Historial de Versiones

Acá se van a detallar los diferentes cambios que se realicen al trabajo luego de cada entrega a fin de tener una mejor trazabilidad.

En el siguiente repositorio es el que contiene el versionado del trabajo:

`https://github.com/ulisescolina/UC-PYLP/tree/master/Integrador`

Versión	Cambios
1.0.0	Primer entrega.

Procesamiento Paralelo y Distribuido

Una respuesta a una creciente demanda de poder de procesamiento

Ramirez Ulises
Universidad Nacional de Misiones
ulisesrcolina@gmail.com

Resumen

En este breve trabajo se busca presentar, describir y experimentar con algunos de los desafíos que acompañan al gran volúmen de datos generados día a día y cómo el procesamiento paralelo y distribuido puede ayudar en el tratamiento de los mismos. Con respecto a la cátedra Paradigmas y Lenguajes de Programación, se puede establecer relación directa con la unidad 1 y 2, estas unidades se enfocan en diferentes problematicas y acercamientos en cuanto a los sistemas de procesamiento paralelo.

Palabras clave: *Multiproceso, programación concurrente, Procesamiento distribuido, Big Data, Apache Hadoop.*

1. Introducción

El manejo de gran cantidad de datos ha ganado atención en los últimos años[1], esto surge a raíz del ritmo acelerado que se generan estos datos y los diferentes escenarios que surgen a la hora de asegurar[2, 3], procesar (lo que se discute en este documento, aunque este a su vez se basa en muchos que van ser citados) y almacenar[4-6] los mismos.

En la actualidad existen mas de 7.8 mil millones de habitantes en el planeta[7], un porcentaje considerable de esos habitantes son usuarios de internet como describe Meeker[8]. Aunque es cierto que la generación de datos viene tomando una tendencia creciente desde hace ya años [9], parte de este crecimiento abrupto de la última decada se puede atribuir, en parte, al gran papel que la tecnología fué adquiriendo con el pasar de los años y su constante evolución: comercio electrónico, publicidad electrónica, pagos electrónicos, etc.[8, 10], las personas conectadas con varios dispositivos (wearables, dispositivos IoT, celulares «especial énfasis», televisores, etc.), sensores produciendo una

ráfaga continua de datos, el avance y la gran disponibilidad de las redes móviles de cara a las redes de siguiente generación (5g)[11], dan como resultado una enorme cantidad de datos.

Esta enorme cantidad de datos acuño un nombre con el pasar de los años, el término es *big data*.

Shönberger y Culkier[12] brindan una síntesis concreta de lo que es big data, sin embargo, como los mismos autores lo mencionan, esto no es nada mas que el inicio.

> "Datos masivos, o cosas que se pueden hacer a gran escala, pero no a una escala inferior, con el fin de extraer nuevas percepciones o crear formas de valor de tal forma que se transformen los mercados."

Transformar estos datos en valor viene ya llevándose a cabo hace décadas, caracterizado con términos que fueron cambiando con el tiempo, conmunmente denominado *inteligencia de negocios* (BI) sobre el cual hay extensa literatura y gran variedad de escenarios en los que se aplicó el mismo con diferentes enfoques[13-19]. El big data, toma estos principios de la inteligencia de negocios y los aplica a una escala mayor.

Aquí, se explora el procesamiento del big data, los desafíos en cuanto a procesamiento que surgieron y siguen surgiendo (aunque también se mencionan brevemente algunas cuestiones relacionadas con los desafíos que intervienten con la seguridad y el almacenamiento de la información), los enfoques a la hora de tratar de encontrar valor a una cantidad de datos masivo teniendo en cuenta diferentes atributos con los que cuentan.

En la sección 2 se describe en mayor profundidad lo que significa el big data y cuales son sus características predominantes, es decir, cuando alguien considera algo como big data, en la sección 3 se introduce lo que es el procesamiento paralelo, se hace especial énfasis en cómo éste permite el tratamiento de los datos haciendo uso eficiente de todos los recursos. A partir de ahí, en la sección 4 se aplica esta idea de utilización eficiente de los recursos y se los traslada a un entorno mayor no limitado a una máquina y con un nivel de acoplamiento inferior.

2. Big Data

Definir *¿Qué es el big data?* es un trabajo no trivial, anteriormente, se brindó un concepto que pretendía dar al lector un primer acercamiento al tema, sin embargo este carece de completitud y especificidad. En la literatura se demuestra que términos difusos como 'grande' hacen aun más difícil el determinar qué es y qué no es big data, esto supone tener que definir variables que se van a tomar en cuenta en diferentes disciplinas con el fin de poder determinar '¿Qué tan grande tiene que ser algo para ser grande?'.

La caracterización principal del big data y por lo tanto la respuesta a la pregunta anterior se tiene en lo que la literatura llama *las 3 Vs*[1][25-27]. Antes

[1]Existen autores que consideran que son 4*V*s[20, 21], 5*V*s [22, 23], e incluso autores que consideran más *V*s [24]. Sin embargo, cabe destacar que estas dos opiniones incluyen las 3*V*s mencionadas anteriormente.

de continuar con las *V*s, es conveniente realizar una aclaración importante: En algunos casos la definición o caracterización aceptada del big data con las *V*s *no es aplicable*, por ejemplo, en el ámbito de la medicina, Baro *et. al* [28] determinan que la cantidad de datos necesarios para ser grande se satisface si se cumple que,

$$\log_{10}(n * p) \geq 7 \tag{1}$$

En donde se tiene que,

- n Cantidad de *individuos estadísticos*
- p Cantidad de variables a ser analizadas dentro del dataset

En este caso específico analiza el significado de ¿Qué es ser big data? para un área en particular, *la medicina*[2], aquí se busca enfocar y delimitar el concepto de grande al área de la salud. El motivo principal detrás de esto es que luego de estudiar los diferentes casos (entendiendo como *casos* las publicaciones en revistas científicas orientadas exclusivamente a la medicina)en un ambiente orientado a la salud, no se satisfacen los criterios de masividad[3], pero aún así existe en el corpus material acerca del big data asociado a la medicina, por lo que se hace útil poseer una definición mas ajustada a dicha disciplina.

2.1. Volúmen

El volúmen hace referencia a la cantidad de datos dentro de un dataset[21, 29], considerado por algunos autores como la característica que brinda los mayores desafíos para el tratamiento con el big data[30].

Se habla de cantidad, y nuevamente se tiene esta idea de *¿Qué tantos datos hacen falta para tener un volúmen lo suficientemente masivo como para ser considerado grande?*

Hacia el 2010, el mundo había creado 1ZB[4] de datos, se menciona esto para tener una idea aproximada de la cantidad de datos manejados hace DIEZ AÑOS ATRAS! (una eternidad en el mundo de las tecnologías de la información), y se estimaba que para el 2020 el mundo habría creado 40ZB[31], sin embargo, para 2018 ya se tenían 33ZB, lo cual llevó al IDC a realizar nuevas consideraciones y se terminó estimando que para 2025 se tendrían 175ZB[32]. Es decir, que en el tramo de 15 años, lo que se consideraba inconmensurable al inicio multiplicaría su tamaño por 175. Hablar de estas magnitudes se va naturalizando a medida que pasa el tiempo, ya en el 2013 se iba haciendo cotidiano hablar de Petabytes (PB) de cara al uso de Exabytes (EB)[30].

2.2. Variedad

Esta es otra de las características principales dentro del big data, y hace referencia a la variedad de datos que conforman un dataset. Este fenómeno tiene lugar por el simple hecho de que existen casi ilimitadas fuentes que pueden

[2]cabe destacar que no es posible asumir que este análisis llevado a cabo por los autores es trasladable a todas las disciplinas

[3]Comparado con la cantidad de consultas que recibe Google, la cantidad de reseñas escritas en Amazon, la cantidad de comentarios que se publican en Twitter, etc.

[4]NOTA: 1ZB equivale a 1×10^9 terabytes, es decir 1000000000TB.

contribuir a un dataset, el cual puede estar compuesto por diferentes tipos y formas de representaciones[30] lo cual hace al big data grande y estos pueden ser resumidos en estructurados, semi-estructurados y no-estructurados[33].

Los datos estructurados y los semi estructurados caben en los Sistemas de Administración de Bases de Datos (DBMS) y/o Data-Warehouses (DW), acá se mencionan los semi estructurados porque estos estan compuestos por archivos tales como XML, en donde se tienen etiquetas para separar diferentes elementos de datos. En cuanto a los datos no estructurados se puede mencionar a la web como una fuente importante de los mismos, un ejemplo primordial aquí es el denominado clickstream[34, 35], aunque menciones honorables van además para los mensajes de textos que se obtienen de las compañias de celulares, datos generados por sensores en dispositivos IoT, etc.

