Diseño de una arquitectura para el procesamiento distribuido de grandes volúmenes de datos

Design of an architecture for the distributed processing of large volumes of data

Huanca Marín, Julio César
juliohuanca@unajma.edu.pe – Universidad Nacional José María Arguedas
https://orcid.org/0000-0001-8714-2623
Quina Quina, Luz Delia
dquina@unajma.edu.pe – Universidad Nacional José María Arguedas
https://orcid.org/0000-0002-5150-2575
Huancahuire Bravo, Claudio Isaias
claudio.huancahuire@unsaac.edu.pe – Universidad Nacional de San Antonio Abad del Cusco
Bravo Mendoza, Guido
gbravo@unamba.edu.pe – Universidad Nacional Micaela Bastidas de Apurímac
https://orcid.org/0000-0001-8852-2097

Recibido el 20/06/21 | Aceptado el 22/07/21
DOI: https://doi.org/10.47190/nric.v3i3.9

Resumen

Actualmente, Big Data se ha convertido en un concepto que está presente en muchas actividades, y su importancia es debido a que es utilizado en diversos aspectos que conduzcan a mejorar decisiones en el campo empresarial y gubernamental. Es posible analizar los grandes volúmenes de datos, tanto estructurados como no estructurados, que a cada día aumentan en los diferentes negocios y campos del conocimiento. Para obtener resultados satisfactorios es importante diseñar una arquitectura físicamente en base a Hardware Commodity (homogénea, heterogénea), escalable horizontalmente y con tolerancia a fallas. De esta manera, actualmente, con la evolución de las herramientas, es conveniente utilizar un híbrido donde la parte lógica trabaja con el Framework Apache Hadoop 2.0, que realiza el procesamiento de datos en paralelo (utilizando YARN), con almacenamiento HDFS (Sistema de Archivos Distribuidos sobre Hadoop) y agregando Spark para el tratamiento en memoria con respuestas en tiempo real y la utilización de recursos gráficos mediante Apache Ambari.

Palabras claves: *Ambari, Hadoop 2.0,YARN, MapReduce, Spark, HDFS, Distribución Paralela.*

Abstract

Currently, Big Data has become a concept that is present in many activities, and its importance is due to the fact that it is used in various aspects that lead to better decisions in the business and government field. It is possible to analyze the large volumes of data, both structured and unstructured, which are increasing every day in different businesses and fields of knowledge. To obtain satisfactory results, it is important to design an architecture physically based on Hardware Commodity (homogeneous, heterogeneous), scalable horizontally and with fault tolerance. In this way, currently, with the evolution of the tools, it is convenient to use a hybrid where the logical part works with the Apache Hadoop 2.0 Framework, which performs data processing in parallel (using YARN), with HDFS storage (File System Distributed on Hadoop) and adding Spark for in-memory treatment with real-time responses and the use of graphical resources through Apache Ambari.

Keywords: *Ambari, Hadoop 2.0, YARN, MapReduce, Spark, HDFS, Parallel Distribution*.

Como citar: Huanca-Marín, J.C., Quina-Quina, L.D., Huancahuire-Bravo, C.I. & Bravo-Mendoza, G. (2021). Diseño de una arquitectura para el procesamiento distribuido de grandes volúmenes de datos. ÑAWPARISUN – Revista de Investigación Científica de Ingenierías, 3(3), 73-77.

Introducción

Datos e información son irrefutablemente esenciales para una toma de decisión óptima en este siglo XXI. Hoy en día no son únicamente con formatos estructurados con equivalencia al 20%, sino con formatos semi-estructurados y no estructurados equivalentes al 80%.

Los datos provenientes de Internet y Web 2.0, específicamente de algunas fuentes como Google, Amazon, Redes Sociales (Facebook, Twitter, WhatsApp), han ido pasando rápidamente desde PetaByte, ExaByte, ZettaByte, YottaByte para llegar a BrontoByte (ver Figura 2). Por ello es inevitable e innegable la proliferación a un ritmo exponencialmente alto, de naturaleza heterogénea y con la velocidad a la que se generan dificultan una toma de decisión óptima.

La solución diseñada en el presente trabajo, no es por medio de una supercomputadora de Empresas como IBM, Amazon Web Service, Oracle, Microsoft Azuze, Cloudera y HortonWorks, sino a través de computadoras de precios módicos, unidas en redes utilizando tecnología Hadoop (código abierto y gratuito) la cual permite procesar a las aplicaciones en miles de nodos escalables, agregando la velocidad de trabajo por el Framework Spark.

