Ejemplos de Aplicabilidad de Giraph y Hadoop para el Procesamiento de Grandes Grafos

Sebastián A. Valenzuela⁽¹⁾, Cristian L. Vidal⁽²⁾, Jenny D. Morales⁽²⁾ y Leopoldo P. López⁽³⁾

(1) Departamento de Computación e Informática, Facultad de Ingeniería, Universidad de Playa Ancha. Avenida Leopoldo Carvallo 270, Playa Ancha. Valparaíso, Chile. (e-mail: sebastian.valenzuela.f@alumnos.upla.cl)

(2) Facultad de Ingeniería, Universidad Autónoma de Chile, Sede Talca, 5 Poniente 1670, Talca-Chile. (e-mail: cristian.vidal@uautonoma.cl; jmoralesb@uautonoma.cl)

(3) Facultad de Economía y Negocios, FEN, Instituto de Investigación y Desarrollo Educacional, IIIDE, Universidad de Talca, Campus Lircay, Avenida Lircay S/N, Talca-Chile. (e-mail: llopez@utalca.cl)

Recibido Ene. 22, 2016; Aceptado Mar. 24, 2016; Versión final Abr. 6, 2016, Publicado Oct. 2016

Resumen

Este artículo presenta una comparativa del rendimiento de las herramientas Hadoop y Giraph para el procesamiento de grandes volúmenes de información o Big Data con el fin de mostrar su utilidad para el procesamiento de Big Graph. El análisis y procesamiento de grandes volúmenes de información representa un verdadero desafío en la actualidad. Ya existen metodologías y herramientas libres para el procesamiento de Big Data como las mencionadas: Hadoop para el procesamiento de grandes volúmenes de datos, principalmente no estructurados, y Giraph para el procesamiento de grandes grafos o Big Graph. En esta comparativa, este trabajo presenta un análisis del costo en tiempo de ejecución práctico de la implementación del algoritmo PageRank, el cual permite clasificar páginas Web según su relevancia, y de algoritmos para encontrar un árbol de expansión mínima en un grafo. Los experimentos muestran que el uso de Giraph para el procesamiento de Big Graph reduce el tiempo de ejecución en un 25% respecto a los resultados con el uso de Hadoop.

Palabras clave: Hadoop, Giraph, Grafos, MapReduce, Big Data, Big Graph.

Applicability of Giraph and Hadoop for the Processing of Big Graph

Abstract

This article presents a comparison of the performance of the tools Hadoop and Giraph for the analysis and processing of large volumes of information or Big Data, with the aim of showing their usefulness for Big Graph processing. The analysis and processing of large volumes of information represents a real challenge nowadays. There already exist Big Data methodologies and free processing tools such as those mentioned above: Hadoop for processing large volumes of data, mainly non-related data, and recently Giraph for processing large graphs or Big Graph. In this comparison, this paper presents an analysis of the execution time cost for the practical implementation of the PageRank algorithm, which classifies Web pages according to their relevance, and of algorithms to find the minimum spanning tree in a graph. Experiments show that the use of Giraph for processing Big Graphs reduces the execution time by 25% with respect to the results obtained using Hadoop.

Keywords: Hadoop, Giraph, Grafos, MapReduce, Big Data, Big Graph

INTRODUCCIÓN

De acuerdo a IBM (IBM, 2015), diariamente se crean 2.5 quintillones de bytes en el mundo (1030 bytes) (IBM, 2015). De manera similar, International Data Corporation IDC (Tuner et al., 2015) estima que el tamaño del “universo digital” era de 4.4 zettabytes en el 2013 y pronostica un crecimiento de diez veces para este valor al 2020. Justamente, ante este crecimiento exponencial de la información, se popularizó el término Big Data. Según Gartner (2015), Big Data se define como: “grandes volúmenes, alta velocidad y gran variedad de información que exigen, formas innovadoras y rentables de procesamiento de la información para mejorar la comprensión y la toma de decisiones”.

Aun cuando exista una tendencia al aumento en los volúmenes de espacio para el almacenamiento de información, no hay un aumento equivalente en la velocidad de acceso y procesamiento de grandes volúmenes de información. Una solución directa a este problema, para reducir el tiempo de acceso y procesamiento de datos, es el uso de clúster de computadores (White, 2015). En este contexto, existen varios sistemas distribuidos que permiten combinar datos desde múltiples fuentes, pero con un alto costo de análisis y desarrollo debido principalmente a la concurrencia, sincronización de tareas, acceso y transferencia de datos (White, 2015). Aquí es donde Hadoop (Hadoop, 2015), un framework opensource basado en MapReduce (Dean y Ghemawat, 2008) es de gran importancia y utilidad.

