Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

SwarmBot: Enjambres de agentes artificiales para el testeo de interacciones en prototipos de redes sociales

Alumno: Jorge Martinez De La Chica

Tutor: Jose María Serrano Chica

Dpto: Informática

UNIVERSIDAD DE JAÉN

D./Dª Jose María Serrano Chica, tutor(es) del Trabajo Fin de Grado titulado: **Swarm-Bot: Enjambres de agentes artificiales para el testeo de interacciones en prototipos de redes sociales**, que presenta Jorge Martinez De La Chica, autoriza(n) su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2021

El estudiante

El tutor

SERRANO CHICA JOSE MARIA - 26490278M

Firmado digitalmente por SERRANO CHICA JOSE MARIA - 26490278M
Fecha: 2021.09.09 14:10:02 +02'00'

Jorge Martinez De La Chica

Jose María Serrano Chica

Agradecimientos

Para este Trabajo de Fin de Grado tengo que agradecer a unas cuantas personas por su apoyo. En primer lugar a mi tutor, por su compresión con la situación que hemos pasado debido a la COVID-19, que aún en vacaciones ha estado atento a mis necesidades y me ha aportado muchas ideas de cara al desarrollo de este trabajo.

También a un amigo en especial, estudiante en la UGR, Andrés Millán, ya que me ha inspirado no solo en la toma de soluciones de este trabajo si no en el enfoque que se expondrá en él. Y por último pero no menos importante a mi familia y pareja que me han animado a finalizar este trabajo a pesar de la situación tanto laboral como personal.

Tabla de contenidos

Lista de figuras

Lista de algoritmos

Capítulo 1

INTRODUCCIÓN

Este Trabajo de Fin de Grado aparece como respuesta a una necesidad que aparece comúnmente en ciertos sectores, a día de hoy existen múltiples redes sociales, y se crean nuevas continuamente, aunque pocas son las que triunfan. A la hora del despliegue de los servidores para estos servicios aparece una incógnita importante "¿Como debo dimensionar mi servidor?", nosotros podemos calcular en base a predicciones que hardware usar para alojar nuestro sistema, pero hasta el día de apertura no tenemos claro que esos cálculos se transladen a la realidad de forma correcta, es más, no podemos saber si el diseño de nuestra plataforma será capaz de soportar esa carga.

El objetivo por tanto es solucionar ese problema, crear una herramienta con la que estos proyectos tengan posibilidad de hacer test de carga que de verdad demuestren que lo desarrollado es robusto y a su vez dar una idea del hardware necesario para desplegarla. Por esto su desarrollo se ha hecho con la idea en mente de ser flexible en cuanto a la red social a aplicarse, y para ello se han implementado métodos que generalicen tanto la conexión como la comunicación con el servidor, de forma que sea versátil al uso de cada usuario.

Esta herramienta se basará en la simulación de conexión de usuarios mediante el uso de bots, este término junto a todo el contexto necesario para entender todo lo que rodea a este trabajo será introducido a continuación

1.1. ¿Qué son las Redes Sociales?

Previamente al desarrollo de esta memoria es necesario ser consciente de en qué contexto trabajamos. Como se indica en los objetivos la idea es medir como un servidor usado para alojar una red social soporta una carga "real", pero ¿que es una red social?

Podemos definir una red social según se indica en Lopez-Cuevas A. [2021]:

> *A social network (SN) is a social structure made up of individuals or organizations, which are connected by one or more specific types of interdependency, such as friendship, kinship, common interest, likes/dislikes, or relationships of beliefs, knowledge or prestige.*

Es decir, podemos considerar una red social como la interconexión entre iguales, provocando interacciones y un intercambio de información. Ésta información que se genera es el verdadero punto de interés para una empresa, puesto que con ella se puede obtener el perfil de usuario de casi cualquier individuo.

Sin embargo esto nos queda escueto para definir que es una red social y que no, porque al final el contacto físico que podemos tener también se podría integrar en esta especificación. Por tanto ahora es necesario dar unas características mínimas que nos permitan diferenciar, tal y como se especifica en Masrom et al. [2021] existen múltiples definiciones según cada autor, pero todos comparten enfoques comunes:

1. La comunicación se establece mediante un medio digital, para ser una red social por tanto debe establecerse la interacción integramente a través de un medio no físico.

2. Debe ser posible establecer vínculos entre usuarios, ya sea como seguidores o como se quiera definir.

3. Debe ser posible navegar a través de los vínculos establecidos entre otros usuarios, es decir, ver los vínculos que establece cada usuario

Con esto en mente ya si somos capaces de identificar que definimos hoy día como red social, pero ¿este concepto ha sido siempre igual?

1.2. Un poco de historia

Simeon Edosomwan [2011]

A pesar de que hoy día cuando escuchamos las palabras “red social” y pensamos en una web o plataforma digital hace no tanto esto no era así.

Para entender la historia de las redes sociales debemos recordar que son, en la definición inicial dada no menciona el medio por el que se forman las relaciones, esto es porque la estructura social en la que se basan las redes sociales podemos llegar a verla incluso en la época de los telégrafos, la década de 1790.

Lógicamente no era lo mismo mandar un pequeño mensaje por telégrafo que poner un post en *Twitter* hoy día, sin embargo ya empezaba a crearse ese concepto de comunicación entre personas. Se cree que los grupos que formaban esta estructura se crean debido a que compartían valores o situaciones similares, una base que se conserva y lo que justifica que se considere este el origen.

Avanzando en el tiempo encontramos que la siguiente evolución o salto sigue sin ser al medio digital, en este caso aparece el teléfono, aunque este era de pago y poca gente tenía liquidez para hacer un uso extenso de él en la década de 1950 aparece el concepto de *phone phreaking*, esto es una técnica que empleaba un sistema (*Blue Box* 1.1) para engañar a los sistemas de la época y conseguir acceso a llamadas de forma gratuita. Este método en su base consiste en crear una frecuencia similar al que hacían las teclas de los teléfonos para hacerles creer que se les había marcado un número, para lo cual debías pagar antes.

Lo importante es que con estos marcadores los teléfonos se vuelven un medio muy bueno para establecer las primeras redes sociales a gran escala, ya que no se depende de telegrafiar y se tiene un acceso más o menos sencillo. Además se llegó a hacer uso de algunos contestadores de empresas para crear los primeros *podcast*.

Tras la aparición del ordenador personal aparecieron las primeras redes entre individuos, esto aún lejos de lo que hoy conocemos como internet, en aquel momento eran redes aisladas entre unos pocos con acceso limitado. Esta situación da lugar a las primeras redes sociales digitales, la aparición de los BBS (*Bulletin Board System*) comienzan esto alrededor de 1980, consistían principalmente en tablones de anuncios creados por la comunidad y que servían como punto de encuentro entre los usuarios. Más tarde el concepto fue evolucionanado y paso a ser lo que se conoce como *Newsgroups* conforme el concepto de conexión evoluciona y se amplia la red de difusión.

Figura 1.1: Blue box usada para *phone phreaking*

En 1988 también aparece lo que se conoce como IRC (*Internet Relay Chat*) como sustitución a un BBS que se centraba en establecer comunicaciones entre usuarios concretos, este nuevo sistema crea una conexión única entre dos usuarios haciendo uso de *USENET* (al igual que los newsgroups) e incluso a día de hoy sigue usandose.

Y para concluir con este pequeño repaso a la historia de redes sociales es importante pasar por el correo electrónico, pues es la transición "final.ªntes de llegar al concepto actual. Aún en su forma primigenia en 1991 cuando internet fue de uso público y en la que para enviar un correo ambos PC deben estar encendidos y conectados a la red para poder establecer una comunicación, hablamos de que ésta es accesible al público general sin necesidad de los conocimientos que requerían los marcadores caseros, por no hablar de los riesgos que suponia su uso.

1.3. Redes Sociales actualmente

Después de todo este tiempo aquí estamos, en una sociedad donde el uso de redes sociales es casi trivial y sin darnos apenas cuenta estamos conectados a 3 o 4 a la vez. En la actualidad los niños de 10 años ya usan *WhatssApp* y algunos ya tienen su cuenta de *Facebook* pero ¿realmente somos capaces de distinguir que estamos en una red social?

Con toda la evolución la estructura social de la que se habla al principio puede en-

contrarse de muchas formas aunque aquí voy a distinguir entre dos grupos: **síncronas** y **asíncronas**.

Consideraremos una red síncrona a aquella en la que se espera una comunicación inmediata por parte de los que establecen la conexión, aquí entrarían las videoconferencias y también incluiré la mensajería instantánea debido a que el objetivo de ésta es habilitar el intercambio de información fluido y rápido entre los individuos que conversan, sin embargo si nos ceñimos a las características dadas antes podemos ver que a estas redes les falta un punto, ser capaces de visualizar los vínculos de otros usuarios, esto ocurre debido a que aquí se prima una comunicación directa y no un enfoque más tradicional. Ejemplos de este tipo tenemos y usamos muchos:

- Videoconferencias: *Discord, Google Meet, Skype, Zoom, etc*; Todas ellas cumplen lo que especificamos, un medio digital mediante el que personas comparten información, aunque los vínculos que se creen sean meramente temporales.
- Mensajería instantánea: *WhatssApp, Telegram, Line, Facebook Messenger, etc*; Aquí ya nos quedamos tan solo con ese enfoque de compartir información, ya que no se establece vínculo ninguno (no te añades a una lista, solo envías a un número o alias)

Por otro lado las redes asíncronas son aquellas que aún con la capacidad de comunicarse al momento están pensadas para una interacción a largo plazo, aquí entrarían las redes sociales más comunes y en las que primero pensamos cuando escuchamos las palabras así como pueden incluirse blogs, correos electrónicos, foros, etc. Aquí si se cumple en su totalidad estas características que se enumeraban anteriormente. Ejemplos de estas hay por todas partes, algunos son:

- *Twitter*
- *Facebook*
- *Youtube*
- *Instagram*
- *TikTok*
- *Snapchat*
- *Google Blogs*
- *Reddit*
- *Outlook*

1.4. ¿Qué son los Bots?

En los objetivos se menciona el uso de bots, sería necesario especificar qué entendemos por bots para así tener una imagen correcta del posterior desarrollo del trabajo. Ferrara et al. [2016] Para empezar, usamos *bot* como abreviatura de los que sería decir software robot, continuando con su definición tenemos que ver que los convierte en lo que son, qué características.

Bots hay de muchos tipos y con capacidades diferentes y metas completamente opuestas en algunos casos, podemos ver algunos que ayudan en el aprendizaje de lenguas extranjeras, bots para redactar artículos periodísticos, etc; sin embargo lo que nos interesa en el ambito que trata este documento es de un tipo específico de bots, **los social bots**.

Estos presentan comportamientos muy similares a los humanos desde un punto de vista externo, la idea es que hablen como personas y actúen como personas, para de esta forma crear individuos ficticios con metas dispares. Desarrollar un bot de este tipo no es trivial, aunque su existencia si fue "predicha" por Alan Turing en la década de 1950, donde expresó que llegaría el día en el que las máquinas serían capaces de engañar al ser humano y hacerse pasar por uno, así nació el **Test de Turing** que sería el "muro" que todos los predecesores de los bots querrían superar.

En cuanto a los objetivos con los que estos social bots aparecen como se ha mencionado anteriormente son dispares, tenemos bots que desempeñan tareas sociales importantes como es el caso que se menciona Geiger [2018], que a pesar de ser comprometido el bot en concreto descrito realiza la tarea de revisar las publicaciones de wikipedia, otros como el que se plantea en este trabajo tienen como meta usar esas cualidades de simulación de individuos para realizar tareas como comprobar cargas a una red social, otros con cualidades similares pueden tener enfoques más cuestionables desde un punto de vista ético, lo cual nos lleva al siguiente punto, ¿qué efecto tienen estos bots sobre nosotros?

1.5. La amenaza de los social bots

Un social bot hoy día puede llegar a ser una amenaza enorme debido a un fenómeno llamado la *espiral del silencio* Ross et al. [2019]. Este efecto se define comúnmente como presión social, y consiste en hacer creer a la persona que su opinión es minoritaria, ante esto la persona se vuelve recelosa de dar su punto de vista, provo-

cando que poco a poco se mimetice con esa opinión general que percibe.

El trabajo por tanto de los social bot que buscan influenciar a la gente a favor de algo es simplemente hacer parecer que son una mayoría de personas reales, de esta forma cualquier usuario que los lea creerá que realmente existe esa mayoría. Si partimos de la base de que los bots ya son capaces de simular conductas humanas a través de redes sociales encontramos que es peligrosamente fácil ejecutar manipulaciones partiendo de esta base, y no es por falta de ejemplos tampoco.

