ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO

BIGDATA: ANÁLISIS Y MEDICIÓN DE SOLUCIONES EMPRESARIALES DE BIG-DATA DESPLEGADAS EN LA NUBE.

Por

Kristhian C. Gomez H.

Maria Paula Bueno M. Luis D. Benavides N.

Proyecto de grado para el
grado de Profesional en Ingeniera de Sistemas.

en la
Facultad de Ingenieria de Sistemas
Departamento de Sistemas y Organizaciones

Febrero 2018

ESCUELA COLOMBIANA DE INGENIERIA JULIO GARAVITO

Abstract

Facultad de Ingenieria de Sistemas
Departamento de Sistemas y Organizaciones

Profesionales en Ingeniera de Sistemas.

Por Kristhian C. Gomez H.
Maria Paula Bueno M. Luis D. Benavides N.

This Article consists of the study of different specialized tools in the field of BIG-DATA, showing the efficiency of each of these, in addition to measuring their efficiency by implementing a new operating tools of distributed and concurrent monitoring, for this case the extra tool implemented It is called EKETAL which works with aspects oriented programming in a distributed way....

Índice general

Capítulo 1

Introducción

En la actualidad están surgiendo nuevas necesidades y aplicaciones que requieren de la innovación de nuevas herramientas tecnológicas que sirvan de forma distribuida y concurrente, por esto se tomó de proyecto a Big Data como tema relevante, el cual se dirige hacia un nuevo enfoque de toma de decisiones mediante la recolección, análisis y gestión de grandes volúmenes de datos, ya sean estructurados, no estructurados y semi-estructurados, que no pueden ser procesados de una manera convencional[1]. Para esto se ha de investigar sobre el uso, arquitectura, construcción, depuración y evolución de soluciones empresariales Big Data desplegadas en la nube. Todo esto para conocer en que entorno se encuentra actualmente el manejo de grandes volúmenes de datos que se envían a diario por la red, que avances hay sobre todo lo relacionado con Big Data, además de conocer e identificar los diferentes tipos de errores que se generan al momento de diseñar herramientas distribuidas y concurrentes, para así plantear alguna solución o apoyar alguna ya planteada, con el fin de mejorar cada día el avance de estas nuevas propuestas de solución.

Este proyecto tiene como objetivo estudiar las diferentes arquitecturas y entender fuentes de errores en el desarrollo de middleware distribuido para Big Data, además de proponer soluciones y mejores prácticas para corrección y detección de errores de desarrollo. (Detección de patrones de eventos distribuidos), al final se dará a conocer las diferentes mediciones de desempeño de las técnicas propuestas, se profundizó en las herramientas Hadoop y AsterixDB y así mostrar que técnicas de procesamiento están funcionando para lograr resolver las necesidades de los usuarios, dependiendo del problema.

El problema radica en que el desarrollo de sistemas distribuidos y concurrentes es una tarea compleja afectada, no solo, por errores de programadores sino por el entorno heterogéneo de sistemas operativos y dispositivos conectados en la nube, la falta de soporte

de las herramientas de desarrollo para modelar la distribución y la concurrencia, y los errores humanos en el desarrollo de aplicaciones. Este proyecto busca estudiar y caracterizar el tipo de errores que comúnmente se presentan en la construcción de middleware para Big-Data. El middleware para Big-Data son las herramientas que se desarrollan para soportar actividades típicas del procesamiento distribuido de grandes volúmenes de datos. Es decir se enfocó este proyecto en los errores de distribución y concurrencia que se encuentran en herramientas como Hadoop, AsterixDB principalmente.

Para empezar a dar una solución al problema planteado se estudió el estado del arte para entender y especializarse en Big-Data y así realizar experimentos con herramientas concretas para adquirir habilidades de manipulación e implementación de soluciones Big-Data para caracterizar el tipo de errores de concurrencia y distribución que se ven en este tipo de herramientas, finalmente probar técnicas de diseño, mejores prácticas y heurísticas para atacar los problemas de concurrencia y distribución que se encuentran en esas soluciones.

En el siguiente artículo se introducirá dos componentes que posee el ecosistema de Big Data apoyados de una breve introducción para cada uno y, definir que componentes parten de ellos con su respectivo funcionamiento. En primer lugar, se iniciará con todo lo relacionado a Data Ingest (extracción de datos), es aquí donde se explicará a detalle que herramientas colaboran a este proceso y cómo llegan a solucionar problemas con el manejo de un volumen alto de datos; como segundo componente a estudiar es el procesamiento, el cual consiste en tomar los datos para analizar y llegar hacer diferentes acciones con relación a ellos, por ejemplo la predicción de ciertos casos, consultas más rápidas y eficientes, además de mostrar la posibilidad de procesar grandes volúmenes de datos a una alta velocidad, con el fin de realizar experimentos interceptando aspectos para la detección de errores en las respectivas herramientas.

Capítulo 2

Estado del Arte

2.1. Programación Orientada a Aspectos (POA)

Para empezar, Un aspecto es una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa (G. Kiczales), entonces la Programación Orientada a Aspectos (POA) es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la POA se pueden capturar los diferentes conceptos que componen una aplicación en entidades bien definidas, de manera apropiada en cada uno de los casos y eliminado las dependencias inherentes entre cada uno de los módulos. De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables.[2]

En las aplicaciones tradicionales, bastaba con un compilador o intérprete que tradujera nuestro programa escrito en un lenguaje de alto nivel a un código directamente entendible por la máquina. En las aplicaciones orientadas a aspectos, sin embargo, además del compilador, hemos de tener el tejedor, que nos combine el código que implementa la funcionalidad básica, con los distintos módulos que implementan los aspectos, pudiendo estar cada aspecto codificado con un lenguaje distinto. Para el lenguaje de aspectos AspectJ, el que utilizamos, hay un compilador llamado ajc, que tiene una opción de preprocesado que permite generar código java, para ser utilizado directamente por un compilador Java compatible con JDK, y también tiene una opción para generar archivos .class, encargándose él de llamar al compilador Java.

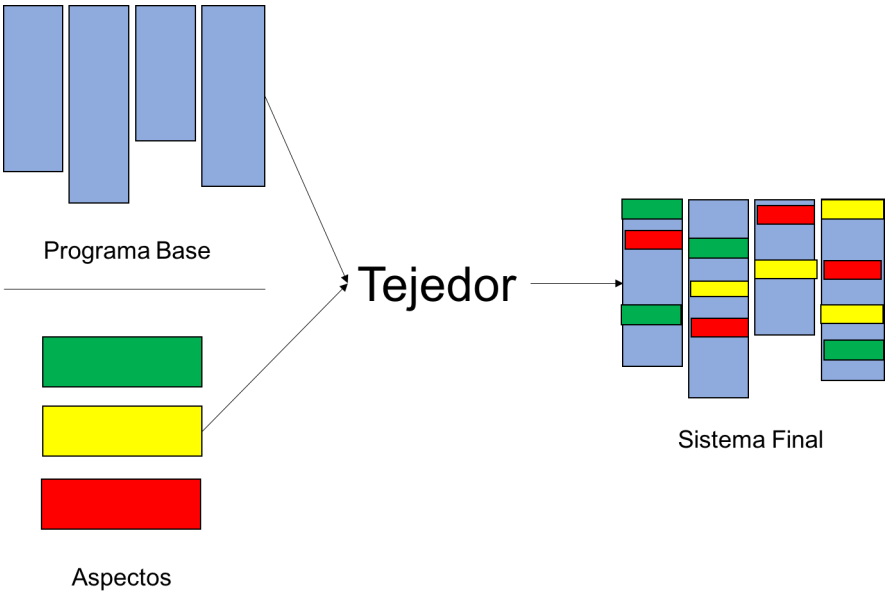

FIGURA 2.1: Programación orientada a aspectos

En definitiva, en AspectJ los aspectos son constructores que trabajan al cortar de forma transversal la modularidad de las clases de forma limpia y cuidadosamente diseñada. Por lo tanto, un aspecto puede afectar a la implementación de un número de métodos en un número de clases, lo que permite capturar la estructura de corte modular de este tipo de conceptos de forma limpia [3].

En general, un aspecto en AspectJ está formado por una serie de elementos:

1. Cortes. Los cortes (pointcut) capturan colecciones de eventos en la ejecución de un programa. Estos eventos pueden ser invocaciones de métodos, invocaciones de constructores, señalización y gestión de excepciones. Los cortes no definen acciones, simplemente describen eventos. Se utilizan para definir el código de los aspectos utilizando avisos.

2. Introducciones. Las introducciones (introduction) se utilizan para introducir elementos completamente nuevos en las clases dadas. Entre estos elementos podemos añadir: Un nuevo método a la clase, un nuevo constructor, un atributo, varios de los elementos anteriores a la vez y varios de los elementos anteriores en varias clases.

3. Avisos. Las declaraciones de avisos (advice) definen partes de la implementación del aspecto que se ejecutan en puntos bien definidos. Estos puntos pueden venir dados bien por cortes con nombre, bien por cortes anónimos. En las dos siguientes figuras se puede ver el mismo aviso, primero definido mediante un corte con nombre, y después mediante uno anónimo.

2.2. Programación Orientada a Aspectos de Eventos Distribuidos

Para poder introducir la programación de eventos distribuidos, se dice que el ambiente de desarrollo para este tipo de lenguajes está enfocado para aquellas aplicaciones que trabajan de forma distribuida y concurrente, además de que necesita de otras herramientas para poder ser compilado de forma correcta.

