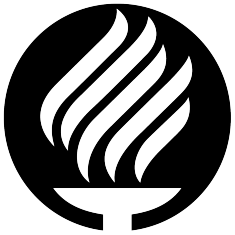
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro
Departamento de Computación y Mecatrónica



Tecnológico de Monterrey

Propuesta de Arquitectura Paralelizable para la generación de Datasets con WebScrapping

por

José Ramón Romero Chávez

Proyecto Integrador para el Desarrollo de Soluciones Empresariales

Ingeniería
en
Sistemas Computacionales

Asesor: Dr. Pedro Peréz Murueta

Santiago de Querétaro, Querétaro, México
27/11/2019

"divide et impera."

Filipo II de Macedonia

Reconocimientos

Agradezco a todas las personas que creyeron, y aún creen en mi, ellos saben quien son.

En esta particular ocasión hago mención a los maestros que me otorgaron su apoyo cuando lo necesité y dejaron una huella en mi durante esta etapa: Fabiola Diaz, Benjamin Valdés, Óscar Hernandez, Rocío Aldeco, Silvana De Gyves, Enrique Calderón y Ruben Stranders.

Una mención en particular a Pedro Pérez, asesor y mentor, gracias por tu paciencia y compartir todo lo que sabes, no se que sería de mi y del Tec sin todo lo que nos has aportado.

Finalmente un agradecimiento a Dios y mis padres, Marisela y Ramón, quienes son mi motivación e inspiración.

En memoria de Edsger Dijkstra, (Róterdam, Países Bajos, 11 de mayo de 1930 - Nuenen, Países Bajos, 6 de agosto de 2002).

Abstract

El reciente interés y desarrollo de herramientas para inteligencia artificial y procesamiento de imágenes (OpenCV, TensorFlow, PyTorch, entre otros), ha motivado la inclusión de este tipo de tecnologías en infinidad de campos de acción.

Sin embargo, una dificultad a la que se enfrentan estos proyectos es la escasez de información con características adecuadas para ser utilizados en los algoritmos más usuales de Machine Learning (Aprendizaje Supervisado).

Durante el desarrollo de esta solución, se explorará el desarrollo de un algoritmo para la recopilación de datasets de imágenes con etiquetado (labelling) .

Índice general

Acrónimos

URL	**U**niform **R**esource **L**ocator
SQL	**S**tructured **Q**uery **L**anguage
HTML	**H**yper**T**ext **M**arkup **L**anguage
TCP	**T**ransmission **C**ontrol Protocol
IP	**I**nternet **P**rotocol

Capítulo 1

Introducción

La búsqueda de imágenes es clave fundamental en cualquier motor de búsqueda (Google, Bing, Yahoo, etc), y nadie a ciencia cierta conoce la cantidad exacta de imágenes en la web [1], ya que conforme el tiempo transcurre, nuevo contenido (no sólo imágenes), es publicado cada segundo [2].

La gran cantidad de información existente ha motivado nuevas e interesantes aplicaciones en infinidad de campos [3], y junto con ellas han aparecido gran variedad de datasets de libre acceso como Google Image Data Sets, Yahoo's Datasets, ImageNet, Kaggle [4], entre muchos otros mas, que en muchas ocasiones, resultan estar limitados.

Como se ha descrito muchas veces [5], la utilidad de un modelo esta directamente relacionada con la calidad y cantidad de la información con la que se alimentan los modelos, en especial al ejecutar algoritmos de aprendizaje supervisado.

Ahora bien en la epoca de la información, donde todo esta conectado a Internet, es difícil concebir la idea de que por falta de información no se puedan desarrollar las ideas de manera adecuada, es por ello que establecer y evaluar una metodología para generación automatizada de datasets, es fundamental para el seguimiento de estos esfuerzos.

MGDrivE

MGDrivE es un marco de trabajo diseñado para servir como campo de prueba en la liberación y evaluación de mosquitos causantes de enfermedades. Estos mosquitos son representados de manera dinámica en un escenario espacial y con interacción con el mismo.

El uso y generación de dataset de imágenes satélites permitirá la mejora e implementación de algoritmos de aprendizaje supervisado dentro de la simulación de la distribución e interacción de los mosquitos con el medio ambiente. [6]

Adicionalmente, los métodos fundamentales y las tecnologías en el área están claramente definidas, tal y como nos lo presenta Gahagan [7] las cuales conforman el núcleo de la investigación computacional y sus derivadas:

- Arquitectura Computacional y diseño: Mejoras en la capacidad y desarrollo de arquitecturas paralelas con capacidades de ejecución determinista.
- Búsqueda, clasificación, predicción y modelado: Progresos en el reconocimiento de patrones, clasificación y herramientas de aproximación, originalmente desde el ámbito de la inteligencia artificial, como árboles de decisión. redes neuronales, algoritmos genéticos entre otros.
- Descubrimiento de Conocimiento: Avances en mecanismos los cuales permitan obtener conocimientos derivados de análisis de datasets.
- Visualización: Avances en la visualización como resultado del análisis de datos, provee nuevas herramientas y acercamientos para ayudar a comprender datasets multidimensionales. Las abstracciones visuales proveen un entendimiento más explícito de los problemas.

Particular mente este ultimo nodo, arquitectura computacional y diseño, es fundamental para el desarrollo y entendimiento de esta tecnología.

Capítulo 2

Estado del Arte

En este capítulo describo conceptos clave para el mejor entendimiento de la solución y trabajos relevantes dentro del área.

Posteriormente una breve mirada a trabajos recientes y relevantes a trabajos relacionados con la elaboración de datasets para finamente realizar una análisis objetivo de las áreas de donde queda trabajo por hacer y contribuir en mayor medida.

2.1. Conocimientos Previos o Contexto del Área

2.1.1. Arquitectura de Software

A través del tiempo, e han ido identificado similitudes que han permitido el descubrimiento y nombramiento de modelos que permiten la resolución de estos mismos problemas.

La arquitectura de software se refiere a las estructuras fundamentales que componen un sistema de software y la interacción que existe entre cada una de estas partes. Es el plano de un sistema, independiente al diseño de las tareas de cada componente [8].

David Garlan y Mary Shaw definen que la arquitectura es un nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema". [9], es por ello que a diferencia de los patrones de diseño, los patrones arquitecturales tienen un dominio

mucho mayor, y enfoques en múltiples áreas, que va desde hardware hasta las implementaciones e código.

Recientemente, los patrones de arquitecturales han sido aplicados de manera exitosa para resolver problemas en el desarrollo de de sistemas de software [10].

2.1.2. Cómputo de Alto Desempeño

Previo a los años 1990, la falta de capacidad y recursos computacionales, restringe el avance de las investigaciones en múltiples campos.

Una vez adentrados en el cómputo de alto desempeño, se encuentran dos paradigmas principales [7]:

Cómputo distribuido: donde los recursos computacionales son distribuidos donde sea que estén disponibles. Esta técnica ha permitido a los investigadores comenzar a atacar y solucionar gran cantidad de problemas en física, biología, entre muchos otros, debido a la facilidad de colaboración interdisciplinaria a pesar de la distancia.

Otro servicio, muy allegado es el de cómputo en la nube (cloud computing), el cual permite el uso de capacidades y recursos computacionales a través de internet sin la necesidad de una gran inversión inicial en hardware, esto gracias a servicios como Amazon EC2.

Cómputo clusterizado: Donde una gran cantidad de procesadores localizados de manera cercana, trabajan en conjunto para formar así una suerte de súper computadora. Este tipo de arquitectura es homóloga a la existente en procesadores multinúcleo con tarjetas de procesamiento gráfico (GPUs).