2.3. Velocidad

En la introducción se hizo mención de la utilidad de herramientas para análisis de datos, más específicamente, se habló de Almacenes de Datos o Data Warehouse/Enterprise Data Warehouse (DW/EDW), y su contribución superlativa a la Inteligencia de Negocios (BI) mencionada en la Sección 1.

Este sistema utilizado en BI es una respuesta para múltiples desafíos que se tienen dentro de una organización[36], en este documento se hace énfasis concretamente en uno de los motivos, el *para dar soporte a las desiciones que se toman a nivel estratégico*. Para lograr ese soporte, la organización sigue una metodología que le va a permitir aplicar estas herramientas analiticas con el fín de extraer el valor de los datos. La gente de Datalytics[5], menciona a lo largo de varias charlas que componen el ciclo de "DataSchool", la importancia de estas arquitecturas clásicas, sin embargo, el acercamiento clásico a la creciente cantidad de datos brinda más desafíos, particularmente el que más resuena en el contexto de este documento es el hecho de que esta arquitectura tradicional es *lenta para reaccionar*[37].

El concepto de velocidad es esto, y es tan importante como los que se hablaron anteriormente. La velocidad a la que se pueda realizar un análisis concreto y tomar decisiones basadas en los resultados es clave. El análisis de un fragmento de información acerca de la competencia, datos estadísticos realizados por algún organismo ajeno a la organización que tiene una periodicidad anual (ej: INDEC), etc. posee una gran potencialidad, sin embargo, si no se analiza oportunamente, el valor que pueda existir en esa información se pierde.

3. Procesamiento Paralelo

Diversos autores[38, 39] describen en profundidad diferentes modelos que extienden a la arquitectura tradicional de Von Neumann para lograr paralelización[6]. Esta se logra mediante la separación de una tarea T, en subtareas

[5]https://www.datalytics.com/

[6]No es el objetivo de este documento dar una descripción minuciosa de los diferentes modelos teóricos de paralelización.

$T_1, T_2, \ldots, T_n$ (Diferentes autores adoptan este concepto con el término de *pipelining* [39, 40]). Luego para lograr el cometido, más de una de las T_n tareas debe realizarse al mismo tiempo, sin malinterpretar la realización de una tarea de manera muy velóz con realizarla al mismo tiempo que otras [40].

Las diferentes soluciones para procesamiento paralelo se desenvolvieron hasta llegar a presentar uno de los siguientes tres tipos de paralelismo[39]:

- Los **Sistemas de Memoria Compartida** que se componen de múltiples unidades de procesamiento unidas a una memoria.
- Los **Sistemas Distribuidos** que se componen de equipos con sus propias unidades de procesamiento y memoria, comunicados a traves de una conexión de red de alta velocidad. Este es el caso que nos compete en este documento y es tratado en la Sección 4.
- Las **Unidades de Procesamiento Gráfico** utilizadas como co-procesadores por su capacidad de paralelizar grandes cantidades de operaciones.

Diferentes enfoques existen a la hora de aplicar el pipelining mencionado anteriormente, y estos no son recientes sino que datan desde la década de 1990. Con el pasar del tiempo se fueron creando más enfoques y especializando los casos generales que se tenían en un principio. Ejemplo de algunas especializaciones son los enfoques que se ven en la cátedra de Paradigmas y Lenguajes de Programación[41], en donde se denomina a este problema *Descomposicion*, y de ahí surgen las diferentes especializaciones: Dominio, Funcion, Recursiva, Mixta.

Para este documento solamente se tienen en cuenta dos de las 3 que se proponen por Fountain [42]: Paralelización de Datos y Paralelización de Funciones.

La paralelización de datos y la paralelización de funciones puede encontrarse en diferentes niveles de la arquitectura del computador, algunos ejemplos van desde, el procesador como lo describe Hyde [42], pasando por el compilador [43], llegando a la aplicación [44], etc. Este documento cubre únicamente a casos que son aplicados a nivel de aplicación.

3.1. Paralelización de Datos

> La idea principal detrás de este enfoque a la hora de aplicar el *pipelining* consiste en que un programa secuencial puede ser transformado a un programa paralelo, realizando ejecuciones de copias idénticas de dicho programa como tareas separadas, a las cuales se les brinda simplemente parte de los datos iniciales [45].

En la figura 1 se presenta una aplicación que está corriendo sobre un núcleo aleatorio en un ordenador. En este ejemplo no se tiene presente la paralelización, D representa los datos sobre los cuales está trabajando la *Aplicación*.

En la figura 2 se presenta la misma aplicación (esta vez se la denomina *app*). En este ejemplo los datos representados por D anteriomente, se dividien en D_n secciones más pequeñas que son distribuidas a distintos núcleos de procesamiento [7]. Una copia idéntica de la aplicación (Figura 1) se encuentra

[7] en el ejemplo se utilizan 4, aunque pueden ser más núcleos dentro de un procesador

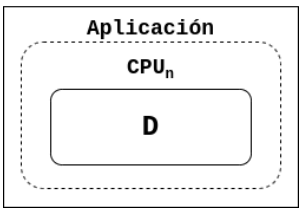

Figura 1: Aplicación sin paralelización

corriendo en cada uno de los cuatro núcleos del ordenador, cada una de estas copias idénticas se encuentra trabajando con datos distintos (ya que a cada una de las copias le toca una sección D_n diferente).

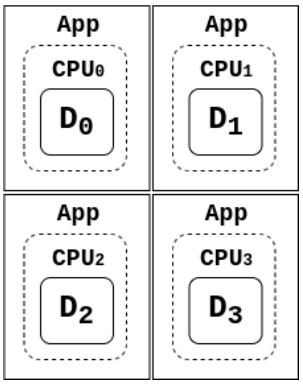

Figura 2: Aplicación con paralelización de datos

Una interpretación alternativa se puede ver en la figura 3, este gráfíco se puede leer como "Una aplicación que se encuentra corriendo sobre distintos nucleos, con distintos fragmentos de los datos originales"

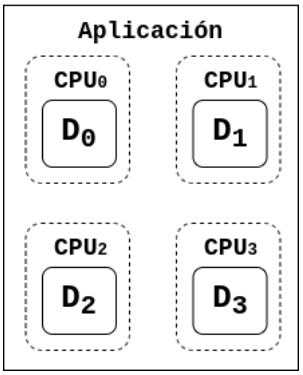

Figura 3: Aplicación con paralelización de datos - Representación alternativa

En la bibliografía existen implementaciones en diferentes sub áreas de las Ciencias de la Computación con este enfoque presente, el procesamiento de imágenes [46], el manejo de bases de datos [47] y la inteligencia artificial [48] por nombrar algunos ejemplos.

Para un ejemplo simple y concreto se propone lo siguiente:

Se presenta la tarea de encontrar el valor mínimo dentro de un arreglo $A = [a_0, a_1, \ldots, a_{n-2}, a_{n-1}]$.

Se puede proceder sin aplicar paralelización de datos, y realizar una aplicación que recorra los n elementos del arreglo, realice las comparaciones necesarias y devuelva el valor mínimo, o se puede optar por la opción con paralelización de datos, una solución con este enfoque será fraccionar el arreglo con A en, por ejemplo, 2 partes A_1 y A_2. Luego se procede a distribuir estas dos partes A_1 y A_2 a diferentes instancias de la aplicación que calcula el mínimo, de esta manera se tendrán dos núcleos de procesamiento que trabajarán en un problema de menor tamaño (ya que cada uno va a trabajar únicamente con el 50 % de los datos), y al final solamente hará falta realizar una comparación entre los mínimos que encuentren las dos instancias de la aplicación.

Cabe destacar que algunos problemas pueden surgir a la hora de particionar datos si esto se hace sin nignun análisis, un ejemplo de esto puede ser considerado en el escenario en donde ocurre la división de una imagen en partes más pequeñas para su análisis en forma paralela [49]. También se hace presente la limitación que impone la ley de Amdhal[8].

3.2. Paralelización de Funciones

Este enfoque a la paralelización toma otro camino a la hora de aplicar el *pipelining*, este acercamiento no fracciona los datos sino que transforma la tarea a un gráfo de dependencias en donde cada nodo corresponde a una operación a ser realizada para cumplir la tarea. Y lo que trata de hacer es ejecutar la mayor cantidad de tareas al mismo tiempo [51-54].