El funcionamiento de este lenguaje es permitir el monitoreo y acción frente a una serie de sucesos que son enviados por diferentes nodos que pueden estar ubicados en diferentes locaciones, este lenguaje es soportado por la herramienta EKETAL, la cual permite las mismas funciones con las que trabajan los aspectos pero en una escala de procesamiento mucho mayor. Eketal es un modelo de programación basado en eventos y un lenguaje para la detección, el monitoreo y la modificación dinámica en tiempo de ejecución de aplicaciones distribuidas y concurrentes, se basa en la programación de autómatas, logrando con esto patrones de eventos distribuidos. Una de las capacidades de este lenguaje es permitir los predicados de localización, quiere decir, se puede definir a que grupo de nodos actuar o sobre cuales no debería actuar, por último es el poder realizar la relación temporal de eventos (Cumple las reglas de Causalidad). Una forma gráfica de mostrar como trabaja se muestra en la siguiente imagen.

2.3. Taxonomía general de soluciones de Big-Data

Vamos a evidenciar que para gran parte de las soluciones de Big-Data de lo posible se enfoca en un ecosistema y un proceso a seguir, para que pueda emplearse correctamente su desarrollo esto se ve mejor en la figura N.4.

El ecosistema de Big Data trabaja sobre aplicaciones distribuidas, además de trabajar sobre el envío de grandes volúmenes de datos por la red. El procedimiento que se maneja internamente inicia por la parte de la ingestión de datos, donde se realiza el proceso de coger los datos crudos, en otras palabras sin ninguna relación de estructuración u orden alguno, pasan estos datos de estado crudo a uno ya estructurado, lo cual da la posibilidad manejar datos no estructurados, semi-estructurados y estructurados, después de dejar todos estos datos ya de una forma más organizada y relacionada entre ellos, para seguir después con el procesamiento de estos, donde ya dependiendo de los requerimientos del usuario se hace el respectivo procedimiento, para dar con resultados tales como consultas, análisis estadísticos, predicciones entre otras más posibilidades de procesamiento, por último se entra a realizar la parte de la organización de este procesamiento para ya ser

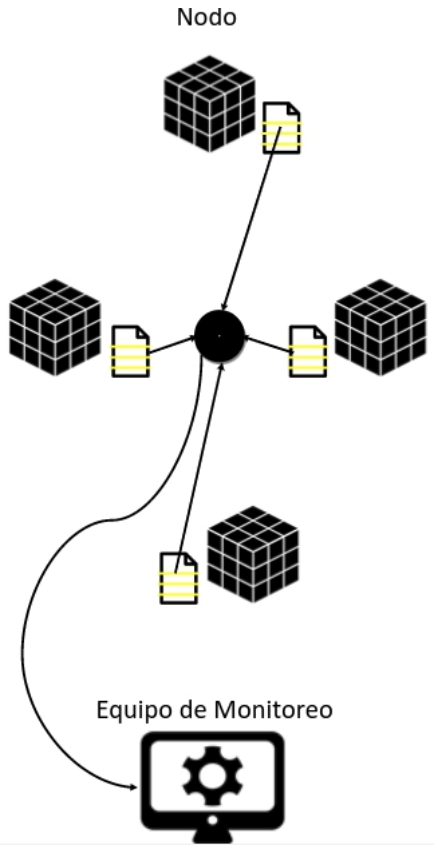

FIGURA 2.2: Programación orientada a eventos distribuidos

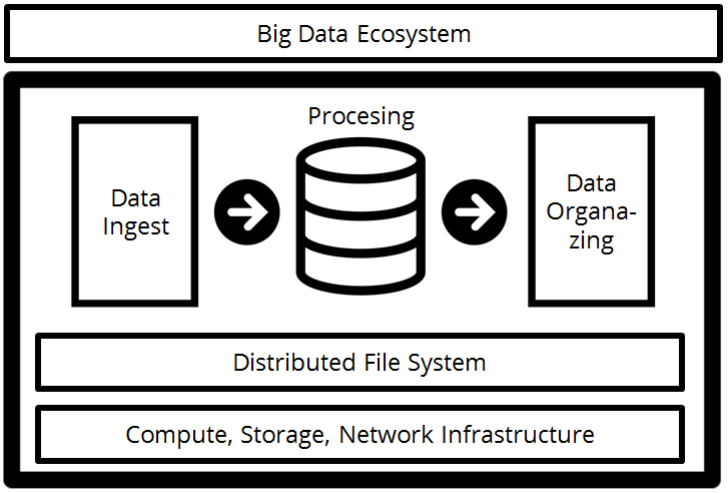

FIGURA 2.3: Ecosistema de Big Data

mostrado al usuario final, donde evidenciará directamente la interacción entre usuario y aplicación pero con apoyo de los dos componentes mencionados anteriormente.[4]

2.4. Data Ingest

Una gran cantidad de información digital se está generando diariamente a través de redes sociales, blogs, comunidades en línea, fuentes de noticias y aplicaciones móviles, así como de una variedad de fuentes en nuestros entornos cada vez más sensibles. Las organizaciones y los investigadores en la mayoría de los dominios reconocen hoy que el valor enorme y la penetración se pueden ganar capturando estos datos y haciéndolo disponible para la consulta y el análisis, y hacerlo es uno de los enfoques principales del movimiento actual de Big Data. La seguridad pública, la salud pública, la seguridad nacional, la aplicación de la ley, la medicina, el mercadeo, la ciencia política y la formulación de políticas gubernamentales pueden ser tremendamente beneficiosas en los ámbitos en los que la disponibilidad oportuna de información derivada de Big Data podría ser tremendamente beneficiosa[1]. La solución radica en crear sistemas de ingestión escalables especializados para poder abrir un número ilimitado de canales para recibir datos. [4]

Hoy en día, las personas utilizan ampliamente redes sociales (por ejemplo, Facebook, Twitter, etc.) y canales multimedia (por ejemplo, Youtube, Wikipedia, etc.) para compartir con sus amigos en todo el mundo: gustos, opiniones, ideas, etc. Millones de tweets sobre marcas, noticias y así sucesivamente, cientos de miles de Facebook "likes", ocurren todos los días. La recopilación de estos datos y la realización de analíticas en los flujos de datos de las redes sociales es uno de los principales desafíos de los grandes datos, ya que podrían lograrse objetivos empresariales muy interesantes, tales como: abordar estrategias de marketing, perfilar los gustos de las personas.[5]

2.4.1. AsterixDB

Entonces se deduce que recientemente existe un grado significativo de atención a las cuestiones de agregación en relación con los grandes datos. Reconocer las técnicas tradicionales de agregación para estudiar su capacidad en el contexto que describe la anatomía de AsterixDB, una plataforma paralela, semi-estructurada para la agregación, el almacenamiento, la indexación, la consulta y el análisis de grandes datos, que combina paradigmas heredados de la gestión de datos semi-estructurada, sistemas de bases de datos paralelas y grandes plataformas de computación de datos como MapReduce y Hadoop[6]. En AsterixDB, la gran agregación de datos desempeña un papel central, y se aborda mediante

implementaciones de bajo nivel que buscan la ingeniería del operador de agregación a través de diferentes combinaciones de clasificación, todos ellos trabajando dentro de un presupuesto de memoria estrictamente limitado.

En este documento se cubre el modelo de datos del sistema, su lenguaje de consulta y su arquitectura de software. También se incluye un resumen del estado actual del proyecto y un primer vistazo a cómo funciona AsterixDB en comparación con las tecnologías alternativas, incluyendo un DBMS relacional paralelo, un popular almacén NoSQL y una popular plataforma de análisis de datos SQL basada en Hadoop. Que ambas tecnologías pueden hacer. A continuación una serie de características que se le atribuyen a AsterixDB:

1. un modelo de datos NoSQL flexible que pudiera manejar escenarios de datos que van desde .esquema de primera.a "ningún esquema";
2. un lenguaje de consulta completa con al menos el poder expresivo de SQL;
3. soporte para el almacenamiento de datos, gestión de datos, y la indexación automática;
4. soporte para una amplia gama de tamaños de consulta, con el coste de procesamiento de consultas es proporcional a la consulta dada;
5. apoyo para la ingestión continua de datos, por lo tanto, la acumulación de grandes volúmenes de datos;
6. la capacidad de escalar con gracia para administrar y consultar grandes volúmenes de datos a través de grupos de productos básicos; y,
7. soporte integrado para los "tipos de datos Big Data"de hoy comunes, tales como texto, temporales y espaciales de datos simples.[7]

La Figura 5 proporciona una visión general de alto nivel de AsterixDB y su arquitectura lógica. Los datos entran a través de cargas, alimentaciones continuas y/o consultas de inserción. Se accede a los datos a través de consultas y la devolución (de forma síncrona o asincrónica) de sus resultados. El controlador de clúster es el punto de entrada lógica para las solicitudes de usuario; Los controladores de nodo y el controlador de nodo de metadatos (MD) proporcionan acceso a los metadatos de AsterixDB y al poder de procesamiento agregado del clúster de una arquitectura distribuida en el que cada nodo es independiente y autosuficiente (shared-nothing) subyacente. La ruta de datos punteada de la figura indica que se está trabajando para agregar soporte para consultas y notificaciones continuas.