Siguiendo la pista de una *colmena* de social bots ya identificada podemos remitirnos a las elecciones de EEUU en 2016, donde el candidato Donald Trump es elegido presidente, durante estas elecciones esta probado que hubo bots apoyando a ambos candidatos, sin embargo la cantidad que había hacia el ganador eran muchos más. Lo amenazante de esto es que unas elecciones fueron ganadas por aquel que más bots tenía de su lado, como si ya no fuese gente la que eligiese a su lider. Shane [2017]

Este mismo grupo también fue usado (o parte) en las elecciones francesas de 2017, donde actuaron contra el actual presidente Emmanuel Macron, en esta ocasión presentando datos falsos que cuestionaban la legitimidad de ciertos actos del político. Finalmente como sabemos esto fue desmentido y al final salio reelegido, sin embargo lleva a cuestionarnos algo, realmente es posible que algo falso pase a ser cierto tan solo porque una mayoría lo valida, y lo que es peor, una falsa mayoría. Redondo [2018]

Como estos ejemplos existen muchos más, que no hacen otra cosa que ilustrar como nuestra sociedad es guiada en base a las masas, somos lo que vemos, y muchos usuarios con conocimientos suficientes aprovechan este hecho para llevarnos a lo a ellos les favorece. Pero no todos los bots son así, internet no esta lleno de *hackers* esperando a que tecleemos nuestra clave del banco para robarnos, existen bots con intenciones no tan malévolas, como es el ejemplo de bots en *Twitter* destinados simplemente a descargar vídeos que se publican en la red.

1.6. La utilidad de los bots

En esta primera parte se ha visto el potencial de los bots, que son capaces de hacer y que es lo primero que una persona piensa cuando se le habla de bots, sin embargo esto no es todo lo que pueden hacer. Los bots ni nacen ni exclusivamente se usan para el mal, los social bots no son distintos, existen múltiples ejemplos sobre desarrollos que han llevado a resultados prometedores con el uso de estos sistemas, como es el caso mencionado anteriormente en Geiger [2018].

Y la realidad es que hoy en día convivimos con muchos bots que nos facilitan la vida, por ejemplo lo son aquellos destinados a enviar ofertas que pueden interesarnos, o bots tan útiles como puede haber en muchas páginas de soporte para solucionar dudas y cuestiones a los clientes. Por mucho miedo que puedan darnos bien usados son un poderoso mecanismo no solo de apoyo para tareas más automáticas como casi todo programa informático, si no como medio de difusión de información.

Ante todo esto el objetivo de este trabajo no es otro más que usar estos bots como apoyo para la simulación, son capaces de imitar a personas y actuar de forma parecida, pues lo que buscamos es sintetizar esa capacidad para crear el entorno perfecto de pruebas para una futura red donde habrá tanto bots como usuarios.

1.7. Objetivos

Los objetivos se visualizan un poco con la idea del capítulo, sin embargo es preciso ser concretos ya que lo que se pretende puede abarcar mucho más de lo que puede contener un trabajo de esta embergadura.

Como objetivo principal se busca desarrollar una herramienta **flexible**, **escalable** y **completa** para simular una **conexión de múltiples usuarios** de forma *natural* a una red social, con la idea de ver si el hardware es el adecuado y si la red en si es capaz de aguantar ese número de conexiones concurrentes.

En esta simulación la prioridad será crear una **carga realista**, no tan solo conexiones sin sentido, si no simular la **interacción de los usuarios** de forma **creíble** y como se podría dar entre seres humanos. Con esto en mente se desarrollará siguiendo un modelo de *eventos* que se explicará más adelante.

Previo a esta simulación también se busca la creación de un espectro de usuarios ficticios que se comporten de formas diversas como lo harían usuarios reales, unos de forma menos activa, otros con más poder de influencia, etc; Con la idea de que esta simulación sea aún más realista.

Como objetivo final pero no menos importante, esta herramienta debe ser capaz de generar la información necesaria para analizar el comportamiento del servidor durante la simulación y ser capaz de detectar problemas que hayan podido ocurrir durante esta (cuellos de botella por ejemplo).

1.8. Planificación temporal

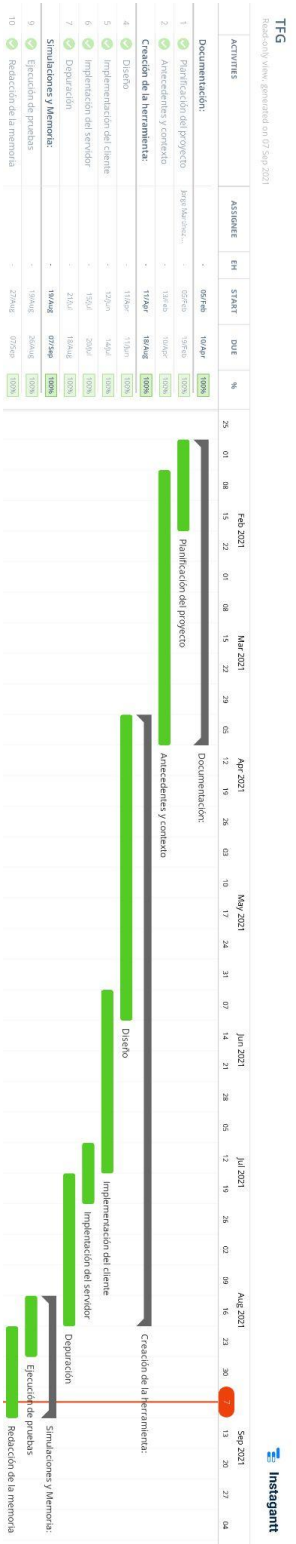

Figura 1.2: Diagrama de Gant del trabajo

Como se puede ver esta es la planificación que he seguido a lo largo de la realización del proyecto, este diagrama (1.2) ha sido generado tras completar las distintas fases del mismo ya que como se puede apreciar ciertos tiempos son más extensos que otros, esto se debe a que como es lógico avanzaba según la carga de trabajo me lo permitía.

Pasando a un análisis del diagrama se puede ver que se le ha dedicado mucho tiempo principalmente a la documentación, unos tres meses, en este tiempo estuve no solo con los antecedentes vistos en esta introducción si no también observando el comportamiento de usuarios y analizando como simularlo, el resultado de lo obtenido se verá más adelante.

El proceso de creación ha sido como es de esperar el grueso de tiempo empleado, esto se debe principalmente a que el diseño ha sido algo complejo de pensar ya que llegué a una idea que avancé mucho pero cuando la tenía casi cerrada vi que era inviable sin un hardware extremadamente exagerado y daría lugar a algo desproporcionado para los objetivos del trabajo. Fue entonces cuando volví atrás y llegué al enfoque finalmente desarrollado, mucho más eficiente y práctico.

Por último se puede ver que el último tiempo ha sido empleado en la elaboración de la memoria, esto es relativo, ahí a lo que hago referencia es la creación del documento formal siguiendo las directrices y un formato adecuado, usando LaTeX, sin embargo la redacción base de todo lo que se contiene en este documento ha sido redactado a lo largo de todo el proceso en un formato mucho más informal, de forma que no se perdieran detalles de ningún proceso.

1.9. Organización del documento

A continuación se podrá ver el proceso incremental seguido para el desarrollo de la herramienta descrita, para ello se seguirán unos pasos de cara a orientar al lector y hacer comprensible el enfoque sin perder el conjunto de todo y a su vez dar a conocer la metodología empleada:

1. Descripción de la estructura: Aquí se hablará del enfoque tomado a la hora de desarrollar toda la herramienta, y las partes que la componen

2. Estudio realizado: Que se ha extraído de la observación de algunas redes sociales convencionales y como se translada eso al desarrollo

3. Síntesis de la información: Como se ha abstraído la información recabada y como se representarán los términos.
4. Elección de tecnologías: Que se usará a lo largo del desarrollo y el porque de estas decisiones
5. Detalles de implementación: Como se han programado los detalles de cada parte del programa y que algorimos se han seguido
6. Proceso de uso: Explicación del entorno de pruebas y como se ha preparado para que sea replicable

Además de todo esto en la parte final se realiza un estudio de como es la carga creada por la aplicación final así como una valoración y puesta en escena de ideas de cara a futuros usos.

Capítulo 2

MATERIALES Y MÉTODOS

En este capítulo desarrollare todo el contexto, material e ideas usadas para la implementación de la herramienta antes descrita, referida como *SocialBench*.

Antes de comenzar con el diseño he de recalcar que en la etapa anterior se investigó la existencia de herramientas similares, aunque no apareció nada similar. Si que existen múltiples herramientas para comprobar el hardware del servidor ante cargas similares a las que puede sufrir con el despliegue de una red social, sin embargo ninguna tiene en cuenta un factor decisivo de esto, y es que esa red social puede tener particularidades a la hora de funcionar, tal vez un uso de memoria excesivo bajo ciertas cargas u optimizaciones para disminuir el impacto respecto a lo previsto, todo esto da lugar a una estimación que puede ser errónea lo que origina problemas en el dimensionamiento, ya sea por exceso o por falta. Es por esto que esta herramienta gana importancia ya que comprueba todos los aspectos de la red social en su conjunto además de ser única en su campo.

2.1. Diseño de la herramienta

2.1.1. Descripción de la estructura

La estructura que regirá *SocialBench* será la de cliente - servidor, esto se debe a que vamos a generar una carga para el servidor pero necesitamos poder medirla afectando lo menos posible a este, a su vez así conseguimos simular incluso las conexiones de los usuarios desde un equipo externo. Es en el cliente donde se ubicará el grueso del desarrollo, en él se llevará a cabo la **generación de usuarios**, la **ejecución**

de la simulación y el **guardado de los datos en tablas** para su posterior análisis. Y será en el servidor donde se medirá cada determinado tiempo el uso de recursos para detectar posibles problemas.

Sin embargo antes de pasar a detalles de la implementación es importante entender el enfoque seguido en todo el programa.

2.1.2. Estudio de la interacción en redes sociales

Previo a la realización de este trabajo se ha desarrollado un periodo de estudio del comportamiento en otras redes sociales. Lo que se ha podido observar es que existen dos conductas generales:

1. **Interacciones rutinarias**: Generan carga superficial y suelen ocasionarse por bots automáticos o usuarios con influencia que buscan ser activos para sus seguidores
2. **Eventos relevantes**: momentos clave en los que se abre debate u ocurre algo en el mundo que insta a la gente a dar su opinión sobre el tema.

Lógicamente para los objetivos de esta herramienta lo relevante son estos eventos, ya que buscamos cargar el servidor por lo que se deshecha esas interacciones rutinarias debido a su bajo impacto [1].

Una vez sabemos que genera interacciones es también interesante conocer como reaccionan los usuarios a estos sucesos, ante esto se han podido dividir los usuarios en dos grupos diferenciados:

1. **Usuarios activos**: generan *posts* ya sea individuales o contestando a otros usuarios.
2. **Usuarios pasivos**: observan sin más el transcurso del evento en la red y a veces dan difusión a algún comentario con el que se sienten afín, pero sin generar contenido propio.

Con esto presente puede aparecer la idea plasmada en *SocialBench*

[1] Se considera que si esa interacción genera debate es un evento en sí

2.1.3. Síntesis del cliente

En este apartado se describirá todos los sistemas usados para la abstracción de los comportamientos observados en la etapa anterior para dar lugar a una simulación realista del comportamiento.

Sistema de etiquetas

En primer lugar se ha elegido clasificar los eventos, esto se debe a que según de que trate cada evento un determinado grupo de usuarios interactuará en él o no. Para ello se ha llegado a un sistema jerárquico que permite no solo clasificar de forma "precisaçada evento si no que también nos da flexibilidad de cara a añadir más categorías en un futuro sin afectar al resto del programa.

En este sistema se consideran cuatro etiquetas o categorías centrales: Política, Ciencias, Cultura y Ocio. La idea de estas es que engloben a categorías más concretas y son el eje central del sistema, es la parte inmutable de él.

Dentro de estas categorías he introducido múltiples especificaciones con motivos de demostración aunque como se indica esto si es flexible y pueden agregrarse y eliminarse con total libertad. *Véase figura 2.1*

2.1.4. Perfil de usuarios

Cada usuario como se ha estudiado tendrá un perfil distinto a otro, ya sea por gustos o por orientaciones propias, con la idea de plasmar esto se ha implementado una estructura de usuario que permita definirlo y que interactúe respecto a esas características.

Los puntos clave de los comportamientos del usuario vienen definidos principalmente por tres datos: su **actividad**, definida como activa o pasiva; sus **gustos** y sus **seguidores**, a quién sigue cada usuario.

El hecho de seleccionar estos datos como los decisivos se debe principalmente a que hoy día factores como la edad o el sexo pueden influenciar en como se expresa una persona, pero no necesariamente en como se comporta en una red social, tenemos ejemplos de periodistas soltando bulos en redes sociales o de niños de doce años dando lecciones de vida como si tuviesen cuarenta, por este motivo me restrinjo

Figura 2.1: Esquema de etiquetas

a factores que si que indican mejor cuando y como interactuará cada usuario.

