulación distribuida, proporcionando una forma de analizar los resultados y detener la ejecución en caso de que se obtengan resultados suficientes o se identifique la necesidad de interrumpir la simulación.

Esta funcionalidad, denominada 'debug', permite ejecutar una orden en las instancias desplegadas que provoca el envío de los datos de depuración al sistema de almacenamiento persistente de Google Cloud Platform (GCP), es decir, al bucket de Google. Cada instancia de la simulación distribuida enviará sus datos de depuración correspondientes al bucket, lo que permitirá recopilar y analizar la información de cada parte de la simulación.

Una vez que los logs se encuentran en el bucket, se pueden examinar detenidamente para analizar los resultados obtenidos hasta el momento. Esta información es crucial para evaluar el progreso de la simulación y determinar si es necesario continuar ejecutándola o si se han alcanzado los resultados deseados.

4.8. Uso de la base de datos

Por último, este TFG se ha integrado con un TFG anterior que aborda la gestión de usuarios y simulaciones utilizando una base de datos. Para facilitar esta integración, se ha utilizado un ORM (Object-Relational Mapping) de Golang, que permite realizar inserciones de objetos en la base de datos de manera más sencilla y eficiente.

Se han añadido dos tablas adicionales a la base de datos existente para gestionar adecuadamente los despliegues para cada usuario. La primera tabla, llamada 'deploy', se encarga de almacenar los registros de los despliegues realizados por cada usuario. Esta tabla guarda información relevante sobre cada despliegue, como el usuario asociado, la configuración utilizada y otros detalles relacionados.

La segunda tabla, llamada "instances", se encarga de almacenar las instancias desplegadas para cada despliegue. Cada registro en la tabla "instances" representa una instancia específica desplegada en el contexto de un despliegue. Los atributos de cada

instancia incluyen la dirección IP externa proporcionada por Google, el ID de la instancia que coincide con el ID de la subred de la simulación distribuida a la que pertenece y el archivo de datos de depuración correspondiente a esa instancia.

Para cada despliegue, se establece una conexión con la base de datos PostgreSQL y se realizan las inserciones necesarias en las tablas 'deploy' e 'instances' para registrar la información relevante del despliegue y las instancias desplegadas.

El uso del ORM de Golang facilita la interacción con la base de datos, ya que permite mapear los objetos del código a las tablas de la base de datos de manera automática y realizar operaciones de inserción de forma más intuitiva. Se ha incluido en el ANEXO E el código SQL de creación de las tablas de la base de datos.

Capítulo 5

Validación y pruebas

5.1. Prueba de Simulación Distribuida

Se ha empleado una red de Petri inicial compuesta por tres subredes para probar el funcionamiento del framework desarrollado. Esto implica un despliegue de tres instancias, cada una ejecutando una de las subredes.

Esta prueba exhaustiva abarca todas las funcionalidades disponibles en el framework. Incluye la evaluación de las necesidades de la simulación distribuida, el análisis del precio de despliegue de instancias en las diferentes zonas de GCP, la estimación de costos de la simulación, el despliegue de instancias, el aprovisionamiento del sistema de almacenamiento persistente de bajo coste con los archivos necesarios para la ejecución de la simulación distribuida, y el aprovisionamiento de las instancias con los archivos correspondientes a la subred asignada.

Durante la ejecución de la simulación distribuida en cada instancia, se realizan inserciones de datos relevantes en la base de datos. Se incluye una fase de depuración para verificar el estado de la simulación en cada instancia, seguida de la recopilación final de los resultados de la simulación distribuida para cada instancia.

Después de completar la simulación, se procede al borrado de las instancias y los archivos, dejando el entorno limpio para la ejecución de futuras simulaciones.

Además, se ha desarrollado una segunda prueba utilizando el proxy junto con la misma red de Petri. Esto implica la creación del proxy antes de crear las instancias, y el aprovisionamiento del proxy con las direcciones de red interna de Google asignadas a cada instancia, así como los archivos Go necesarios para enviar comandos a las instancias. También se incluye la fase de borrado del proxy al finalizar la simulación.

Estas pruebas exhaustivas permiten verificar el correcto funcionamiento del framework en diferentes escenarios, tanto con despliegue directo en las instancias como con el uso de un proxy.

5.2. Prueba de volumen

Se realizaron pruebas para analizar la relación entre la cantidad de instancias desplegadas y el tiempo de despliegue utilizando la llamada a la API de creación masiva de instancias proporcionada por GCP.

Como se puede observar en los resultados, el despliegue utilizando la llamada a la API "bulkInsert".^{es} considerablemente más rápido que el proceso de borrado, el cual se realiza de forma iterativa. Para mejorar el tiempo de borrado, se ha aprovechado el uso de las goroutines de Golang. Al ejecutar el proceso de borrado en un dispositivo con mayor capacidad de CPU y aumentar el número de goroutines utilizadas, se puede lograr una reducción significativa en el tiempo necesario para completar el borrado.

Estas pruebas demuestran que, desde una perspectiva económica, a Google le interesa que los clientes dispongan rápidamente de las instancias para poder iniciar el cobro lo antes posible. Por otro lado, existe una tendencia a retrasar el proceso de borrado, ya que durante este período, el cliente continúa generando cargos, lo que aumenta los ingresos para Google.