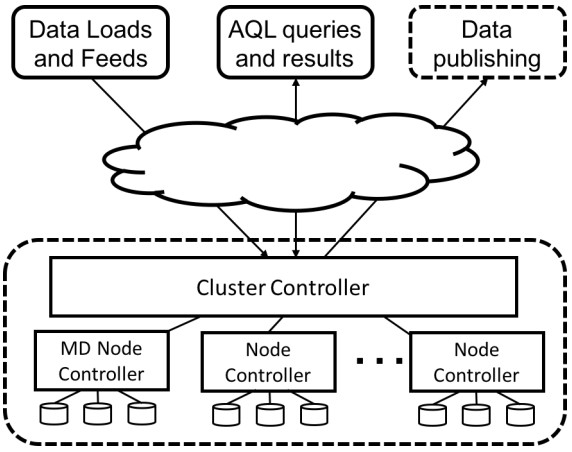

FIGURA 2.4: Arquitectura de AsterixDB

```
create dataverse TinySocial;
use dataverse TinySocial;
```

FIGURA 2.5: Creando TinySocial

A continuación el lado de definición de datos de AsterixDB, incluyendo su modelo de datos flexible basado en JSON (ADM).

2.4.1.1. Data Definition

Se describirá las características de definición de datos de AsterixDB, para esto ilustraremos a través de un escenario pequeño basado en información sobre usuarios y sus mensajes de una popular red social hipotética llamada TinySocial.

2.4.1.2. Dataverses, Datatypes, and Datasets

Para empezar definimos que es Dataverse o üniverso de datos.[es] un sector dentro del cual uno puede crear y gestionar los tipos, conjuntos de datos, funciones y otros artefactos para una aplicación determinada. Inicialmente, una instancia de AsterixDB no contiene datos distintos de los catálogos de sistema, que viven en un Dataverse definido por el sistema (el Metadata Dataverse). Para almacenar datos en AsterixDB, se crea un Dataverse y luego se rellena con los Datatypes y Datasets deseados. Un Datatype especifica lo que su definidor quiere que AsterixDB conozca, a priori, sobre un tipo de datos que se le pedirá que almacene. Un conjunto de datos es una colección almacenada de instancias de un tipo de datos, y uno puede definir conjuntos de datos múltiples de un tipo de datos dado. AsterixDB asegura que los datos insertados en un conjunto de datos se ajustan a su tipo especificado.

```
create type EmploymentType as open {
 organization-name: string,
 start-date:date,
 end-date:date?
}

create type MugshotUserType as {
 id: int32,
 alias: string,
 name: string,
 user-since: datetime,
 address: {
        street: string,
        city: string,
        state: string,
        zip: string,
        country: string },
 friend-ids: {{ int32 }},
 employment: [EmploymentType] }

create type MugshotMessageType as closed{
 message-id: int32,
 author-id: int32,
 timestamp: datetime,
 in-response-to: int32?,
 sender-location: point?,
 tags: {{ string }},
 message: string }
```

FIGURA 2.6: Ejemplo de definición de tipos en AsterixDB

AsterixDB se dirige a datos semi-estructurados, su modelo de datos proporciona el concepto de tipos de datos abiertos o cerrados. Al definir un tipo, el definidor puede utilizar Datatypes abiertos y decir AsterixDB tan poco o tanto sobre sus datos de antemano como ellos desean. (Cuanto más AsterixDB sepa sobre el conjunto de datos, menos necesita almacenar en cada instancia datos individuales). Las instancias de tipos de datos abiertos tienen contenido adicional, más allá de lo que especifica el tipo, siempre que contengan al menos la información prescrita por la definición del tipo de datos. Los tipos abiertos permiten que los datos varíen de una instancia a otra, dejando "margen de maniobra"para las variaciones a nivel de instancia, así como para la evolución de la aplicación (en términos de lo que se puede almacenar en el futuro). Para restringir los objetos en un conjunto de datos para contener sólo lo que diga un tipo de datos, con nada extra en instancias, se puede optar por definir un tipo de datos cerrado para ese conjunto de datos, AsterixDB evita que los usuarios almacenen objetos con datos extra o que falten ilegalmente en un conjunto de datos como éste. Por lo tanto, el subconjunto cerrado es relacional. Los tipos de datos están abiertos por defecto y cerrados sólo si su definición lo dice, la razón de esta elección es que muchos analistas de Big Data no parecen favorecer el diseño de esquema a priori y el diseño de ADM se dirige a datos semi-estructurados.

Abriendo un paréntesis, ADM es lo que uno obtiene extendiendo JSON con un conjunto más grande de tipos de datos (por ejemplo, datetime) y construcciones adicionales de modelado de datos (por ejemplo, bolsas) extraídas de bases de datos de objetos y luego

dándole un lenguaje de esquema. A diferencia de AsterixDB, las actuales plataformas de datos basadas en JSON, como REDIS, no ofrecen la opción de definir todo (o parte) del esquema de sus datos.

La última instrucción create type en la figura 7 define un tipo de datos para almacenar el contenido de un mensaje TinySocial. En este caso, dado que la definición de tipo para los mensajes de TinySocial se dice cerrada, los campos que lista serán los únicos campos que las instancias de este tipo podrán contener en Datasets.

Como el objetivo de AsterixDB es almacenar y consultar no sólo datos grandes, sino grandes datos semi-estructurados. La mayoría de los campos en las sentencias de tipo de creación anteriores podrían omitirse, por ejemplo, un cambio sería el tamaño de los datos en el disco. AsterixDB almacena información sobre los campos definidos a priori como metadatos separados. Un cambio lógico sería que AsterixDB fuera más flexible sobre el contenido de los registros en los conjuntos de datos resultantes, ya que hace cumplir sólo los detalles especificados del tipo de datos asociado con un conjunto de datos determinado. Los campos que deben especificarse a priori son los campos de clave primaria.

Otra característica importante de AsterixDB para gestionar los datos grandes de hoy es su soporte incorporado para tipos y funciones primitivas avanzadas útiles, específicamente aquellas relacionadas con espacio, tiempo y texto. La Tabla 1 enumera algunos de los tipos primitivos ADM avanzados, así como algunas de sus funciones AQL correspondientes. Una lista completa y más detalles se pueden encontrar en la documentación de AsterixDB.

2.4.1.3. Data Manipulation

El lenguaje de consulta para AsterixDB es AQL (Asterix Query Language). Dada la naturaleza de ADM, este lenguaje es capaz de manejar bien el anidamiento y la falta de esquemas a priori. XQuery tenía requisitos similares de XML, así que su base AQL es libremente en XQuery. ADM es más simple que XML y la compatibilidad XPath era irrelevante. XQuery fue co-diseñado por una diversa banda de diseñadores de lenguaje experimentados (SQL, programación funcional y expertos en XML) y se evitó el tener que revisar muchos de los mismos problemas y cometer errores. Comenzar desde SQL habría sido más desordenado, sintáctico y semánticamente, ya que el ANSI SQL fue diseñado para datos planos. Las sub consultas a menudo tienen scalarat-runtime-else-semántica de errores y su tratamiento de anidamiento para su función de tabla anidada

es complejo. Además, dado que AQL no está basado en SQL, AsterixDB es "NoSQL compliant".

2.4.1.4. Estado y Rendimiento

Hasta la fecha, ha habido tres versiones públicas de AsterixDB. La versión beta (0.8.0) apareció en junio de 2013[7]. Fue el primer lanzamiento y mostró errores, pero algo de espacio para la mejora. Las liberaciones subsiguientes (0.8.3, 0.8.5) han salido aproximadamente en intervalos de cinco meses; estos han sido los lanzamientos de estabilización con un rendimiento mejorado más algunas características menores que nuestros çlientesïniciales necesitaban. AsterixDB está en su infancia y tiene las limitaciones correspondientes. Una limitación, típica de los primeros sistemas, es la ausencia de un optimizador de consultas basado en costos. En cambio, tiene un conjunto de reglas bastante sofisticadas pero "seguras"para determinar la forma general de un plan de consulta físico y su paralelización y movimiento de datos. El optimizador realiza un seguimiento de la partición de datos y sólo mueve los datos como requieren los cambios en el paralelismo o partición.

2.4.2. Propuesta Futura

Data ingest consiste en mover datos - y especialmente datos no estructurados - desde dónde se origina, en un sistema dónde puede ser almacenado y analizado como Hadoop. Puede ser continua o asincrónica, en tiempo real o en conjunto o ambas (arquitectura lambda) dependiendo de las características de la fuente y el destino. En muchos escenarios, la fuente y el destino pueden no tener la misma sincronización de datos, formato o protocolo y requerirá algún tipo de transformación o conversión que pueda ser utilizada por el sistema de destino.

A medida que crece el número de dispositivos IoT, tanto el volumen como la varianza de las fuentes de datos se están expandiendo rápidamente, fuentes que ahora necesitan ser acomodadas ya menudo en tiempo real. Sin embargo, extraer los datos de tal manera que pueda ser utilizado por el sistema de destino es un desafío significativo en términos de tiempo y recursos. Hacer que la ingesta de datos sea lo más eficiente posible ayuda a concentrar los recursos en el análisis de grandes datos, en lugar de los esfuerzos mundanos de preparación y transformación de datos.

Los grandes problemas de hacer data ingest se pueden resumir en que las herramientas diseñadas con propósito y sobre-diseñadas hacen que los datos grandes consuman complejos, consumen mucho tiempo y son caros, la escritura de scripts personalizados y la

combinación de varios productos para adquirir e ingerir datos asociados con las grandes soluciones de ingesta de datos tardan demasiado tiempo y evitan la toma de decisiones puntuales requeridas para el entorno empresarial actual y las interfaces de línea de comandos para las herramientas existentes crear dependencias en los desarrolladores y los grilletes de acceso a los datos y la toma de decisiones.