La meta es la misma, la paralelización de tareas computacionalmente intensivas.

Recordando que siempre es necesario tener en cuenta que no todos los algoritmos y programas se pueden paralelizar, por lo que es necesario que el programador esté al tanto de la relación que existe entre los hilos lógicos y su abstracción relacionada con los núcleos y procesadores, su comunicación. y sincronización, especialmente considerar dependencias, condiciones de carrera, inconsistencias de memoria, “deadlocks” y otros conflictos, para evaluar si la paralelización es una opción de optimización.

2.1.3. Patrón Arquitectónico: Maestro-Esclavo

El patrón de arquitectura Maestro-Esclavo, es un modelo en el cual se describa la interacción donde un elemento del sistema (maestro) tiene poder de manera unidireccional con uno o mas entidades(esclavos)(procesos, threads, objetos). [11]

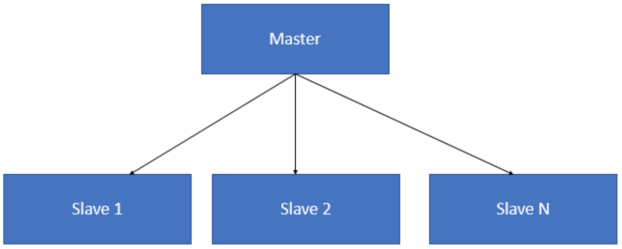


FIGURA 2.1: La arquitectura maestro esclavo es mostrada en este diagrama [12]

El elemento maestro, es el encargado de la distribución de la carga entre los distintos nodos esclavos y al concluir, sumarizar o realizar alguna tarea con el resultado de los esfuerzos realizados por los esclavos. [13]

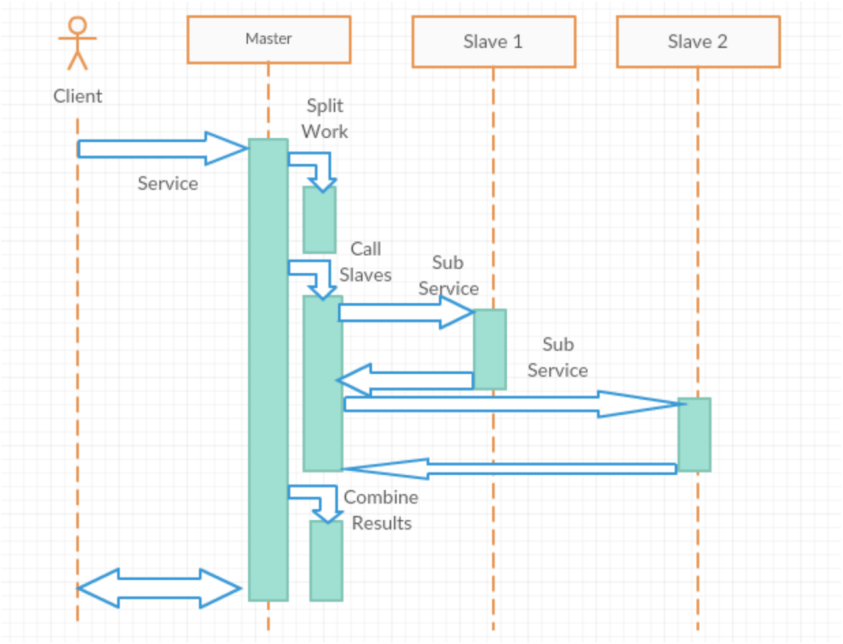


FIGURA 2.2: Pipeline de interacción en Crowdsourcing [A/]

El patrón maestro-esclavo esta basado en el principio de divide y conquista. La coordinación del trabajo es un aspecto separado del trabajo como tal. Cada uno de los elementos esclavos

trabajan en un ambiente de insolación, donde solo tienen comunicación con el nodo maestro pero no entre ellos.

Este patrón es únicamente aplicable si el problema especifico se puede descomponer en partes de funcionalidad igual.

Uno de los principales problemas que recaen en este patrón es la latencia, es decir, la suma de retardos temporales dentro de una red generado por las interacciones definidas dentro de la misma como topología, medio, tamaño de paquete, etc.(Particularmente en sistemas de con procesamiento real esto resulta crítico)

2.1.4. Comunicación por Sockets

De acuerdo a Oracle[14]:

> Un socket es un punto final de un enlace de comunicación bidireccional entre dos programas que se ejecutan en la red. Un socket está vinculado a un número de puerto para que la capa TCP pueda identificar la aplicación a la que están destinados los datos para ser enviados.

Normalmente, un servidor se ejecuta en una computadora específica y tiene un socket vinculado a un número de puerto específico. El servidor solo espera, escuchando el socket para que un cliente haga una solicitud de conexión.

En el lado del cliente: el cliente conoce el nombre de host de la máquina en la que se ejecuta el servidor y el número de puerto en el que el servidor está escuchando. Para realizar una solicitud de conexión, el cliente intenta encontrarse con el servidor en la máquina y el puerto del servidor. El cliente también necesita identificarse con el servidor para que se una a un número de puerto local que utilizará durante esta conexión. Esto generalmente lo asigna el sistema.

Si todo va bien, el servidor acepta la conexión. Una vez aceptado, el servidor obtiene un nuevo socket vinculado al mismo puerto local y también tiene su punto final remoto establecido en la dirección y el puerto del cliente. Necesita un nuevo socket para poder seguir escuchando el socket original para solicitudes de conexión mientras atiende las necesidades del cliente conectado.

En el lado del cliente, si se acepta la conexión, se crea un socket con éxito y el cliente puede usar el socket para comunicarse con el servidor.

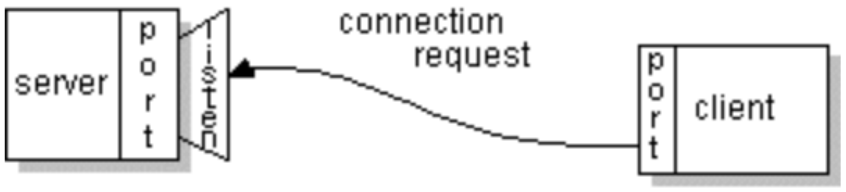


FIGURA 2.3: Solicitud de conexión a servidor

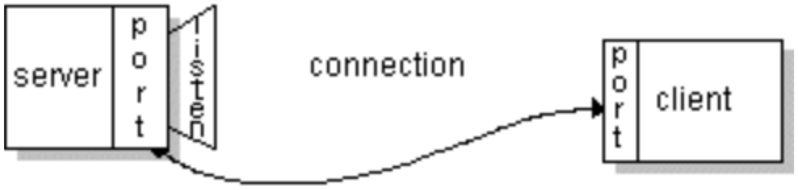


FIGURA 2.4: Conexión establecida entre cliente-servidor

El cliente y el servidor ahora pueden comunicarse escribiendo o leyendo desde sus sockets.

Un punto final es una combinación de una dirección IP y un número de puerto. Cada conexión TCP se puede identificar de forma exclusiva por sus dos puntos finales. De esa manera, puede tener múltiples conexiones entre su host y el servidor.

2.1.5. WebCrawling

Zhang [2] nos ha mostrado como ha sido el proceso de evolución de la búsqueda de imágenes por internet, donde en los años 1990 la recuperación de imágenes se encontraba completamente basada en textos gracias a un manejador de bases de datos relacionales (RDBMS), y su posterior extracción de información e indexado permitió a empresas lanzar sus motores de búsqueda con acceso a millones de resultados a través de Internet.