Figura 1. Datos en ascendencia para Big Data, fuente consultec [1].

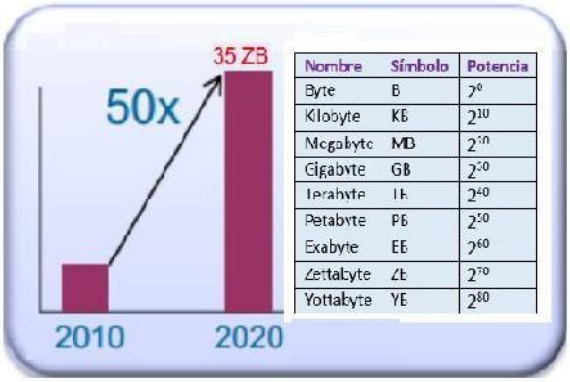

Figura 2. Datos en ascendencia para Big Data, fuente IBM [12].

Trabajos Relacionados

Para Big Data, una herramienta de almacenamiento y procesamiento es Apache Hadoop en su versión 1 pero surgiendo limitaciones de eficiencia en escalabilidad, costos, tolerancia a fallos. Así que se mejora con Hadoop versión 2 o denominado YARN (otro administrador de recursos) separando la administración de recursos con trabajos de tareas exclusiva del modelo maestro-esclavo.

Generación de Excel, SQL, Datawarehouse, DataMinig, dan inicio a una generación exclusivamente de datos no estructurados exorbitantes sobre el Framework Hadoop [10].

Los Framework Hadoop y Spark permanecen por su performance en este tipo de almacenamiento y procesamiento en paralelo y distribuido sobre miles de nodos escalables horizontalmente generando así un clúster [3, 4].

La velocidad de Spark en ejecución es hasta 100 veces más rápido que Hadoop o 10 veces más rápido en disco. La comparación de estos dos Framework es crítica en tiempo de ejecución sus características en mención en la Tabla 1 [5].

Tabla 1
Comparativa de Hadoop y Spark [5].

Descripción	***MapReduce***	***Spark***
Almacenamiento	Sobre disco	Sobre memoria
Procesa datos	Batch	Hibrido Batch y stream
Procesa en tiempo real	No	si
Código escrito	Scala, java y python	Complejo y largo
Tiempo de ejecución	Lento	100 veces más

Metodologia

La metodología propuesta contempla varios aspectos, que se presentan a continuación:

Diseño del clúster

Hadoop permite el procesamiento distribuido de grandes conjuntos de datos que usa modelos de programación simple, diseñado para escalar desde servidores únicos a miles de nodos, presentamos el diseño en la Figura 3.

Este diseño de basa en modelo maestro-esclavo en donde el maestro está configurado con la IP 192.168.30.11 los esclavos con IP 192.168.30.12 y 192.168.30.13 máscara de red 255.255.255.0 y puerta de enlace 192.168.30.1. Pero la comunicación de este clúster va más allá de una configuración de red clase C. Está configurado mediante el protocolo SSH con el comando ssh keygen, generando así claves públicas (id_rsa.pub) y privadas (id_rsa), ubicados dentro de la carpeta .ssh, luego con el comando cp copiamos en un archivo denominado autorized_keys en todos los nodos del clúster para su comunicación segura y dicho archivo se envía de nodo a nodo con el comando scp.

Editar con editores (nano, gedit o vi) los archivos core.site.xml para denominar el nombre del maestro,

hdfs.site.xml para la cantidad de réplicas en bloques y yarn-site.xml, para el proceso map, suffle-sort y reduce.

Levantamos el Framework Hadoop. Primero YARN con el comando start-yarn.sh procede a ejecutar los servicios ResourceManager, NameNode, SecondaryNameNode y Jps. Segundo HDFS con el comando hdfs-dfs.sh procede a ejecutar DataNode, NodeManager y Jps, mostrar estos servicios con el comando jps.