Eventos como elemento de acción

Según lo plasmado los eventos serán los que desencadenantes de la acción durante la simulación, estos eventos aparecerán en cierto momento con unas etiquetas determinadas y los usuarios afines a ellos comenzarán a interactuar, desencadenando que se comience a visualizar el evento y los seguidores de cada usuario original tengan la posibilidad de interactuar también.

2.2. Implementación de la herramienta

Con esta base se pasa a la implementación de la herramienta, y a pesar de la síntesis ya realizada se llegará a un nuevo punto de abstracción. Una de los puntos claves de esta será la generación de aleatorios, ya que se necesita determinar múltiples datos que no podemos prever, y otro es el control de los hilos, debido a que para generar muchas acciones concurrentes es necesario tener en cuenta los hilos de ejecución disponibles para no sobrecargar el equipo que la lanza.

2.2.1. Tecnologías usadas

Antes de comenzar con detalles dentro de las estructuras de datos y algoritmos involucrados dentro del desarrollo es preciso conocer que tecnologías serán usadas, ya que condicionarán todo el proceso.

En primer lugar el lenguaje escogido para el servidor es *Python*, esto se debe a que se requiere un script simple y ligero que monitorice los recursos y *Python* ofrece librerías y una rapidez de desarrollo que no podemos conseguir con otros lenguajes compilados, y unas posibilidades a las que no alcanzan otras opciones interpretadas.

Para el cliente sin embargo es necesario una opción que nos de más control sin afectar al peso de la aplicación, ya que la simulación en si puede llegar a ser muy lenta si no se optimiza bien. Por todo esto el lenguaje escogido es *C++*, los motivos además de cumplir lo ya expuesto son la contención de la solución, al compilarlo las librerías y todo lo usado queda en un paquete, y tener conocimientos previos de este lenguaje desde un punto de vista personal.

En el propio cliente aparecerán también tecnologías como es el *multi-threading*, ya antes comentado; la generación de archivos para guardar los datos recogidos y logs que ayuden a los análisis posteriores; intrepretación de archivos estructurados json, usados para el intercambio de información y por último el protocolo Socket como canal de comunicación entre cliente y servidor.

Por último, con motivos de test y demostración, se requiere una red social sobre la que lanzar estos bots, el problema que se presenta es las cuestiones éticas introducidas al comienzo del documento así como los problemas legales que podría acarrear debido a las prohibiciones de la mayoría de redes respecto a bots. Además con la estructura ya comentada se requiere un despliegue en local, o al menos tener acceso al servidor, cosa que por ejemplo *Twitter* no permitiría. Por esto se ha elegido como

red de pruebas *Mastodon*, una red social basada en Nodos, de código abierto, que permite bots en sus bases legales, y autoriza a cualquier usuario a crear su propio nodo. Con esta solución es posible tener un entorno sobre el que trabajar cómodamente gracias a su API y sobre el que realizar todas las pruebas Mas.

El entorno de desarrollo y posterior despliegue será el sistema *Ubuntu*, tanto en cliente como en servidor. Esto se debe a que en el lado servidor *Mastodon* esta creado para correr en este sistema y sus herramientas por tanto están adaptadas para él. Por otro lado el cliente tiene la limitación también de este entorno debido a que la única librería presente para el lenguaje escogido se encuentra enlazada a él por dependencias, esta librería es necesaria debido a que simplifica el desarrollo bastante al abstraer las comunicaciones de más bajo nivel con la API. A pesar de esto y gracias a la aparición en estos años de WSL *Windows Subsystem for Linux* podemos ser capaces de usar la herramienta en una variedad de equipos amplia.

2.2.2. Estructuras datos y algoritmos

Usuarios como conjunto

Los usuarios como se comenta en el análisis son un elemento fundamental del desarrollo de la acción, principalmente se basa en los tres factores descritos, aunque a la hora de la implementación se han decidido establecer los parámetros de edad y género de cara a una futura modificación del algoritmo de generación que se explicará mas adelante.

Además de estos parámetros, para poder desarrollar la actividad los usuarios contaran con su correo y contraseña para poder iniciar sesión, así como la clave de acceso dada por la API para poder enviar mensajes al servidor.

La generación de usuarios puede ser uno de los procesos más complejos según se enfoque, en este caso he optado por una alternativa intermedia, es decir se dará capacidad al usuario de la aplicación de influir en los parámetros de generación pero no se dejará establecer una generación manual. La idea con esto es crear una variedad de usuarios balanceada según se quiera dirigida a conjuntos grandes de usuarios, es por esto que se desestima darle al usuario darle potestad para generar casos individuales ya que el impacto finalmente lo tendrá el conjunto.

Este algoritmo toma como entrada la configuración y los datos de inicio de sesión para darnos cada usuario. Lo más destacable antes de entrar en detalle es que todas

las distribuciones usadas para los aleatorios son uniformes, esto se debe principalmente a que en el generador no se busca una exactitud completa actualmente, si no dotar de datos a cada usuario de forma que se creen perfiles uniformes, es decir, como se ha indicado las pequeñas individualidades no afectan en la simulación que se quiere llevar a cabo, lo que importa es que hace, no como lo hace.

Algoritmo 1: Generación de usuarios

Data: Archivo de configuración
Data: Archivo de datos de Inicio de sesión
Result: Usuario
Género $\leftarrow$ Aleatorio con distribución uniforme;
Edad $\leftarrow$ Aleatorio con distribución uniforme en el rango de 11 a 70;
Credenciales $\leftarrow$ Datos inicio de sesión;
Importancia $\leftarrow$ Aleatorio uniforme entre 0 y 1;
if *Importancia* $<$ *Umbral Importancia* **then**
  Se le resta $0{,}2$
else
  Se mantiene igual
Número de seguidores $\leftarrow$ Importancia $*$ Total Usuarios;
while *Falten seguidores* **do**
  Seguidor $\leftarrow$ Aleatorio del total
Perfil $\leftarrow$ Aleatorio con probabilidad en configuración;
Número de gustos $\leftarrow$ Etiquetas medias en configuración;
if *Aleatorio entre* $0{,}7$ *y* $0{,}3$ **then**
  Se mantiene;
else
  Se reduce o aumenta según aleatorio uniforme entre etiquetas totales;
Genera etiquetas(Número de gustos);

Dejando a un lado estos aspectos generales aparece el factor de importancia, perfil, seguidores y gustos. El primero se ha generado de forma ponderada, esto se debe a que en toda red social existen personas con mucha influencia, y entre estas y el resto existe un escalón de diferencia, lo que se intenta es que el máximo de un individuo medio sea de $0,5$ como tope, y en caso de ser una persona influyente se dispare a $0,7$ o más, por tanto se le asigna un valor de $0,7$ al umbral de importancia. Este primer factor es importante puesto que los seguidores se basan directamente en él, en este caso se intenta crear una red fuertemente enlazada, que es el "peorçaso de simulación que se puede dar, ya que una acción puede desencadenar muchísimas reacciones.

Por otro lado tenemos el perfil y número de gustos, en el caso del perfil tomamos

un aleatorio y basamos las probabilidades según se indique en el archivo de configuración, esto esta hecho para los casos en los que la carga varía según si los usuarios son pasivos o activos. En cuanto a los gustos en este caso vemos que llamamos a otra función, esto se debe a que se ha separado su comportamiento de cara a generar una lista distribuida y con fundamento para cada usuario, ya que determinará cuando puede actuar de forma directa.

Esta función toma como mínimo dos gustos por usuario y un máximo de once debido a que es el tope de gustos posibles según el diseño. La probabilidad de cada etiqueta dentro de su grupo esta distribuida de forma uniforme sin embargo se contempla que sea distinto y pueda ajustarse.

Primero se genera una probabilidad de que el usuario sea activo en política, esto se establece como una probabilidad uniforme entre las dos alternativas. En caso de ser positivo se asigna la orientación y se resta un gusto. A continuación se pasan a generar el resto de gustos, restringiendo el número de categorías según cuantos gustos resten al usuario.

Algoritmo 2: Generación de gustos

```
Data: Numero de gustos a generar
Result: Lista de gustos
Activo en política ← Aleatorio uniforme;
if Activo en política then
    Etiqueta 1 ← Orientación aleatoria;
    Número de gustos − 1;
Probabilidad de etiquetas por grupo ← Aleatorios según mínimos;
Orden etiquetas ← Barajado aleatorio;
while Número de gustos ≠ 0 do
    if Etiquetas restantes del grupo then
        Etiqueta i ← Aleatoria del grupo;
    else
        Avanzo de grupo;
```

El modo para evitar que siempre se tengan etiquetas de los mismos grupos es un sistema de barajado junto a probabilidades acumuladas. Esta metodología se ha desarrollado de forma que se asigna un porcentaje de etiquetas por grupo y posteriormente se decide de forma aleatoria el orden de los grupos, así se genera diversidad.

Control de eventos

En una red social los *eventos* aparecen y desaparecen en función de la actualidad en la mayoría de los casos, o en acciones de usuarios determinados. Sin embargo para la simulación he decidido crear la figura del **gestor de eventos**, un elemento de control que hace de actualidad así como limitador para evitar que la creación de eventos sea desmesurada y se adapte según el equipo donde se ejecute el cliente.

La idea de este gestor es cada determinado tiempo lanzar una probabilidad de crear un evento, en caso positivo se pasa a agregar a la ejecución un hilo por evento, de forma que se ejecute concurrente al resto de eventos activos en la red. Esta generación asigna los valores necesarios para el evento, la duración, etiquetas implicadas y usuarios iniciales.

Por otro lado cuando se sobrepasan los eventos máximos y se va a crear un nuevo evento se lanza un aleatorio, si este cumple los requisitos se elimina de la ejecución el evento más antiguo y se crea uno nuevo, en caso negativo se desestima el intento de crear el nuevo evento y se pasa a la siguiente iteración. Esta mecánica se introduce como he indicado antes para desempeñar la función de limitar el número de eventos, de cara a no sobrecargar ni la red social ni el equipo que ejecuta la simulación.

Estructura del evento

Esto representa la unidad de ejecución en la aplicación, aquí es donde se desarrolla la acción de envío por parte de los usuarios. Estos eventos deben generarse de forma dinámica para que varíen mucho y así crear cargas realistas, pero de ello se encarga el gestor. El propio evento consta de cuatro parámetros que serán determinantes a la hora de definir el transcurso del mismo:

- **Actividad** (0-1): Cumplirá dos roles, en primer lugar nos permite controlar la duración del evento, una actividad larga dará lugar a más interacciones. Por otro lado generará una actividad decremental, esto se debe a que también será la probabilidad de que un usuario envíe una interacción al servidor.
- **Factor de disminución** (Entero): Será la constante que definirá cuanto reduce la actividad cada interacción, esta será definida por el usuario en la configuración.
- **Temática**: Será la etiqueta del evento, es decir, sobre que tema se trata ese evento, esto servirá no solo para definir los usuarios iniciales que ya creamos si

no para penalizar la posibilidad de que haya interacciones por parte de usuarios que no se interesan en este tema.

- **Usuarios iniciales**: Estos serán el principio de la actividad y en esencia los que decidirán si el evento continua o no, en caso de que usuarios poco relevantes sean este conjunto inicial pocos seguidores mantendrán el evento vivo y a pesar de su actividad este terminará prematuramente.

Como puede apreciarse el proceso es simple (*Página 23*), se comprueba si un usuario realiza una interacción, en caso positivo se determina según el perfil y si es un usuario base o ha entrado al evento por otro usuario. A continuación se ajusta la actividad como he comentado anteriormente y se suma el post, por último se comprueban que seguidores continuarán el evento y cuales no.

Cabe destacar la presencia del término *Penalización*, esta variable entra para limitar la actuación de usuarios. Un usuario puede interactuar en un evento porque otro al que sigue lo ha hecho, pero si no es un tema que le interese es menos probable que esto ocurra, o puede que el usuario original solo diese *Like* algo que tiene menos valor que un mensaje propio, es por esto que se deja la ventana para que esto pueda darse, pero evitando que sea un suceso común. Los penalizadores aplicados son constantes basadas en la gravedad de cada suceso, en caso de ser por un perfil pasivo se aplica $0{,}5$, debido a que hace la mitad de probable que otro usuario interactúe con eso por lo general, en caso de ser un tema que no corresponde con el usuario es de $0{,}75$ debido a que es aún menos probable. El caso de ambos penalizadores aplicados se da por hecho que el usuario nunca interactuara ya que será ver el mensaje de otro con un *Like* sobre algo que no le interesa.