Observamos una tabla con las mediciones tomadas.

Número de instancias	Tiempo de creación(bulk)	Tiempo de borrado (10 gorutinas)	Tiempo de borrado (40 gorutinas)
10	29.003536257s	56.308817699s	
50	40.326825824s	4m23.264089945s	
100	32.44116711s	8m26.037068489s	
1000	1m5.327291933s		20m56.0976644737s
2300	4m18.719773661s		43m35.64657388383

Tabla 5.1: Tabla de tiempos de creación y borrado de instancias

Para comparar también la ventaja en la velocidad de despliegue obtenida, se han hecho pruebas de despliegue iterativo de instancias mediante llamadas individuales de creación de instancias a la API de GCP obteniéndose los siguientes resultados:

Número de instancias	Tiempo de creación (Iterativo)
1	24.748385768s
3	1m12.785467654s
10	4m54.243454564s
20	9m02.153746474s

Tabla 5.2: Tabla de tiempos de creación iterativa de instancias

5.3. Prueba de velocidad de borrado

Se ha comparado el proceso de borrado entre hacerlo de forma iterativa y hacerlo de forma paralelizada. Para ello se ha creado un programa en Go que despliega una

cantidad determinada de instancias y las borra midiendo el tiempo necesario para ello. Se obtienen los siguientes resultados:

Número de instancias	Borrado Iterativo	Borrado con Paralelización (10 Gorutinas)
3	2m45.537283563s	50.639472239s
10	8m12.647859395s	53.638495747s
15	13m50.465738474s	1m45.748585858s

Tabla 5.3: Tabla de tiempos de borrado de instancias de forma iterativa y paralelizada

Cómo se ha mencionado antes, al ejecutar ambos procesos en un dispositivo con mayor capacidad de CPU y aumentar el número de goroutines utilizadas, se puede lograr una reducción significativa en los tiempos.

5.4. Prueba de latencia

Se han realizado pruebas de latencia entre las máquinas utilizadas en el framework. Para ello se ha realizado un despliegue y se han tomado mediadas de latencia.

Origen	Destino	Latencia (RTT)
Local	Instancia	300.505 ms
Local	Proxy	202.522 ms
Proxy	Instance	0.519 ms

Tabla 5.4: Tabla de tiempos de de latencia entre una terminal en una red externa a GCP e instancias de la red interna de GCP

Los resultados obtenidos confirman los beneficios significativos del uso de un proxy en términos de latencia, especialmente gracias al aprovechamiento de la red interna de Google. Estos resultados validan las suposiciones y consideraciones desarrolladas en la sección de arquitectura del proyecto.

Durante las pruebas, se ha comparado la latencia entre las comunicaciones directas entre las máquinas y las comunicaciones a través del proxy. Se observa una reducción notable en la latencia cuando se utilizó el proxy y se establecieron las conexiones dentro de la red interna de Google. Esta reducción se debe a la menor distancia y al tráfico de red interno optimizado en comparación con las comunicaciones externas.

Los resultados de estas pruebas proporcionan evidencia concreta de la eficacia del proxy en la reducción de la latencia y mejoramiento de las comunicaciones en la simulación distribuida. Estos hallazgos respaldan las decisiones de diseño y arquitectura del framework, que se basaron en la utilización de la red interna de Google para optimizar el rendimiento y la eficiencia en las comunicaciones.

Capítulo 6

Conclusiones

En conclusión, el presente Trabajo de Fin de Grado ha logrado cumplir satisfactoriamente los objetivos propuestos de diseñar y desarrollar un framework de despliegue de simulaciones distribuidas en la nube. El framework se ha enfocado en aspectos clave como la velocidad de despliegue de un gran número de máquinas virtuales y la consideración central del coste de despliegue en la nube.

Durante el desarrollo del framework, se ha tenido en cuenta la gestión básica de datos, comunicación y almacenamiento, abordando de manera efectiva aspectos cruciales para el despliegue del código de simulación, los modelos a simular y la recuperación de resultados. Se han implementado funcionalidades adicionales, como la selección de la zona más económica para el despliegue, la identificación de límites de cuotas de recursos, la depuración de la ejecución y la integración con una base de datos existente para la gestión de usuarios y simulaciones.

Las pruebas exhaustivas realizadas han demostrado el rendimiento y escalabilidad del framework. Se ha comprobado su capacidad para manejar un gran número de instancias en paralelo, reduciendo significativamente el tiempo de despliegue gracias a la creación masiva de instancias. Además, se ha observado una comunicación eficiente y de baja latencia entre las instancias, optimizando el rendimiento de la simulación distribuida.

En términos económicos, el framework ha proporcionado funcionalidades para analizar el coste de despliegue en diferentes zonas de Google Cloud Platform (GCP), permitiendo a los usuarios seleccionar la zona más económica y optimizar los gastos asociados.

En general, el framework ha cumplido con los objetivos propuestos, ofreciendo una solución especializada para el despliegue de simulaciones distribuidas en la nube. Se ha demostrado su eficacia en términos de velocidad de despliegue, costes y gestión de recursos. El desarrollo de funcionalidades adicionales ha enriquecido su capacidad y versatilidad, ofreciendo una solución completa que aborda los desafíos clave en la

simulación distribuida.