La herramienta HORTONWORKS que maneja la forma más rápida de abordar muchos problemas de ingesta de datos grandes hoy en día control en tiempo real e interactivo de los flujos de datos, aceleración de la recopilación y el movimiento de datos para aumentar el ROI de los grandes datos visibilidad operativa en tiempo real, feedback y control, agilidad y capacidad de respuesta en tiempo real, toma de decisiones en tiempo real a partir de fuentes de datos Logrado mediante la eliminación de la dependencia y los retrasos inherentes a un enfoque de codificación y secuencias de comandos personalizados Off-the-shelf, la programación basada en flujo de grandes infraestructuras de datos. Seguro, la recogida de datos fiables y priorizados sobre geográficamente dispersos, ancho de banda variable procedencia Una cadena de custodia para el cumplimiento de los datos y la "valoración"de los datos y la optimización del flujo de datos y la resolución de problemas

2.5. Procesamiento

En el ecosistema de Big Data es la etapa en la cual los datos son procesados y analizados con el apoyo de herramientas como Hadoop entre otras, estas soportan aplicaciones distribuidas, las cuales trabajan con grandes cantidades de datos con el beneficio de poder transferir los datos en un corto intervalo de tiempo, mejorando la eficiencia de las aplicaciones. Más adelante explicaremos a profundidad como logran estas herramientas para poder procesar toda esa gran cantidad de datos de una forma muy ingeniosa y fácil de entender en ciertos casos, para iniciar la parte de procesamiento se dará a conocer una de las principales soluciones al manejo de datos a gran escala y muy conocido por gran cantidad de personas, esta solución es llamada MapReduce de la cual se desglosan mas herramientas basadas en esta, daremos a conocer dos de estas derivaciones, además de una alternativa diferente a MapReduce el cual logra su mismo propósito.

2.5.1. MapReduce

El Map Reduce esta pensado para la solución practica y el mejoramiento de aquellos problemas que son o pueden llegar a ser paralelizados, toca tener en cuenta que no todos

los problemas se pueden resolver con este Framework de forma eficiente y en otros casos más extremos no puede resolverlos. Además de que esta herramienta utiliza un sistema de archivos distribuidos HDFS.

La estructura que maneja Map Reduce se conforma de la siguiente manera: El maestro cuenta con un servidor JobTracker y varios servidores TaskTracker que toman el rol de esclavos; el funcionamiento de estos componentes se realiza de la siguiente manera, el JobTracker es el punto de interacción entre los usuarios y el Map Reduce, donde los usuarios mandan las tareas o trabajos Map Reduce al JobTracker donde los pone en fila y los ejecuta en orden de llegada un ejemplo más ilustrativo es imaginar una cadena de producción.

El maestro gestiona la asignación y delega las tareas a los esclavos, los esclavos ejecutan las tareas bajo la supervisión y manejo del maestro, estos esclavos son los que manejan el movimiento de datos entre la fase de Map() a la de Reduce().

* (si se presenta un error en una función map se hace la detección de ellos en el mismo momento que se han de presentar colocando a otro esclavo a realizar la función que no se termino de ejecutar y así permitiendo al sistema estar activo todo el tiempo.)

2.5.1.1. Función Map()

Esta función se paraleliza, el cual es un conjunto de archivos de entrada que se divide en varias tareas llamado filesplit; se crean bloques de tamaño de 128 MB las cuales se distribuyen a los nodos de los TaskTracker en este caso los esclavos, estos pueden realizar la misma tarea si lo hiciera falta.

2.5.1.2. Función Reduce()

Esta función se realiza de forma paralela para cada uno de los grupos que produce la función Map(), esta función es llamada una vez para cada clave única de salida de la función Map() junto con esa clave lo que hace es pasarse todos los valores asociados que posee esa clave para realizar alguna fusión para que se genere un conjunto más pequeño de los valores y así mejorar la eficiencia frente al problema que se quiere resolver.

A pesar de que el Map Reduce logra muchas cosas con las bases que tiene da la posibilidad de explotar aun más, dado que tiene más extensiones para realizar aun más tareas con un cierto propósito y de estas mismas hay gran diversidad para la solución de

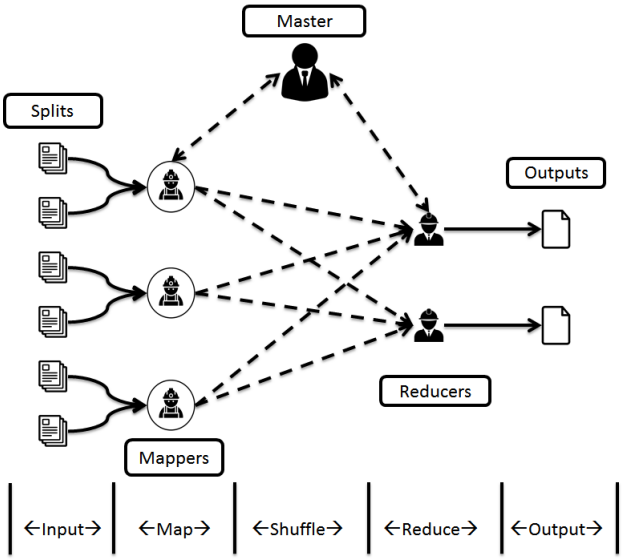

FIGURA 2.7: Arquitectura de MapReduce

problemas, claro esta que se producen ciertos efectos secundarios como por ejemplo la no compatibilidad con ciertos tipos de extensiones así provocando retrasos al momento de trabajar.

Se emplea MapReduce en aquellos problemas de tipo concurrente entre los que se encuentran involucrados grandes conjuntos de datos que deben ser procesados por una gran cantidad de computadoras. El procesamiento paralelo puede ocurrir con el empleo de datos almacenados tanto en sistemas de archivos o en una base de datos estructuradas. Es por esta razón por la que se emplea en aplicaciones que poseen datos a gran escala, tales como aplicaciones paralelas, indexación web, minería de datos y simulación científica.

2.5.1.3. Ejemplo

El ejemplo más simple es el WordCount (el Hello World del MapReduce) sobre un fichero o ficheros de texto.

2.5.2. Conclusión De MapReduce

Esta herramienta da la posibilidad de crear soluciones de manera más eficiente y más eficaces, dado que brinda una fácil programación como para aquellos que no poseen experiencia en sistemas paralelos o de distribución, por debajo de todo esto posee algo de detalles de la paralelización, tolerancia a fallos, optimización de la localidad, Y balanceo

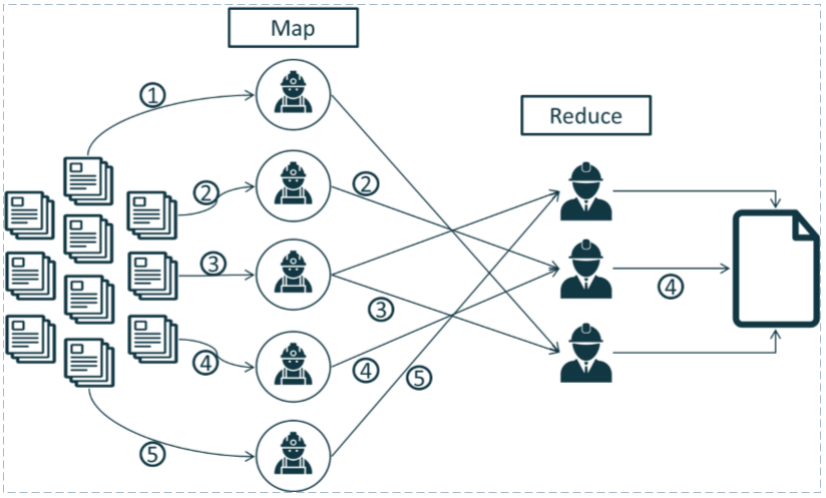

FIGURA 2.8: Ejemplo de procesamiento de Word Count.

de carga. por lo tanto no se puede tomar a la ligera esta metodología dado que se puede explotar de muchas maneras y cumplir con un fin[8]. [9]

2.5.3. Optimización de MapReduce a partir de Extreme Learn Machine con ComMapReduce

Uno de las derivaciones de MapReduce el cual cumple el mismo propósito pero de forma más eficiente a partir de ciertos casos de aprendizaje esta derivación da soluciones que se hacen sobre la nube, se habla de dos componentes utilizados para esta soluciones los cuales son ComMapReduce y ELM más adelante se ha de describir brevemente cada uno de estos componentes, también se a de encontrar las propuestas sugeridas por los autores de esta herramienta y que resultados se dieron del haber aplicado estos componentes juntos.[10]

En la actualidad cada vez hay mas volúmenes de datos que se necesitan transferir por medio de la red, lo cual requiere bastante tiempo y muchos requerimientos para poder ser enviados, por lo tanto se están necesitando soluciones para este tipo de inconvenientes, diferentes soluciones se han propuesto y cada día se busca la forma para poder optimizarlas y volver cada vez más fácil procesar este gran volumen de datos y a una gran velocidad.