Algoritmo 3: Ejecución del evento

```
Data: Actividad
Data: Temática
Data: Factor de disminución
Data: Usuarios iniciales
Pila Usuarios ← Usuarios iniciales;
Pila Respuestas ← Null;
while Pila Usuarios no este vacía do
    Penalización ← 0;
    Probabilidad de interacción ← Aleatorio uniforme 0 - 1;
    if Probabilidad de interacción ≤ Actividad then
        if Perfil del usuario activo then
            if Pila Respuesta ≠ Null then
                Envía una respuesta;
            else
                Envía un comentario directo;
        else
            if Pila Respuesta ≠ Null then
                Da Like una respuesta;
            else
                Envía un comentario directo;
            Penalización + = Penalización por Like;
        Actividad ← Actividad - 1/Factor de disminución;
        Número de posts++;
        for Seguidores del usuario do
            if Seguidor coincide en gusto con el evento then
                if Aleatorio > Actividad ∗ (1 - Penalización) then
                    Pila Usuarios.add(seguidor);
                    Pila Respuesta.add(usuario);
            else
                if Penalización < Penalización por Like then
                    if Aleatorio > Actividad ∗ (1 - Penalización por Etiqueta) then
                        Pila Usuarios.add(seguidor);
                        Pila Respuesta.add(usuario);
    Pila Usuarios.pop;
    Pila Respuestas.pop;
```

Registro de acciones y errores

De cara a un correcto seguimiento de que realiza la aplicación es crucial que se genere un archivo que registre todas las acciones de forma que se pueda ver que ocurre en cada momento. Para ello se ha establecido un sistema de escritor de registro, este será un proceso independiente por el cuál los eventos envían sus *logs* completos y este los escribe, pero no solo los eventos si no el gestor y todas las partes del programa que ejecuten acciones relevantes. Este enfoque surge debido a la necesidad de tener un orden cronológico sin alargar la ejecución posteriormente para escribirlo, también surge el problema de exclusión mutua si hacemos que cada hilo escriba su propio registro por lo que se necesita un elemento común a todos para evitar esto.

En consecuencia surge un proceso que es nutrido de información mediante un canal seguro que contempla la exclusión mutua para evitar sobrescribir información y que reúne todas las fuentes posibles en solo una.

Parámetros de la aplicación

Se ha nombrado múltiples veces un archivo de configuración a lo largo de la memoria, sin embargo falta explicar que habrá exactamente en este fichero y porque contiene tanta información vital.

En primer lugar se crea este método para definir variables con la intención de dotar al sistema de versatilidad, con este archivo el usuario puede especificar que parámetros son los adecuados para su simulación. Otros enfoques posibles podrían haber sido la creación de una interfaz para modificarlo de una forma más cómoda, sin embargo esta idea se descarta debido a que se pretende la creación de una herramienta específica para usuarios con conocimiento y que sea rápida de usar, una interfaz crea interrupciones entre el objetivo y el usuario, también limita la ejecución a sistemas con entornos gráficos ya que una interfaz en terminal se hace aún más lenta. También existe la alternativa de pasar algunos parámetros como argumentos, esto se descarta debido a que complica innecesariamente la llamada del comando y hace obligatoria la especificación de ajustes cada vez que se ejecuta todo.

Una vez queda explicado el porque de esta metodología doy paso a la especificación de cada parámetro presente en este archivo:

- **Usuarios**: Número entero de usuarios que se usaran en la simulación, es importante que el número no supere la cantidad de credenciales presentes en el

archivo destinado a ello ya que se ignorarán y se llegará al limite de este.

- **MaxEventos**: Número entero que determina el limite de eventos concurrentes que puede haber durante la ejecución, es importante no excederse con este parámetro ya que como veremos a continuación en los experimentos puede llegar a sobrecargar mucho el servidor según que configuración tengamos.

- **TiempoSim**: Tiempo en minutos que durará la simulación completa.

- **ProbPasivo**: Probabilidad de que un usuario sea pasivo (0,0 - 1,0), es decir que sea más espectador de los eventos o entre a interactuar, un valor alto aquí hará que los eventos sean menos duraderos ya que la interacción se demorará menos por los penalizadores que se aplicarán más comúnmente.

- **EtiquetasMedias**: Número entero de gustos promedios por cada usuario.

- **ProbabilidadEvento**: Probabilidad de que se genere un evento en cada iteración del gestor (0,0 - 1,0). Un valor alto dará lugar a muchas colisiones de eventos y una carga sostenida alta mientras que un valor más bajo creará cargas más suaves a lo largo del tiempo. Es ideal para ajustar simulaciones de distintos escenarios.

- **URL**: Dirección donde se aloja el servidor de la red social, en este caso de Mastodon, sin protocolo solo la dirección o en caso de no tener DNS la ip.

- **IP**: Dirección ip de donde se aloja el servicio de monitorización, no DNS si no ip estática. Este parámetro aparece así debido a que puede ser que ejecutemos la simulación en una estancia local por lo que sea más fácil acceder así, además de evitar que se abran puertos en el equipo servidor y así estar más protegido.

- **TiempoEntEvento**: Tiempo en segundos que se espera entre iteraciones del gestor para intentar generar nuevos eventos, un valor alto dará lugar a eventos más separados en el tiempo mientras que uno bajo puede originar que los eventos acaben de forma prematura y mantener una carga alta.

- **FactorDisminucion**: Número entero que como se ha explicado anteriormente determina cuanto se reduce la actividad por iteración, a más alto más se alarga la duración de estos.

- **SegundosEntreMediciones**: Tiempo en segundos que se especifica al servidor para que tome medidas del uso de recursos, determinará el impacto del monitor en el *host* por lo que es importante que no sea muy bajo.

Salida y su información

Por último dentro de las especificaciones de la herramienta queda destacar como se recogen los datos que estarán presentes en el informe final que da como salida la aplicación.

Como se ha indicado antes la conexión entre cliente y servidor se establece mediante el uso de *sockets*, en un primer lugar se establece la conexión y se indican dos parámetros al monitor, cuanto durará la simulación y cada cuanto debe tomar muestras. Esta información es tomada en el servidor y guardada en RAM hasta que la simulación finaliza, es entonces cuando se guarda una copia en formato *json* y a su vez se envía al cliente. El cliente al finalizar de cerrar correctamente todos los procesos recoge los datos enviados y los combina con el registro generado en el gestor de eventos, todo esto formará un *CSV* final que será nuestro archivo de salida.

La estructura de este archivo es simple, en las primeras columnas tenemos la lecturas del monitor, CPU, RAM y disco; a continuación tenemos la media de interacciones por evento y la duración media de cada uno. Los datos del servidor son obvios dado que lo que queremos ver es el impacto de la simulación en la red, por otro lado estos últimos se especifícan de cara a darnos una idea de que carga ha generado los parámetros indicados en esa configuración, esto nos permite identificar si es similar a lo que esperamos de la red o por otro lado debemos hacer modificaciones para ajustarla más.

Por otro lado están los logs, estos archivos buscan dejar un registro de todo lo que ocurre a lo largo de la simulación con todo lujo de detalles. Su estructura es simple, se escribe un mensaje cada vez que ocurre una acción relevante, por ejemplo si el gestor encuentra que se superen el número máximo de eventos concurrentes. En el caso de llevar un control de cada evento su registro se realiza todo junto, aunque cada evento ocurra a la vez que otros, esto se hace para hacer más legible y facilitar el seguimiento de cada uno sin crear múltitud de archivos para ubicar cada uno.

```
Iniciando simulacion
Evento en creacion
0: evento iniciado
0: evento terminado con 0 interacciones
Evento en creacion
Evento en creacion
Evento en creacion
2: evento iniciado
2 || usuario98 postea (penalizado)
2 || usuario94 hace rt una respuesta de usuario98
2 || usuario68 hace rt una respuesta de usuario94
2 || usuario92 postea una respuesta a usuario68
2 || usuario33 hace rt una respuesta de usuario92
2 || usuario92 postea una respuesta a usuario33
2 || usuario57 postea una respuesta a usuario92
2 || usuario59 postea una respuesta a usuario57
2 || usuario72 hace rt una respuesta de usuario59
2 || usuario56 hace rt una respuesta de usuario72
2 || usuario41 hace rt una respuesta de usuario72
2 || usuario52 postea una respuesta a usuario41
2 || usuario75 hace rt una respuesta de usuario52
2 || usuario1 hace rt una respuesta de usuario52
2 || usuario55 hace rt una respuesta de usuario1
2 || usuario48 postea una respuesta a usuario55
2 || usuario42 postea una respuesta a usuario48
2 || usuario93 postea una respuesta a usuario42
2 || usuario12 hace rt una respuesta de usuario93
2 || usuario40 hace rt una respuesta de usuario12
2 || usuario93 postea una respuesta a usuario40
2 || usuario99 hace rt una respuesta de usuario93
```

Figura 2.2: Ejemplo de Log

2.3. Proceso de puesta en funcionamiento

Una vez queda explicada la idea tras todo el planteamiento de cada proceso podemos tener una idea de como es la herramienta propuesta, pero lo ideal sería ponerla en práctica en un entorno lo más real posible, es por ello que ahora expondré ese entorno de demo que he creado para ilustrar este proyecto, tanto que alternativas tenía como el proceso de despliegue de todo lo necesario para hacerlo funcionar.

2.3.1. ¿Por que Mastodon?

En el apartado Tecnologías usadas hago mención a la red social Mastodon, antes de pasar a explicar más la estructura de esta red es preciso hacerse la pregunta con la que introduzco este apartado.

Para entender porque he escogido Mastodon como entorno de pruebas es necesario analizar las alternativas presentes, en un primer lugar se nos ocurriría lanzar la simulación a una red social ya creada, sin embargo esto presenta dos problemas principales, por un lado hoy en día la gran mayoría de redes sociales prohiben explicitamente los bots en sus términos. Por otro lado no tendría acceso al servidor por lo que no puedo ver el impacto de la simulación. Como punto extra cabe destacar que podría incidir en un ataque de denegación de servicio (*DDoS*) lo cuál también infringe una serie de reglamentos.

Otra alternativa posible que tenía era desarrollar yo mismo una red social básica para ilustrar el funcionamiento de las pruebas, el problema que esto trae consigo es que este desarrollo ya superaría la envergadura de este trabajo ya de por sí, además esta red social necesitaría de un desarrollo preciso para ser eficiente ya que si no será incapaz de aguantar siquiera el test más suave que le pueda lanzar.

Por esto quedo limitado a redes sociales que yo pueda desplegar, con posibilidad de insertar bots y optimizada para soportar una carga media sin dar lugar a muchos errores. Ante estos requisitos aparece Mastodon, que yo haya podido encontrar la única red social que cumple estos requisitos y más. No solo es lo que necesito si no que tambien proporciona una API la cuál ya tiene librerías desarrolladas en los lenguajes más comunes lo cuál facilita el desarrollo y hace que me sea invisible a la hora de planificar y programar toda la aplicación.

2.3.2. Despliegue en el servidor

Una vez queda explicado porque se usará Mastodon es momento de explicar su despliegue. Este proceso antes de todo requiere de un entorno basado en Linux, en este caso usaré una Máquina Virtual de Ubuntu, aunque lo ideal sería lanzarlo en un servidor pero no cuento con ese hardware a mi disposición y también me permite adaptar la configuración para las distintas pruebas que se harán.

Una vez tengo instalado Ubuntu he seguido una guía online para el proceso completo, para agilizar esto y no especificar comandos en exceso voy a citarla Ins y comentar el proceso en rasgos más generales junto a las modificaciones que he tenido que realizar ya que esta algo obsoleta. Primero las etapas del proceso son las siguientes:

1. Requisitos previos
2. Instalación de Node.js y yarn
3. Instalación y configuración de PostgreSQL
4. Instalación de Ruby
5. Configuración de Mastodon
6. Configuración de Nginx
7. Creación de demonios para lanzado automático

Quizás lo más importante de todo el proceso sean los requisitos, es importante tener un dominio funcionando para que Mastodon opere, en mi caso use el DNS gratuito de No-Ip. Con este he podido crear el nombre *ottoand.ddns.net* que será el acceso a mi red Mastodon. Por lo demás solo cambiará el hecho de que ejecuto Ubuntu en su última versión en lugar de Debian, pero eso no supone ningún problema.

Una vez cumplidos los requisitos avanzamos por los pasos hasta la instalación de Ruby, esta guía como comento usa una versión de Ruby bastante antigua, en concreto la 2.6.1 , el problema aparece con que Mastodon ha sido actualizado, aunque su desarrollo parece que se estancó hace un tiempo. Por todo esto la versión necesaria para ejecutar el Mastodon actual es distinta a esa, en concreto es la 2.7.2 la que uso yo para que funcione, esto es importante ya que las librerías posteriores no funcionan en la versión que se indica dando lugares a errores en la instalación.

A continuación llega el momento de instalar Mastodon y configurarlo, aquí di con un problema que no estoy seguro si es por usar Ubuntu o es por la versión, el caso

es que Mastodon necesita de tener instalado *Redis*, esto no es otra cosa si no una pseudo BBDD principalmente usada para guardar tablas en cache para un uso rápido, ideal en una red social para accesos de usuarios por ejemplo. El problema que surge es que mi versión de Ubuntu no lo incorporaba por lo que saltaba un error no explicito a lo cuál tuve que depurar para ver donde fallaba.