Esta idea de distribuir los datos a diferentes unidades funcionales que se encarguen de realizar una tarea sobre los datos mencionados se presenta en la figura 4.

Aquí se puede observar que lo que se subdivide no son los datos, sino que lo que se está subdividiendo es la funcionalidad que compone a la tarea T, aquí cada o_n es una operación claramente delimitada dentro de la aplicación, y D representa a los datos con los que se va a trabajar.

Un ejemplo concreto para este enfoque se presenta a continuación:

Dada una imagen, se presenta la tarea de determinar si en dicha imagen se encuentra presente algún rostro, vehículo, gato o planta.

Por lo que se puede decir que la tarea T se compone de la siguiente manera

$$T = [o_0, o_1, o_2, o_3] \tag{2}$$

En donde se tiene que,

[8] "El incremento máximo de velocidad (la cantidad máxima de procesadores que pueden ser utilizados de manera efectiva) es la inversa de la fracción de tiempo que la tarea toma para finalizar en un solo hilo" [50]

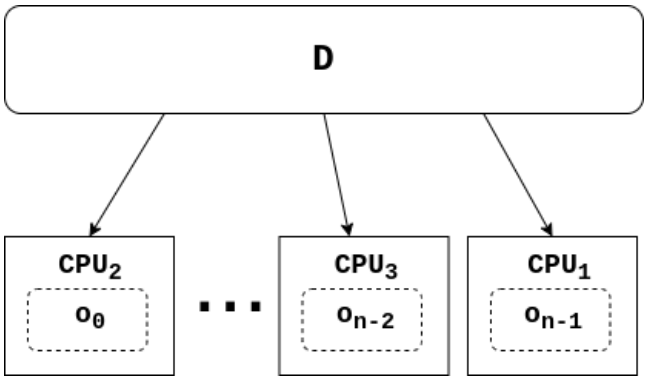

Figura 4: Aplicación con paralelización de funciones

o_0 Buscar rostro
o_1 Buscar vehículo
o_2 Buscar planta
o_3 Buscar gato

Cada o_n en la ecuación 2 representa un nodo del gráfo de mencionado anteriormente. Ahora toca analizar y determinar cuales de las o_n operaciones se pueden realizar al mismo tiempo.

Una cuestion esencial en este ejemplo es identificar el hecho de que ninguna operación o_n depende de alguna operación. Esto es visible a la hora de realizar diversas preguntas con respecto al contexto,

- ¿Es necesario determinar si existe o no un rostro para poder iniciar el análisis y determinar si la imagen contiene un vehiculo, gato o planta? No, entonces, *Buscar rostro* es independiente a las demás operaciones.
- ¿Es necesario determinar si existe un vehiculo en la imagen para poder iniciar el análisis y determinar si la imagen contiene un rostro, gato o planta? No, entonces, *Buscar vehículo* es independiente a las demás operaciones.

El análisis anterior debe continuar hasta que se analicen las o_n operaciones. Habrán veces en donde dos o más operaciones no podrán ser ejecutadas de manera paralela, esto puede darse porque una (también pueden ser varias) de ellas depende de los resultados que se obtengan de otra u otras operaciones.

En el caso del ejemplo para la sección, esta apreciación y el análisis que se realizó permite definir que todas las operaciones que componen a la tarea T se pueden ejecutar de manera paralela por el mismo dato.

4. Procesamiento Distribuido

Este tiene muchas similitudes con los sistemas de Procesamiento Paralelo, es más, es válido decir que los Sistemas Paralelos y los Sistemas Distribuidos son un entrelazamiento de redes, que tienen la caracteristica de ser continuas[9],

[9] o un continuum de redes. `https://dictionary.cambridge.org/es/diccionario/ingles/continuum`

lo que significa que el parámetro que determina si el sistema es distribuido es el promedio de distancias entre los nodos de procesamiento [40]. El desafío que propone Hyde, y también diferencia a los dos sistemas es el de poder determinar el estado exacto de todo el sistema. Este es un desafío no-trivial, ya que dicho estado no puede ser computado instantaneamente por el hecho de que cada operación dentro de la red podría requerir travesías por la misma.

Esta caracteristica mencionada hace que sea muy complicado establecer una unica definición de que significa un sistema distribuido. Para el propósito del documento se hace útil una definición brindada por Tanenbaum *et. al.* [55], la cual sugiere que,

> "Un sistema distribuido es una colección de computadoras independientes que parecen ser un único sistema coherente ante los usuarios."

Aquí hay dos cuestiones extremadamente importantes que son de interés para el documento. La primera, se habla de una *colección de computadoras* y se hace énfasis en el hecho de que estos ordenadores son independientes, lo cual llevan a otro aspecto a considerar para esta colección de máquinas, el hecho de que son *fácilmente escalables* y *tolerante a fallos*. La segunda, se habla de los usuarios ya sea que se esté hablando de *personas o programas* y como para estos es transparente el hecho de que están lidiando con un sistma distribuido, es decir, estos tienen la sensación de utilizar un sistema compuesto por un único ordenador.

Además de estas dos cuestiones que se hacen presentes en la definición presentada, existen otros aspectos dentro de los sistemas distribuidos que son importantes para el estudio de los mismos en profundidad, sin embargo, no es el objetivo de este documento estudiar los sistemas distribuidos en profundidad, a continuación se presenta un punto de referencia para que el lector pueda seguir estudiando a los sistemas distribuidos y sus diferentes aspectos, características y desafíos los cuales no son pocos [56-58].

Referencias para el estudio de Sistemas Distribuidos

[56] Andrew S. Tanenbaum. *Distributed Operating Systems*. USA: Pearson Prentice Hall, 1994. ISBN: 0-13-219908-4.

[57] Andrew S. Tanenbaum y Maaren van Steen. *Distributed System Principles and Paradigms*. USA: Pearson Prentice Hall, 2007. ISBN: 0-13-239227-5.

[58] Abraham Siberschatz, Peter Baer Galvin y Greg Gagne. «Distributed Systems». En: *Operating Systems Concepts*. Ninth Edition. USA: Wiley, 2013. Cap. 17, págs. 741-772. ISBN: 978-1-118-06333-0.

5. MapReduce

En las secciones anteriores, se habló de Procesamiento Paralelo y Distribuido, se mencionaron algunas características (particularmente las más afínes a

este documento), en esta sección, se busca aplicar las cuestiones que se mencionaron con anterioridad, y empezamos a hablar del núcleo del documento *MapReduce*.

MapReduce (MR), es un modelo de programación; una forma de hacer las cosas; un paradigma. Este modelo fue introducido por Google en la primer mitad del 2000 [59] y la implementación del mismo se asocia con grandes datasets y es aplicable a una gran cantidad de tareas del mundo real [60]. En esta sección se avanza sobre una introducción acerca del modelo que presenta MapReduce y luego se aplicarán los conceptos en una de las varias herramientas que aplican el modelo en cuestion, la herramienta a la que se hacer referencia es un Framework para Big Data, su nombre es *Hadoop*.

5.1. Funciones `Map` y `Reduce`

Un vistazo general: en este modelo los usuarios que implementan MR definen dos funciones principales, una función *Map* y una función *Reduce*, el sistema que da soporte a la aplicación implementada se encarga de paralelizar la ejecución de la misma a lo largo y ancho de un sistema distribuido (cluster) de ordenadores, este cluster cuenta con las características de los sistemas distribuidos mencionados en la sección 4. Sistemas como estos permitían que Google (en el 2008) pueda procesar 20PB por día [60].

Las implementaciones de diferentes cargas de trabajo llevadas a cabo en el 2004 por Google con este nuevo modelo de programación sentaba sus bases sobre un desarrollo anterior extremadamente importante, el Sistema de Archivos de Google (GFS, por sus siglas en inglés) [61], el cual, en resumidas cuentas, es un sistema de archivos distribuidos (lo cual está en línea con lo presentado anteriormente con respecto a la "paralelización de datos").

5.1.1. Map

Overview Esta función `map`, se distribuye automáticamente sobre el sistema distribuido, y puede trabajar ya sobre *chunks* (partición) de datos que se encuentran dentro de los diferentes nodos, a los cuales se les distribuye el procesamiento de dicha función.