Administramos el clúster desde el nodo maestro con la url y puerto nodo1:8088/cluster y los esclavos nodo1:50070

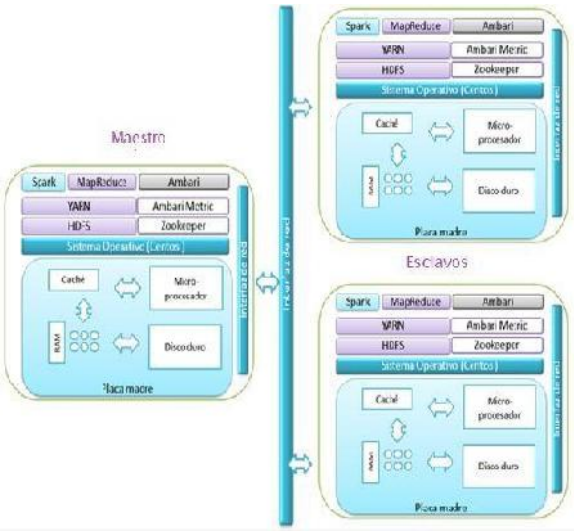

Figura 3. Diseño de clúster, fuente propia.

Hadoop version 2

Se compone de:

A. Hadoop Common.

Las librerías o bibliotecas Hadoop.

B. HDFS

Está diseñado para almacenar de forma fiable archivos muy grandes en bloques, todos los bloques excepto el último bloque son del mismo tamaño, si en caso excediese a la configuración de bloques a 128, 256 MB. Una aplicación puede específicar el número de réplicas de un archivo. El nodo maestro toma todas las decisiones de replicación.

C. YARN (Map, Suffle-Sort y Reduce)

Proporciona una solución a los problemas, como es el soporte distribuido y la programación paralela en 3 procesos.

1. Proceso MAP: Tareas que se ejecutan en los nodos, cada tarea MAP ataca a un solo bloque de datos HDFS.

2. Proceso SHUFFLE Y SORT: Ordena y consolida los datos intermedios (temporales) que generan los MAP, como clave-valor

Proceso REDUCER: Operan sobre los datos intermedios Shuffle/Sort Genera la salida final.

Spark

A. Usa HDFS para almacenar los datos.

B. Se ejecuta en memoria.

C. Tiene un Direccionamiento adelantado un motor a cíclico de ejecución gráfica 3 (DAG).

D. Usa Resilient Distributed Dataset (implementa estructuras de datos en memoria) y almacena los datos en caches.

También puede ser utilizado para trabajos en lote, streaming, iterativos y Maching Learning

Diseño actual

El rendimiento del diseño depende principalmente de sus parámetros de configuración de modo que podemos reducir el tiempo de ejecución y procesamiento en disco.

Su ajuste basado principalmente en la CPU, velocidad deE/S de disco y componentes de tráfico de red.

Spark mejora el rendimiento en un 32,97%, con la arquitectura de la Tabla 2.

Tabla 2
Recurso de clúster homogéneo.

N°	Descripción	Características
1	Interfaz Virtual	Oracle Virtual Box 5.0.16
2	Sistema Operativo	Centos 64 bits
3	Disco Duro	100 GB
4	Arquitectura	64 bit
5	Procesador	Intel Core i5-2410M
6	RAM	2 GB
7	Hadoop	V. 2.7.2
8	Java	JDK 1.8
9	Red	Clase C

Los archivos de configuración como mapred-site.xml, core-site.xml, hdfs-site.xml se configurán por valores predeterminados con factor de duplicación 2 y tamaño de bloque predeterminado 128 MB.

Utilizamos datos de un libro de Don Quijote de la macha, hicimos un conjunto de experimentos para evaluar el rendimiento en la Tabla 3 y Figura 4.

Tabla 3
Cantidad de palabras, en ambos Framework.

N° de palabras	Hadoop (en seg)	Spark (en seg)
100	84	31.73
1000	95	33.17
10000	101	38.19
100000	108	39.84
1000000	122	42.52

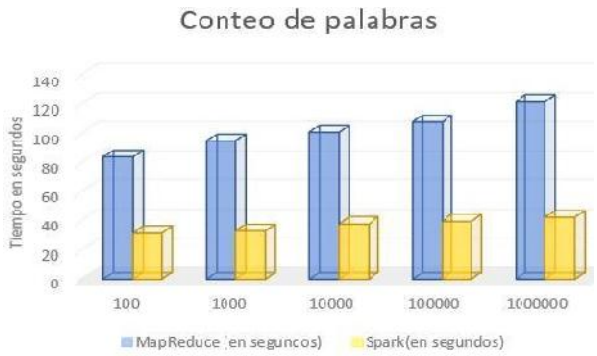

Figura 4. Resultado de rendimiento

Diseño previo

El rendimiento de este diseño se basa en la Tabla 4. La configuración experimental de esto fue en un único nodo con la siguiente información, como se muestra a continuación:

Tabla 4
Recurso de un simple nodo [9].