Por último queda la instalación y configuración del servicio web, en este caso Nginx, que principalmente consiste en validar un certificado para poder usar el protocolo HTTPS en la red social. Este proceso ha cambiado a lo largo del tiempo, esto se debe a que antes con insertar el certificado y arrancar era suficiente, sin embargo actualmente es necesario iniciar primero y verificar ese certificado. El problema surge en que una de las directrices de Mastodon para iniciar es cargar ese certificado por lo cuál choca y no arranca el servicio, y por tanto no se puede validar, es un ciclo. La solución a esto fue simple, eliminé estas directrices del archivo de configuración de Mastodon, inicie el servicio, valide el certificado y reinicie con las directrices originales.

Con esto hecho y los *demonios* creados queda una máquina virtual lista para funcionar con mi nodo de Mastodon operativo y accesible.

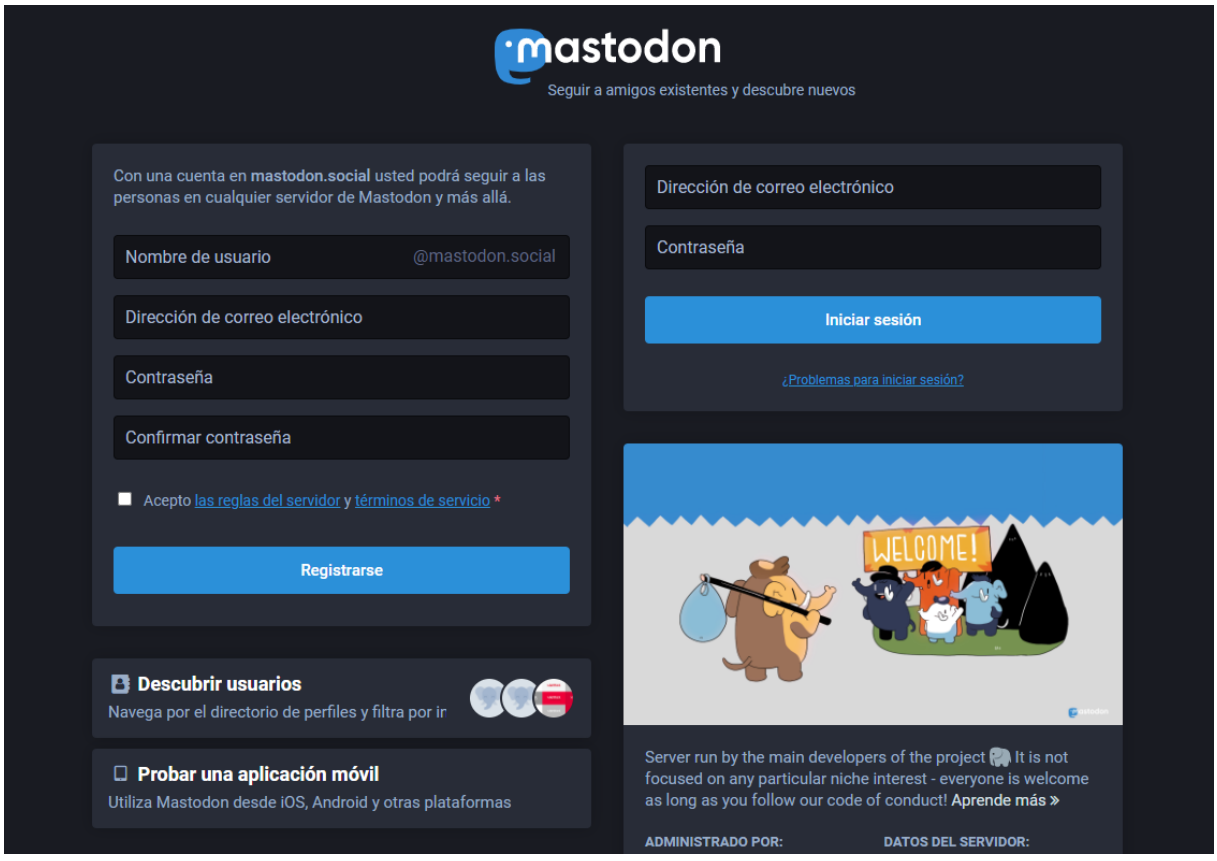

Figura 2.3: Página de inicio de sesión de Mastodon

2.3.3. Implementación del cliente

Con el nodo listo y la herramienta terminada solo quedaría hacer las pruebas y test para ver que efectivamente todo funciona como debe. Para esto primero se debe configurar el script que se lanzará en el servidor.

Este pequeño script de *Python* como comente busca monitorizar el uso de recursos del servidor y mandarlo al cliente, para hacer esto requiere de que se le de la IP del equipo en el que se usará. Este proceso he encontrado mucho más cómodo definirlo en el mismo script, tan solo es definir la variable indicada al principio. Con esto la ejecución del mismo es muy rápida ya que consiste en una mera llamada al script y se lanzará en el puerto establecido por defecto, el 8082.

Todo el código fuente tanto de servidor como del cliente esta subido tanto en gitlab como en drive junto a los archivos necesarios para su compilación.

Con esto operativo en nuestro servidor podemos hacer alguna prueba de conexión pero en principio todo debe conectarse y ya funcionará la herramienta, así que solo nos quedará pasar a la acción.

Capítulo 3

RESULTADOS

En este capítulo se busca ilustrar el funcionamiento completo de la herramienta a través de un posible caso real. Imaginemos que somos una empresa que quiere desplegar sus propios nodos de Mastodon, vamos a ver cuál sería la configuración concreta para alojar a una media de 100 usuarios por nodo. Esta baja cifra la cuento debido a que existen muchos accesos a la red de Mastodon por lo que la carga se balancea mucho y al final las conexiones reales al servidor descienden mucho en cada nodo concreto.

Iteracion	Uso CPU	Uso RAM	Numero de Lecturas	Numero de escrituras	Tiempo de lectura	Tiempo de escritura
0	0,0%	1 723	40 908	958 565	18 218	29 276
1	21,8%	1 756	40 921	958 639	18 396	29 337
2	34,6%	1 783	40 921	958 639	18 662	29 452
3	29,3%	1 777	40 921	958 639	18 906	29 540
4	1,8%	1 777	40 921	958 639	18 931	29 548
5	0,2%	1 777	40 921	958 639	18 940	29 551
6	10,0%	1 778	40 921	958 639	19 037	29 597
7	26,2%	1 767	40 921	958 639	19 536	29 998
8	20,8%	1 767	40 921	958 639	19 821	30 177
9	10,3%	1 768	40 923	958 665	19 930	30 301
10	11,1%	1 768	40 923	958 665	20 083	30 380

Figura 3.1: Ejemplo de Informe

Estas simulaciones como he indicado se lanzarán en una máquina virtual debido a que no dispongo de hardware específico de servidor, de ahí que ciertos datos puedan ser anómalos. Antes de empezar con las pruebas es necesario saber la estructura con la que se presentarán los datos. De forma original la herramienta nos da un formato

CSV que es transformado a tabla de forma trivial, sin embargo la lectura de esta es difícil como se puede ver en este ejemplo simple. *Véase 3.1*.

Es solo este ejemplo y queda algo difícil de ver, es por esto que iré transformando estos datos en gráficos que recalquen y visualicen los datos mejor según de que esté hablando. Con esto en mente paso a las pruebas directamente.

3.1. Primera configuración de hardware

Las simulaciones que he lanzado han sido de diez minutos cada una tomando mediciones cada diez segundos para no saturar con el monitor al servidor. En un principio comienzo con la siguiente configuración:

- **CPU**: 1 procesador x 2 núcleos
- **RAM**: 4 GBs
- **Disco Duro**: 50 GBs
- **Red**: 300 MBs simétricos en modo puente

Como se puede observar la configuración de CPU y RAM es baja ya que el objetivo es intentar optimizar todo lo posible el hardware para reducir gastos. El disco duro se pone en 50 GBs ya que para las pruebas no es necesario mucho espacío, en la configuración final si se llegarán a capacidades más altas. En cuanto a red es el máximo de lo disponible aunque no supondrá un cuello de botella como se verá más adelante.

Con esto definido se lanza el primer test, para este se usa una configuración que adjuntaré de base, y posteriormente iré especificando únicamente las modificaciones que se hagan al mismo.

- usuarios = 100
- maxEventos = 15
- tiempoSim = 10
- probPasivo = 0.5
- etiquetasMedias = 7

- probabilidadEvento = 0.8
- tiempoEntEvento = 15
- factorDisminucion = 500
- segundosEntreMediciones = 10

Como se puede apreciar, de por si esta configuración busca varios factores, en primer lugar expande mucho las opciones de muchos eventos concurrentes y de gran longitud gracias a un factor de disminución muy alto (recordar que a más alto menos disminuye el tiempo del evento) y una alta probabilidad de evento en cada iteración, por otro lado las etiquetas medias son altas lo cuál hace que los usuarios sean más propicios a participar en los eventos, aunque dejamos que sea mitad y mitad de usuarios activos y pasivos.

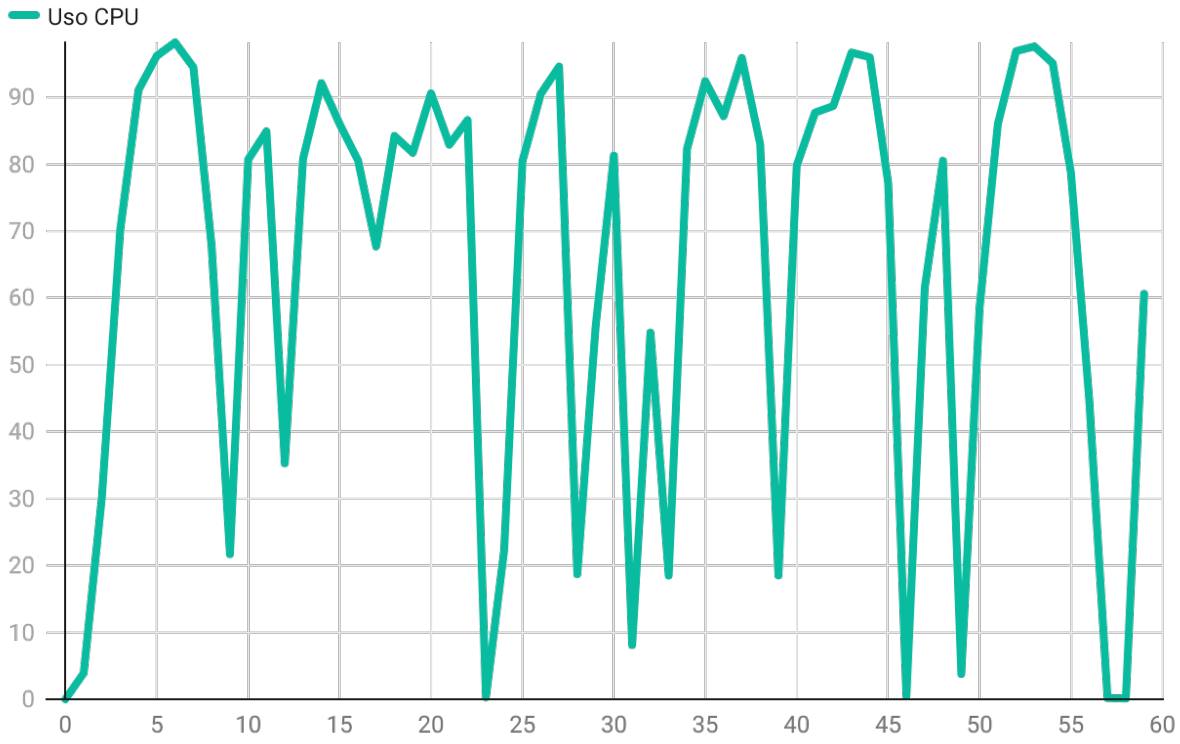

Figura 3.2: Test 1 - CPU

Ante los resultados 3.2 lo primero que podemos ver que la CPU definitivamente necesita de más núcleos o puede que otro segundo procesador. La actividad vemos que es intermitente pero llegamos a resultados muy altos, si bien para una carga bastante alta, aunque no la que más, hablamos de unas 244 interacciones medias por evento en esos diez minutos Esta carga supone como si cada usuario interactuase dos

y hasta tres veces por evento lo cuál deja un margen importante de cara a la predicción inicial, sin embargo he decidido aumentar el hardware para crear más margen en caso de querer expandir los usuarios en cada nodo. Sin embargo antes de hacer las modificaciones es conveniente ver como se han comportado el resto de componentes.