El presente TFG sienta las bases para futuras investigaciones y mejoras en el ámbito del despliegue de simulaciones distribuidas en la nube, contribuyendo al avance y la aplicación de tecnologías en esta área. Algunos aspectos no han recibido una atención exhaustiva, como la exploración de oportunidades para enriquecer la operativa del proxy con funcionalidades adicionales. Existen posibilidades de mejora y expansión en cuanto a dichos aspectos lo cual permite el uso de este TFG como una base para TFGs futuros.

Capítulo 7

Bibliografía

- [1] Gcp console. <https://console.cloud.google.com/>.
- [2] Documentación de pulumi. <https://www.pulumi.com/docs/>.
- [3] Documentación de golang. <https://go.dev/doc/>.
- [4] Documentación de terraform. [https://developer.hashicorp.com/terraform?product_intent = terraform](https://developer.hashicorp.com/terraform?product_intent=terraform).
- [5] Documentación sobre creación masiva de instancias en gcp. <https://cloud.google.com/compute/docs/instances/multiple/about-bulk-creation?hl=es-419>.
- [6] Zongheng Yang, Zhanghao Wu, Michael Luo, Wei-Lin Chiang, Romil Bhardwaj, Woosuk Kwon, Siyuan Zhuang, Frank Sifei Luan, Gautam Mittal, Scott Shenker, and Ion Stoica. Skypilot: An intercloud broker for sky computing. In Mahesh Balakrishnan and Manya Ghobadi, editors, *20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023*, pages 437–455. USENIX Association, 2023.
- [7] Gcp instance pricing. <https://cloud.google.com/compute/all-pricing?hl=es-419>.
- [8] Andrés María Gilgado Barrachina, Unai Arronategui Arribalzaga, and José Ángel Bañares Bañares. Diseño e implementación de un servicio web para gestión de simulaciones distribuidas. <https://zaguan.unizar.es/record/126172>, 2022.
- [9] Documentación de slurm. <https://slurm.schedmd.com/documentation.html>.
- [10] Gcp storage. <https://cloud.google.com/storage/docs/creating-buckets?hl=es-419>.
- [11] Gcp preemptible and spot instances. <https://cloud.google.com/compute/docs/instances/spot?hl=es-419>.
- [12] Gcp cuotas. <https://cloud.google.com/vpc/docs/quota?hl=es-419>.

Lista de Figuras

3.1. Diagrama de arquitectura del framework utilizando el proxy	14
3.2. Diagrama de secuencia de la funcionalidad de despliegue.	18
3.3. Diagrama de secuencia de la funcionalidad de ejecución de la simulación distribuida.	19
3.4. Diagrama de secuencia de la funcionalidad de ejecución de la simulación distribuida utilizando el proxy.	20
3.5. Diagrama de secuencia de la funcionalidad de destrucción de las instancias y limpieza del entorno.	21
3.6. Diagrama de secuencia de la funcionalidad de destrucción de las instancias y limpieza del entorno mediante el uso del proxy.	22
A.1. Diagrama de Gantt del TFG.	39
D.1. Ejecución del comando Price.	45
D.2. Ejecución del comando Availability.	45
D.3. Ejecución del comando Deploy.	46
D.4. Instancias tras ejecución del comando Deploy.	46
D.5. Ejecución del comando Exec.	46
D.6. Bucket tras ejecución del comando Exec.	47
D.7. Bucket tras ejecución del comando Debug.	47
D.8. Instancias tras ejecución del comando Exec-proxy.	48
D.9. Ejecución del comando Exec-proxy.	48

Lista de Tablas

- 5.1. Tabla de tiempos de creación y borrado de instancias 30
- 5.2. Tabla de tiempos de creación iterativa de instancias 30
- 5.3. Tabla de tiempos de borrado de instancias de forma iterativa y paralelizada 31
- 5.4. Tabla de tiempos de de latencia entre una terminal en una red externa
a GCP e instancias de la red interna de GCP 31

Anexos

Anexo A

Diagrama de Gantt

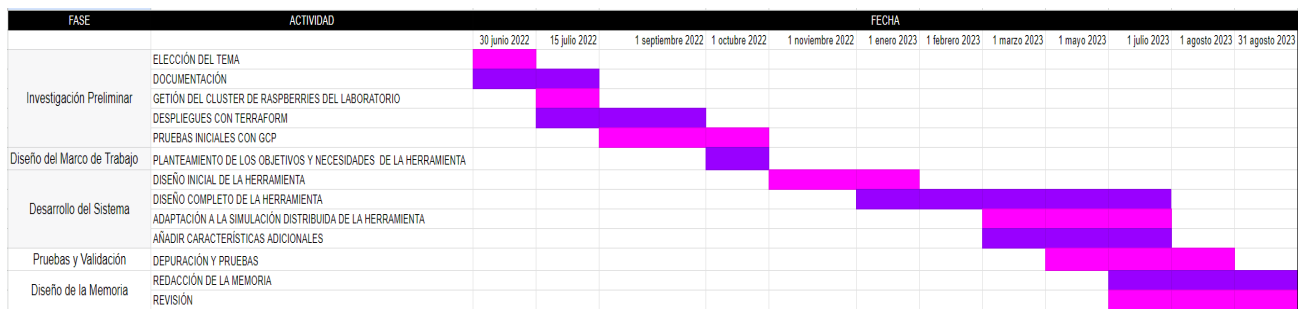


Figura A.1: Diagrama de Gantt del TFG.