N°	Descripción	Características
1	Interfaz Virtual	Oracle Virtual Box 5.0.16
2	Sistema Operativo	Cloudera 5.9.0
3	Disco Duro	1 TB
4	Arquitectura	64 bit
5	Procesador	Intel Core i5- 3230M
6	RAM	8 GB

Su análisis de conteo de palabras en Hadoop Map Reduce y Spark: el programa de conteo de palabras se usa para contar la frecuencia de cada palabra que está ocurriendo en un documento. La Figura 5 es la representación gráfica del tiempo de ejecución tanto en Hadoop como en Spark. En este experimento, aumentamos el número de palabras en cada iteración y registramos las lecturas. Tabla 5.

Tabla 5
Cantidad de palabras, en ambos Framework [9].

N° de palabras	Hadoop(en seg)	Spark(en seg)
100	79	28.841
1000	91	31.185
10000	96	35.181
100000	103	36.969
1000000	116	39.569

Figura 5. Resultado de rendimiento [9].

Comparación entre diseños

Nodos

- Diseño propio usa en total de 3 nodos, uno como servidor y dos como esclavos.
- Diseño previo usa 1 nodo pre configurado de Cloudera.

RAM

- Diseño propio usa en cada nodo 1.5 GB de RAM.
- Diseño previo usa un nodo de 8 GB de RAM.

Arquitectura

- Diseño propio usa arquitectura de 32 bits.
- Diseño previo usa arquitectura de 64 bits.
- Disco Duro o Diseño propio usa 100 GB
- Diseño previo usa 1TB.
- Red o Diseño propio, Clase C.
- Diseño previo sin Red.

Porque Hadoop tiene 3 tipos de evaluación, en un nodo, en virtualización y en clúster real.

Conclusión

Se ha realizado un estudio de herramientas existentes, y se propone el diseño de una arquitectura que emplea herramientas que pueden trabajar en conjunto para mejorar el desempeño en el análisis, procesamiento y tratamiento de la exorbitante cantidad de información digital que existe actualmente. Logrando de esta manera optimizar, en comparativa con alternativas anteriores, donde se usan nodos preconfigurados. La presente investigación se desarrolla desde la instalación, configuración y puesta en marcha. Usualmente los modelos predefinidos usan, en el aspecto lógico más memoria, y en el aspecto físico demasiado recurso, por lo tanto, se ha visto conveniente emplear un modelo simplificado que permite utilizarse en máquinas de poco recurso, hasta de arquitecturas de 32 bits y hasta con 1,5GB de RAM y en sistemas operativos heterogéneos.

La arquitectura propuesta es para ser utilizada en clústeres, y representa un inicio para para trabajo Interactivo, Maching Learning, Internet de las cosas, Deep Learning e Inteligencia Artificial.

Referencias

[1] www.consultec.es.

[2] Sarah Shaikh,Deepali Vora, ”YARN versus MapReduce A comparative study”, IEEE. 2016.

[3] Vladyslav Taran, Oleg Alienin, Sergii Stirenko, and Yuri Gordienko ”Performance Evaluation of Distributed Computing Environments with Hadoop and Spark Frameworks ”, IEEE International Young Scientists Forum on Applied Physics and Engineering. 2017.

[4] I. Chebbi, W. Boulila, N. Mellouli ”A comparison of big remote sensing data processing with Hadoop MapReduce and Spark ” ,Advanced Technologies for Signal and Image Processing (ATSIP), 2018 4th International Conference-IEEE. 2018.

[5] Yassir Samadi, Mostapha Zbakh ”Comparative study betweenHadoop and Spark based on Hibench benchmarks”,Cloud Computing Technologies and Applications (CloudTech), 2016 2nd International Conference on IEEE 2016.

[6] Yunping Feng, Haopeng Chen ”Optimization of spark storagesolutions” ,Progress in Informatics and Computing (PIC), 2016 International Conference on IEEE 2016.

[7] Bilal Akil, Ying Zhou, “On the usability of Hadoop MapReduce, Apache Spark and Apache flink for data science”,Big Data (Big Data), 2017 IEEE International Conference on IEEE 2017.

[8] [8] Yan Li, Hongbo Wang, Yangyang Li “Research on query analysis and optimization based on spark”, Computer Science and Network Technology (ICCSNT), 2017 6th International Conference on IEEE 2017.

[9] Akaash Vishal Hazarika “Performance Comparison of Hadoop and Spark Engine”, International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) IEEE 2017.

[10] http://hadoop.apache.org/

[11] https://spark.apache.org/

[12] https://www.ibm.com