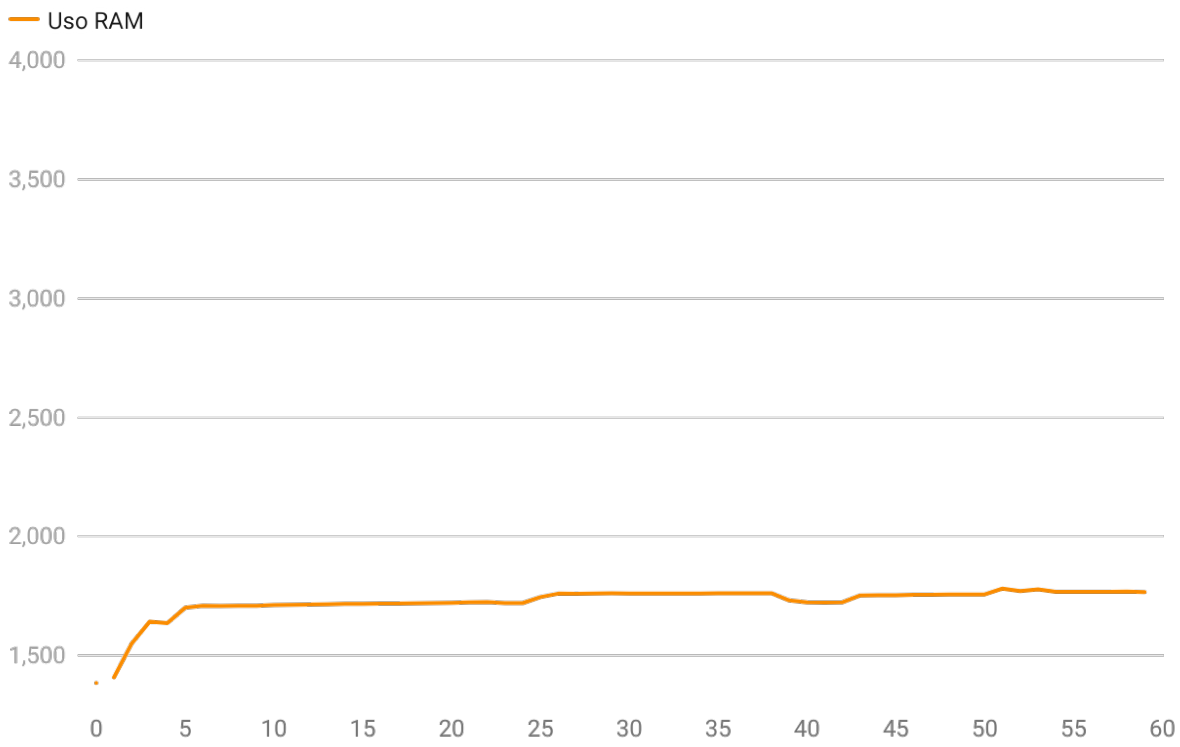

Figura 3.3: Test 1 - RAM

Como vemos 3.3 la RAM a pesar de tener 4 GBs disponibles se estanca en 1.7 GBs por lo que podríamos pensar que bajar su capacidad es una buena forma de optimizar, aunque por ahora se mantendrá en esa capacidad a falta de ver si es por un cuello de botella en CPU o realmente no se requiere más.

El disco también parece sufrir a lo largo del tiempo de forma lineal aproximadamente 3.4, esto considero que es debido al impacto de la virtualización, aunque no estoy completamente seguro. Podría pensarse que fuese la CPU pero como veremos más adelante este comportamiento se mantiene en las distintas lecturas más adelante.

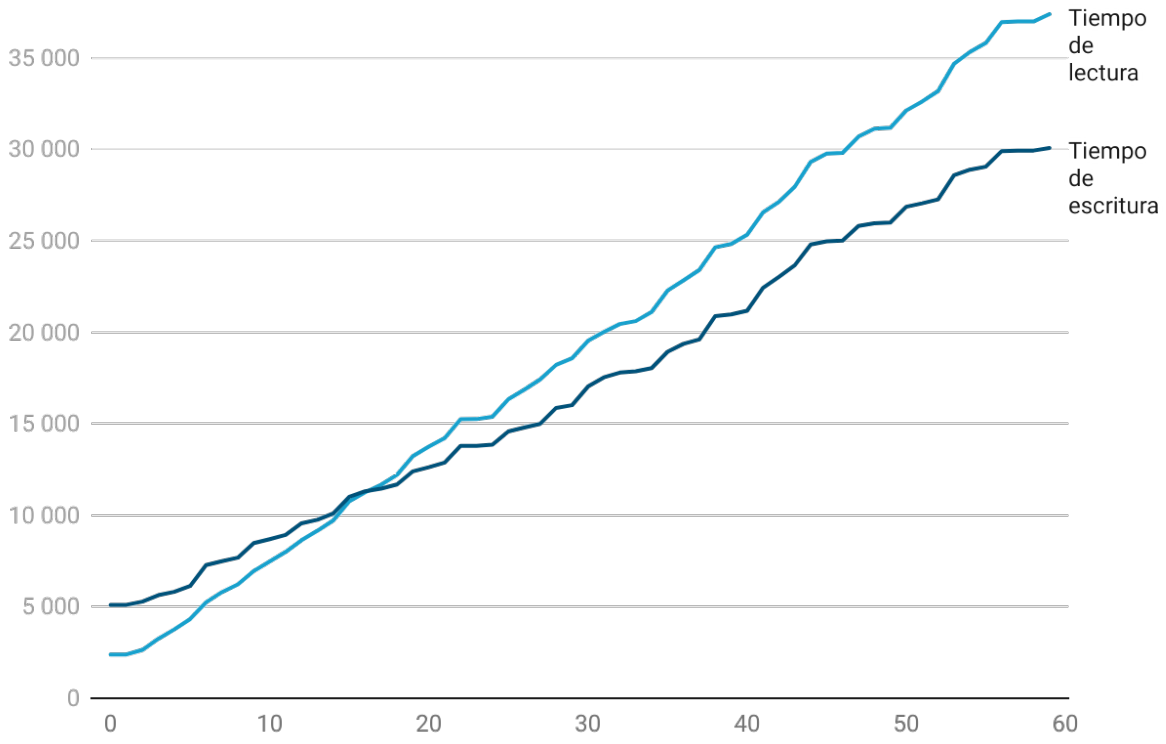

Figura 3.4: Test 1 - Disco

3.2. Ampliación de núcleos

En el siguiente test he probado a **aumentar en dos núcleos más al hardware** para ver si el cuello de botella se alivia, por lo demás he mantenido la RAM como he indicado antes y he dejado la configuración sin alterar para ver que tal el comportamiento.

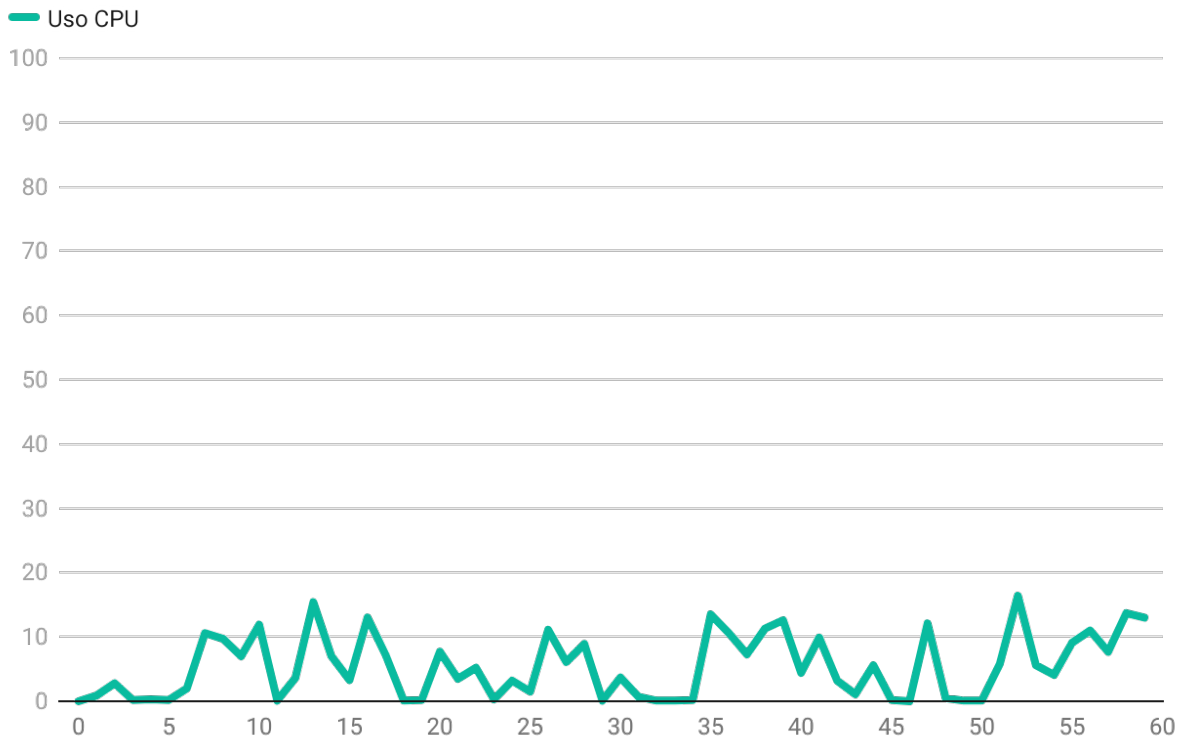

Figura 3.5: Test 2 - CPU

Como se puede apreciar 3.5 ahora el uso de CPU es muchísimo menor, esto puede deberse a dos posibilidades. Puede ser que la implementación de Mastodon no sea escalable, algo poco probable, o puede ser que efectivamente se reparta la carga entre los 4 núcleos y se reduzca en gran medida la saturación. Lo que sabemos con certeza es que esta configuración elimina el cuello de botella en CPU, así que faltaría ver el impacto de esto en el resto de componentes.

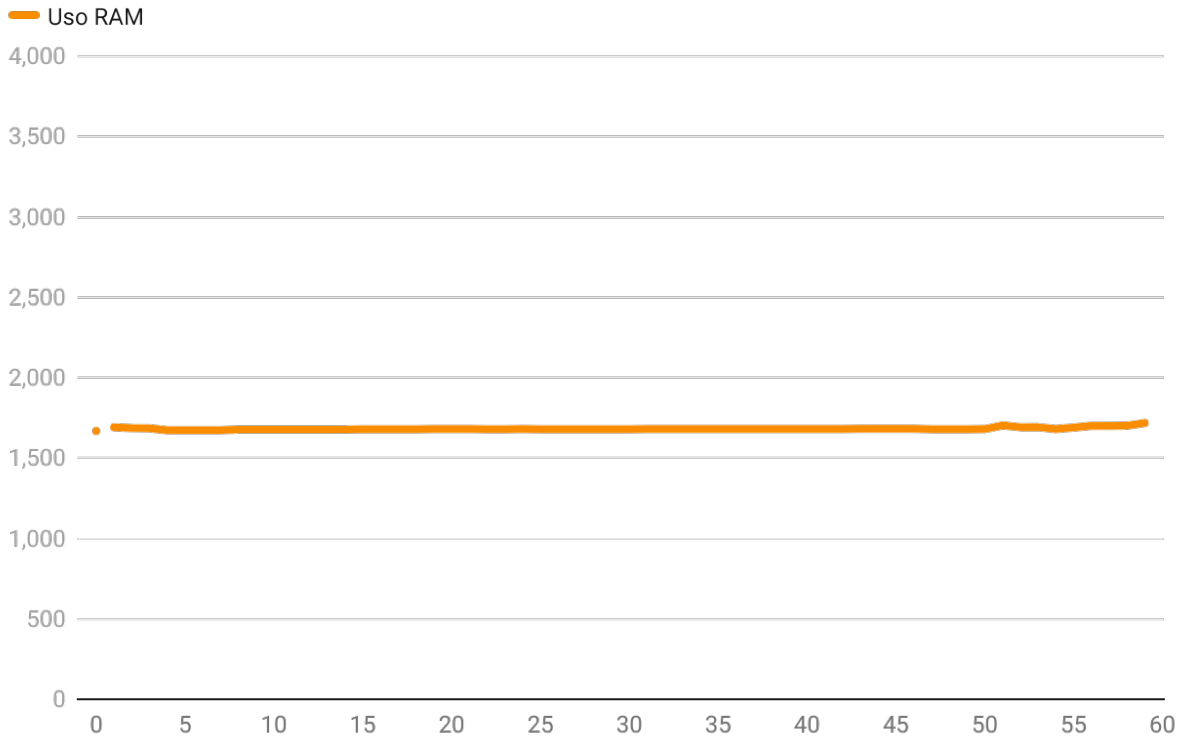

Figura 3.6: Test 2 - RAM

Como adelantaba la RAM se mantiene en un registro similar (3.6) si no igual al anterior caso, por lo que definitivamente la CPU no era un factor limitador a la hora de que se usase más, por lo que casi que podríamos concluir que la cantidad ideal de RAM para cada nodo es de 2 GBs ya que no parece expandirse más. En el caso de disco (3.7) tenemos unos datos que llaman la atención debido a que son mucho peores que los anteriores, esto se puede deber o bien a que el cuello de botella en CPU limitaba las lecturas/escrituras, o bien el hecho de ser una máquina virtual afecta negativamente a esto, lo cuál no sería extraño. Para comprobar esto he lanzado un segundo test con la misma configuración para ver cuanto oscilan estos datos.