Un web crawler [15] es un programa que recupera y accede a paginas web de manera recursiva, accediendo a los links que existen dentro de un conjunto inicial conocido como semillas y almacenando el contenido.

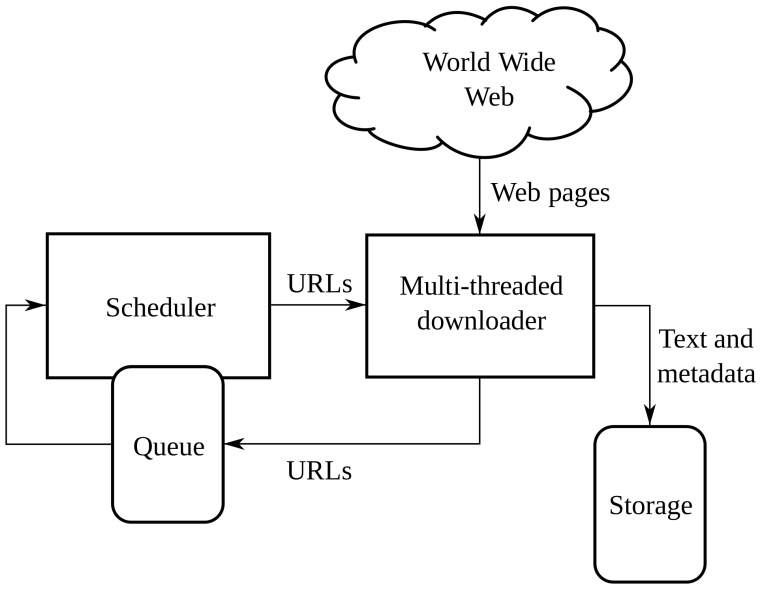


FIGURA 2.5: Arquitectura genérica un WebCrawler [16]

2.1.6. WebScrapping

Un aspecto muy valioso a considerar es que dentro de la web, es que la información siempre viene con metadata cómo: URL, nombres de archivo y demás información circundante a ella que es de utilidad para su proceso de etiquetado y búsqueda [17].

Una variación mas reciente de los WebCrawlers son los WebScrapper, que están enfocados en buscar cierto tipo de información dentro de los archivos recuperados[18].

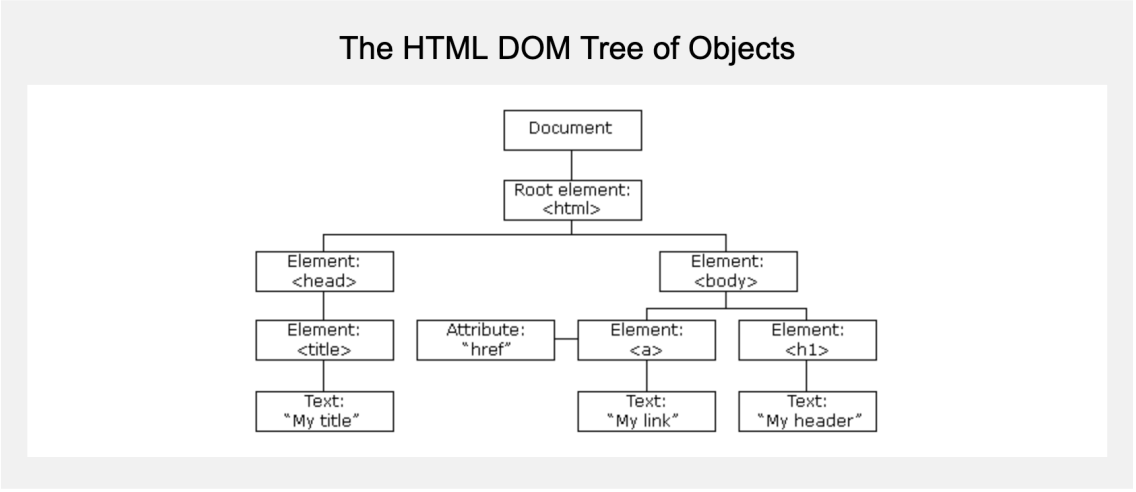


FIGURA 2.6: Árbol de DOMS [19]

El scrapping hace uso de los DOMS que componen un archivo HTML para generar una estructura jerárquica no cíclica (árbol) para la búsqueda de atributos o contenidos de manera

estructurada

2.2. Trabajos Relevantes en Área de Generación de Datasets

Se han liderado gran cantidad de esfuerzos para la creación de algoritmos de generación de datasets haciendo uso de la información existente en la web.

Zhang,[20] desarrolló un algoritmo de generación y etiquetado de personas, haciendo uso de una imagen input, se identificaron similitudes en textos circundantes alrededor de la imagen, dando como resultado el nombre de la persona (label)

Fisher [16] propuso un proceso para la adquisición de dataset etiquetado (labeled) de imágenes haciendo uso de deep learning y algunos humanos en procesos intermedios.

J. Zhang [21] introdujo el DIRS,un método para aplicar la recuperación de imágenes basada en contenido a conjuntos de datos de imágenes masivas. DIRS se implementó en Hadoop utilizando el modelo de computación MapReduce y las imágenes y sus características se almacenaron en bases de datos.

Ahora bien, más cercano al problema de las imágenes satelitales, Gao [22] enfocó su esfuerzo en la creación de "gazetteers" con base a análisis geográficos.

Los "Gazetteers" contienen información con respecto al área analizada, sin embargo la diferencia propuesta fue el uso de un sistema basado en Hadoop para la recolección de data sobre los lugares.

Finalmeent Asmat[23], presenta un esfuerzo de recuperación de imagenes usando Hadoop Apache Nutch, sin embargo dadas las características de los sistemas usado no fue posible ejecutar las tareas de modo no pseudo-distribuido.

2.3. Análisis del estado actual

Tras revisar los trabajos, es claro que la tendencia en todos y escenarios ha sido recaer en una arquitectura paralelizada dada la gran cantidad de información a procesar, adicional a ello el uso de información desde internet o datases preestablecidos ha jugado un rol importante.

Sin embargo, a pesar de los esfuerzos realizado no se ha logrado encontrar un algoritmo que permita la creación de datasets de una manera automatizada y generalizada.

Capítulo 3

Problemática y Contexto específico

El problema del cual se derivó todo el trabajo e investigación realizado fue debido a la necesidad de un dataset etiquetado de imágenes satelitales para el proyecto de simulación MGDrivE.

Tras analizar los trabajos relacionados a la generación de datasets y tendencias actuales, fue posible observar que la falta de datasets estructurados resulta ser una problemática muy común dentro de la investigación[24].

Planteando la siguiente situación:

Dado un dataset de imágenes a procesar, un modelo de aprendizaje lo toma como entrada a su proceso buscando la identificación de características.

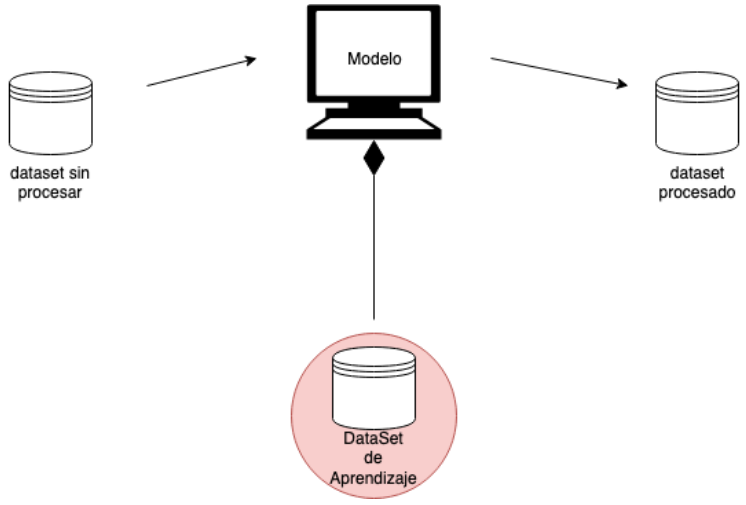


Figura 3.1: Pipeline de procesamiento

Una de las preguntas que surgen es ¿Cómo se ha generado el modelo que procesa el nuevo dataset?.