El presente diagrama de Gantt representa la planificación y programación de actividades para la realización del Trabajo de Fin de Grado.

Anexo B

Horas de dedicación

En el marco de la realización de este Trabajo de Fin de Grado se llevó a cabo la siguiente asignación de tiempo y recursos para abordar las diversas etapas del proyecto de manera efectiva. A lo largo de un período de un año y dos meses, se dedicó esfuerzo a cada fase clave del proyecto con el objetivo de alcanzar los objetivos establecidos. A continuación, se presenta un resumen de las horas de dedicación para cada etapa:

- Investigación Preliminar (60 horas)
- Diseño del Marco de Trabajo (50 horas)
- Desarrollo del Sistema (150 horas)
- Pruebas y Validación (80 horas)
- Diseño de la Memoria (30 horas)

Anexo C

Aspectos relevantes de los comandos del framework

C.1. Deploy

Con respecto a la función de despliegue desarrollada en Golang cabe destacar las propiedades de las instancias que van a ser desplegadas:

```
bulkInsertReq := &compute.BulkInsertInstanceResource{
    ...
    InstanceProperties: &compute.InstanceProperties{
        ...
        Scheduling: &compute.Scheduling{
            Preemptible: true,
            //ProvisioningModel: "SPOT",
        },
        ...
    }
    ...
}
```

En este contexto, la propiedad Scheduling permite configurar la planificación de la instancia. Permite configurar la instancia cómo instancia preemptible e instancia spot.

```
op, err := computeService.Instances.BulkInsert(projectID, Zone, bulkInsertReq).Do()
if err != nil {
    log.Fatalf("Error creating instances: %v", err)
}
waitForOperation(computeService, projectID, Zone, op.Name)
```

Se ejecuta la llamada a la API de GCP de inserción masiva pasando la configuración de las instancias como parámetro. Se ha implementado también una función "waitForOperation" que espera hasta que la llamada a la API se ha completado.

C.2. Exec

En la función exec se va a incidir en el script bash que se ejecuta en las instancias de GCP desplegadas que inicia la ejecución de la simulación:

```
#!/bin/bash
hostname=$(hostname)
last_number=$(echo "${hostname}" | sed 's/.*-\([0-9]*\)$/\1/')
new_num=$((last_number - 1))
gsutil cp gs://my-bucket-angel-tfg/$zone/ejemplo_simulador_i_${new_num}.lefr /
/home/angel
gsutil cp gs://my-bucket-angel-tfg/$zone/ip_port.txt /home/angel

while IFS=: read -r ip puerto; do
    ips_puertos+=("$ip:$puerto")
done < ip_port.txt

ips_puertos_str=$(IFS=','; echo "${ips_puertos[*]}")
./distrisim ejemplo_simulador_i_${new_num}.lefr ${new_num} 5 /
["$ips_puertos_str"] >> output_${hostname}.txt
gsutil cp output_${hostname}.txt gs://my-bucket-angel-tfg/$zone/
```

Este script bash captura el nombre del host, extrae su número y realiza una serie de tareas, incluyendo la copia de archivos desde GCS, la extracción de las ips del resto de instancias de la simulación, la ejecución del simulador distribuido con los parámetros necesarios y la copia de los resultados obtenidos a GCS.

C.3. Destroy

De la función Destroy desarrollada en golang cabe destacar el bucle encargado de la eliminación de la instancias:

```
instancesListCall := computeService.Instances.List(config.ProjectID, config.Zone)
instancesList, err := instancesListCall.Do()
if err != nil {
```

```

log.Fatalf("Error al listar las instancias: %v", err)
}
var wg sync.WaitGroup
for _, instance := range instancesList.Items {
wg.Add(1)
go func(instanceName string) {
defer wg.Done()
instanceDeleteCall := computeService.Instances.Delete
    (config.ProjectID, config.Zone, instanceName)
op, err := instanceDeleteCall.Do()
if err != nil {
log.Printf("Error al eliminar la instancia %v: %v", instanceName, err)
} else {
fmt.Printf("Instancia %v eliminada.\n", instanceName)
}
waitForOperation(computeService, config.ProjectID, config.Zone, op.Name)
}(instance.Name)
}
wg.Wait()

```

El fragmento de código utiliza la API de Google Compute Engine para listar y eliminar instancias de máquinas virtuales en paralelo.

Lista las instancias de la zona deseada, crea goroutines para eliminar cada instancia y utiliza una estructura de WaitGroup para asegurarse de que todas las goroutines de eliminación finalicen antes de continuar.

C.4. Price

Para la funcionalidad que muestra el precio del despliegue de las diferentes instancias se ha tratado la web de GCP que dispone de los precios. Para ello se han utilizado dos librerías: "github.com/antchfx/htmlquery" y "github.com/PuerkitoBio/goquery" las cuales permiten hacer un tratamiento del html de la web y disponer de la información de la misma en código. Son unas de las librerías más completas para esta finalidad.