En la figura 8 se encuentra la arquitectura que se maneja en ELM-CMR, donde más adelante se dará una breve descripción de las herramientas utilizadas para la solución por los autores:

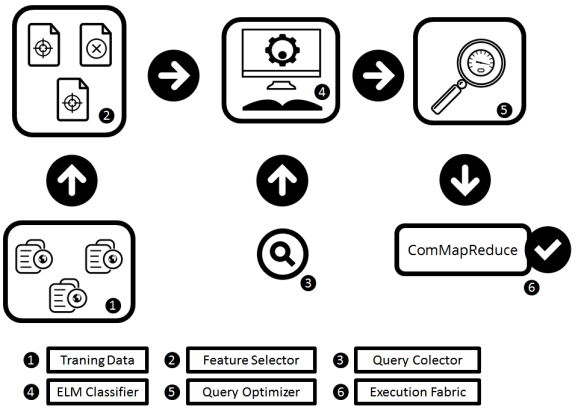

FIGURA 2.9: Arquitectura de ELM-CMR.

El procedimiento el cual funciona es que primero se han de seleccionar los datos de entrenamiento a que se refiere con este termino, se refiere a que son los casos mas generales para la solución del problema a resolver, así dado el caso que se presente ese caso se solucione de manera automática, el siguiente paso es revisar y seleccionar cuales casos son los más eficientes y entrar a una parte de aprendizaje de maquina, donde guardara esos casos y apoyarse de ellos para la solución de los problemas emergentes de los usuarios, después de ello logra optimizar las diferentes consultas que tiene que hacer para encontrar los datos necesarios a enviar y organizarlos, todo esto para ser enviados al procesamiento de datos y mostrar la solución del problema.

2.5.3.1. ComMapReduce

ComMapReduce es una mejora exitosa del Framework MapReduce(MR), sigue manteniendo la estructura y las cualidades que posee MR como por ejemplo la escalabilidad que posee, la tolerancia a fallos, lo rentable y fácil de usar.

Las características que posee ComMapReduce son:

* Maneja diferentes estrategias de comunicación para poder manejar de manera eficiente los datos.
* A este Framework se le agrega un nuevo nodo denominado Coordinador el cual se encarga de almacenar y brindar una información sobre los datos que se están enviando en el proceso.
* Lo que se hace en este Framework es filtrar los datos pocos prometedores
* No solo puede procesar las aplicaciones de datos masivos de una sola pasada si no que también se implementa para aplicaciones de análisis de datos masivos iterativos.

2.5.3.2. ELM (Extreme Learning Machine)

Tiene 5 componentes principales:

* Selector de Características: Examina los datos de entrenamiento lo que se refiere a datos base de aprendizaje y selecciona las características que pueden afectar a la consulta y así evitar estos.
* Selector de Funciones: Selecciona las funciones más recomendables para poder realizar las consultas esto depende de la aplicación.
* El clasificador ELM: Ejecuta patrones de implementación para elegir un orden de de ejecución
* El optimizador de Consultas.
* La Ejecución: es aquí donde se se envían los datos a ejecución en el ComMapReduce.

2.5.4. Propuestas por parte de los autores

* Una eficiente optimización del pensamiento de consultas basado en ELM en el Framework de ComMapReduce, este se le llama ELM-CMP
* Dos implementaciones una de ellas es sobre una sola consulta y la otra es con múltiples consultas a la vez.

2.5.5. Conclusión de sección ELM-CMR

Esta herramienta va a facilitar el trabajo de desarrollar aplicaciones las cuales se enfocan en el manejo de grandes volúmenes de datos y el manejo de consultas, todo esto para dar buenos productos finales a los usuarios y mejorar la rutina diaria de estos.

2.6. ComMapReduce Una mejora de MapReduce

Este es la segunda derivación de MapReduceen el cual se hablaran de 2 temas principales con los que trabaja ComMapReduce(CMR) los cuales son, las estrategias básicas de comunicación donde se vera a más profundidad a que se están refiriendo, otro de esos

temas son las estrategias de comunicación de optimización donde se da una explicación de como funcionan estas estrategias. [11]

Un problema detectado en la actualidad es la poca cantidad de algoritmos distribuidos para el almacenamiento y procesamiento de datos, he de venir la propuesta de los autores para desarrollar e implementar nuevas soluciones que satisfagan esta necesidad del cual hablaremos de CMR. Los autores de esta solución hicieron las siguientes propuestas para llevar acabo una herramienta mas eficiente para las necesidades actuales, las cuales son:

* Un Framework con mecanismos de comunicación simples y ligeros.
* Se proponen tres estrategias de comunicación básicas y dos estrategias de comunicación de optimización para procesar eficazmente aplicaciones relacionadas con Big Data.
* La implementación de CMR en tres aplicaciones, las cuales se dan para demostrar los amplios campos de aplicación que tiene este Framework
* Realización de extensos experimentos para evaluar el Framework CMR.

2.6.1. MapReduce

Para poder hablar más concretamente del funcionamiento de CMR se debe de hablar de su antecesor MapReduce, el cual también es un Framework de programación en paralelo que procesa datos a gran escala.

Su estructura esta dividido en tres componentes, el nodo maestro, los nodos Map y los nodos Reduce donde el papel del nodo maestro es el de encargarse de asignar tareas a los nodos mapeadores y asignarles los mapeadores su correspondiente reductor, la función del nodo Map es definida por el usuario donde obtiene ciertos datos de entrada y en la salida de datos produce tuplas intermedias en forma paralela escritas sobre el disco local para que el nodo reductor se encargue del siguiente proceso, el nodo reductor se ejecuta inmediatamente que el nodo mapeador termina lo que hace es obtener todo los datos intermedios de forma remota y aplica su función ya predefinida por el usuario y manda un archivo con los datos resultantes.

Hay una fase en este proceso que se llama barajado es donde obtiene los datos intermedios el nodo reductor, en esta fase se supone un considerable gasto de rendimiento debido a la transferencia de grandes volúmenes de datos que ocupan mayor parte de acceso al disco y

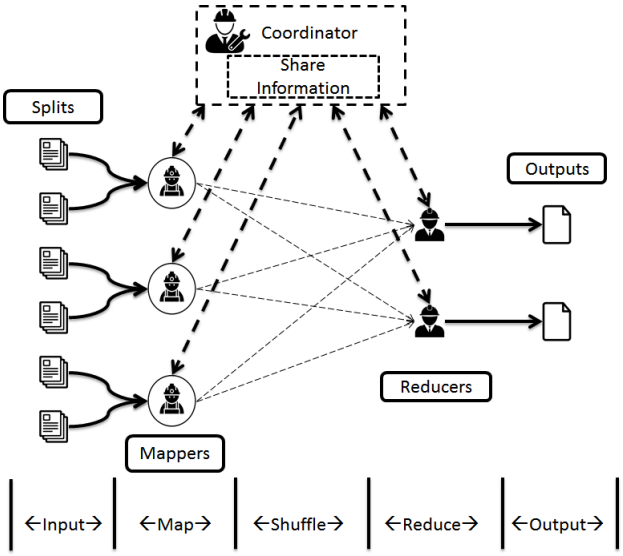

FIGURA 2.10: Arquitectura de ComMapReduce.

el ancho de banda de la red. Para disminuir el volumen de datos transferidos en esta fase. MapReduce utiliza una optimización denominada Combiner, es una compresión local de datos donde no tiene procesamiento de información de los otros mapeadores, por lo tanto no puede desempeñar un papel eficiente en la compresión global de datos.(Para mirar su funcionamiento global Observar la Figura 9)

2.6.2. ComMapReduce(CMR)

ComMapReduce hereda el marco básico de MapReduce, toma Hadoop Distributed File System (HDFS), la implementación de código abierto de MapReduce para almacenar los datos de entrada originales y resultados finales de cada aplicación. En su arquitectura cuanta con los mismos componentes de Maestro, Mapeadores y reductores los cuales cumplen los mismos roles que hay en MapReduce original. por lo tanto en su arquitectura no se a de notar el nodo maestro como se muestra en la Figura 10. Para evitar ejercer una presión adicional sobre el nodo maestro con relación a los mecanismos de comunicación y al obtener la información compartida entre componentes, se implementa un nuevo nodo denominado nodo coordinador, el cual se despliega en CMR para almacenar y generar información compartida.

El nodo coordinador puede comunicarse con los mapeadores y reductores con mecanismos de comunicación sencillos y ligeros, este nodo coordinador también puede recibir y almacenar algunas variables temporales y después generar un paquete de mensajes con la información compartida para las aplicaciones, además cada mapeador obtiene la

información compartida del nodo coordinador para poder evitar los datos poco prometedores. luego de el proceso de interacción entre el nodo coordinador y los mapeadores, se disminuye considerablemente el numero de reductores necesarios para terminar el proceso, así mejorando los tiempos de respuesta de este Framework.

ComMapReduce es fácil de usar solo mediante una interfaz de programación para asegurarse de como generar la información compartida, esta información compartida puede ser cualquier tipo descrito por Infotype:

* La función send se utiliza para enviar esta información compartida.
* La función receive se utiliza para recibir los múltiples datos que llegan.
* La búsqueda de funciones se utiliza para identificar la información compartida con las diferentes estrategias de comunicación, estas estrategias se explicaran mas adelante.

Las características de CMR son las siguientes, este Framework es tolerable a fallos como lo es MapReduce sin verse afectado por tener un nodo de mas en este caso el coordinador, además de ser igual de escalable a MapReduce gracias a sus métodos de comunicación.

vspace3mm

* Nodo coordinador:los tiempos de complejidad del nodo coordinador es el mismo del que emplea el nodo maestro, de tal manera no se ve afectada la escalabilidad, El uso de memoria es mucho menor que el nodo maestro, para almacenar solamente la información compartida y otras variables de estrategias de comunicación.