Independientemente la técnica de machine learning utilizada en el modelo, es necesaria información que sirva para su creación y posterior evaluación, y tal y como fue descrito en la introducción el comportamiento del modelo esta directamente relacionado con la calidad y la cantidad de información con la que es entrenado.

Las tecnologías implementadas hasta el momento hacen uso de múltiples tecnologías, sin embargo ninguna logra automatizar por completo la recolección de información

3.1. Humanos en el Loop

El uso de sistemas como Amazon Mechanical Turk como lo propuso Fisher[17] dentro del pipeline para la generación de datasets, trae consigo muchas implicaciones, siendo la mas importante la varianza que existe entre la percepción de cada uno de los humanos involucrados (sesgo cognitivo), causando así un problema en la calidad de los resultados.

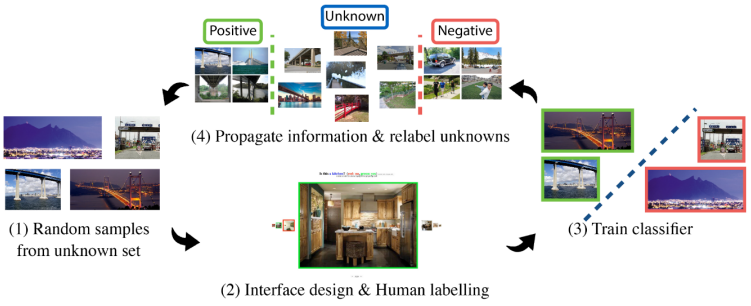


FIGURA 3.2: Interacciona genérica entre Maestros y esclavos [A/]

Adicional a ello la dependencia directa del los servicios de crowdsourcing, aumenta tiempo en pipeline de la generación de datasets.

3.2. Ecosistema Hadoop

El uso y configuración de un sistema basado en Hadoop de manera apropiada (paralelización real) implica el entendimiento de los cuatro componentes principales de Hadoop (HDFS, MapReduce, YARN, and Hadoop Common) y las interacciones y dependencias internas de cada

uno de ellos (Spark, HIVE, HBase, Lucene, Zookeeper, Oozie).[25]

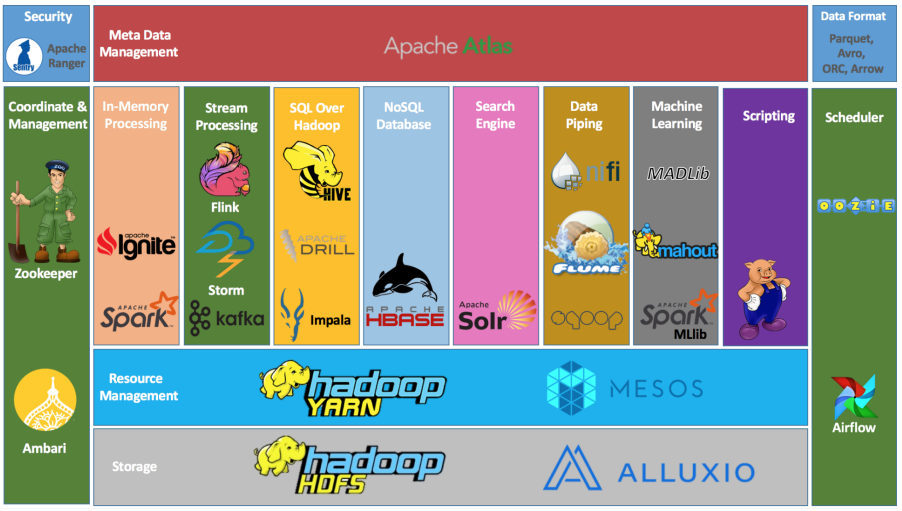


FIGURA 3.3: Ecosistema Hadoop [26]

Las combinaciones y posibilidades son verdaderamente abrumadoras, derivando en una curva de aprendizaje muy pronunciada.

3.3. Apache Nutch

Apache Nutch [27], es el WebCrawler apoyado por Apache Software Foundation, que dentro de su documentación afirma una integración completa con el ecosistema Hadoop. Sin embargo, existen múltiples inconsistencias y ambigüedades dentro de su documentación que dificultan el uso del mismo, esto debido a que por su naturaleza de software libre (FLOSS) no ha recibido la atención adecuada.

FIGURA 3.4: Logotipo Apache Nutch [27]

Es por ello que mi propuesta de solución ante estas problemáticas es una arquitectura paralelizable y con facilidad de abstracción y manipulación que permita generar datasets a partir de webscrapping

Capítulo 4

Desarrollo y propuesta de arquitectura

Con lo anteriormente descrito, entrare de lleno al desarrollo y descripción de esta arquitectura.

4.1. Arquitectura paralelizada

Como ya fue mencionado previamente, una de las características de un buen dataset es el tamaño del mismo, teniendo en consideración y la metodología de "divide y conquista"de derivar un problema en partes más pequeñas, es que se decidió implementar una solución que divida la tarea que implica el webscrapping siguiendo el patrón de arquitectura de maestro-esclavo.

4.1.1. Tecnologías

- Python3: Python3 fue elegido como lenguaje de programacion para desarrollar los componentes de software para la arquitectura presentada debido a su filosofía que favorece a la legibilidad dentro del código tanto para proyectos de pequeñas dimensiones y viceversa [28][29]. las características previamente descritas han favorecido a la creación y desarrollo de los paquetes y utilidades del Python Package Index.

  Los siguientes paquetes son claves para el funcionamiento de la arquitectura propuesta

  - beautifulsoup4: provee métodos para la navegación, búsqueda y alteración de un árbol de parseo.[30]

- requests: Permite enviar solicitudes HTTP/1.1 de una manera simplificada. [31]

- MongoDB: para la gestión y almacenamiento de la información recuperad, se ha decidido usar un esquema NoSQL; esto debido a la naturaleza no estructurada de la información que puede ser recuperada desde internet, adicionalmente, un esquema des este tipo favorece la escalabilidad de el sistema y las complejidades de inserción y recuperación de información son mínimas.[32]