Según Lotz (2015), el procesamiento de Big Data está muy asociado a información no estructurada, esto es, datos directamente no relacionados. Sin embargo, en las redes sociales actuales, como Facebook y Twitter, existen vínculos entre los nodos componentes de las mismas, para lo cual su representación directa como grafos es adecuada. Sin embargo, por la estructura y asociaciones de grafos, MapReduce y Hadoop no son óptimos para su procesamiento, por el alto costo de la entrada/salida de información y la posibilidad de requerir muchas fases de MapReduce encadenadas. En este contexto, Giraph (Giraph, 2015) es una contraparte opensource de Pregel (Malewicz et al, 2010), sistema de procesamiento de grafos desarrollado por Google, para el procesamiento de grandes grafos.

Pese a que la generación de datos crece a un ritmo del 60% anual, las tecnologías relacionadas con Big Data como Hadoop y Giraph aún están en fase de desarrollo en Chile. Según IDC Chile, “Aun falta dar a conocer que es lo que puede hacer Big Data por las compañías nacionales y como podría llegar a impactar en las diversas líneas de negocios”, y que solo el 17% de las empresas están implementando o piensan implementar Big Data en el corto plazo en Chile (IDC1, 2015; IDC2, 2015). Justamente, el principal objetivo de este trabajo, es presentar una revisión de la aplicación de Hadoop y Giraph para el procesamiento de grafos, y así resaltar las ventajas prácticas que tiene Giraph sobre Hadoop al momento de resolver problemas con datos relacionados y que usualmente se solucionan mediante algoritmos iterativos, como son las estructuras de grafos. Para esto se realizará una comparativa entre Hadoop y Giraph del rendimiento de soluciones algorítmicas a un problema clásico (árbol de expansión mínima) y a un problema más actual (clasificación de páginas web). Cabe señalar que esta investigación es una extensión de (Valenzuela y Vidal, 2015) con el fin de reafirmar los resultados y conclusiones ya obtenidas, además de mostrar su aplicabilidad para el procesamiento de Big Data en Chile.

BIGDATA Y GRAFOS

Se describe una breve historia de la herramienta HADOOP y se explica su estructura y funcionamiento. Luego se introduce el concepto de grafos y se detalla la filosofía de funcionamiento de Pregel y Giraph.

Hadoop

Hadoop (Hadoop, 2015) es un framework opensource (marco de trabajo de código abierto) creado con el fin de conseguir una computación segura, escalable y distribuida. Hadoop está basado en los documentos de Google para MapReduce (Dean y Ghemawat, 2008) y Google File System (GFS) (Ghemawat et al., 2003) y permite el procesamiento distribuido de grandes conjuntos de datos por medio de clúster de computadoras usando simples modelos de programación.

MapReduce

Hadoop implementa un paradigma computacional llamado MapReduce. Algorítmicamente está basado en el concepto de “divide y vencerás”, es decir, divide el problema en pequeños trozos para su procesamiento en paralelo y así obtener soluciones en un entorno distribuido (Gunarathne, 2015). MapReduce toma un conjunto de pares key-value iniciales, y produce un conjunto final de pares key-value, y su modelo de programación está compuesto por una función Map y una función Reduce: Map recibe un par key-value de entrada y produce un conjunto de pares key-value intermedios, entonces MapReduce agrupa todos los

valores intermedios asociados con la misma key intermedia como entrada de la función Reduce. De esta forma, la función Reduce acepta una key intermedia y un conjunto de valores para esa key con el fin de mezclar estos valores y así formar un nuevo conjunto de salida final (Dean y Ghemawat, 2008).

Normalmente, una solución en MapReduce se realiza en cinco etapas. Concretamente, i) splitting, los datos se dividen en múltiples partes y se entregan a cada mapper; ii) mapper, ejecuta la función map encargada de procesar los datos; iii) combiner, funciona directamente sobre la salida de los mappers para la agregación local; iv) shuffle, responsable de barajar y ordenar los pares key-value para la función reduce, y v) reduce, en el último paso se reducen todos los valores con la misma key intermedia, para generar pares key-value finales (Dean y Ghemawat, 2008). Figura 1 (Dean y Ghemawat, 2008) muestra este proceso de forma general.

Fig. 1: Flujo de ejecución MapReduce.