Anexo D

Ejecución del framework

A continuación se va a describir un flujo de ejecución del framework para una simulación distribuida que consta de 3 subredes. Iniciamos por ejecutar el comando 'price' el cual muestra los precios de despliegue en cada zona para el tipo de máquina descrita en el archivo de configuración. Los datos se presentan en 3 tablas, la primera muestra la zona con el precio más económico en cada región, la segunda muestra la zona con el precio más económico en cada continente y la tercera muestra los precios de todas las zonas. Se muestra la zona más económica para desplegar y una predicción del coste que va a tener la simulación si desplegamos en la zona habiendo establecido previamente una estimación de la duración de la simulación en el archivo de configuración. Figura: D.1

Después vamos a ejecutar el comando Availability el cual muestra las cuotas disponibles en la zona establecida en el archivo de configuración. Figura: D.2

Después ejecutamos el comando deploy el cuál despliega las instancias y carga los ficheros necesarios para la simulación. Figura: D.4 y D.3

Después ejecutamos el comando Exec que es el encargado de ejecutar la simulación distribuida en cada una de las instancias. Al finalizar el proceso, cada una de las instancias envía al Bucket de GCP el resultado de la ejecución de la simulación. Figura: D.6 y D.5

Cómo podemos observar sólo 2 de las 3 instancias han finalizado satisfactoriamente y han logrado enviar los resultados al Bucket de GCP. Por lo que se va a ejecutar el comando Debug para que todas envíen el estado actual de la ejecución a un directorio del Bucket. Aquí podemos observar el estado de la ejecución en las tres instancias evaluando el fichero de salida. Figura: D.7 Finalmente ejecutamos el comando Destroy el cual elimina todos los recursos desplegados y los ficheros generados para la simulación. Finalmente se comprueba que los recursos se han eliminado satisfactoriamente. A su vez se pueden ver en la base de datos las filas que se han creado relacionadas con la simulación.

```

angel@angel-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Desktop/TFG/tool/tool$ go run mytool.go price
Hourly cost in USD of n1-standard-1 instance:
-----
Region: asiaeast, Subregion: asiaeast1, Price: 0.00880
Region: asianortheast, Subregion: asianortheast1, Price: 0.01081
Region: asiasouth, Subregion: asiasouth1, Price: 0.00772
Region: asiasoutheast, Subregion: asiasoutheast1, Price: 0.00935
Region: australiasoutheast, Subregion: australiasoutheast1, Price: 0.01094
Region: europenorth, Subregion: europenorth1, Price: 0.01100
Region: europewest, Subregion: europewest1, Price: 0.00891
Region: northamericanortheast, Subregion: northamericanortheast1, Price: 0.01100
Region: southamericaeast, Subregion: southamericaeast1, Price: 0.01510
Region: uscentral, Subregion: uscentral1, Price: 0.01000
Region: useast, Subregion: useast1, Price: 0.01000
Region: uswest, Subregion: uswest1, Price: 0.01000
-----
Cont: asia, Subregion: asiasouth1, Price: 0.00772
Cont: australia, Subregion: australiasoutheast1, Price: 0.01094
Cont: europe, Subregion: europewest1, Price: 0.00891
Cont: northamerica, Subregion: northamericanortheast1, Price: 0.01100
Cont: southamerica, Subregion: southamericaeast1, Price: 0.01510
Cont: us, Subregion: uswest1, Price: 0.01000
-----
Cheapest zone: asiasouth1 with 0.00772 USD/hour
-----
A deploy of 3 instances and 2 hours will have a price of: 0.04632 USD
-----
+-----+-----+-----+-----+-----+
| MACHINE TYPE | VIRTUAL CPUS | MEMORY | PRICE (USD) | SPOT PRICE* (USD) |
+-----+-----+-----+-----+-----+
| n1-standard-1 | 1 | 3.75GB | asiaeast1: 0.0550 asiaeast2: 0.0665 asianortheast1: 0.0610 asianortheast2: 0.0610 asianortheast3: 0.0610 asiasouth1: 0.0570 asiasouth2: 0.0586 asiasoutheast1: 0.0586 asiasoutheast2: 0.0586 australiasoutheast1: 0.0674 australiasoutheast2: 0.0674 europecentral2: 0.0612 europenorth1: 0.0523 europewest1: 0.0523 europewest2: 0.0612 europewest3: 0.0612 europewest4: 0.0523 europewest6: 0.0665 northamericanortheast1: 0.0523 northamericanortheast2: 0.0523 southamericaeast1: 0.0754 uscentral1: 0.04749975 useast1: 0.04749975 useast4: 0.0535 uswest1: 0.04749975 uswest2: 0.0571 uswest3: 0.0571 uswest4: 0.0535 | asiaeast1: 0.0088 asiaeast2: 0.01400 asianortheast1: 0.01081 asianortheast2: 0.01325 asianortheast3: 0.01081 asiasouth1: 0.00772 asiasouth2: 0.01203 asiasoutheast1: 0.00935 asiasoutheast2: 0.01024 australiasoutheast1: 0.01094 australiasoutheast2: 0.01349 europecentral2: 0.01230 europenorth1: 0.01100 europewest1: 0.00891 europewest2: 0.01230 europewest3: 0.00978 europewest4: 0.00893 europewest6: 0.01400 northamericanortheast1: 0.01100 northamericanortheast2: 0.00891 southamericaeast1: 0.0151 uscentral1: 0.01 useast1: 0.01 useast4: 0.01070 uswest1: 0.01 uswest2: 0.01202 uswest3: 0.00974 uswest4: 0.01070 |
+-----+-----+-----+-----+-----+
angel@angel-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Desktop/TFG/tool/tool$