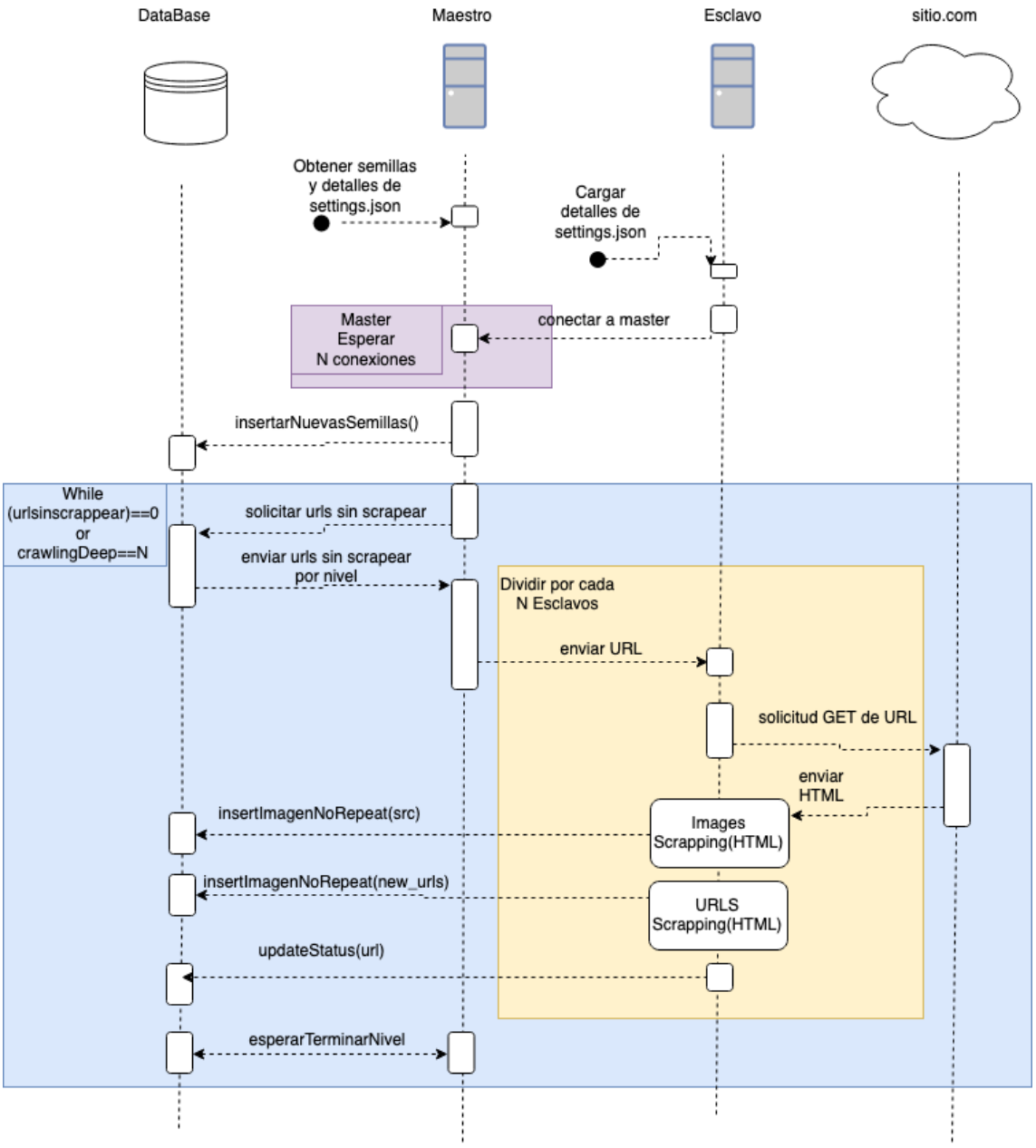


Figura 4.1: Propuesta divide y conquista

4.1.2. Interacción y comunicación

La interacción se puede observar en la figura 4.1 y consta de los siguientes componentes y tareas:

- Base de datos NoSQL: que consta de las colecciones 'urls' e 'imágenes', con limitantes de unicidad.
- Nodo maestro: Divisor de la carga de trabajo (URLS a analizar)
- Nodos esclavos: Solicitar y recuperar HTML del URL recibido, realizar scrapping de URLS e imágenes y actualizar la base de datos.

La comunicación entre cada uno de estas entidades se realiza gracias a comunicación vía sockets (IP, puerto TCP), esta característica permite que cada una de estas instancias puedan estar localizadas en distintas posiciones geográficas.

Las características clave del funcionamiento de esta arquitectura se encuentran definidas dentro del archivo 'settings.json'.

```
{
    "seeds": [
        "http://0.0.0.0:8000"
    ],
    "kwords": [
        "mexican"
    ],
    "master": {
        "ip": "127.0.0.1",
        "port": 7000,
        "db": 7050,
        "id": "master"
    },
    "slaves": 6,
    "depth": 10,
    "force_downlooad":true
}
```

FIGURA 4.2: Archivo de configuración

En el podemos encontrar los siguientes aspectos a configurar.

- Lista de las semillas para iniciar el crawling
- Lista de las palabras clave a buscar dentro del HTML.
- Identificadores de conexión.
- Variables para la ejecución del algoritmo.

4.2. Algoritmo de Scrapping Imágenes

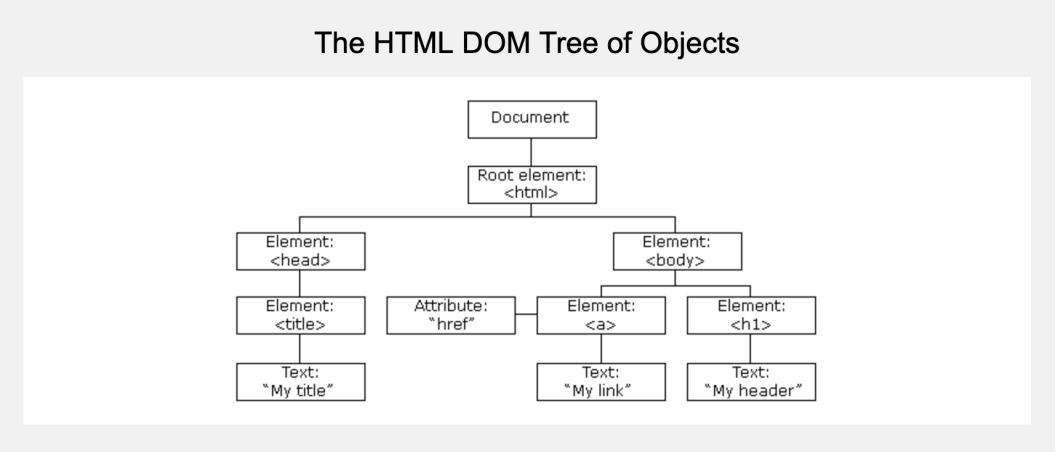


FIGURA 4.3: Árbol de DOMS [19]

Previo al la descripción del algoritmo de identificación y etiquetado de imágenes, cabe recordar que uno de los procesos clave del webscrapping es la generación del árbol de DOMS del HTML e iterar dentro del mismo, para simplificar estas dos tareas se hizo uso de la librería BeautifulSoup y requests.

Algorithm 1: Recuperar Imagen y Tags

Result: Write here the result

KEYWORDS = set retrived from settings;

requestContent = requests.get(url) ;

tree = BeautifulSoup(requestContent, html.parser) ;

allImagesNodes= tree.find(ïmg") ;

tags = [];

foreach $node \in allImagesNodes$ **do**

$surroundingText$=node.parent.text ;

$tags$= $(KEYWORDS \cap surroundingText) \cup (KEYWORDS \cap node.alt) \cup (KEYWORDS \cap node.src$

if *tags* **then**

StoreImg(src,tags)

end

end

La abstracción clave de este algoritmo es que la inserción de la imagen al dataset se ve determinada por la inspección del texto circundante a cada imagen (esto gracias a acceder al

padre y elementos hermanos del nodo imagen), su nombre ('src') y texto alternativo ('alt') en intersección con las palabras claves que se encuentran definidas de dentro de nuestro archivo 'settings.json'.

4.3. Crawling con profundidad limitada

De igual manera que ha sido posible recuperar as imágenes dentro del HTML recuperado, es posible obtener los URL contenidos dentro del mismo, recordando que la estructura genérica es la siguiente:

<a href="newsite.com">Text </ a >

Parece bastante simple la idea de ir buscando recursivamente URLs dentro del contenido un URL, sin embargo, esto se traduce a una complejidad factorial dado un input, debido a la cantidad incierta de derivaciones del mismo . (O(N!))