1. **fork:** El sistema distribuido se encarga de realizar la replicación de la aplicación del usuario sobre el conjunto de máquinas, esta aplicación se divide en copias que son *workers* y una especial que se denomina *master*[10]. Este último es el encargado de dividir las tareas a las copias que son workers.
2. **Asig. Map:** El *master* asigna las tareas a los workers. Los cuales, cabe aclarar, trabajan sobre un o varios chunks de datos locales al nodo.
3. **Lectura:** Un *worker* al que se le asigna una tarea `map`, lee los contenidos de los *archivos de entrada*, interpreta estos datos en formato clave-valor y se encarga de distribuir estos pares a la función map. Los valores que se produzcan por la función *map*, se almacenan en un buffer de memoria.
4. **Escritura Local:** Periodicamente, los pares que se almacenan en el buffer de memoria, pasan a ser escritos al disco de manera particionada. Las

[10] Esto es análogo a lo que se estudia en la cátedra con OpenMPI.

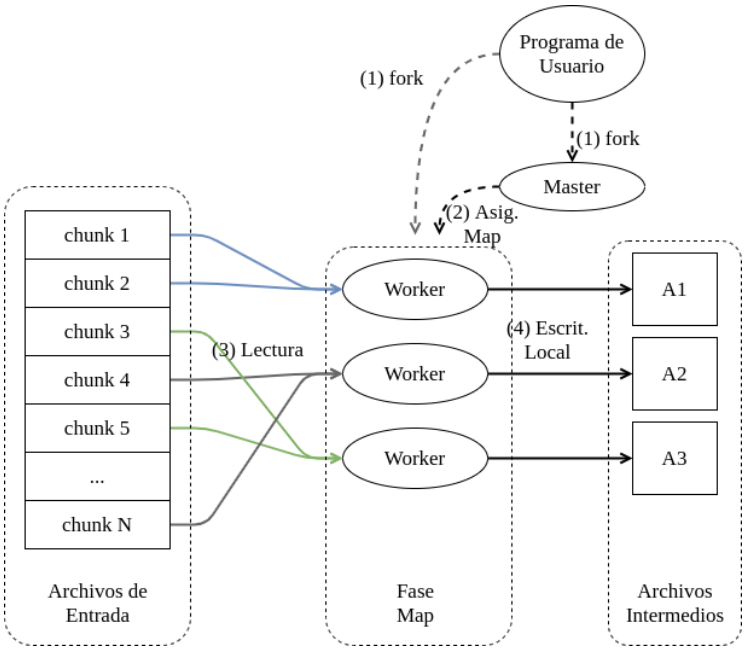

Figura 5: MapReduce: Fase `map`

ubicaciones de las particiones se comunican al *master* el cual será responsable más adelante de distribuir esta info. a los *workers* que son parte del *reduce*.

5.1.2. Reduce

Overview Esta función, al igual que los *mappers* se distribuye automáticamente a través del sistema distribuido y trabaja sobre los archivos intermedios que resultan de los mencionados *mappers*, los *workers* que se encargan de realizar la etapa reduce leen estos archivos desde los nodos de los *mappers* y se encargan de procesar los datos asociados a cada clave-valor en orden.

Las etapas que son llevadas a cabo en la fase de *reduce*, se pueden ver en la figura 6.

2. **Asig. Reduce:** El *master* se encarga de distribuir la carga de reducer información correspondiente a la ubicación de los archivos intermedios a los *workers* que trabajan como reducers.
5. **Lectura Remota:** Al recibir la información del *master* acerca de la ubicación de los *archivos intermedios*, los *workers* proceden a la lectura de los mismos a través de RPC. El worker ahora procede a realizar las tareas de *reduce*.
6. **Escritura:** Luego de haber procesado los datos de las claves que le fueron asignadas al *reducer*, este pasa a guardar los resultados a los archivos de salida *S1* y *S2*.

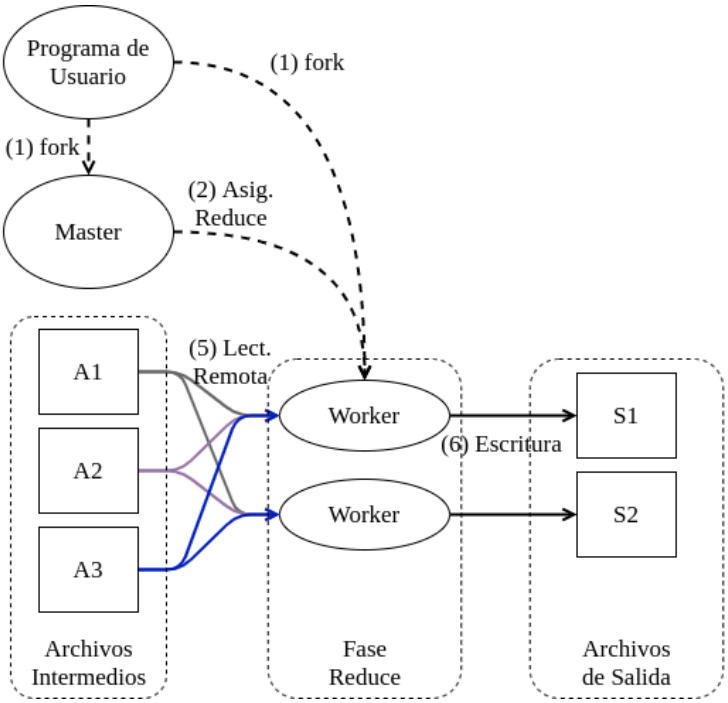

Figura 6: MapReduce: Fase `reduce`

6. Hadoop

Introducción

Apache Hadoop es una herramienta que surge luego de dos grandes contribuciones de las que se estuvieron hablando en las secciones anteriores, estas dos son, el modelo de programación MapReduce [59, 60], y GFS [61].

Este es un framework para Big Data implementado en Java, esta implementación consiste de dos capas principales, la primer capa es para almacenamiento de datos, esta capa es un sistema de archivos distribuidos que está basado en GFS, que se denomina Hadoop File System (HDFS), la segunda capa, ataca el problema del procesamiento de los datos y se denomina *Hadoop MapReduce Framework* [62].

Este framework trabaja exclusivamente sobre pares clave valor («`K, V`»), se puede resumir este funcionamiento de la siguiente manera []:

`input` <k1, k2>→ `mapper` → <k2, v2>→ `reducer` → <k3, v3>

Implementación: Mapper y Reducer

Para demostrar de manera menos abstracta y teórica el funcionamiento del framework, ahora se procede a la implementación de un *mapper* y un *reducer* extremadamente sencillos. El propósito del código brindado en este ejemplo es el de contar la cantidad de palabras en diferentes archivos distribuidos en el HDFS. En los listings 1 y 2 se tiene el código respectivo para cada uno de estos.

La implementación completa (no solamente del mapper y el reducer) se puede encontrar en el Anexo: **Implementación sobre HDFS**.

```
public static class TokenizerMapper
  extends Mapper<Object, Text, Text, IntWritable>{

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(
      Object key,
      Text value,
      Context context
      ) throws IOException, InterruptedException {

    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
    }
  }
}
```

Listing 1: Hadoop MapReduce: Mapper

Lo primero a notar con la implementación de la clase creada, es el hecho de que extiende de `Mapper` [63].

```
extends Mapper<Object, Text, Text, IntWritable>
```

La clase Mapper abstrae dentro del framework la distribución de tareas a los diferentes nodos en la etapa *Map*, esta clase se instancia una vez por cada *InputSplit*[11] [12].

Lo siguiente a prestar atención, es la reimplementación del método `map`, este método se aplica a cada registro dentro del InputSplit, la funcionalidad por defecto es devolver el mismo registro.

```
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {
  StringTokenizer itr = new StringTokenizer(value.toString());
  while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
  }
}
```

[11] Representa el/los datos a ser procesados por un mapper individual

[12] https://hadoop.apache.org/docs/r2.7.4/api/org/apache/hadoop/mapreduce/InputSplit.html

Los primeros dos parametros reiteran lo mencionado anteriormente, el uso de clave-valor `(K, V)`. Aquí, `Object key` representa a la clave del InputSplit que le toca al mapper instanciado, y de la misma manera, `Text value` representa al dato del InputSplit asociado al mapper. Lo que queda pendiente es el `Context context`, este permite la comunicación con el resto del *sistema distribuido*, este objeto contiene información referente a la configuarción del trabajo actual, reporte de progreso, mensajes de estado a nivel de aplicación, heartbeats.