HDFS

Hadoop incluye un sistema de archivos distribuido llamado Hadoop Distributed File System (HDFS) que puede manejar grandes cantidades de datos. El patrón más eficiente de procesamiento de datos es escribir una vez y leer muchas veces (White, 2015). Como en un sistema de archivo de un único disco, los archivos en HDFS se dividen en bloques, que se almacenan como unidades independientes. Un clúster HDFS tiene dos tipos de nodos que operan en un patrón master-worker: un Namenode (el máster) y varios Datanodes (workers) (Gunarathne, 2015). El Namenode almacena y administra los metadatos del sistema de archivos, y conoce los Datanodes en la que se encuentran todos los bloques reales para un archivo dado (NameNode, 2015). Cuando se recuperan los datos, el cliente se contacta con el Namenode para obtener la lista de los lugares de los datos solicitados y luego se contacta directamente con el Datanode para recuperar los datos reales (Gunarathne, 2015). Figura 2 muestra este proceso. Tanto MapReduce y HDFS están diseñados de manera que los errores se gestionan automáticamente por el framework de Hadoop (Hadoop2, 2015).

Fig. 2: Arquitectura HDFS.

En la figura 2: Client' data: Datos del cliente - Read/write: Leer/escribir – Client: Cliente - Metadata Access: Acceso a metadatos - Stores: almacena - For every block in HDFS: Metadata [Name.replication]: Por cada bloque en HDFS: [Nombre.replica] – Data Access: Acceso a datos – Heartbeat and block report: Latido y reporte de bloqueo – Data: Datos – Replication: Replicación.

Grafos y Pregel / Giraph

Los grafos son un tipo de datos abstracto finito, que consiste en un conjunto de aristas y nodos o vértice. Una arista entre dos nodos x e y puede ser descrita matemáticamente como arista (x, y) (Lotz, 2015). Justo, Lotz (2005) también señala que, en el análisis de Big Data, el procesamiento de grafos es considerado computacionalmente difícil debido a su naturaleza irregular. Además, tal y como indican (Brin y Page, 2015), procesamiento de grafos es no adecuado con sistemas de propósito general como MapReduce.

Para el procesamiento de grandes grafos (Big Grafos), existen Pregel y Giraph los que se basan en el modelo 'Paralelo Síncrono a Granel' (Bulk Synchronous Parallel - BSP) introducidos por Valiant (1990). BSP es un modelo de computación paralela, en el cual los cálculos son divididos en superpasos (supersteps) separados por barreras globales (global barriers) (Han et al., 2014). Esta idea se representa en la Figura 3.

Fig. 3: Modelo BSP, con tres superpasos y tres workers.

Pregel

En el 2010 Google anunció Pregel (Malewicz et al., 2010), un framework para el procesamiento distribuido de grafos basado en BSP. Su objetivo era proporcionar un cierto nivel de abstracción para que los programadores no tengan que hacer frente a la gestión de memoria distribuida o a la sincronización (Voulgaris y Martella, 2015).

El paradigma utilizado por Pregel es "pensar como vértice". El cálculo se especifica en términos de lo que cada vértice tiene que calcular; las aristas son los canales de comunicación para la transmisión de los resultados desde un vértice a otro. En cada superpaso, un vértice puede ejecutar una función definida por el usuario y cambiar su estado de activo a inactivo. Un vértice puede votar para detener un superpaso y ser despertado cuando reciba un mensaje (ver Figura 4) (Han et al., 2014).

Fig. 4: Estado del vértice (Malewicz et al., 2010)

Giraph

Apache Giraph (Giraph, 2015) es una alternativa opensource de Pregel. En Pregel, para apoyar el multihilo, a cada worker se le asigna varias particiones del grafo. Durante cada superpaso, un par de worker disponibles calcula hilos con particiones no calculadas. Cada worker mantiene su propio almacén de mensajes para guardar todos los mensajes entrantes. Para reducir la contención en el almacén y utilizar los recursos de red eficientemente, cada hilo de cómputo tiene un buffer caché para todos los mensajes salientes (Han y Daudjee, 2015). Para implementar el modelo BSP, los workers mantienen dos almacenes en cada superpaso, uno para los mensajes previos y otro para los actuales (Han y Daudjee, 2015). Giraph soporta diferentes estructuras de datos para la lista de vértices adyacente (Han et al., 2014). Es importante destacar que Giraph se ejecuta como tareas MapReduce y usa Zookeeper para coordinar las barreras globales (Han y Daudjee, 2015).