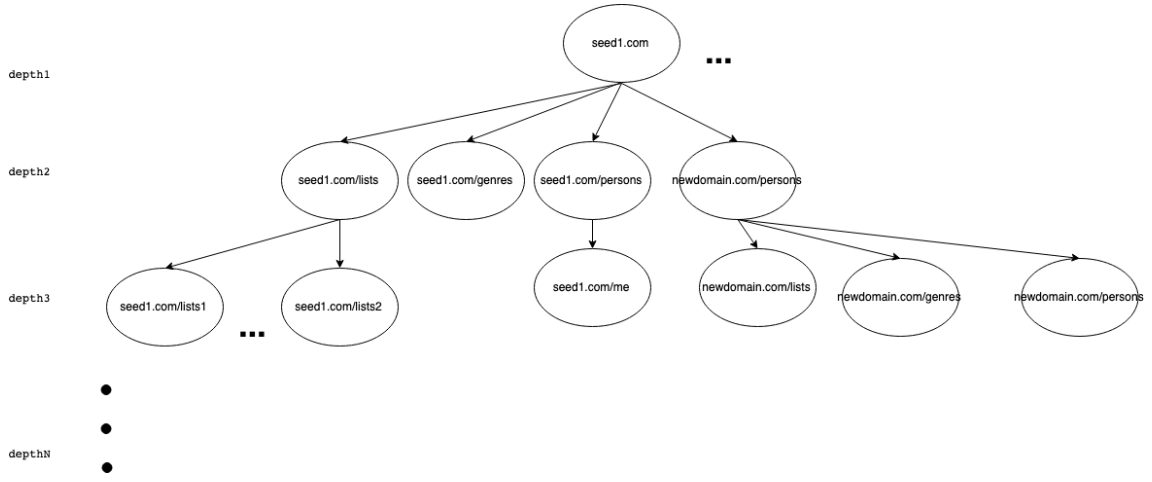


FIGURA 4.4: Expansión de una semilla

Para esta implementación la manera de limitar la expansión del crawling,por medio de la profundidad del árbol que va generando tras cada iteración.

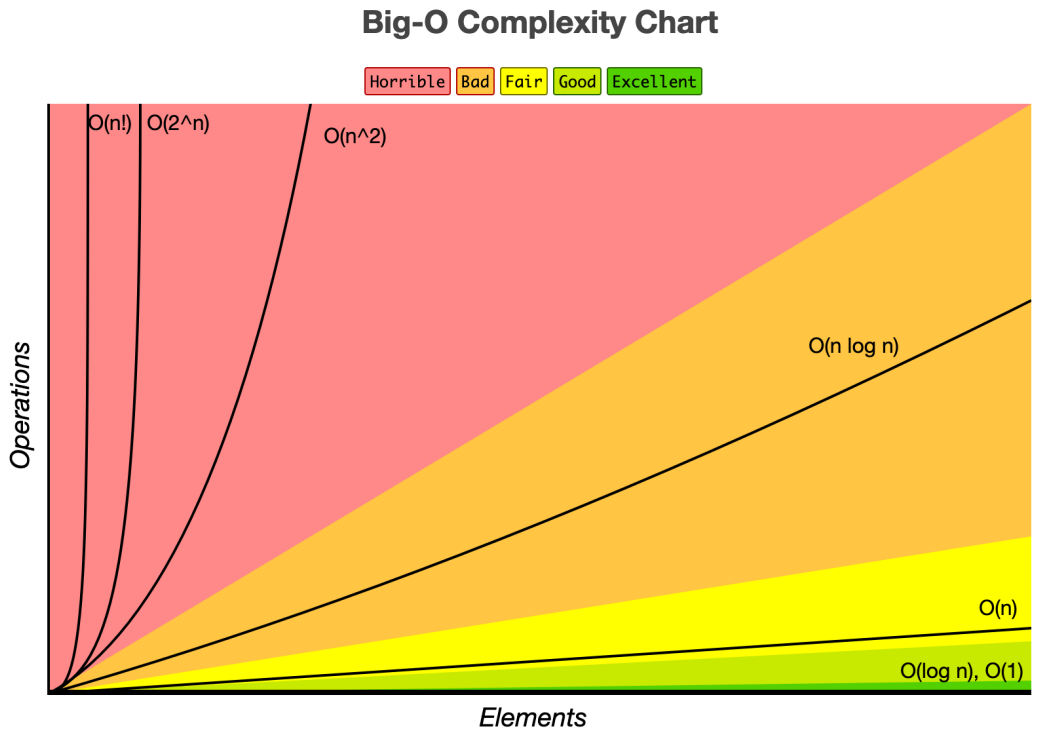


FIGURA 4.5: Complejidad Big-O

Capítulo 5

Metodología de Evaluación

Dada la complejidad e implicaciones que fueron descritas en el capitulo anterior con respecto al la crecimiento del crawling, se decidió probar el algoritmo y arquitectura dentro de un ambiente controlado y estructurado.

Como ya lo mencionamos, nuestra meta es generar un dataset etiquetado a través de webscrapping, es por ello que se realizo el proceso inverso para generar un escenario de pruebas: Generar un servicio web a partir de un dataset de imágenes etiquetadas.

Usando un dataset etiquetado de Kaggle [34], se generaron archivos HTML que cuya trazabilidad es mapeable con una estructura de árbol con una aridad de máxima entre 3-5 dentro de sus subárboles.

El dataset consta de una galería de 8355 imágenes de pinturas de los 50 artistas mas reconocidos de todos los tiempos y su descripción.

El estado del arte muestra que la forma de medir una arquitectura paralelizada es compararla con su implementación secuencial-iterativa (mono-nodo) para obtener el 'SpeedUp', la relación entre los tiempos de ejecución de la arquitectura serial con respecto a la paralela.

Adicional a lo anterior, probaremos el algoritmo para la clasificación de acuerdo a la información del HTML mostrando los aciertos obtenidos con respecto al etiquetado del dataset original.

La implementación permite modo pseudo-distribuido (misma computadora, distintos procesos) o completamente distribuido (distintos equipos).

Estas son las características del los equipos usados y su rol en el las configuraciones de prueba.