La información contenida en `value` se transforma en un iterador, que contiene en cada uno de sus elementos a una palabra (para este caso particular), al recorrer este iterador, se escribe otro par `(K2, V2)`, la clave pasa a ser la palabra, y el contador pasa a ser un numero (1), en un ejemplo concreto esto significa que, para el par `(hadoop, 1)` la palabra "hadoop", tiene 1 repetición. Al terminar el recorrido se finaliza con la "Emisión" de estos pares (K2, V2) intermedios. Ahora la etapa *Reduce* puede acceder a estos pares (K2, V2).

```
1  public static class IntSumReducer
2    extends Reducer<Text,IntWritable,Text,IntWritable> {
3
4    private IntWritable result = new IntWritable();
5
6    public void reduce(
7        Text key,
8        Iterable<IntWritable> values,
9        Context context
10       ) throws IOException, InterruptedException {
11
12     int sum = 0;
13     for (IntWritable val : values) {
14       sum += val.get();
15     }
16     result.set(sum);
17     context.write(key, result);
18   }
19 }
```

Listing 2: Hadoop MapReduce: Reducer

Al igual que en el mapper, acá se inicia prestando atención a la definición de la clase *Reducer*, en este caso.

```
extends Reducer<Text,IntWritable,Text,IntWritable> {
```

Al igual que la etapa anterior, el reducer, acepta pares (K2,V2). La diferencia es que estos pares, son el resultado de las operaciones realizadas por el *Mapper* en el listing 1.

```
public void reduce(Text key, Iterable<IntWritable> values, Context context)
    throws IOException, InterruptedException {
```

```
  int sum = 0;
  for (IntWritable val : values) {
    sum += val.get();
  }
  result.set(sum);
  context.write(key, result);
}
```

Nuevamente, aquí se cuenta con la clave `Text key` y el valor `Iterable<IntWritable>values`, este último, posee todas las instancias de una clave específica. Para ilustrar este contenido de mejor manera, se presenta un ejemplo, aquí se tiene el par,

```
(K, V) = (‘‘hadoop’’, Iterable [1, 1, 1, 1, 1])
```

Descomponiendo el ejemplo anterior se tiene lo siguiente,

```
(‘‘hadoop’’, 1)
(‘‘hadoop’’, 1)
(‘‘hadoop’’, 1)
(‘‘hadoop’’, 1)
```

Estos múltiples pares (K, V) resultaron de la etapa mapper (que se recuerda que pueden estar en *archivos intermedios* separados en distintos nodos), que se unen en la etapa número 5, explicada en la subsección 5.1.2 *Reduce*.

Resumiendo el significado puntual de lo visto en términos simples es,

Iterable<IntWritable>values = **Todos los valores con la clave “hadoop”**

Recordando que los valores son siempre 1, estos son siempre el mismo valor por el problema que se está atacando con el mapper en este ejemplo particular, no necesariamente son **siempre** 1 en todos los problemas.

7. Conclusiones y Futuros trabajos

El ecosistema alrededor de Hadoop es grande, con proyectos dentro de la incubadora de Apache que contribuyen como pequeños bloques de funcionalidad que agregan a lo desarrollado aquí, algunos proyectos de trascendencia dentro de este ecosistema son Spark, Hive, Pig, HBase, Yarn, ZooKeeper, etc.

Este framework crece día a día, por lo que se aclara que este trabajo no es una revisión profunda del software en cuestión, ni tampoco se apuntaba a serlo, lo presentado es un pequeño vistazo, un vistazo que sirve para poner en perspectiva lo que se puede lograr, estableciendo paralelismos con lo visto en la cátedra en cuanto al procesamiento paralelo y distribuido.

En los experimentos a lo largo del articulo no se demuestra la utildad en un alto volúmen de datos, lo cual `sí` era el punto para este trabajo, el brindar un vistazo de cómo el procesamiento distribuido y paralelo, teniendo

en cuenta estos enfoques de paralelización de datos y funciones, permite que las operaciones sobre grandes volúmenes de datos sean capaces de escalar con el agregado de mas nodos al cluster (aunque no siempre será la respuesta a un problema, siendo en algunas partes su uso es contraproducente, y es notado en la literatura acerca del tema), sin embargo, estos experimentos que demuestran esta capacidad mencionada pueden ser revisados en la bibliografía listada. Los impedimentos que actuaron como barrera para la inclusión estos experimentos a una escala mayor en el presente trabajo tienen que ver con la falta de recursos.

Con la falta de recursos en mente, en la siguiente subsección se presenta una pequeña propuesta que puede ser de mucha ayuda para los que deseen seguir estudiando los sistemas distribuidos y su procesamiento en una escala mayor.

7.1. Propuesta para futuros trabajos

En esta sub sección se hace mención de una propuesta para la incorporación de un pequeño cluster para el estudio y la aplicación de diferentes procesos distribuidos.

Uno de los atractivos de estos frameworks es la capacidad que tienen de funcionar en lo que se llama "comodity hardware", es decir, una máquina modesta es capaz de ser parte del cluster.

Entonces el uso de máquinas que se encuentren en deshuso dentro del módulo pueden ser de extrema utilidad para alguien que necesite los recursos para la implementación de esta red interconectada de máquinas para procesar datos. Aquí no necesariamente se debe hablar de Hadoop, o cualquier herramienta mencionada que forme parte del ecosistema, sino que se puede generalizar aún más esta situación y se puede tratar de poner en funcionamiento un cluster de cómputo distribuido de código abierto, como por ejemplo OpenStack [13], que será de utilidad para un número importante de cátedras[14] y será un recurso extremadamente importante para el estudiante que se interese por el tema y no tenga los recursos para acceder a los equipos o tiempo de computo en Google/Amazon/Microsoft.

[13] `https://www.openstack.org/`

[14] Arquitectura de Computadores, Comunicación y Redes I y II, Sistemas Operativos, Sistemas Distribuidos, Paradigmas y Lenguajes de Programación

Referencias

[1] *NewVantage Partners Big Data and AI Executive Survey 2019*. 2019. URL: https://www.tcs.com/content/dam/tcs-bts/pdf/insights/Big-Data-Executive-Survey-2019-Findings-Updated-010219-1.pdf (visitado 16-06-2021).

[2] Danda B. Rawat, Ronald Doku y Moses Garuba. «Cybersecurity in Big Data Era: From Securing Big Data to Data-Driven Security». En: *IEEE Transactions on Services Computing* (2019). ISSN: 1939-1374. DOI: 10.1109/TSC.2019.2907247.

[3] Murat Kantarcioglu. «Securing Big Data: New Access Control Challenges and Approaches». En: *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*. New York, NY, USA. DOI: 10.1145/3322431.3326330.

[4] M H Padgavankar y Dr S R Gupta. «Big Data Storage and Challenges». En: 5 (2014), pág. 6.

[5] Rajeev Agrawal y Christopher Nyamful. «Challenges of big data storage and management». En: *Global Journal of Information Technology: Emerging Technologies* 6.1 (). ISSN: 2301-2617. DOI: 10.18844/gjit.v6i1.383.

[6] Hongming Cai y col. «IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges». En: *IEEE Internet of Things Journal* 4.1 (2017). ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2619369.

[7] Worldometer. *Current World Population*. 11 de jun. de 2021. URL: https://www.worldometers.info/world-population/.

[8] Mary Meeker. *Internet Trends 2019*. 2019. URL: https://bit.ly/3pPfE2L (visitado 11-06-2021).

[9] Peter Lyman y Hal R. Varian. *How Much Information?* 2003. URL: https://www2.sims.berkeley.edu/research/projects/how-much-info-2003/ (visitado 04-07-2021).

[10] Miquel Pellicer. *Las claves del informe 'Internet Trends 2019' de Mary Meeker*. 2019. URL: https://miquelpellicer.com/2019/06/claves-informe-internet-trends-2019-mary-meeker/ (visitado 11-06-2021).

[11] Kashif Sultan, Hazrat Ali y Zhongshan Zhang. «Big Data Perspective and Challenges in Next Generation Networks». En: *Future Internet* 10.7 (2018). DOI: 10.3390/fi10070056.

[12] Viktor Mayer-Shönberger y Kenneth Cukier. *Big data. A Revolution that will transform how we Live, Work and Think*. Boston, Massachusetts: Houghton Mifflin Harcourt, 2016. ISBN: 0544227751.

[13] Paola Verónica Britos. «Procesos de Explotación de Información Basados en Sistemas Inteligentes». Tesis doct. Universidad Nacional de La Plata, 2008. 234 págs. DOI: 10.35537/10915/4142. URL: http://sedici.unlp.edu.ar/handle/10915/4142.

[14] Atika Qazi y col. «Enhancing Business Intelligence by Means of Suggestive Reviews». En: *The Scientific World Journal* 2014 (). DOI: 10.1155/2014/879323. URL: https://www.hindawi.com/journals/tswj/2014/879323/.