El cálculo en Giraph está organizado en una serie de superpasos idealizados como una iteración de un determinado algoritmo. En cada superpaso, un vértice puede enviar mensajes hacia los otros vértices, acceder al valor de su vértice o arista, y votar para detenerse. Al comienzo de la computación (superpaso 0), todos los vértices se inician con estado activo. Un vértice vota para detenerse porque decidió, desde su punto de vista local, que su trabajo está hecho y que el cálculo puede concluir. La entrega de un mensaje cambia el estado de un vértice de inactivo a activo. Giraph termina el cálculo cuando todos los vértices se encuentran inactivos y no existen mensajes para enviar (Martella et al., 2015).

ALGORITMOS

Para la comparación de rendimiento en el procesamiento de grafos entre Hadoop y Giraph, se consideran dos situaciones algorítmicas:

Clasificación de páginas - PageRank: PageRank es un algoritmo creado por Google (Brin y Page, 2015) para clasificar páginas web, basado en la idea de que las páginas web más importantes probablemente reciben más enlaces desde otras páginas web (Han et al., 2014). La web se conceptualiza como un grafo dirigido, donde las páginas son nodos, y existen arcos entre las páginas si es que hay uno o más enlaces (links) entre estas páginas. Típicamente, PageRank es iterado hasta la convergencia, es decir, cuando los valores de PageRank para cada nodo ya no cambian. Por lo tanto, al final de cada iteración, PageRank debe comprobar si se ha alcanzado la convergencia. Por otra parte también se pueden incluir la ejecución de un número fijo de iteraciones (Lin y Dyer, 2010).

Árbol de recubrimiento mínimo – Algoritmo de Kruskal y Boruvka: Para obtener un árbol de recubrimiento mínimo de un grafo, usualmente se utilizan los algoritmos de Kruskal y Prim (Kruskal, 1956; Prim, 1957). En este trabajo, se utiliza una implementación del algoritmo de Kruskal en Hadoop, y una implementación que es parte de soluciones ejemplos de Giraph para encontrar un árbol de expansión mínima distribuido, algoritmo Boruvka (Han et al., 2014).

Para realizar las pruebas de ejecución de estas soluciones, por las limitaciones de hardware, se decidió utilizar Hadoop en modo “Single Node Setup” (Hadoop3, 2015) en una máquina con dos CPUs y 8 GB de memoria RAM. Además, se utilizó Hadoop 2.4.0 y Giraph 1.1 en Ubuntu 15.10. Para la comparación, se realizarán pruebas de PageRank con implementaciones en Hadoop y Giraph, y de Kruskal y Boruvka en Hadoop y Giraph, respectivamente. Además, se utilizara un conjunto de datos representativo, constituido por un grafo con 31 vértices y 82 aristas de peso uno para el caso de PageRank. Si bien estos datos son pequeños, en términos de Big Data, estos son extrapolables en el sentido comparativo entre ambas herramientas en términos de tiempo de ejecución para cada una de las tres comparaciones.

RESULTADOS

PageRank

Tanto PageRank en Hadoop como en Giraph fueron definidos con 30 iteraciones/superpasos respectivamente. Los tiempos de cálculo de cada una de estas pruebas se representan en la figura 5 para Hadoop y figura 6 para Giraph.

Kruskal y Boruvka

El algoritmo de Kruskal encontró el árbol recubridor mínimo (MST por sus siglas en inglés) en torno a los 3 segundos en Hadoop (figura 7) en una sola iteración, mientras que su implementación en Giraph encontró la solución en torno a 0.7 segundos en 2 superpasos (figura 8).

Fig. 5: Resultados del algoritmo PageRank en Hadoop.

Fig. 6: Resultados del algoritmo PageRank en Giraph.

Como una forma de justificar los resultados obtenidos, con el fin de procesar todo el grafo, MapReduce realiza muchas tareas encadenadas hasta lograr su objetivo, y cada tarea implica entradas y salidas a disco. Una iteración en MapReduce puede ser idealizada como un superpaso en el enfoque Giraph (Lotz, 2015). Asumiendo esto, se puede realizar una aproximación estableciendo un tiempo promedio por iteración entre el tiempo total y la cantidad de iteraciones/superpasos, para cada una de las cuatro pruebas. Estos resultados se presentan en la tabla 1. Figura 9 muestra una comparación de los tiempos promedios para los cálculos efectuados por Hadoop y Giraph con el algoritmo de Kruskal y Boruvka, respectivamente, para encontrar el árbol de expansión mínima de un grafo.