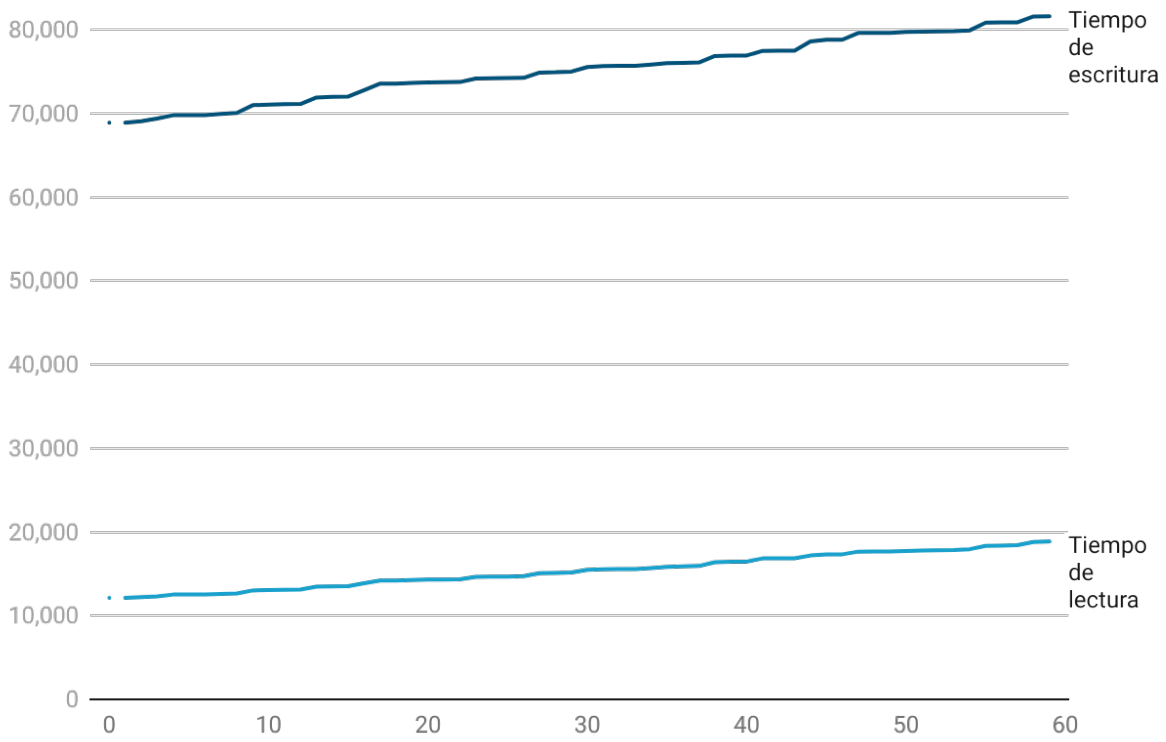

Figura 3.7: Test 2 - Disco

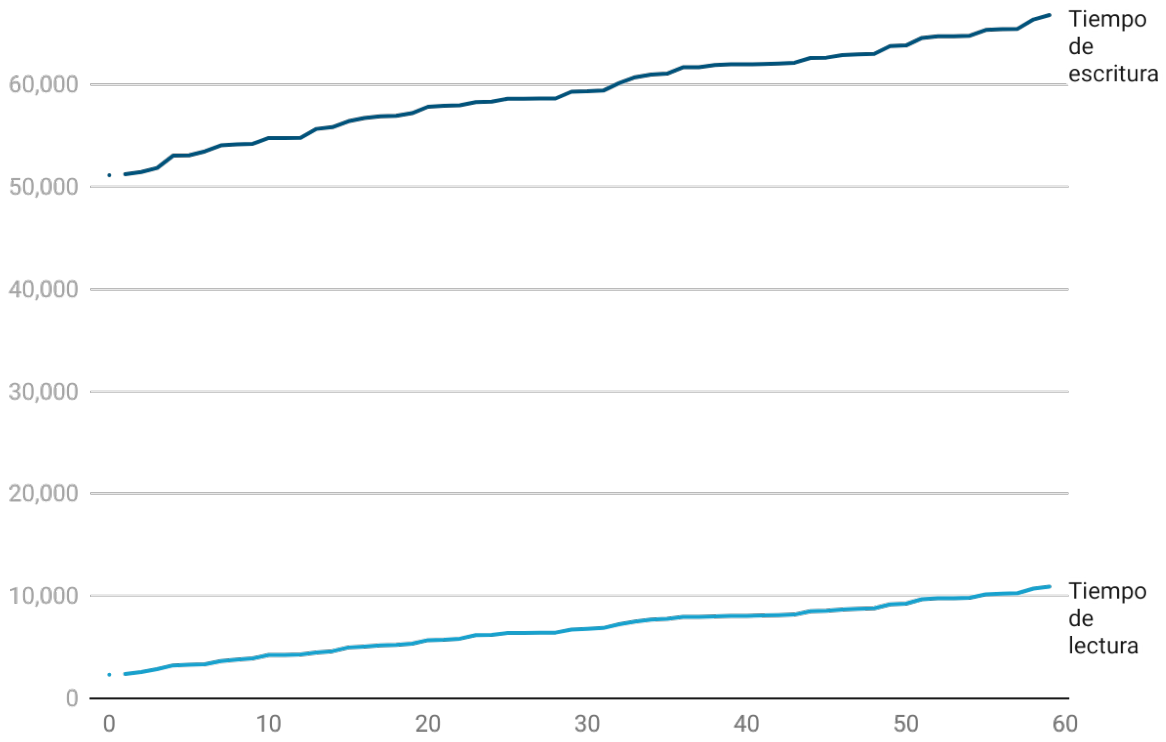

Figura 3.8: Test 3 - Disco

Y en efecto vemos (3.8) unas lecturas menores a las anteriores sin que se varíe nada de la configuración anterior, esto nos muestra que en efecto el ser una máquina virtual hace que lo que se ejecute en el host modifique los tiempos de lectura de nuestro servidor. Siendo conocedores de esto podemos ver que el tiempo de lectura de disco es algo irrelevante en nuestras pruebas por lo que pasaremos a omitirlo de cara a ofrecer datos de mayor exactitud, aunque como destaque al inicio el disco en el despliegue real sería de unas características muy distintas al usado en estas pruebas.

3.3. Alternativas a las opciones convencionales

Con esto podría dar por completadas las simulaciones para el despliegue, sin embargo hoy en día existe una opción casi más económica según de que hablemos, y es usar varios procesadores de menor potencia para alojar el servicio, es por ello que antes de concluir he decidido primero lanzar una carga inferior a esta configuración y posteriormente a la misma para ver como varía y si es viable tener en cuenta esta opción. La duda aparece ya que cuando estamos ante un sistema multiprocesador la RAM se divide entre los dos, y si uno quiere acceder al espacio de memoria del otro debe consultar el dato al otro procesador, con la consecuente perdida de rendimiento que esto ocasiona.

Pasando a la primera configuración para **dos procesadores de dos núcleos cada uno** he creado una carga menor para comenzar, respecto a lo anterior cambia lo siguiente:

- **maxEventos** = 10 (antes era 15)
- **probPasivo** = 0.3 (antes era 0.5)
- **tiempoEntEvento** = 10 (antes era 15)
- **factorDisminucion** = 300 (antes era 500)

Con esta configuración se crea una carga de unas 162 interacciones medias por evento, algo mucho menos a la anterior. Sin embargo a pesar de esto como vemos en la figura 3.9 el porcentaje de uso de estos núcleos es un modo intermedio entre lo que teníamos antes con un solo procesador y dos núcleos, esto refleja esa perdida de rendimiento que comentaba antes. En el caso del uso de memoria RAM como era esperable se mantiene exactamente igual que en los anteriores casos por lo que este cambio no afecta a ese uso. Por último y como anotación en este caso los tiempos de

lectura aumentan, posiblemente reflejando no solo los inconvenientes de la máquina virtual si no también que es menos eficiente esta configuración.

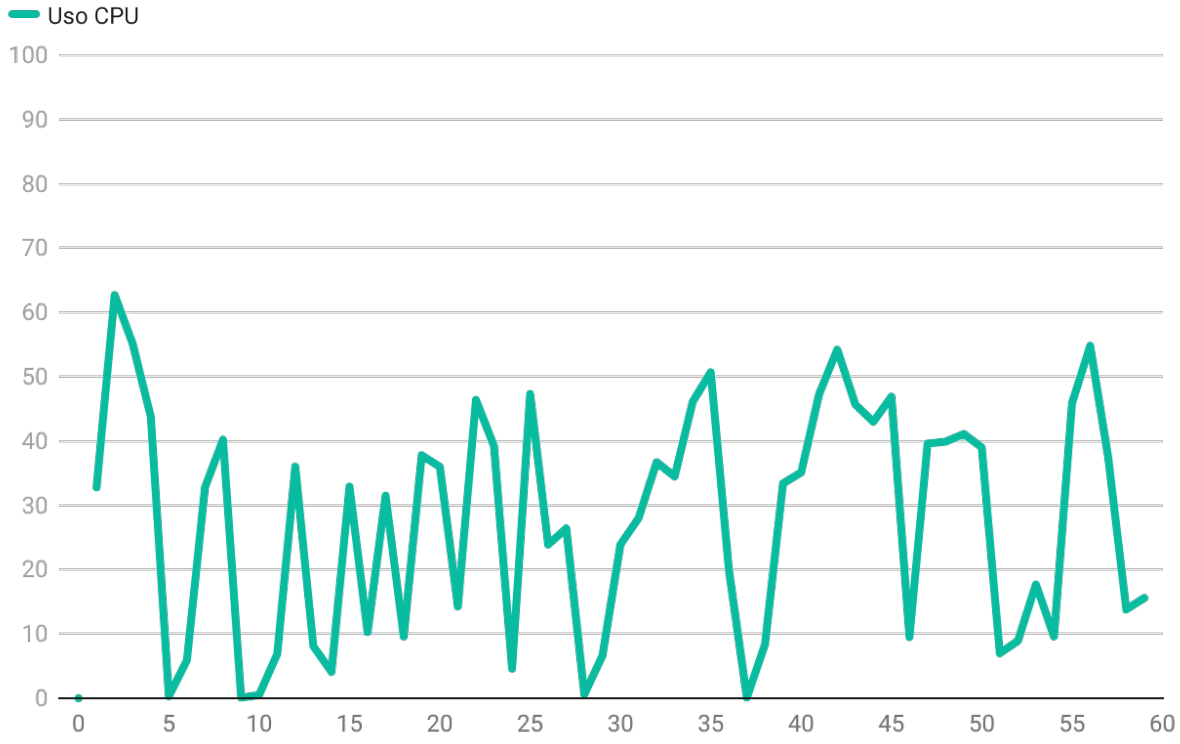

Figura 3.9: Test 4 - CPU

Ahora veamos como responde esta combinación de hardware ante la misma carga que los anteriores. En este caso (3.10) se da algo bastante curioso, el uso de CPU no sube, si no que baja. Esto puede dar a pensar en un primer lugar que existe algún tipo de error en las mediciones, pero parémonos a pensar por un momento en que tipo de carga estamos dando. Con la configuración inicial lo que buscamos es muchos eventos concurrentes, al reducir esta carga lo que hemos hecho es reducir el número de eventos concurrentes, por lo que el impacto es directo por evento. Esto puede llevarnos a una hipótesis, cuando tenemos menor número de eventos la acción se concentra de forma poco balanceada por lo que el uso sube debido a los defectos de la combinación de hardware, mientras que al tener más se balancea mejor y no se da esta casuística.

Esta hipótesis queda muy bien plasmada en papel pero es necesario reflejarlo con un último experimento, en este caso voy a reducir la carga a unas 100 interacciones medias por evento, para ello reduciré mucho la vida de los eventos bajando el factor de 500 a 200 y daré un tiempo entre evento de 5 en lugar de 15 para compensar, de esta forma lo que creo es menor carga por evento pero mantengo la densidad de eventos, así que el resultado esperable es un uso de CPU similar o algo superior al que tenía con esta configuración de hardware en mi primer test.