```

Figura D.1: Ejecución del comando Price.

```

angel@angel-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Desktop/TFG/tool/tool$ go run mytool.go availability
Metric: DISKS_TOTAL_GB
Limit: 204800.000000
Usage: 0.000000

Metric: IN_USE_ADDRESSES
Limit: 2300.000000
Usage: 0.000000

Metric: INSTANCES
Limit: 24000.000000
Usage: 0.000000

Metric: PREEMPTIBLE_CPUS
Limit: 10000.000000
Usage: 0.000000

```

Figura D.2: Ejecución del comando Availability.

Ahora veremos la versión de la ejecución en la que se utiliza el proxy.

El primer paso; la ejecución del comando 'deploy' se ejecuta cómo en el caso anterior.

```
angel@angel-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Desktop/TFG/tool/tool$ go run mytool.go deploy
2023/08/31 00:19:52 proyectodespliegue1
2023/08/31 00:19:52 Creating 3 instances of type n1-standard-1 in project proyectodespliegue1 and zone us-central1-a ...
2023/08/31 00:20:00 Waiting for operation operation-1693433994846-6042b53b6a03f-6bb1509d-3173b1fa to complete...
2023/08/31 00:20:12 Operation operation-1693433994846-6042b53b6a03f-6bb1509d-3173b1fa completed successfully
2023/08/31 00:20:12 Created 3 instances!
Copying SSH key to
Copying SSH key to 34.72.97.61
Copying SSH key to 34.134.11.40
Copying SSH key to 34.69.74.151
```

Figura D.3: Ejecución del comando Deploy.

Instancias de VM [CREAR INSTANCIA](#) [IMPORTAR VM](#) [ACTUALIZAR](#)

INSTANCIAS OBSERVABILIDAD PROGRAMAS DE LAS INSTANCIAS

Instancias de VM

Filtro Ingresar el nombre o el valor de la propiedad

☐	Estado	Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
☐	✓	instance1-us-central1-a-001	us-central1-a			10.128.0.16 (nic0)	34.68.67.249 (nic0)	SSH ▾ ⋮
☐	✓	instance1-us-central1-a-002	us-central1-a			10.128.0.17 (nic0)	34.171.66.180 (nic0)	SSH ▾ ⋮
☐	✓	instance1-us-central1-a-003	us-central1-a			10.128.0.18 (nic0)	34.27.162.143 (nic0)	SSH ▾ ⋮

Figura D.4: Instancias tras ejecución del comando Deploy.

```
angel@angel-HP-Pavilion-Gaming-Laptop-15-ec0xxx:~/Desktop/TFG/tool/tool$ go run mytool.go exec
Instance 34.69.74.151, External IP,
Instance 34.134.11.40, External IP,
█
```

Figura D.5: Ejecución del comando Exec.

Después se ejecuta el comando 'exec-proxy', figura D.9 ,el cual se encarga de desplegar la instancia que va a funcionar cómo proxy, de aprovisionarla con los ficheros necesarios y de enviar la orden al resto de instancias de que ejecuten su parte de la simulación. Cómo podemos observar en la figura D.8 se ven las 3 instancias 'worker' y la instancia debug que va a actuar cómo proxy.

EL contenido del bucket tras la ejecución del comando 'exec-proxy' es similar al del comando 'exec'. Y si ejecutamos el comando debug se obtendrá también el mismo resultado que en caso anterior. Finalmente ejecutamos el comando 'destroy-proxy' el cual elimina todos los recursos desplegados y los ficheros generados para la simulación. Finalmente se comprueba que los recursos se han eliminado satisfactoriamente. A su vez se pueden ver en la base de datos las filas que se han creado relacionadas con la simulación.

← Detalles del bucket

my-bucket-angel-tfg

Ubicación	Clase de almacenamiento	Acceso público	Protección
us (varias regiones en Estados Unidos)	Standard	Sujeto a LCA de objeto	Ninguno