[15] Yanhui Su, Per Backlund y Henrik Engström. «Business Intelligence Challenges for Independent Game Publishing». En: *International Journal of Computer Games Technology* 2020 (). DOI: 10.1155/2020/5395187. URL: https://www.hindawi.com/journals/ijcgt/2020/5395187/.

[16] Mohamed O. Khozium y Norah S. Farooqi. «Cooperative Business Intelligence Model Using a Multiagent Platform». En: *Scientific Programming* 2020 (). DOI: 10.1155/2020/8898719. URL: https://www.hindawi.com/journals/sp/2020/8898719/.

[17] Cempírek Václav y col. «Utilization of Business Intelligence Tools in Cargo Control». En: *Transportation Research Procedia* 53 (). ISSN: 2352-1465. DOI: 10.1016/j.trpro.2021.02.028. URL: https://www.sciencedirect.com/science/article/pii/S2352146521001885.

[18] Gloria Phillips-Wren, Mary Daly y Frada Burstein. «Reconciling business intelligence, analytics and decision support systems: More data, deeper insight». En: *Decision Support Systems* 146 (). DOI: 10.1016/j.dss.2021.113560. URL: https://www.sciencedirect.com/science/article/pii/S0167923621000701.

[19] Florian Schwade. «Social Collaboration Analytics Framework: A framework for providing business intelligence on collaboration in the digital workplace». En: *Decision Support Systems* (). ISSN: 0167-9236. DOI: 10.1016/j.dss.2021.113587. URL: https://www.sciencedirect.com/science/article/pii/S016792362100097X.

[20] Satriyo Wibowo y Tesar Sandikapura. «Improving Data Security, Interoperability, and Veracity using Blockchain for One Data Governance, Case Study of Local Tax Big Data». En: *2019 International Conference on ICT for Smart Society (ICISS)*. Vol. 7. ISSN: 2640-0545, págs. 1-6. DOI: 10.1109/ICISS48059.2019.8969805.

[21] Maryam Ghasemaghaei y Goran Calic. «Does big data enhance firm innovation competency? The mediating role of data-driven insights». En: 104 (), págs. 69-84. ISSN: 0148-2963. DOI: 10.1016/j.jbusres.2019.07.006.

[22] Pascal Hitzler y A. Krzysztof Janowicz. *Linked Data, Big Data, and the 4th Paradigm Editorial*. 2013. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.9076%5C&rep=rep1%5C&type=pdf.

[23] Shen Yin y Okyay Kaynak. «Big Data for Modern Industry: Challenges and Trends [Point of View]». En: *Proceedings of the IEEE* 103.2 (), págs. 143-146. ISSN: 1558-2256. DOI: 10.1109/JPROC.2015.2388958.

[24] Stephen Bonner y col. «Chapter 14 - Exploring the Evolution of Big Data Technologies». En: *Software Architecture for Big Data and the Cloud*. Ed. por Ivan Mistrik y col. Boston: Morgan Kaufmann, 1 de ene. de 2017, págs. 253-283. ISBN: 978-0-12-805467-3. DOI: 10.1016/B978-0-12-805467-3.00014-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128054673000144.

[25] *Big Data Analytics*. URL: https://www.ibm.com/analytics/hadoop/big-data-analytics (visitado 12-07-2021).

[26] *¿Qué es big data? – Amazon Web Services (AWS)*. Amazon Web Services, Inc. URL: https://aws.amazon.com/es/big-data/what-is-big-data/ (visitado 12-07-2021).

[27] *What Is Big Data? | Oracle*. URL: https://www.oracle.com/big-data/what-is-big-data/ (visitado 12-07-2021).

[28] Emilie Baro y col. «Toward a Literature-Driven Definition of Big Data in Healthcare». En: *BioMed Research International* 2015 (). DOI: 10.1155/2015/639021.

[29] Maryam Ghasemaghaei. «Understanding the impact of big data on firm performance: The necessity of conceptually differentiating among big data characteristics». En: *International Journal of Information Management* 57 (). ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2019.102055.

[30] Dunren Che, Mejdl Safran y Zhiyong Peng. «From Big Data to Big Data Mining: Challenges, Issues, and Opportunities». En: vol. 7827. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, págs. 1-15. ISBN: 978-3-642-40269-2 978-3-642-40270-8. DOI: 10.1007/978-3-642-40270-8_1.

[31] Son K. Lam y col. «Leveraging Frontline Employees' Small Data and Firm-Level Big Data in Frontline Management: An Absorptive Capacity Perspective». En: *Journal of Service Research* 20.1 (), págs. 12-28. ISSN: 1094-6705, 1552-7379. DOI: 10.1177/1094670516679271.

[32] Gil Press. *6 Predictions About Data In 2020 And The Coming Decade*. Forbes. URL: https://www.forbes.com/sites/gilpress/2020/01/06/6-predictions-about-data-in-2020-and-the-coming-decade/ (visitado 15-07-2021).

[33] Senem Sagiroglu y Duygu Sinanc. «Big data: A review». En: 2013, págs. 42-47. ISBN: 978-1-4673-6403-4. DOI: 10.1109/CTS.2013.6567202.

[34] Philip Russom. *Big Data Analytics*. 2011.

[35] Bill Albert, Tom Tullis y Donna Tedesco. «Beyond the Usability Lab. Chapter 5 - Data Preparation». En: (2010). URL: https://www.sciencedirect.com/book/9780123748928/beyond-the-usability-lab.

[36] Datalytics. *Datalytics DataSchool. Arquitectura de Datos basadas en Data Warehouse*. 10 de sep. de 2020.

[37] Datalytics. *Datalytics DataSchool. Arquitectura de Datos Modernas*. 17 de sep. de 2020. URL: https://www.youtube.com/watch?v=wWNZa-Ez8IU.

[38] Dan I. Moldovan. *Parallel Processing. From Applications to Systems*. San Mateo, California: Morgan Kaufmann Publishers, 1993.

[39] Roman Trobec y col. *Introduction to Parallel Computing. From Algorithms to Programming on State-of-the-Art Platforms*. Switzerland: Springer Nature Switzerland, 2018. DOI: 10.1007.978-3-319-9883307_1.

[40] D. C. Hyde. *Introduction to the Principles of Parallel Computation. Chapter 1: Introduction*. Lewisburg, PA 17837: Department of Computer Science, Bucknell University, 1998.

[41] Germán A. J. Pautsch y Esteban R. Martini. *Paradigmas y Lenguajes de Programación - Unidad II - Metodologías de Programación Paralela*. 2019.

[42] Terry J. Fountain. *Parallel Computing. Principles and Practice*. Cambridge: Cambridge University Press, 1994.

[43] Terry W. Clark, Reinhard v. Hanxleden y Ken Kennedy. «Experiences in Data-Parallel Programming». En: *Scientific Programming* 6.1 (1997), págs. 153-158. ISSN: 1058-9244. DOI: 10.1155/1997/260463.

[44] Lex Wolters, Gerard Cats y Nils Gustafsson. «Data-Parallel Numerical Weather Forecasting». En: *Scientific Programming* 4.3 (1995), págs. 141-153. ISSN: 1058-9244. DOI: 10.1155/1995/692717.

[45] Magne Haveraaen. «Machine and Collection Abstractions for User-Implemented Data-Parallel Programming». En: *Scientific Programming* 8.4 (2000), págs. 231-246. ISSN: 1058-9244. DOI: 10.1155/2000/485607.

[46] Yi Pang, Lifeng Sun y Shiqiang Yang. «Data parallelization of Kd-tree ray tracing on the Cell Broadband Engine». En: *2009 IEEE International Conference on Multimedia and Expo*. 2009 IEEE International Conference on Multimedia and Expo. ISSN: 1945-788X. Jun. de 2009, págs. 1246-1249. DOI: 10.1109/ICME.2009.5202727.

[47] Victor Zakharov y col. «Architecture of Software-Hardware Complex for Searching Images in Database». En: *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). ISSN: 2376-6565. Ene. de 2019, págs. 1735-1739. DOI: 10.1109/EIConRus.2019.8657241.

[48] Nadine Hajj, Yara Rizk y Mariette Awad. «A MapReduce Cortical Algorithms Implementation for Unsupervised Learning of Big Data». En: *Procedia Computer Science* 53 (2015), págs. 327-334. ISSN: 18770509. DOI: 10.1016/j.procs.2015.07.310. URL: https://linkinghub.elsevier.com/retrieve/pii/S187705091501813X (visitado 26-07-2021).