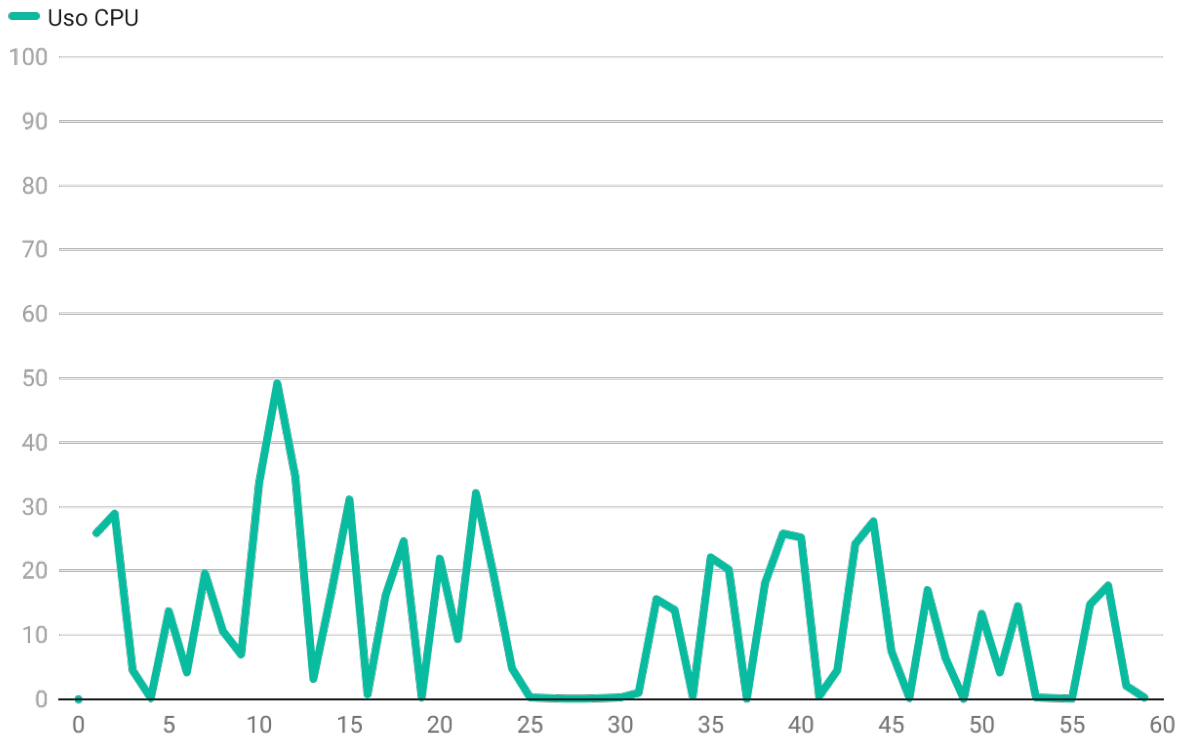

Figura 3.10: Test 5 - CPU

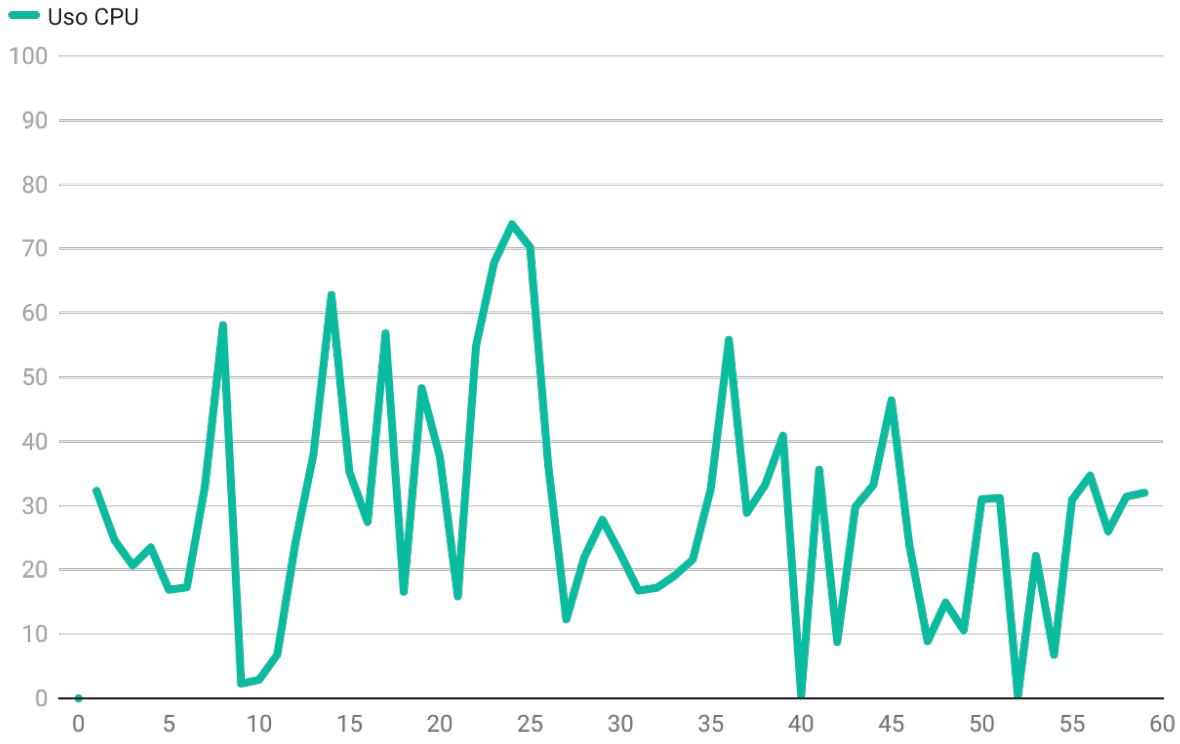

Figura 3.11: Test 6 - CPU

Con la gráfica 3.11 queda reflejado que estaba en lo correcto, tenemos un uso de media mayor al del primer test, lo cuál confirma la hipótesis planteada y deja en evidencia que esta configuración crece de forma inversa a la carga que se le aplica.

Con esto podemos concluir de cara a nuestra simulación que existen dos configuraciones ideales:

- **Un solo procesador de mayor potencia**: Siendo esta la forma que en todos los apartados obtiene un mejor rendimiento y más margen de escalabilidad
- **Dos procesadores de menor potencia**: Siendo esta la forma más económica con un rendimiento aceptable además de un margen para crecer dentro de los planes impuestos al inicio.

3.4. Apéndice: Conclusiones

Y para resumir un poco todas las pruebas llevadas a cabo así como dejar constancia de las configuraciones usadas voy a incluir una tabla a modo de resumen con toda la información relevante 3.1. Como se ha llegado a la conclusión de que los datos respectivos a tiempos de lectura y escritura dependen mucho del estado de la máquina virtual no están incluidos, así como los apartados de configuración fijos por la máquina (IP y dominio) y los inherentes del problema propuesto (usuarios, tiempo y tiempo entre mediciones).

Test	1	2	3	4	5	6
MaxEventos	15	15	15	10	15	15
ProbPasivo	0.5	0.5	0.5	0.3	0.5	0.5
EtiquetasMedias	7	7	7	7	7	7
ProbabilidadEvento	0.8	0.8	0.8	0.8	0.8	0.8
TiempoEntEvento (seg)	15	15	15	10	15	5
FactorDisminucion	500	500	500	300	500	200
Interacciones medias	244.53	259.21	265.81	162.23	238.52	98.13
Media Uso CPU (%)	64.72	5.44	6.67	27.18	11.86	28.56
Media RAM (MBs)	1724.3	1682.41	1665.85	1753.47	1737.39	1797.31

Tabla. 3.1: Tabla resumen de las pruebas

También una carpeta con todos los archivos generados a lo largo de las distintas pruebas hechas en este apartado, con el objetivo de dejar constancia de todos los resultados así como logs de lo ocurrido y no contemplado en este análisis (cuya estructura queda explicada en el apartado "Salida y su información").

Capítulo 4

CONCLUSIONES

Para concluir con este proyecto considero importante hacer un repaso de los objetivos propuestos al inicio y ver de que manera han sido cumplidos. Es relevante destacar antes las dificultades que este trabajo traía consigo, todo originado por un factor que a su vez lo hace interesante, este no es otro que su innovación. Como he destacado antes la herramienta desarrollada no cuenta con referencias claras actualmente por lo que ha sido un enfoque completamente original en base a un estudio profundo de la carga que sufren las redes sociales en su uso diario y un conocimiento previo para sintetizar y diseñar algo que se asemeje.

El primer objetivo que se colocaba era el de crear una solución **flexible, escalable y completa**. Empezando por el primer punto, una herramienta flexible es aquella que permite actuar según las directrices que se dan, escalable hace referencia a que puede crecer sin problema siempre que los recursos sean suficientes y completa implica que contemple todos los factores importantes. Para estos tres puntos *SocialBench* tiene su factor que lo cumple, la flexibilidad se consigue con un archivo de configuración que permite adaptarse a todas las cargas que queramos así como la facilidad y modularización necesaria que hace probar cualquier red algo casi trivial siempre que se conozca el método de comunicación. En el apartado de escalabilidad, la herramienta siempre lo será ya que sus únicas limitaciones son las impuestas por el hardware (número de hilos) y por el usuario (número de cuentas para acceder). Y por último se cumple el hecho de ser completa ya que la carga generada tiene en cuenta a todos los tipos de usuario que puede haber, haciendo que esten presentes siempre que se quiera.

En segundo lugar tenemos el objetivo de crear una **carga realista**, esto en concreto se cumple gracias al sistema de eventos desarrollado ya que da realismo a las inter-

acciones de los usuarios, no son aleatorias si no basadas en el grafo social generado. Además esta carga es adaptable según que se considere realista para cada entorno ya que lo mismo para un servidor de *Facebook* 7000 interacciones en 10 minutos no es realista.

Y como tercer y último objetivo se estableció la creación de una herramienta que **aportase información necesaria para ver el comportamiento del servidor** durante la simulación, esto se consigue desde dos puntos distintos. En un primer punto y de forma clara al finalizar la ejecución se genera un fichero CSV a modo de informe lo cuál da muchos datos, sin embargo otro modo que aunque más lioso es también más preciso es la creación de un log de todas las acciones que ocurren durante la ejecución de los distintos eventos. Es por esto que este objetivo se cumple de forma satisfactoria sin lugar a dudas.

Con los objetivos propuestos cumplidos solo nos queda pensar a futuro, ¿que margen de mejora tiene esta herramienta? La respuesta es mucho. Actualmente esta aplicación es capaz de generar una carga realista sobre un servidor, pero las actuaciones que tiene son textos sin sentido más allá de indicar que de que evento se trata y que id tienen asignado. Esto puede llevarnos a ver de que son bots en un sentido, debido a que simulan comportamientos humanos, sin embargo el comportamiento que simulan esta incompleto por lo que a nuestros ojos no muestran capacidades comunicativas ninguna. Una evolución para estos por tanto sería dotarlos de la capacidad de crear publicaciones realistas, sin que esto ponga en riesgo las capacidades ya disponibles.

Por otro lado otra modificación interesante sería en la generación de los usuarios, actualmente se ha dejado operativa una estructura de datos compleja con mucha información de cada uno pero cuyo uso en la generación es inútil, esto podría tenerse en cuenta para crear perfiles más acordes y con más similitud a los usuarios de una red social.

Y con esto quisiera cerrar la memoria de este Trabajo de Fin de Grado, he aprendido mucho a lo largo del desarrollo de este, conceptos tan específicos como pueden ser el funcionamiento interno de una red social, hasta conocimientos tan genéricos como puede ser la influencia del hardware en despliegues específicos y como aunque a priori algo pueda parecer mejor no será siempre lo óptimo para determinados casos.

Bibliografía

Vega L. Lopez-Cuevas A., Mendez-Vazquez A. Probabilistic reasoning system for social influence analysis in online social networks. *Social Network Analysis and Mining*, 11, 2021. doi: 10.1007/s13278-020-00705-z.

M.B. Masrom, A.H. Busalim, H. Abuhassna, and N.H.N. Mahmood. Understanding students' behavior in online social networks: a systematic literature review. 18(1), 2021. doi: 10.1186/s41239-021-00240-7.

Doriane Kouame Simeon Edosomwan, Sitalaskshmi Kalangot Prakasan. The history of social media and its impact on business. *JAME*, 16, 2011. URL https://web.archive.org/web/20210425213155/http://www.minot.com/tom/SocialMedia.pdf.

Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, June 2016. ISSN 0001-0782. doi: 10.1145/2818717. URL https://doi.org/10.1145/2818717.

R. Stuart Geiger. The lives of bots, 2018.

Björn Ross, Laura Pilz, Benjamin Cabrera, Florian Brachten, German Neubaum, and Stefan Stieglitz. Are social bots a real threat? an agent-based model of the spiral of silence to analyse the impact of manipulative actors in social networks. *European Journal of Information Systems*, 28(4):394–412, 2019. doi: 10.1080/0960085X.2018.1560920.

Scott Shane. Rusia creó perfiles falsos de estadounidenses para influenciar en las elecciones. *The New York Times*, 2017. URL https://www.nytimes.com/es/2017/09/19/espanol/rusia-facebook-perfiles-falsos-elecciones-eeuu.html.

Mónica Redondo. Los bots en twitter influyeron en el resultado del brexit y de las elecciones de ee.uu. *Hipertextual*, 2018. URL https://hipertextual.com/2018/05/bots-twitter-brexit-trump.

Mastodon. https://joinmastodon.org/.

How to install mastodon on debian 10. https://www.howtoforge.com/how-to-install-mastodon-social-network-on-debian-10/.