2.6.3. CMR Estrategias de comunicación

Se han diseñaron 3 estrategias de comunicación básicos en la actualidad para ilustrar como se comunican entre el nodo coordinador y los otros nodos, para obtener una información compartida global, en estos momentos se encontraron 2 estrategias de optimización para ampliar las formas de recibir y generar la información compartida.ç

Estrategia de comunicación Lazy: Se realiza la función Map la información local se comparte toda al coordinador, el coordinador elige cual es el mejor parámetro global para ser procesado y filtrado a partir de este nuevo parámetro establecido, así después enviar la información compartida global de nuevo al los nodos de Map y seguir con el proceso de enviar la nueva información a los nodos reduce reduce.

Estrategia de comunicación Eager: La información producida por los mapeadores pasa de la siguiente manera, toma el valor mas óptimo que posee este nodo y se mantiene ese valor como temporal en el nodo coordinador, de este nodo envía de nuevo este dato y filtra en el nodo que fue enviado el dato para realizar el filtro de datos nada prometedores, esa información resultante se envía directamente al nodo reductor, ya para el siguiente nodo mapeador lo que hace es enviar su dato mas óptimo comparar con el dato temporal que se encuentra en el nodo coordinador y elegir cual es el mas indicado, de ahí volver a realizar el mismo procedimiento mencionado por cada uno de los nodos mapeadores, con el beneficio de reducir considerablemente la cantidad de nodos asignados para la parte de reducción.

Estrategia de comunicación Híbrida: Esta estrategia acoge a las dos anteriores de la siguiente manera, se genera un primer grupo de nodos mapeadores para realizar el proceso de comunicación lazy de hay se hace todo el proceso normal, teniendo en cuenta que en el nodo coordinador se encuentra el dato mas óptimo de ese grupo para después ser comparado por el dato mas óptimo del siguiente grupo de nodos mapeadores y volver a repetir este procedimiento hasta terminar todos los grupos de nodos que hallan, de esta manera se ve evidenciada la unión entre la comunicación Lazy y Eager.

2.6.4. CMR Estrategias de comunicación de optimización

Estrategia de optimización Prepositive: Después de completar una parte de los mapeadores es generada una información compartida global mediante cualquier estrategia de comunicación y se coloca como parámetro de entrada a los siguientes nodos mapea dores para que el proceso de filtrado sea mas rápido y no tener que analizar datos de mas.

Estrategia de optimización Pospositive: Después de completar una parte de los datos mapeadores con alguna estrategia de comunicación básica el nodo coordinador guarda el dato mas óptimo como temporal, a partir de esto los mapeadores resultantes hacen su trabajo normal y envían sus datos a la fase de reduce en esta fase se utiliza

el dato temporal óptimo que guarda el nodo coordinar como parámetro de apoyo para filtrar los datos que ya están en la fase reduce así logra reducir la cantidad de datos a procesar en esta parte.

2.7. Alternativas de procesamiento

En la actualidad se ha identificado que no se puede depender todo el tiempo de una sola herramienta para apoyar el trabajo a diario de los usuarios, lo cual nos lleva a buscar nuevas alternativas que cumplan este fin, Para MapReduce Se encontró una solución alternativa llamada Dryad lo cual a su manera logra hacer el mismo propósito, se explicara mas detalladamente de que logra hacer y como lo hace.

2.7.1. Dryad: Distribución de programas paralelos

Dryad es un FrameWork que le permite al usuario utilizar los recursos de un clúster de computadora definiendo los parámetros necesarios que necesite cada nodo para ejecutar programas con funciones que trabajan en paralelo. El usuario puede definir con cuantas maquinas quiere trabajar, definiendo los recursos que van a utilizar cada uno de estos, ya sea con que cada una de ellas cuente con múltiples procesadores o núcleos, otra de las cualidades que posee esta herramienta es lo sencillo de trabajar incluso para personas que no conozcan de como ejecutar aplicaciones que trabajen con procesos en paralelo.

el usuario puede escribir varias funciones secuenciales y los conecta usando canales unidireccionales a estos nos referimos con canales de un solo sentido. El cálculo está estructurado como un grafo dirigido: las funciones son vértices de los grafos, mientras que los canales son bordes de los grafos. Un trabajo Dryad es un generador de grafos que puede sintetizar cualquier grafo acíclico dirigido. Estos gráficos pueden incluso cambiar durante la ejecución, en respuesta a eventos importantes en el cálculo.

Dryad completa a otros marcos computacionales, como Google MapReduce, o el álgebra relacional. Además, Dryad maneja la creación y gestión de trabajos, la gestión de recursos, la supervisión y visualización de trabajos, la tolerancia a fallos, la re-ejecución, la programación y la contabilidad. [12]

2.8. Propuesta para Procesamiento

En estos momentos se ha de apoyar más a las alternativas que se están planteando o ya existentes, con el fin de lograr una buena diversidad de soluciones para el buen

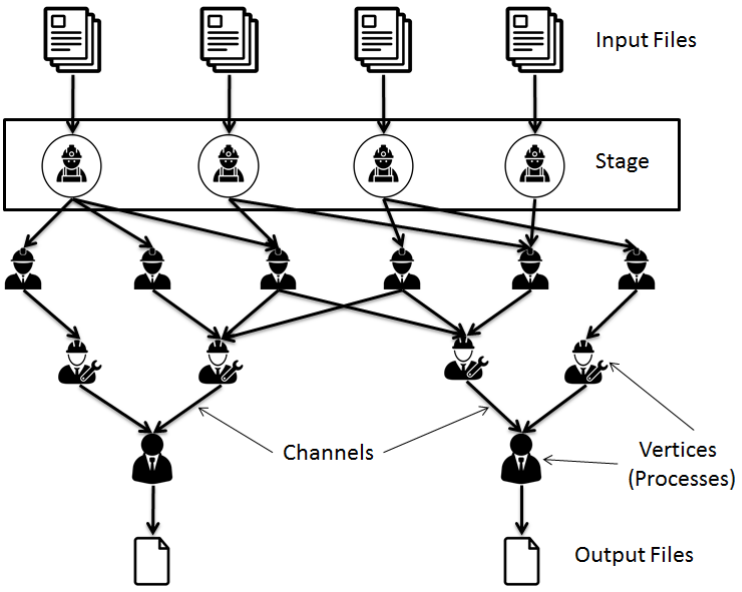

FIGURA 2.11: Arquitectura de Dryad.

procesamiento de grandes volúmenes de datos, así dando le la posibilidad a diferentes desarrolladores y gente del común, familiarizarse, trabajar sobre ellos y dar a conocerlos.

La idea a proponer es la siguiente: Formar una herramienta que aprenda progresivamente de los casos con los que llegue a trabajar de tal manera si se presentan en algún momento, que pueda resolverlos de forma autónoma, además de que cuando entre hacer el procesamientos de estos tenga la oportunidad de escoger cual de los diferentes procedimientos para procesar información es el más conveniente a realizar y entrar a aplicar lo. En pocas palabras Hacer un híbrido multi-diverso el cual tenga las posibilidades de extender y moldear su estructura para poder satisfacer los requerimientos del usuario, a que nos referimos con multi-diverso a que pueda acoplarse las diferentes soluciones que hay y pueden llegar haber, y aplicarlos en diferentes dispositivos que se encuentran en la actualidad, nos referimos también a híbrido, con el fin de que se logre tomar lo mejor que hay como base en este caso las mejores soluciones relacionadas con MapReduce y Lo mejor que hay en las alternativas de este, así logrando a que pueda abarcar más campo y mejorar en muchos aspectos lo que hay actualmente.

Capítulo 3

Experimentos Simples

3.1. Experimento con AsterixDB

Se describirá el apoyo para la ingestión de datos en AsterixDB, una fuente abierta en un sistema de gestión de grandes volúmenes de datos (BDMS) que proporciona una plataforma para el almacenamiento y análisis de grandes volúmenes de datos semi-estructurados. La "alimentación" de los datos es un nuevo mecanismo para que los datos continuos llegan en un BDMS de fuentes externas y poblen gradualmente un conjunto de datos persistido y los índices asociados. Para esto se mostrará cómo añadir un nuevo componente BDMS arquitectónico, llamado 'data feed', que hace que un sistema de Big Data sea el encargado de la funcionalidad que solía vivir fuera.[13]

Alentadas por los bajos costos de almacenamiento, las empresas hoy en día buscan recolectar y persistir los datos disponibles y analizarlos con el tiempo para extraer información útil. Los departamentos de marketing utilizan feeds de Twitter para realizar análisis del sentimiento para obtener comentarios de los usuarios finales sobre los productos de su empresa. Como otro ejemplo, las empresas de servicios públicos han lanzado metros que miden el consumo de agua, gas y electricidad y generan grandes volúmenes de datos de intervalo que se analizan a lo largo del tiempo.

Los sistemas de gestión de datos tradicionales requieren la carga de datos y la creación de índices antes de que los datos puedan someterse a consultas analíticas. Para mantener el ritmo con el rápido movimiento"de datos, un sistema de gestión de grandes volúmenes de datos (BDMS) debe ser capaz de ingerir y conservar los datos sobre una base continua. Un flujo de datos de un origen externo en almacenamiento persistente (indexado) dentro de un BDMS será referido como data feed. La tarea de mantener el flujo continuo de datos es en lo sucesivo, datos de gestión de alimentación.

```
create type RawTweet as open{
 tweetId: string,
 user: TwitterUser,
 location-lat: double?,
 location-long: double?,
 send-time: string,
 message-text: string,
};

create type TweeterUser as open{
 screen-name: string,
 lang: string,
 friendsCount: int32,
 statusCount: int32,
 followersCount: int32,
 name: string,
};
```

FIGURA 3.1: Definiendo los datatipos

3.1.1. Desafíos en la gestión de alimentación de datos

1. Una facilidad en la .ªlimentación.ªl momento de hacer ingestión debe ser lo suficientemente genérica para trabajar con una variedad de fuentes de datos y aplicaciones de alto nivel.