[49] T. Oshitani y T. Watanabe. «Parallel map recognition with information propagation mechanism». En: *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*. Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318). Sep. de 1999, págs. 717-720. DOI: 10.1109/ICDAR.1999.791888.

[50] David P. Rodgers. «Improvements in multiprocessor system design». En: *ACM SIGARCH Computer Architecture News* 13.3 (1 de jun. de 1985), págs. 225-231. ISSN: 0163-5964. DOI: 10.1145/327070.327215. URL: https://doi.org/10.1145/327070.327215.

[51] Xiaofu Meng y col. «Luminance and Chrominance Parallelization of H.264/AVC Decoding on a Multi-core Processor». En: *2013 IEEE Eighth International Conference on Networking, Architecture and Storage*. 2013 IEEE Eighth International Conference on Networking, Architecture and Storage. Jul. de 2013, págs. 252-256. DOI: 10.1109/NAS.2013.39.

[52] Mengjie Liu y col. «Joint Two-Tier Network Function Parallelization on Multicore Platform». En: *IEEE Transactions on Network and Service Management* 16.3 (sep. de 2019), págs. 990-1004. ISSN: 1932-4537. DOI: 10.1109/TNSM.2019.2920012.

[53] Jianhong Zhou, Gang Feng y Yi Gao. «Network Function Parallelization for High Reliability and Low Latency Services». En: *IEEE Access* 8 (2020), págs. 75894-75905. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2988719.

[54] I-Chieh Lin, Yu-Hsuan Yeh y Kate Ching-Ju Lin. «Toward Optimal Partial Parallelization for Service Function Chaining». En: *IEEE/ACM Transactions on Networking* (2021), págs. 1-12. ISSN: 1558-2566. DOI: 10.1109/TNET.2021.3075709.

[55] Andrew S. Tanenbaum y Maaren van Steen. «Introduction». En: *Distributed System Principles and Paradigms*. USA: Pearson Prentice Hall, 2007. Cap. 1, págs. 1-32. ISBN: 0-13-239227-5.

[56] Andrew S. Tanenbaum. *Distributed Operating Systems*. USA: Pearson Prentice Hall, 1994. ISBN: 0-13-219908-4.

[57] Andrew S. Tanenbaum y Maaren van Steen. *Distributed System Principles and Paradigms*. USA: Pearson Prentice Hall, 2007. ISBN: 0-13-239227-5.

[58] Abraham Siberschatz, Peter Baer Galvin y Greg Gagne. «Distributed Systems». En: *Operating Systems Concepts*. Ninth Edition. USA: Wiley, 2013. Cap. 17, págs. 741-772. ISBN: 978-1-118-06333-0.

[59] Jeffrey Dean y Sanjay Ghemawat. *MapReduce: simplified data processing on large clusters*. 2004. URL: https://www.usenix.org/legacy/event/osdi04/tech/full_papers/dean/dean_html/.

[60] Jeffrey Dean y Sanjay Ghemawat. «MapReduce: simplified data processing on large clusters». En: *Communications of the ACM* 51.1 (1 de ene. de 2008), págs. 107-113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492. URL: https://doi.org/10.1145/1327452.1327492.

[61] Sanjay Ghemawat, Howard Gobioff y Shun-Tak Leung. «The Google file system». En: *Proceedings of the nineteenth ACM symposium on Operating systems principles*. SOSP '03. New York, NY, USA: Association for Computing Machinery, 19 de oct. de 2003, págs. 29-43. ISBN: 978-1-58113-757-6. DOI: 10.1145/945445.945450. URL: https://doi.org/10.1145/945445.945450.

[62] Kyong-Ha Lee y col. «Parallel data processing with MapReduce: a survey». En: *ACM SIGMOD Record* 40.4 (11 de ene. de 2012), págs. 11-20. ISSN: 0163-5808. DOI: 10.1145/2094114.2094118. URL: https://doi.org/10.1145/2094114.2094118.

[63] *Mapper (Apache Hadoop Main 2.7.4 API)*. URL: https://hadoop.apache.org/docs/r2.7.4/api/org/apache/hadoop/mapreduce/Mapper.html (visitado 17-08-2021).

Siglas

BI Business Intelligence. 3, 5

DBMS Data Base Management Systems. 5
DW Data Warehouse. 5

EB Exabyte. 4, 26
EDW Enterprise Data Warehouse. 5

GB Gibabyte. 26
GFS The Google File System. 11, 13

HDFS Hadoop File System. 13

INDEC Instituto Nacional De Estadística y Censos. 5
IoT Internet of Things. 2, 5

JAR Java Archive Format. 28, 29
JVM Java Virtual Machine. 29

MR MapReduce. 11

PB Petabyte. 4, 11, 26

TB Terabyte. 4, 26

ZB Zettabyte. 4, 26

Glosario

clickstream *Flujo de clicks*, es una bitacora detallada de como los usuarios navegan una página web al realizar una tarea. 5

cluster *Grupo* o también llamado *Granja de servidores*, es un término que se aplica a los sistemas distribuidos y hace referencia a un conjunto de máquinas interconectadas por una red de alta velocidad. 11, 17, 27–29, 31

continuum *Continuo*, algo que cambia gradualmente o en pequeños incrementos sin ningún pico evidente. 9

dataset *Conjunto de datos*, sobre los cuales se realizan experimentos. Las conclusiones que los investigadores definan sobre un cierto tema, se da mediante los experimentos realizados sobre uno o más conjuntos de datos. 4, 5, 11

framework *Marco de trabajo*, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. 11, 13, 14, 16, 17, 27, 28

heartbeat *Latido*, es una señal periódica generada por software para indicar que el funcionamiento está funcionando adecuadamente, o para la sincronización con otras partes del sistema. 15

pipelining En Ciencias de la Computación, este hace referencia a una organización en la cual pasos sucesivos de una secuencias de instrucciones son ejecutadas por diferentes módulos, esto para que otra instrucción pueda iniciar antes que una instrucción anterior finalice. 6

wearables *Vestible*, En el contexto de la tecnología, hace referencia a un dispositivo que se pueda *vestir*. Ej: relojes inteligentes. 2

Anexo: Terabytes ⇒ Petabytes ⇒ Exabytes ⇒ Zettabytes

A veces, tratar magnitudes tan extremas no permite transmitir de manera comprensible lo masivo o lo ínfimo de tales magnitudes, esto es frecuentemente el caso al hablar de cantidad de datos tales como los Terabytes, Petabytes, Exabytes o Zettabytes.

Si el lector es una persona que esta familiarizada con los medios de almacenamiento y su tamaño, resulta sencilla la explicación de que un Terabyte es mayor que un Gigabyte, esto puede deberse a que el Gigabyte es una unidad que se utiliza cotidianamente, por ejemplo, el tamaño de los pendrives, la memoria del celular, el tamaño de discos en ordenadores, etc. Sin embargo, esta explicación puede volverse un desafío si se intenta realizar la explicación a una persona que no posee conocimiento alguno acerca de este tema, o incluso si la capacidad que se intenta explicar es lo suficientemente masiva como por ejemplo los mencionados Zettabytes (ZB).

En este anexo, se propone un ejemplo para que el lector (independientemente de su nivel en el uso de tecnología) se familiarice y tenga una mejor concepción la masividad a la hora de hablar del salto desde Gigabytes a Zettabytes. En este ejemplo no se van a utilizar conceptos de las ciencias de la computación, se va a tratar con segundos, minutos, horas, días y años, conceptos que se puede asumir son familiares y que son asimilados por todos los lectores.

Para iniciar, podemos partir desde el Gigabyte (GB), y podemos decir que un GB en este anexo va a contar como 1 seg. Partiendo de esa premisa, el siguiente cuadro posee las equivalencias en segundos para cada unidad mencionada anteriormente, utilizando, minutos, horas, días y años a medida que se haga necesario.

Unidad	Segundo(s)	Minutos	Horas	Días	Años
Gibabyte (GB)	1	—	—	—	—
Terabyte (TB)	1024	17	—	—	—
Petabyte (PB)	1048576	—	291	12	—
Exabyte (EB)	1073741824	—	—	12428	34
Zettabyte (ZB)	1099511627776	—	—	—	34865

Cuadro 1: Escala de unidades con su equivalencia temporal

Es decir, si se compara a el Gibabyte con el Zettabyte, se ve la masividad de la que se está hablando, y se hace palpable al extrapolar estos valores tan grandes a un concepto cotidiano como el tiempo. 1GB = 1 segundo y tenemos que 1ZB = 34865 **AÑOS**. Se puede quere acortar la diferencia y comparar dos unidades más cercanas como por ejemplo el Petabyte y el Zettabyte. Pero incluso con este acercamiento la diferencia es extrema ya que se tiene que 1PB = 12 Días mientras que 1ZB es 34865 **Años**.