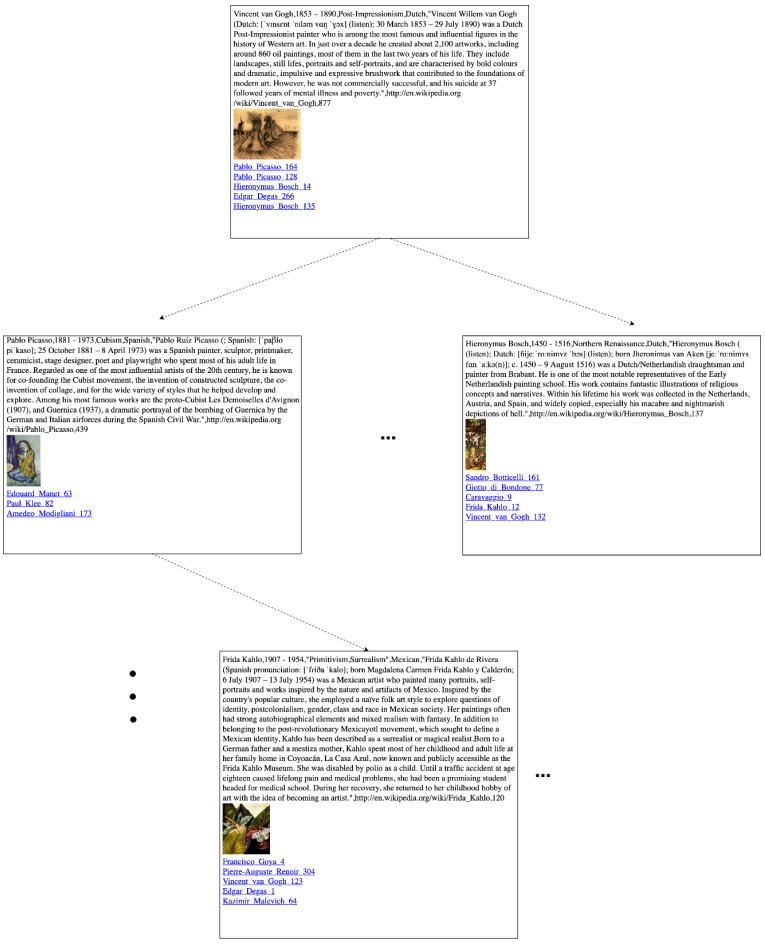


FIGURA 5.1: Servicio web de prueba

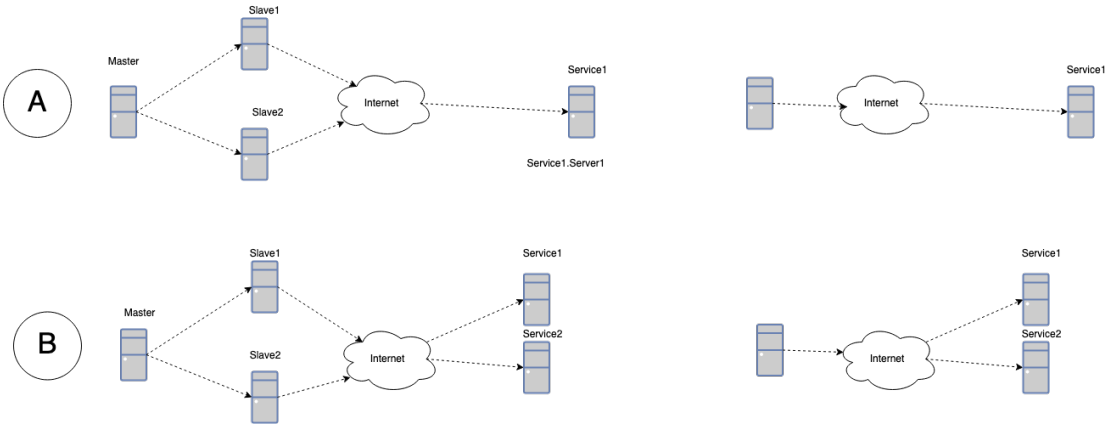


FIGURA 5.2: Configuraciones de prueba

- MacBook Pro (Retina, 13-inch, Early 2015) macOS Mojave 10.14 (18A391) Processor Name: Intel Core i5 o Processor Speed: 2.7 GHz Number of Processors: 1 Total Number of Cores: 2 with hyperthreading o L2 Cache (per Core): 256 KB L3 Cache: 3 MB Memory: 8 GB 1867 MHz DDR3 (Servidor, Master)

- Lenovo C440, Windows 10 , Intel(R) Pentium(R) CPU G2020 @ 2.90GHz, 8 RAM (Master)(2 cores)(Slave)
- HP-qsk5gg2, Windows 10, 64bits, RAM 4GB, AMD A6-5200 APU with Radeon(TM) HD Graphics, 2GHz (Slave)
- Conexion Ethernet CAT5 entre Nodo Maestro, Router y Esclavo Router TL-WR840N V6.20 300Mbps Wireless N Speed
- Conexión Internet 80 Mbps Salida

  La prueba constaba en la recuperación e identificación de imágenes dentro de la galería que cumplan el criterio de ser de autoría de pintores mexicanos.

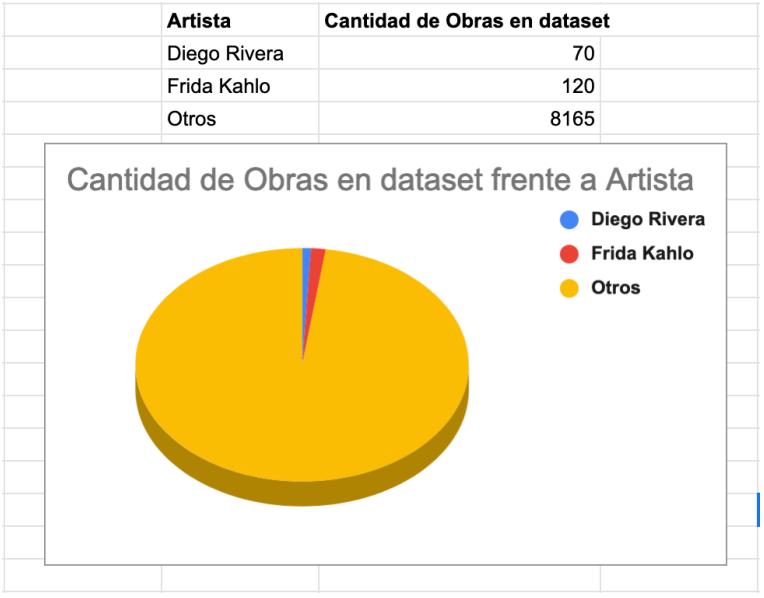


FIGURA 5.3: Obras dentro del set de pruebas

Capítulo 6

Resultados

Al ejecutar el el algoritmo de crawling y scrapping, se logra un recorrido por completo del contenido en las instancias de servidores de prueba.

De 190 imágenes etiquetadas con nombres de artistas de nacionalidad mexicana, 190 fueron recuperadas con éxito, 380 si se cubren 2 instancias del servicio. (100 % de precisión)

Se detecto un 'speedUp' aumento en la velocidad de hasta x1.34

Se realizaron 5 pruebas de las configuraciones descritas en la figura 5.2, adicionalmente se agregaron nodos en modo pseudo-distribuido.

	Tiempo en minutos para terminar	
Procesos	**Un Servicio web**	**Dos Servicios web**
1	2,79	4,58
2	2,51	4,02
4	2,30	3,60
6	2,20	3,40

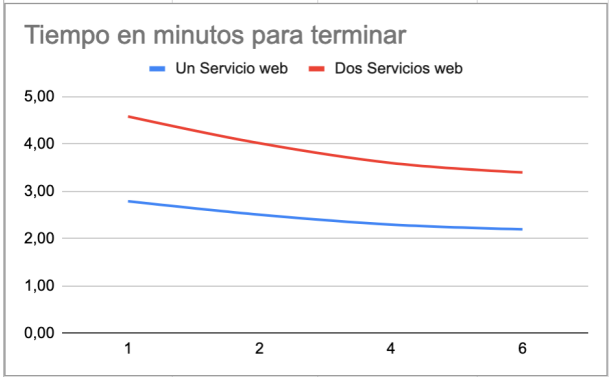


FIGURA 6.1: Variación del tiempo de espera con respecto a los procesos utilizados

Capítulo 7

Conclusiones del Trabajo y trabajo Futuro

7.1. Conclusiones

- Existe una mejora al ejecutar esta tarea en paralelo, sin embargo, los componentes de físicos juegan un papel de suma importancia.
- El etiquetado y recolección con el uso de palabras claves resultó un éxito.
    - El medio y la distancia de comunicación entre cada uno de los componentes (maestro, esclavo, servicios) juega un rol crucial en el comportamiento de este tipo de arquitecturas.
    - **Al ejecutar en modo pseudo-distribuido, se deben tener en consideración las características del equipo como el número de núcleos, al igual que considerar la posibilidad de condiciones de carrera o deadlocks.**
- Es necesario conocer cual es la aplicación y frameworks final, para hacer los ajustes a la configuración (ubicación de base de datos, esquema guardado de las imágenes)

7.2. Trabajo Futuro

- Evaluar la posibilidad de desarrollar e implementar algún algoritmo para la selección 'inteligente' de las imágenes y URL a expandir, algunas alternativas podrían ser el desarrollo

e implementación de heurísticas para el procesamiento de lenguaje natural, y con ello limitar el crecimiento horizontal de la red de links.