2. Múltiples aplicaciones deseen consumir los datos ingeridos y que tal vez los datos que llegan a ser procesados/persistidos sean de manera diferente. Es conveniente recibir un único flujo de datos desde un origen externo y, sin embargo, transformarlo en múltiples maneras para manejar diferentes aplicaciones simultáneamente.

3. Las fluctuaciones de las tasas de llegada, junto con las consultas sobre los datos persistentes, implican una diversa demanda de recursos. El sistema debe ofrecer escalabilidad al poder ingerir volúmenes de datos cada vez más grandes (posiblemente de varias fuentes) a través de la adición de recursos.

4. El sistema debe demostrar la elasticidad para satisfacer la demanda de recursos.

5. Se espera que la ingestión de datos se ejecute en un gran grupo de hardware de productos básicos que puede estar propenso a fallas de hardware. Es conveniente ofrecer el grado deseado de robustez en el manejo de los fallos al tiempo que minimiza la pérdida de datos.[13]

3.1.2. 'Data Feed' Básico

AQL ha incorporado soporte para fuentes de datos. En esta sección, describiremos de manera teórica como un usuario final puede modelar una fuente de datos y sus datos se conservan/indexado en un dataset (anteriormente explicado) de AsterixDB.

```
create feeed TwitterFeed using TwitterAdaptor
(“api“ = “pull“, “query“=“Obama“, “interval“=60);
```

FIGURA 3.2: Definiendo un adaptador de 'alimentación'

3.1.3. Recogida de Datos: Adaptadores de 'alimentación'

La funcionalidad de establecer una conexión con un origen de datos y recibir, analizar y traducir sus datos en registros de ADM (para el almacenamiento interior AsterixDB) está contenida en un adaptador de alimentación. Un adaptador de alimentación es una implementación de una interfaz y sus detalles son específicos para cada origen de datos. Un adaptador puede ser opcionalmente parámetros dados para configurar su comportamiento en tiempo de ejecución. Dependiendo del protocolo de transferencia de datos/APIs ofrecidas por el origen de datos, un 'feed' adaptador pueden operar en un modo de pull o push. Modo push implica sólo una solicitud inicial por el adaptador a la fuente de datos para configurar la conexión. Una vez que la conexión esté autorizado, el origen de datos "pushes.ªl adaptador de datos sin alguna solicitud posterior por el adaptador. En contraste, cuando operan en un modo de extracción, el adaptador se hace una solicitud por separado cada vez para recibir datos.

AsterixDB actualmente proporciona adaptadores integrados para varias fuentes de datos populares como: Twitter, CNN y RSS. Adicionalmente AsterixDB proporciona un adaptador basado en socket genérico que puede ser usado para ingerir datos que se dirige a un socket prescrito. La siguiente imagen muestra como el TwitterFeed contiene los tweets que contienen la palabra .ºbama". Como se ha configurado, el adaptador se hace una solicitud de datos cada minuto.

Para realizar el experimento incluimos un estudio del impacto de la ingestión de los parámetros sobre el comportamiento en tiempo de ejecución (rendimiento y latencia) bajo diferentes condiciones de carga de trabajo.

Configuración Experimental: Se realizó en un clúster de 3 nodos. Cada nodo tenía un procesador Intel 2,26GHz con procesador de cuatro núcleos, 8GB de RAM y un disco duro de 300 GB. Escribimos un stand-alone tweet generador (TweetGen) que puede emitir tweets sintéticos (JSON) a un tipo (Tweets por segundo - TWP) que sigue un patrón configurable. El datatype RawTweet creado en la figura 11 mostraba el equivalente ADM a la representación por un tweet. A continuación, escribimos un adaptador basado en socket personalizado- TweetGenAdaptor. El adaptador está configurado con

```
create feeed TweetGenFeed using TweetGenAdaptor
("datasource" = "10 . 1. 0. 1 :9000 , 10.1.0.2:9000");
```

FIGURA 3.3: Adaptador de 'alimentación'

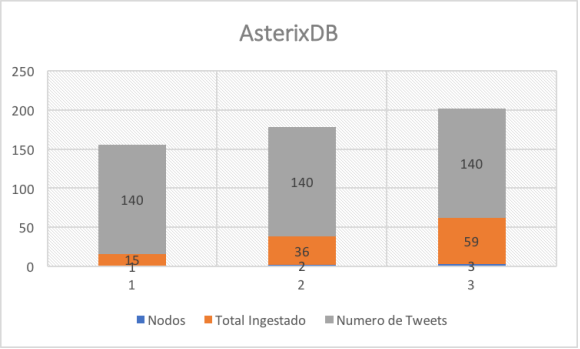

FIGURA 3.4: Escalabilidad: Número de registros (tweets) correctamente ingeridos (persistidos e indexados) a medida que aumenta el tamaño del clúster.

la ubicación(s) (dirección de socket) donde la(s) instancia(s) de TweetGen/se están ejecutando. Cada instancia de TweetGen recibe una solicitud de datos desde una instancia correspondiente TweetGenAdaptor, permitiendo así la ingestión de datos en paralelo.

Los resultados obtenidos se dan en la siguiente imagen.

3.2. Medición y Experimentos con Hadoop

En esta sección se hablara sobre las características, ejecuciones y diferentes actividades que se realizaron al momento de emplear ciertos ejemplos desarrollados sobre Hadoop, para este caso se ejecutó un ejemplo llamado WordCount el cual de forma resumida cuenta el número de palabras y concurrencias que posee cada una de ellas dentro de un documento de texto, claro está que esto se desarrolla a mayor escala en casos de la vida real. Para este ejemplo lo decidimos dividir en 3 pruebas, donde cada prueba tiene un número de palabras diferentes las cuales están de una hasta 80000 palabras, ejecutándose en diferentes números de nodos; Todos estos experimentos se realizaron de forma local, así que todas las pruebas se rigen sobre las siguientes especificaciones que posee la máquina. La máquina posee, un procesador Intel Pentium de 2,00 GHz, De Memoria RAM 3,00 GB, Hadoop se encuentra en modo Pseudo-Distribuido lo cual permite generar varios Nodos de forma local.

En la figura N.18 muestra los diferentes resultados obtenidos sobre la ejecución del ejemplo, donde se muestran los tiempos de ejecución correspondientes a su archivo de prueba, además de la cantidad de nodos que se utilizaron para ejecutarla.

# Nodos	Tests	# palabras	Tiempo De Ejecución
1 Nodo	Prueba 1	27.000	14.10 s
	Prueba 2	54.000	15.78 s
	Prueba 3	80.000	18.38s
2 Nodos	Prueba 1	27.000	10.23 s
	Prueba 2	54.000	11.89 s
	Prueba 3	80.000	13.10 s
3 Nodos	Prueba 1	27.000	7.32 s
	Prueba 2	54.000	9.90 s
	Prueba 3	80.000	11,07 s
4 Nodos	Prueba 1	27.000	4.56 s
	Prueba 2	54.000	6,92 s
	Prueba 3	80.000	9.03 s

FIGURA 3.5: Tabla de datos de ejecución.

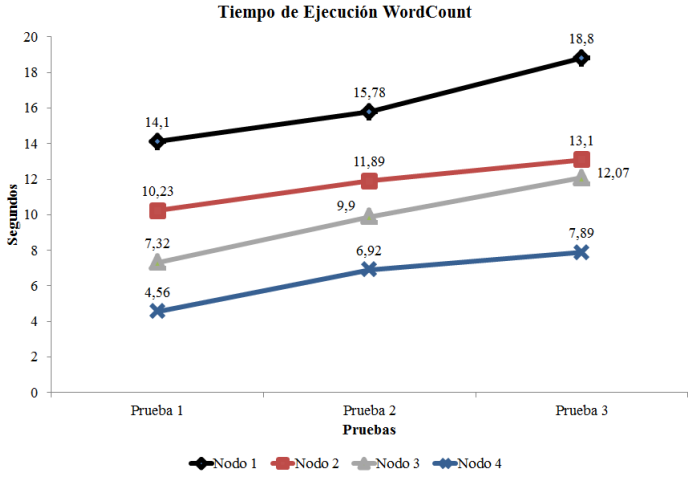

FIGURA 3.6: Tabla de datos de ejecución.

Además se evidencia en la figura 19 la relación entre la diferencia de tiempos y la similitud en la forma de ejecución, hay que tener en cuenta que algunos de los datos pueden tener un margen de error al momento de ejecutar el ejemplo, algunos de estos factores son, los otros procesos que se están ejecutando por parte del sistema operativo o de aplicaciones, o el momento de que un nodo al procesar suceda algún error, entre otros.