Claramente, al comparar los tiempos asociados a PageRank, la solución Giraph es por lo menos 4 veces más rápida que Hadoop. Esta diferencia aumenta a 8 veces en el caso de Kruskal en Hadoop y Boruvka en Giraph para encontrar un árbol de expansión mínima de un grafo como muestra la Figura 8.

Fig. 7: Resultados del algoritmo Kruskal y Boruvka en Hadoop.

Fig. 8: Resultados del algoritmo Kruskal y Boruvka en Giraph.

Fig. 9: Comparativa entre Hadoop y Giraph para los algoritmos de PageRank y Kruskal.

Tabla 1. Resultados de Experimentos de Tiempo de Ejecución de Algoritmos PageRank, Kruskal y Boruvka en Hadoop y Giraph.

<i>Prueba</i>	<i>Tiempo total (ms)</i>	<i>Iteraciones superpasos</i>	<i>Promedio (ms)</i>
PageRank Hadoop	32.238	30	1.074,6
PageRank Giraph	7.925	30	264,2
Kruskal Hadoop	3.096	1	3.096
Boruvka Giraph	718	2	359

CONCLUSIONES

De acuerdo a los resultados, Giraph presenta una eficiencia 4 veces superior sobre Hadoop a lo que se refiere al tiempo de cómputo del algoritmo de PageRank. Esta ventaja aumenta a casi a 8 veces cuando se trata de un algoritmo para encontrar el árbol de expansión mínima. Esto se debe principalmente a la ventaja que posee Giraph al poder mantener una comunicación global entre cada superpaso sin requerir entradas y salidas E/S a disco ya que cada nodo puede intercambiar mensajes con los otros (Lin y Dyer, 2010). Esto, por supuesto, no es posible en Hadoop, ya que MapReduce no provee ninguna herramienta para la comunicación global y directa entre los participantes en cada una de sus iteraciones. Además, la naturaleza de los grafos hace que el cómputo de cada nodo dependa de sus nodos vecinos, lo que implica que soluciones MapReduce tengan que realizar muchas iteraciones encadenadas, lo que genera un excesivo tráfico E/S.

También se debe dar a entender que Giraph no es una herramienta que viene a reemplazar a Hadoop, más bien debe ser visto como un complemento para un problema determinado. Hadoop posee un ecosistema (White, 2015) muy completo, que se expande día a día, por esta razón es importante conocer las ventajas que tienen cada una de estas nuevas tecnologías para poder asimilarlas de forma correcta en Chile.

Tal y como señalan (IDC1, 2015; IDC2, 2015), existe un gran problema producto del poco conocimiento existente en Chile sobre los inconvenientes generados por la Big Data, como así también, la poca información existente para el uso de estas nuevas herramientas. Esto es algo que debe cambiar en el futuro próximo, ya que Chile no está exento de los grandes problemas relacionados con la Big Data.

Como el foco principal de este trabajo es una comparativa práctica entre Hadoop y Giraph para problemas de grafos, un trabajo futuro de los autores es realizar una comparativa entre Giraph y herramientas actuales de BigData como Apache Spark (Spark, 2016) y Flink (Flink, 2014), en escenarios reales y de investigación tal y como procesamiento de modelos, foco actual de Doctorado de uno de los autores. Dado que el foco principal de Giraph es procesamiento de grafos, se requiere comprobar sus potenciales ventajas de rendimiento.

REFERENCIAS

- Brin, S. y L. Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Computer Science Department, Stanford University, Stanford (en línea), 1999. <http://infolab.stanford.edu/~backrub/google.html>. Acceso: 26 de Mayo de 2015
- Dean, J. y S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, Magazine Communication of the ACM – 50th anniversary issue: 1958 – 2008, 51 (1), pp. 197-113, Enero (2008)
- Flink, *Apache Flink* (en línea: <https://flink.apache.org/>, acceso: 4 de Abril 2016), The Apache Software Foundation (2014)
- Gartner, *Gartner IT Glossary* (en línea). <http://www.gartner.com/it-glossary/big-data>. Acceso: 8 de Mayo (2015)
- Ghemawat, S., H. Gobioff y ST. Leung, *The Google File System*, Proceedings of SOSP'03, ACM SIGOPS, 37 (5), pp 29-43, Diciembre (2003)
- Giraph, Apache Giraph (en línea), 2015. <http://giraph.apache.org/>. Acceso: 30 de Abril (2015)
- Gunarathne, T., *Hadoop MapReduce v2 Cookbook*, 2nd Edition, Packt Publishing (2015)
- Hadoop, *Apache Hadoop* (en línea), 2015. <https://hadoop.apache.org/>. Acceso: 30 de Abril (2015)
- Hadoop2, Hadoop Wiki - Apache Hadoop (en línea), 2011. <http://wiki.apache.org/hadoop/>. Acceso: 16 de Mayo (2015)