OBJETOS CONFIGURACIÓN PERMISOS PROTECCIÓN CICLO DE VIDA OBSERVABILIDAD INFORMES DE INVENTARIO

Depósitos > my-bucket-angel-tfg > us-central1-a

SUBIR ARCHIVOS SUBIR CARPETA CREAR CARPETA TRANSFERIR LOS DATOS ADMINISTRAR CONSERVACIONES DESCARGAR BORRAR

Filtrar solo por prefijo de nombre Filtro Filtrar objetos y carpetas

☐	Nombre	Tamaño	Tipo	Fecha de creación	Clase de almacenamiento	Última modificación	Acceso
☐	ejemplo_simulador.coc	362 B	application/octet-stream	30 ago 2023 00:59:52	Standard	30 ago 2023 00:59:52	No público
☐	ejemplo_simulador.ir	5.7 KB	application/octet-stream	30 ago 2023 00:59:52	Standard	30 ago 2023 00:59:52	No público
☐	ejemplo_simulador.pi	246 B	application/octet-stream	30 ago 2023 00:59:53	Standard	30 ago 2023 00:59:53	No público
☐	ejemplo_simulador_i_0.lefo	630 B	application/octet-stream	30 ago 2023 00:59:54	Standard	30 ago 2023 00:59:54	No público
☐	ejemplo_simulador_i_0.lefr	372 B	application/octet-stream	30 ago 2023 00:59:55	Standard	30 ago 2023 00:59:55	No público
☐	ejemplo_simulador_i_1.lefo	743 B	application/octet-stream	30 ago 2023 00:59:55	Standard	30 ago 2023 00:59:55	No público
☐	ejemplo_simulador_i_1.lefr	363 B	application/octet-stream	30 ago 2023 00:59:56	Standard	30 ago 2023 00:59:56	No público
☐	ejemplo_simulador_i_2.lefo	624 B	application/octet-stream	30 ago 2023 00:59:57	Standard	30 ago 2023 00:59:57	No público
☐	ejemplo_simulador_i_2.lefr	366 B	application/octet-stream	30 ago 2023 00:59:58	Standard	30 ago 2023 00:59:58	No público
☐	ip_port.txt	51 B	text/plain	30 ago 2023 01:00:00	Standard	30 ago 2023 01:00:00	No público
☐	output.txt	41 B	text/plain	30 ago 2023 01:00:03	Standard	30 ago 2023 01:00:03	No público
☐	output_instance1-us-central1-a-001.txt	15.2 KB	text/plain	30 ago 2023 01:00:58	Standard	30 ago 2023 01:00:58	No público
☐	output_instance1-us-central1-a-003.txt	615 B	text/plain	30 ago 2023 01:00:58	Standard	30 ago 2023 01:00:58	No público

Figura D.6: Bucket tras ejecución del comando Exec.

← Detalles del bucket

my-bucket-angel-tfg

Ubicación	Clase de almacenamiento	Acceso público	Protección
us (varias regiones en Estados Unidos)	Standard	Sujeto a LCA de objeto	Ninguno

OBJETOS CONFIGURACIÓN PERMISOS PROTECCIÓN CICLO DE VIDA OBSERVABILIDAD INFORMES DE INVENTARIO

Depósitos > my-bucket-angel-tfg > debug > us-central1-a

SUBIR ARCHIVOS SUBIR CARPETA CREAR CARPETA TRANSFERIR LOS DATOS ADMINISTRAR CONSERVACIONES DESCARGAR BORRAR

Filtrar solo por prefijo de nombre Filtro Filtrar objetos y carpetas

☐	Nombre	Tamaño	Tipo	Fecha de creación	Clase de almacenamiento	Última modificación	Acceso público
☐	output_instance1-us-central1-a-001.txt	15.2 KB	text/plain	30 ago 2023 01:07:10	Standard	30 ago 2023 01:07:10	No público
☐	output_instance1-us-central1-a-002.txt	45.3 KB	text/plain	30 ago 2023 01:07:09	Standard	30 ago 2023 01:07:09	No público
☐	output_instance1-us-central1-a-003.txt	615 B	text/plain	30 ago 2023 01:07:10	Standard	30 ago 2023 01:07:10	No público

Figura D.7: Bucket tras ejecución del comando Debug.

Instancias de VM									
CREAR INSTANCIA IMPORTAR VM ACTUALIZAR									
INSTANCIAS OBSERVABILIDAD PROGRAMAS DE LAS INSTANCIAS									
Instancias de VM									
Filtro Ingresar el nombre o el valor de la propiedad									
☐	Estado	Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar	
☐	✓	debug-instance-us-central1-a	us-central1-a			10.128.0.31 (nic0)	34.134.11.40 (nic0)	SSH	⌵ ⋮
☐	✓	instance1-us-central1-a-001	us-central1-a			10.128.0.28 (nic0)	34.67.211.206 (nic0)	SSH	⌵ ⋮
☐	✓	instance1-us-central1-a-002	us-central1-a			10.128.0.29 (nic0)	34.68.67.249 (nic0)	SSH	⌵ ⋮
☐	✓	instance1-us-central1-a-003	us-central1-a			10.128.0.30 (nic0)	34.69.74.151 (nic0)	SSH	⌵ ⋮

Figura D.8: Instancias tras ejecución del comando Exec-proxy.

```
angel@angel-HP-Pavilion-Gaming-Laptop-15-ecbxxx:~/Desktop/TFG/tool/tool$ go run mytool.go exec-proxy
2023/08/31 01:01:41 Waiting for operation operation-1693436499339-6042be8fe2744-4226c8d9-ba896598 to complete...
2023/08/31 01:02:15 Operation operation-1693436499339-6042be8fe2744-4226c8d9-ba896598 completed successfully
Copying SSH key to 34.134.11.40
```

Figura D.9: Ejecución del comando Exec-proxy.