3.3. Medición y Experimentos en Hadoop con EKETAL

En esta sección se hablara sobre las características, ejecuciones y diferentes actividades que se emplearon sobre ciertos ejemplos desarrollados en Hadoop, para este caso se ejecutó una función llamada WordCount el cual de forma resumida, cuenta el número de palabras y concurrencias que posee cada una de ellas dentro de un documento de

```
package org.apache.hadoop.yarn.server.nodemanager;
import org.apache.hadoop.yarn.api.records.ApplicationId;

public aspect AspectCounter {
  Integer counter = 0;
  pointcut aspectcont() : execution
(* ...transition(ApplicationImpl, Application-
Event));
  before() : aspectcont() {
     counter++;
      System.out.println("...: "+counter);
  }
```

FIGURA 3.7: Ejemplo de Aspectos

texto. Claro está que esto se desarrolla a mayor escala en casos de la vida real, pero ahora acoplado con una herramienta llamada EKETAL la cual cuenta con la capacidad de monitorear y analizar eventos distribuidos y concurrentes, un ejemplo de es como hadoop donde esta desarrolla procesamientos de computo a una gran escala. Para este ejemplo se decidió dividir en 3 pruebas, donde cada prueba tiene un número de palabras diferentes las cuales están en un rango de una hasta 80000 palabras, ejecutándose en diferentes números de nodos; Todos estos experimentos se realizaron de forma local y en ambientes de maquinas virtuales en la nube, así que todas las pruebas se rigen sobre las siguientes especificaciones que posee la máquina. La máquina posee, un procesador Intel Pentium de 2,00 GHz, Memoria RAM 3,00 GB, Hadoop se encuentra en modo Pseudo-Distribuido lo cual permite generar varios Nodos de forma local y su versión es la 2.7.4 y las especificaciones más básicas para las maquinas virtuales que posee AWS(Amazon Web Services). Uno de los ejemplos sencillos que se elaboraron y ejecutaron para esta investigación fueron de la siguiente forma, en la figura 19 se muestra un pequeño ejemplo de como es la sintaxis de un aspecto y en la figura 20 la sintaxis de un evento distribuido.

En la figura 21 Muestra los resultados obtenidos de la ejecución de hadoop con EKETAL y al mismo tiempo comparando al tiempo de ejecución sin esta respuesta.

Además se evidencio la relación entre el numero de palabras con el tiempo de ejecución como se puede notar en la figura 22, hay que tener en cuenta que algunos de los datos pueden tener un margen de error al momento de ejecutar el ejemplo, algunos de estos factores son, los otros procesos que se están ejecutando por parte del sistema operativo o de aplicaciones, o el momento de que un nodo al procesar suceda algún error, entre otros.

Después del haber hecho el análisis a esta herramienta podemos concluir que el potencial que posee puede llegar a un punto donde va a ser necesario implementar esta herramienta

```
package org.apache.hadoop.yarn.server.nodemanager;
import ...security.ApplicationACLsManager;
import ...containermanager.ContainerManagerImpl;
import ...containermanager.application.*;
import org.apache.hadoop.yarn.api.records.ApplicationId;
import org.apache.hadoop.security.Credentials;
eventclass ENodeManager{
    automaton yarnContainer(){
        start initial: (startContainer -> container);
        end container: (startContainer -> container);
    }automaton yarnConstructor(){
        start initial: (constructorEvent -> last);
        end last;
    }group localGroup{
        localhost}
    event startContainer(): execution(* ...transition
(ApplicationImpl, ApplicationEvent));
    event constructorEvent(): call(NodeManager.^new());
    reaction before yarnConstructor.last{
        long time = System.currentTimeMillis();
        System.out.println(System.currentTimeMillis());}
    counter:int
    reaction before yarnContainer.container{
        counter+=1;
        System.out.println("Number of containers launched:
"+counter);}}}
```

FIGURA 3.8: Ejemplo de EKETAL

# Nodos	Tests	# de Palabras	Tiempo De Ejecución	Tiempo De Ejecución con EKETAL
1 Nodo(s)	Prueba 1	27000	14.10 s	14.99 s
	Prueba 2	54000	15.78 s	16.37 s
	Prueba 3	80000	18.38 s	18.38 s
2 Nodo(s)	Prueba 1	27000	10.23 s	10.89 s
	Prueba 2	54000	11.89 s	12.53 s
	Prueba 3	80000	13.10 s	14.21 s
3 Nodo(s)	Prueba 1	27000	7.32 s	8.02 s
	Prueba 2	54000	9.90 s	11.02 s
	Prueba 3	80000	11.07 s	13.01 s

FIGURA 3.9: Resultados de Ejecución con EKETAL

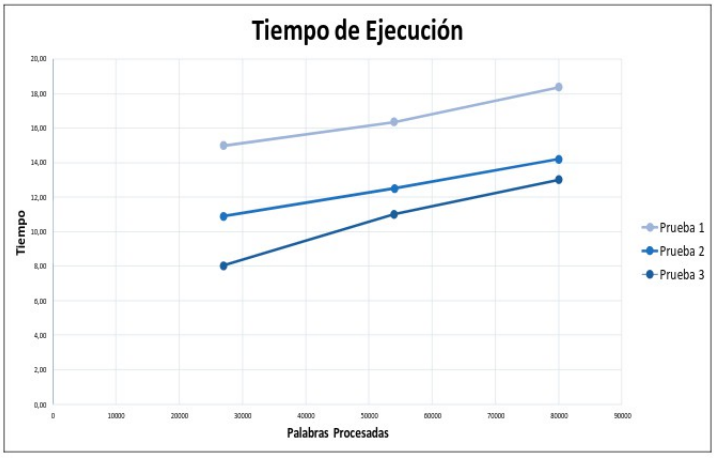

FIGURA 3.10: Comparaciones de pruebas de Ejecución con EKETAL

en muchas aplicaciones de tipo concurrente y distribuido, además de que se comporta en tiempo de ejecución acorde a lo que se esperaba, claro esta que esta herramienta debe en algún momento evolucionar más para poder soportar sistemas aún más grandes que van a llegar a existir en un futuro.

Capítulo 4

Conclusión

Para finalizar esta investigación y después de haber profundizado sobre el amplio campo de Big-Data, se pudo conocer diferentes herramientas muy útiles para el manejo correcto de grandes volúmenes de datos, además de poder conocer componente por componente cada una de estas herramientas; también diferentes alternativas ya sean derivadas unas de otras, o alternativas nuevas que cumplen un el mismo propósito, pero dentro de todo esto falta profundizar más sobre los errores que se encuentran ya descubiertos e identificar tipos de errores aún escondidos, para poder llegar a conocerlos e incluso provocarlos debemos llegar a conocer como se emplea su ejecución, para esto llevamos a cabo ciertos experimentos bajo ciertas condiciones las cuales evidencian a como se sujeta estas herramientas según el caso propuesto, a parir de todo lo anterior da a concluir que cualquier persona puede conocer fácilmente sobre lo que es Big Data, además de que se puede conocer que rendimiento tiene el poder llegar a emplear estas soluciones según las necesidades que se estén recurriendo,también de evidenciar el potencial que posee y el que falta por explotar para que todas las nuevas soluciones sean más eficientes y más fáciles de manejar.

Bibliografía

[1] Jeffrey D. Ullman Alfredo Cuzzocrea, Domenico Saccà. Big data: A research agenda. *Proceedings of the 17th International Database Engineering & Applications Symposium*, pages 198–203, 2013.

[2] Juan Manuel Nieto Moreno. Introducción a la programación orientada a aspectos. 2016.

[3] Antonia Mª Reina Quintero. Visión general de la programación orientada a aspectos. 2000.

[4] A. Maheshwari. Big data essentials. 1st ed., 2016.

[5] Carmen De Maio Giuseppe Fenza Vincenzo Loia Mimmo Parente Alfredo Cuzzocrea, Giuseppe Fenza. Olap analysis of multidimensional tweet streams for supporting advanced analytics. *Proceedings of the 31st Annual ACM Symposium on Applied Computing Pages*, pages 992–999, 2016.

[6] Alfredo Cuzzocrea. Aggregation and multidimensional analysis of big data for large-scale scientific applications: Models, issues, analytics, and beyond. *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, 2015.

[7] Hotham Altwaijry Alexander Behm Vinayak Borkar Yingyi Bu Michael Carey Inci Cetindil Madhusudan Cheelangi Khurram Faraaz Eugenia Gabrielova Raman Grover Zachary Heilbron Young-Seok Kim Chen Li Guangqiang Li Ji Mahn Ok Nicola Onose Pouria Pirzadeh Vassilis Tsotras Rares Vernica Jian Wen Till Westmann Sattam Alsubaiee, Yasser Altowim. Asterixdb: A scalable, open source bdms. *Proceedings of the VLDB Endowment*, 7(14):1905–1916., 2014.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 1–13, 2004.

[9] Ashlee Vance. «hadoop, a free software program, finds uses beyond search». 2009.

[10] Xin J. Ding L. and Wang G. An efficient query processing optimization based on elm in the cloud. neural computing and applications. 27(1), 2014.

[11] Xin J. Wang X. Huang S. Zhang R. Ding L., Wang G. Commapreduce: An improvement of mapreduce with lightweight communication mechanisms. *Data & Knowledge Engineering*, 88:224–247., 2013.

[12] Yuan Yu. Andrew Birrell. Dennis Fetterly. Michael Isard., Mihai Budiu. Dryad: Distributed data-parallel programs from sequential building blocks. *Proceedings of the 2007 Eurosys Conference*, 224-247.:14 pag, Proceedings of the 2007 Eurosys Conference. doi: AssociationforComputingMachinery,Inc.

[13] Michael Carey Raman Grover. Data ingestion in asterixdb. *Proc. of the Int'l. Conf. on Extending Database Technology*, pages 605–616, 2015.