- **Desarrollar una arquitectura de mayor tamaño con entidades (computadoras) dedicadas y de características idénticas con el fin de comparar resultados actuales.**
- Evaluar la migración del algoritmo y la arquitectura haciendo uso de algún otro lenguaje de programación.
- Trabajar en la detección de contenido con derechos de autor y el control de solicitudes basándose en el estándar de 'robots.txt'.

Bibliografía

[1] Xin-Jing Wang, Lei Zhang, and Wei-Ying Ma. "Duplicate-search- based image annotation using web-scale data."Proceedings of the IEEE 100.9 (2012): 2705-2721.

[2] Lei Zhang, and Yong Rui. Ïmage search—from thousands to billions in 20 years..ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 9.1s (2013)

[3] D. Shapiro, Çan Artificial Intelligence Generate Corporate Strategy?", Forbes.com, 2019. [Online]. Available: https://www.forbes.com/sites/danielshapiro1/2019/08/19/can-artificial-intelligence-generate-corporate-strategy/ffcceb9559fc. [Accessed: 23- Aug- 2019].

[4] A. Ali, R. Ali, A. Khatak and M. Aslam, "Large Scale Image Dataset Construction Using Distributed Crawling with Hadoop YARN", 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), 2018. Available: 10.1109/scis-isis.2018.00075 [Accessed 24 August 2019].

[5] J. Z. H et al., .A Survey on Cleaning Dirty Data Using Machine Learning Paradigm for Big Data Analytics", Indonesian Journal of Electrical Engineering and Computer Science, vol. 10, no. 3, p. 1234, 2018. Available: 10.11591/ijeecs.v10.i3.pp1234-1243 [Accessed 24 August 2019].

[6] Hector M. Sanchez C., Sean L. Wu, Jared Bennett, and John M. Marshall (2019). MGDrivE: A modular simulation framework for the spread of gene drives through spatially-explicit mosquito populations.

[7] Gahegan, M., 1999. What is geocomputation? Trans. GIS 3, 203–206.

[8] Clements, Paul; Felix Bachmann; Len Bass; David Garlan; James Ivers; Reed Little; Paulo Merson; Robert Nord; Judith Stafford (2010). Documenting Software Architectures: Views and Beyond, Second Edition. Boston: Addison-Wesley.

[9] David Garlan and Mary Shaw. 1994. An Introduction to Software Architecture. Technical Report. Carnegie Mellon Univ., Pittsburgh, PA, USA.

[10] A3- R.G. Lavender, D.C. Schmidt, Active object—An object be-havioral pattern for concurrent programming, in: Proc. 2nd Pat-tern Languages of Programs Conference, Monticello, IL, Sep-tember 6–8, 1995.

[11] S. Baker, Design patterns for network development, UNIXReview (1997) 33–37

[12] What is master/slave? - Definition from WhatIs.com". Description of the Microsoft Computer Browser Service from Microsoft KnowledgeBase Information on Browser Operation from Microsoft KnowledgeBase

[13] M. Richards, Software Architecture Patterns. O'Reilly Media, Inc., 2015.

[14] "10 Common Software Architectural Patterns in a nutshell", Medium, 2019. [Online]. Available: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013. [Accessed: 27- Nov- 2019].

[15] "What Is a Socket? (The Java™ Tutorials ¿Custom Networking ¿All About Sockets)", Docs.oracle.com, 2019. [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html. [Accessed: 27- Nov- 2019].

[16] D. Evans, "WebCrawler", Udacity.com, 2019. [Online]. Available: https://www.youtube.com/watch?v=CDXOcvUNBaA. [Accessed: 27- Nov- 2019].

[17] "Web crawler", En.wikipedia.org, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Web$_crawler$.[$Accessed : 27 - Nov - 2019$].

[18] Fisher, Yu, "LSUN: Construction of a Large-scale ImageDataset using Deep Learning with Humans in the Loop."

[19] Vargiu, E., Urru, M. (2012). Exploiting web scraping in a collaborative filtering- based approach to web advertising. Artificial Intelligence Research, 2(1). doi:10.5430/air.v2n1p44

[20] "What is HTML DOM", W3schools.com, 2019. [Online]. Available: https://www.w3schools.com/whatis/whatis$_h tmldom.asp$.[$Accessed : 27 - Nov - 2019$].

[21] Xiao Zhang, Lei Zhang, Xin-Jing Wang, and Heung-Yeung Shum. "Finding celebrities in billions of web images.ÏEEE Transactions on Multimedia 14.4 (2012): 995-1007.

[22] J. Zhang, X. Liu, J. Luo, and Bo Lang. "Dirs: Distributed image retrieval system based on mapreduce."Pervasive Computing and Applications (ICPCA), 2010 5th International Conference on. IEEE, 2010.

[23] S. Gao, L. Li, W. Li, K. Janowicz, and Y. Zhang. Çonstructing gazetteers from volunteered big geo-data based on Hadoop.Çomputers, Environment and Urban Systems (2014).

[24] A. Ali, R. Ali, A. Khatak and M. Aslam, "Large Scale Image Dataset Construction Using Distributed Crawling with Hadoop YARN", 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), 2018. Available: 10.1109/scis-isis.2018.00075 [Accessed 24 August 2019].

[25] "How Bad Data Practice Is Leading To Bad Research", Forbes.com, 2019. [Online]. Available: https://www.forbes.com/sites/kalevleetaru/2018/02/19/how-bad-data-practice-is-leading-to-bad-research/59ecb1bb1c35. [Accessed: 27- Nov- 2019].

[26] .ªpache Hadoop", Hadoop.apache.org, 2019. [Online]. Available: https://hadoop.apache.org/. [Accessed: 27- Nov- 2019].

[27] "Hadoop Enviroment", Packtpub.com, 2019. [Online]. Available: https://subscription.packtpub.com/book/application$_d evelopment$/9781788995092/1/$ch01lvl1sec14$/$overview-of - the - hadoop - ecosystem$.[$Accessed : 27 - Nov - 2019$].

[28] .ªpache Nutch^{TM} -", Nutch.apache.org, 2019. [Online]. Available: http://nutch.apache.org/. [Accessed: 27- Nov- 2019].

[29] Kuhlman, Dave. .ª Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.

[30] "Python.org", Python.org, 2019. [Online]. Available: https://www.python.org/. [Accessed: 27- Nov- 2019].

[31] L. Richardson, "Beautiful Soup: We called him Tortoise because he taught us.", Crummy.com, 2019. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/. [Accessed: 27- Nov- 2019].

[32] Requests: HTTP for Humans™ — Requests 2.22.0 documentation", 2.python-requests.org, 2019. [Online]. Available: https://2.python-requests.org/en/master/. [Accessed: 27- Nov- 2019].

[33] "When To Use NoSQL Database", MongoDB, 2019. [Online]. Available: https://www.mongodb.com/scale/when-to-use-nosql-database. [Accessed: 27- Nov- 2019].

[34] "Best Artworks of All Time", Kaggle.com, 2019. [Online]. Available: https://www.kaggle.com/ikarus777/best-artworks-of-all-time. [Accessed: 27- Nov- 2019].