Anexo E

Base de datos

Tomando como referencia la base de datos utilizada en un TFG anterior, se han creado las siguientes tablas con el fin de gestionar las simulaciones ejecutadas por los usuarios, permitiendo almacenar datos tanto de la simulación como de las instancias desplegadas en la misma.

```
CREATE TABLE IF NOT EXISTS petrinet (  
    id serial PRIMARY KEY,  
    short_name VARCHAR(255) UNIQUE,  
    long_name VARCHAR(255),  
    xml TEXT,  
    image TEXT  
);
```

```
CREATE TABLE IF NOT EXISTS user_users (  
    uuid UUID PRIMARY KEY,  
    username VARCHAR(255),  
    password VARCHAR(255)  
);
```

```
CREATE TABLE IF NOT EXISTS compilation (  
    id serial PRIMARY KEY,  
    id_file VARCHAR(500),  
    petrinet_id INTEGER,  
    date_compilation TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    logs BYTEA,  
    logs_pretty TEXT,  
    time_to FLOAT,  
    CONSTRAINT fk_petrinet
```

```

        FOREIGN KEY(petrinet_id) REFERENCES petrinet(id)
        ON DELETE CASCADE
    );

CREATE TABLE IF NOT EXISTS petrinet_user (
    user_id UUID,
    petrinet_id INTEGER,
    CONSTRAINT fk_petrinet_user
        FOREIGN KEY(petrinet_id) REFERENCES petrinet(id)
        ON DELETE CASCADE,
    CONSTRAINT fk_user_petrinet
        FOREIGN KEY(user_id) REFERENCES user_users(uuid)
        ON DELETE CASCADE,
    PRIMARY KEY(user_id,petrinet_id)
);

CREATE TABLE IF NOT EXISTS component (
    id serial PRIMARY KEY,
    short_name TEXT,
    name TEXT,
    type_component TEXT,
    component_parent INTEGER,
    petrinet_id INTEGER,
    x INTEGER,
    y INTEGER,
    CONSTRAINT fk_petrinet_id_composite
        FOREIGN KEY(petrinet_id) REFERENCES petrinet(id)
        ON DELETE CASCADE,
    CONSTRAINT fk_parent_component_composite
        FOREIGN KEY(component_parent) REFERENCES component(id)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS place (
    id serial PRIMARY KEY,
    short_name TEXT,
    name TEXT,
    parent INTEGER,

```

```

marks INTEGER,
x INTEGER,
y INTEGER,
isport VARCHAR(50),
CONSTRAINT fk_parent_place
    FOREIGN KEY(parent) REFERENCES component(id)
    ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS transition (
    id serial PRIMARY KEY,
    short_name TEXT,
    name TEXT,
    parent INTEGER,
    time INTEGER,
    x INTEGER,
    y INTEGER,
    isport VARCHAR(50),
    CONSTRAINT fk_parent_transition
    FOREIGN KEY(parent) REFERENCES component(id)
    ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS inputport (
    id serial PRIMARY KEY,
    name TEXT,
    parent INTEGER,
    CONSTRAINT fk_places_inputport
    FOREIGN KEY(parent) REFERENCES component(id)
    ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS outputport (
    id serial PRIMARY KEY,
    name TEXT,
    parent INTEGER,
    CONSTRAINT fk_places_outputport
    FOREIGN KEY(parent) REFERENCES component(id)
    ON DELETE CASCADE
);

```

```

CREATE TABLE IF NOT EXISTS inoutport (
    id serial PRIMARY KEY,
    name TEXT,
    parent INTEGER,
    CONSTRAINT fk_places_inoutport
        FOREIGN KEY(parent) REFERENCES component(id)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS links (
    id serial PRIMARY KEY,
    parent INTEGER,
    source INTEGER,
    target INTEGER,
    type_link TEXT,
    weight INTEGER,
    parentsource INTEGER,
    parenttarget INTEGER,
    CONSTRAINT fk_link_petri
        FOREIGN KEY(parent) REFERENCES component(id)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS simulation (
    id serial PRIMARY KEY,
    petrinet_id INTEGER,
    date_simulation TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    logs TEXT,
    time_to FLOAT,
    CONSTRAINT fk_petrinet
        FOREIGN KEY(petrinet_id) REFERENCES petrinet(id)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS keypubs (
    id serial PRIMARY KEY,
    user_id UUID,

```

```

    keypub TEXT,
    CONSTRAINT fk_user_pub
        FOREIGN KEY(user_id) REFERENCES user_users(uuid)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS deploy (
    id serial PRIMARY KEY,
    simulation_id INTEGER,
    zone VARCHAR(255),
    project VARCHAR(255),
    machine_type VARCHAR(255),
    vm_instances INTEGER,
    CONSTRAINT fk_simulation
        FOREIGN KEY(simulation_id) REFERENCES simulation(id)
        ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS instance (
    id INTEGER PRIMARY KEY,
    external_ip VARCHAR(255),
    internal_ip VARCHAR(255),
    deploy_id INTEGER,
    logs_file VARCHAR(500),
    CONSTRAINT fk_deploy
        FOREIGN KEY(deploy_id) REFERENCES deploy(id)
        ON DELETE CASCADE
);

```

La últimas dos tablas almacenan los datos mencionados.

Anexo F

Líneas de código

A continuación se detalla el número de líneas de código de los diferentes lenguajes de programación utilizados en el TFG:

- Líneas en Golang: 1754
- Líneas en SQL: 176
- Líneas en script bash: 50
- Líneas en manifest de terraform: 80