- Hadoop3, *Hadoop 1.2.1 Documentation - Single Node Setup* (en línea), 2015. URL: https://hadoop.apache.org/docs/r1.2.1/single_node_setup.html. Acceso: 2 de Junio de 2015
- Han, M. y K. Daudjee, *Giraph Unchained: Barrierless Asynchronous Parallel Execution in Pregel-like Graph Processing Systems*, Proceedings of the VLDB Endowment, 8 (9), pp. 950-961 (2015)
- Han, M., K. Daudjee, K. Ammar, M. T. Özsu, X. Wang y T. Jin, *An Experimental Comparison of Pregel-like Graph Processing Systems*, Proceedings of the VLDB Endowment, 7 (12), pp. 1047-1058 (2014)
- IBM, *Bringing big data to the enterprise* (en línea: <http://www.ibm.com/software/data/bigdata/what-is-big-data.html>, acceso: 4 de Mayo 2015) (2015)
- IDC1, *IDC: Analyze the Future - MOBILE FIRST: LA TENDENCIA QUE DOMINARÁ LA INDUSTRIA TIC ESTE AÑO* (en línea: <http://cl.idclatin.com/releases/news.aspx?id=1609>, acceso: 11 de Mayo 2015), Marzo (2014)
- IDC2, *IDC: Analyze the Future - BIG DATA REGISTRARÁ UNA INVERSIÓN MUNDIAL DE US\$12.6 MIL MILLONES EN 2014* (en línea: <http://cl.idclatin.com/releases/news.aspx?id=1666>, acceso: 11 de Mayo 2015) Mayo (2014)
- Kruskal, J. B., *On the shortest spanning subtree of a graph and the traveling salesman problem*, Proceedings of the American Mathematical society, 7(1), pp. 48-50 (1956)
- Lin, J. y C. Dyer, *Data-Intensive Text Processing with MapReduce*, Synthesis Lectures on Human Language Technologies, 3 (1), pp. 1-177 (2010)
- Lotz, M. A., *Dynamic Graph Computations using Parallel Distributed Computing Solutions*, School of Electronic Engineering and Computer Science, University of London (en línea: <http://www.marcolotz.com/wp-content/uploads/2014/05/LotzReport.pdf>, acceso: 18 de Mayo 2015) (2013)
- Malewicz, G., M. Auster, J. Bik, J. Dehner, I. Hor, N. Leiser G. Czajkowski, *Pregel: A System For Large-Scale Graph Processing*, in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, ACM, 2010, pp. 135-146 (2010)
- Martella, C., D. Logothetis, y R. Shaposhnik, *Practica Graph Analytics with Apache Giraph*, Apress, 1 edition (2015)
- NameNode, *Hadoop Wiki – NameNode* (en línea: <http://wiki.apache.org/hadoop/NameNode>, acceso: 16 de Mayo 2015) (2011)
- Prim, R. C., *Shortest connection networks and some generalizations*, Bell system technical journal, 36(6), pp. 1389-1401 (1957)
- Spark, *Spark: Lightning-fast cluster computing* (en línea: <http://spark.apache.org/>, acceso: 4 de Abril 2016), Apache Spark, Marzo (2016)
- Turner, V., J. F. Gantz, D. Reinsel, S. Minton, *The Digital Universe of Opportunities: Rich data and the increasing value of the internet of things*, International Data Corporation, White Paper, IDC_1672. (2014)
- Valenzuela, S. y C. Vidal, *Evaluación de Hadoop y Giraph para el Procesamiento de Grafos*, Jornadas Chilenas de la Computación, XXVII Encuentro Chileno de Computación, Santiago, Chile, Noviembre (2015)
- Valiant, L. G., *A bridging model for parallel computation*, Communications of the ACM, 33 (8), pp. 103-111 (1990)
- Voulgaris, S. y C. Martella, *A Scalable graph pattern matching engine on top of Apache Giraph*, (en línea), 2014. <http://homepages.cwi.nl/~boncz/msc/2014-SinzianaFilip.pdf>, acceso: 2 de Junio 2015), Vrije Universiteit, Amsterdam (2014)
- White, T., *Hadoop: The Definitive Guide*, O'Reilly Media, 4 edition (2015)