Anexo: Implementación sobre HDFS

En esta sección de anexo se presenta la implementación del ejemplo explicado en la sección 6 *Hadoop*.

Que se cubre en este anexo: En este anexo se cubren, las partes necesarias en el código además de las clases mapper y reducer explicadas, los paquetes necesarios, la forma de compilación, la forma de ejecutar el paquete compilado sobre un cluster y se incluyen además los resultados de una demo realizada sobre un cluster.

Que no se cubre en este anexo: En este anexo no se cubre la instalación/configuración del cluster Hadoop. Cabe mencionar que es posible comprar poder de cómputo a proveedores como Amazon[15] o Microsoft[16] y tener funcionando un cluster Hadoop en minutos, sin ninguna configuración, esperando a que se le adjunten trabajos a realizar y datos sobre los cuales realizar los trabajos.

Partes necesarias además del Mapper y Reducer

En los ejemplos, únicamente se mencionaron las clases Mapper y Reducer, estas los objetos de estas clases son las que se distribuyen a los diferentes nodos workers para que puedan procesar los datos, sin embargo, faltan más partes para que este ejemplo pueda compilar. El ejemplo completo puede ser encontrado en el repositorio que se encuentra en la siguiente nota al pie de página[17]

En el listing 3 se pueden notar dos cuestiones que resaltan y que son completamente necesarias a la hora de construir un Mapper y un Reducer para trabajar, no solamente sobre el Framework Hadoop MapReduce, sino que también algo necesario para los proyectos de Java.

Primero se atiende lo segundo, lo de un proyecto Java. Un proyecto Java siempre tiene este punto de inicio llamado el método `main()` que se encuentra dentro de una clase objetivo para el compilador. En este caso, la clase objetivo se define como `WordCount`, dentro de esta se tienen a las implementaciones del mapper y el reducer, y además se cuenta con el método main que juega un rol importante en la primera parte del argumento de *cuestiones necesarias para trabajar sobre el framework Hadoop*. Estas cuestiones necesarias tienen que ver con las acciones necesarias para poder comunicarle al cluster las clases que extienden de Mapper y Reducer, la creación de un "trabajo" y determinar de dónde sacar el input y hacia donde mandar el output una vez terminado el trabajo, todo esto se implementa (en este ejemplo en particular) en el método `main()`.

[15] `https://aws.amazon.com/es/emr/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc`

[16] `https://docs.microsoft.com/en-us/samples/azure-samples/durablefunctions-mapreduce-dotnet/big-data-processing-serverless-mapreduce-on-azure/`

[17] https://github.com/ulisescolina/UC-PYLP/tree/master/Integrador/cuerpo/codigo/hadoop-integrador

```java
public class WordCount {
  /*
   * Acá se tiene que incluír la clase mapper tratada
   * anteriormente en la sección Hadoop
   * */

  //-------------------------------------------------

  /*
   * Acá se tiene que incluír la clase reducer tratada
   * anteriormente en la sección Hadoop
   * */

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Listing 3: Clase envoltorio WordCount

Paquetes necesarios

Se puede notar que en diferentes partes de los listings 1, 2 y 3 se hace uso de diferentes clases que no se encuentran definidas en ningun sitio en el código presentado, por ejemplo se tienen a las clases más prominentes ya mencionadas, como `Mapper` y `Reducer`, pero también notar que se tienen otras clases y tipos no nativos de Java que son también implementaciones de Hadoop, estos son `Configuration`, `Job`, `FileInputFormat`, `FileOutputFormat`, `IntWritable`, etc. En el listing 4 se lista la forma sintácticamente correcta de incluir los paquetes específicos de Hadoop.

Compilación

Una vez que se tiene implementado el Mapper y el Reducer, se tiene definida la clase que los envuelve y definida claramente las clases que se deben distribuir entre los *workers* para un trabajo determinado, queda armar el paquete que en última instancia será ejecutado sobre el framework sobre el cluster, en este caso, será armar un paquete `JAR`.

```
1 import org.apache.hadoop.conf.Configuration;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapreduce.Job;
6 import org.apache.hadoop.mapreduce.Mapper;
7 import org.apache.hadoop.mapreduce.Reducer;
8 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
9 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Listing 4: Paquetes Hadoop que se necesitan incluir

Acá se definen dos formas de realizar este procedimiento.

Desde Hadoop

Hadoop, cuenta con las herramientas para realizar la compilación de un proyecto, sin embargo este proceso nos dará unicamente el bytecode que puede correr sobre una Java Virtual Machine (JVM), por lo que serán necesarios dos pasos para la compilación exitosa de un proyecto.

```
\$ hadoop com.sun.tools.javac.Main WordCount.java
```

Siendo `WordCount.java` el archivo que contiene el proyecto. una vez terminado este procedimiento, es necesario empaquetar los archivos resultantes en un JAR, esto se logra de la siguiente manera.

```
\$ jar cf wc.jar WordCount*.class
```

Este último paso resultará en un archivo denominado `wc.jar`. El cual contendrá el proyecto compilado y listo para ser ejecutado sobre el cluster

Proyecto Maven

Esta segunda opción utiliza un método de construcción ampliamente aceptado y reconocido en la comunidad de desarrolladores Java, el proyecto Maven[18], esta es una herramienta acompaña al desarrollador Java a lo largo del ciclo de vida del software. Este ejemplo asume que existe una instalación configurada de Maven.

En el ejemplo que se lista a continuación, se está utilizando un concepto dentro de Maven llamado *Archetype*[19], lo que, en pocas palabras, permite que se cree la estructura de un proyecto a partir de una plantilla existente, y el resultado está listo para la compilación.

[18] https://maven.apache.org/
[19] https://maven.apache.org/archetype/index.html

```
\$ mvn archetype:generate \
-DgroupId=com.pylp.integrador \
-DartifactId=hadoop-integrador \
-DarchetypeArtifactId=maven-archetype-quickstart \
-DinteractiveMode=false
```

Este comando, crea la estructura de un proyecto Java listo para ser compilado. El subdirectorio `src` contiene el código necesario para el proyecto, y el subdirectorio `test` contiene los casos de prueba para el código en `src`. Se puede ver la implementación del código en el repositorio deo proyecto, en este se realiaron algunas modificaciones necesarias al archivo `pom.xml` que actúa como archivo de configuración de la compilación y el empaquetado de las clases. Las modificaciones se muestran en el siguiente listing

```
  <!-- ... -->
  <!-- Acá se tienen más cosas -->
  <!-- ... -->

  <packaging>jar</packaging>

  <!-- ... -->
  <!-- Acá se tienen más cosas -->
  <!-- ... -->
  <!-- Se importan los artefactos de Hadoop desde el repositorio
       de Maven -->
  <dependencies>
    <!-- https://mvnrepository.com/artifact/
                org.apache.hadoop/hadoop-common -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.8.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/
                org.apache.hadoop/hadoop-mapreduce-client-core -->
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-mapreduce-client-core</artifactId>
        <version>2.8.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/
                 org.apache.hadoop/hadoop-core -->
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-core</artifactId>
      <version>0.20.2</version>
    </dependency>
  </dependencies>
```

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <source>1.6</source>
          <target>1.6</target>
        </configuration>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

Con todo esto en orden se puede proceder a la compilación con Maven en una simple linea.

```
\$ mvn clean package
```

Esto va a dar como resultado un nuevo directorio que contendrá las partes resultantes de la compilación, en la raíz del directorio (con la configuración que se tiene en el repositorio mencionado), se va a tener un archivo `.jar`, este archivo es el que se utilizará para ejecutar los trabajos en el cluster

Ejecución del paquete y resultados

Una vez se finalizan los pasos anteriores, solamente queda ejecutar el paquete obtenido sobre la plataforma Hadoop. Para continuar con esto, se asume la existencia de los archivos que van a ser el input del Mapper en el directorio `./pylp/input`. Es necesario recalcar que el directorio que funcionará como output no se encuentre creado. Con todo esto en su lugar, se procede a ejecutar lo implementado de la siguiente manera

```
\$ hadoop jar \
wc.jar \  # Archivo jar con la implementacion de Mapper y Reducer
WordCount  \ # Clase envoltorio dentro del proyecto
pylp/input/ \ # Directorio que tiene los inputs
pylp/output/ # Directorio que funcionará como output
```