MAKALAH ANALISIS BIG DATA

PENGENALAN HBASE

Dosen Pengampu: Sevi Nurafni ST., M.Si., M.Sc.

Disusun Oleh Ryan Fadhilah Faizal Hakim

Program Studi Sains Data

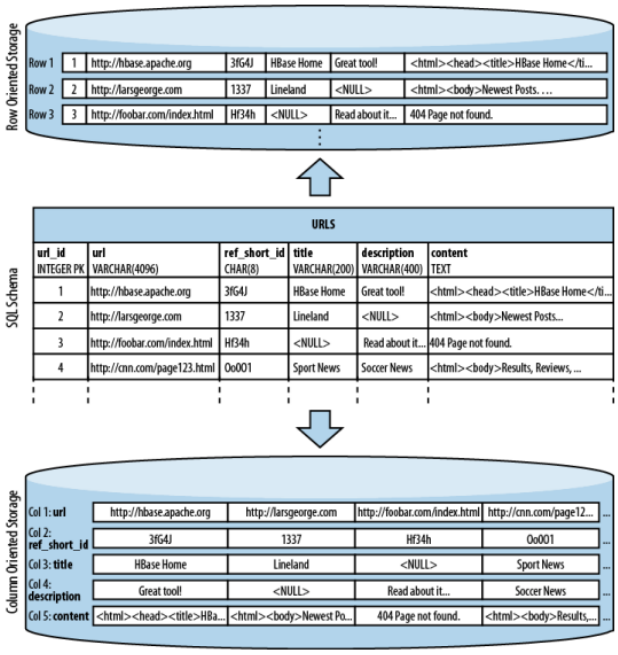
Fakultas Sains dan Teknologi

A. Apa yang dimaksud dengan basis data non-relasional dan basis data berorientasi kolom?

Basis data non-relasional adalah database yang tidak menggunakan skema tabular baris dan kolom yang ditemukan di sebagian besar sistem database tradisional[1]. Database non-relasional menggunakan model penyimpanan yang dioptimalkan untuk persyaratan spesifik dari jenis data yang disimpan[1]. Misalnya, data dapat sebagai pasangan key/value sederhana, seperti dokumen JSON, atau sebagai grafik yang terdiri dari tepi dan simpul[1]. Kemampuan untuk memproses dan mengatur berbagai jenis informasi secara berdampingan membuat database non-relasional jauh lebih fleksibel dibandingkan database relasional[2].

Istilah *NoSQL* mengacu pada penyimpanan data yang tidak menggunakan SQL untuk kueri[1]. Database non-relasional sering kali bekerja lebih cepat karena kueri tidak harus melihat beberapa tabel untuk memberikan jawaban[2].

Basis data berorientasi kolom (Columnar Database) adalah Database Management System (DBMS) yang membantu menyimpan data dalam kolom, bukan baris[3]. Ini bertanggung jawab untuk mempercepat waktu yang diperlukan untuk mengembalikan kueri tertentu[3]. Motif utama dari Columnar Database adalah untuk membaca dan menulis data secara efektif[3]. Basis data Kolom dan Baris adalah beberapa metode yang digunakan untuk memproses analisis big data dan data werehousing[3]. Namun memiliki pendekatan yang berbeda satu sama lain[3]. Misalnya dalam Basis data Baris, kolom untuk setiap catatan baru disimpan dalam baris panjang[3]. Sedangkan Basis data Kolom, setiap bidang memiliki kumpulan kolomnya sendiri[3]. Alasan untuk menyimpan nilai per kolom sebagai gantinya didasarkan pada asumsi bahwa, untuk kueri tertentu, tidak semua nilai diperlukan[4].



Gambar 1: Column-oriented and row-oriented storage layout[4]

B. Bagaimana data direpresentasikan oleh tabel HBase?

Data yang direpresentasikan oleh tabel HBase diantaranya[5]:

- Key-value stores: Penyimpanan ini menyediakan cara untuk menyimpan segala jenis data tanpa harus menggunakan skema. Ini berbeda dengan basis data relasional, di mana Anda perlu mendefinisikan skema (struktur tabel) sebelum data dimasukkan. Karena key-value stores tidak memerlukan skema, Anda memiliki fleksibilitas besar untuk menyimpan data dalam banyak format. Dalam key-value store, sebuah baris hanya terdiri dari kunci (pengenal) dan nilai, yang bisa berupa apa saja mulai dari nilai integer hingga string data biner besar[5].
- Column family stores: Di sini Anda memiliki basis data di mana kolom dikelompokkan menjadi keluarga kolom dan disimpan bersama di disk. Sebagian besar basis data ini sebenarnya bukan berorientasi kolom, karena mereka didasarkan pada makalah BigTable dari Google, yang menyimpan data sebagai peta yang diurutkan secara multidimensi[5].
- Document stores: Penawaran ini mengandalkan koleksi dokumen yang dikodekan dan diformat secara serupa untuk meningkatkan efisiensi[5]. Document stores memungkinkan dokumen individu dalam koleksi untuk hanya mencakup sebagian kecil dari bidang, sehingga hanya data yang dibutuhkan yang disimpan. Untuk set data yang jarang, di mana banyak bidang sering tidak diisi, ini dapat berarti penghematan ruang yang signifikan[5].
- Graph databases: Di sini Anda memiliki basis data yang menyimpan struktur grafik - representasi yang menunjukkan koleksi entitas (simpul atau node) dan hubungan mereka (tepi) satu sama lain. Struktur ini memungkinkan basis data grafik sangat cocok untuk menyimpan struktur kompleks, seperti hubungan tautan antara semua halaman web yang diketahui[5].

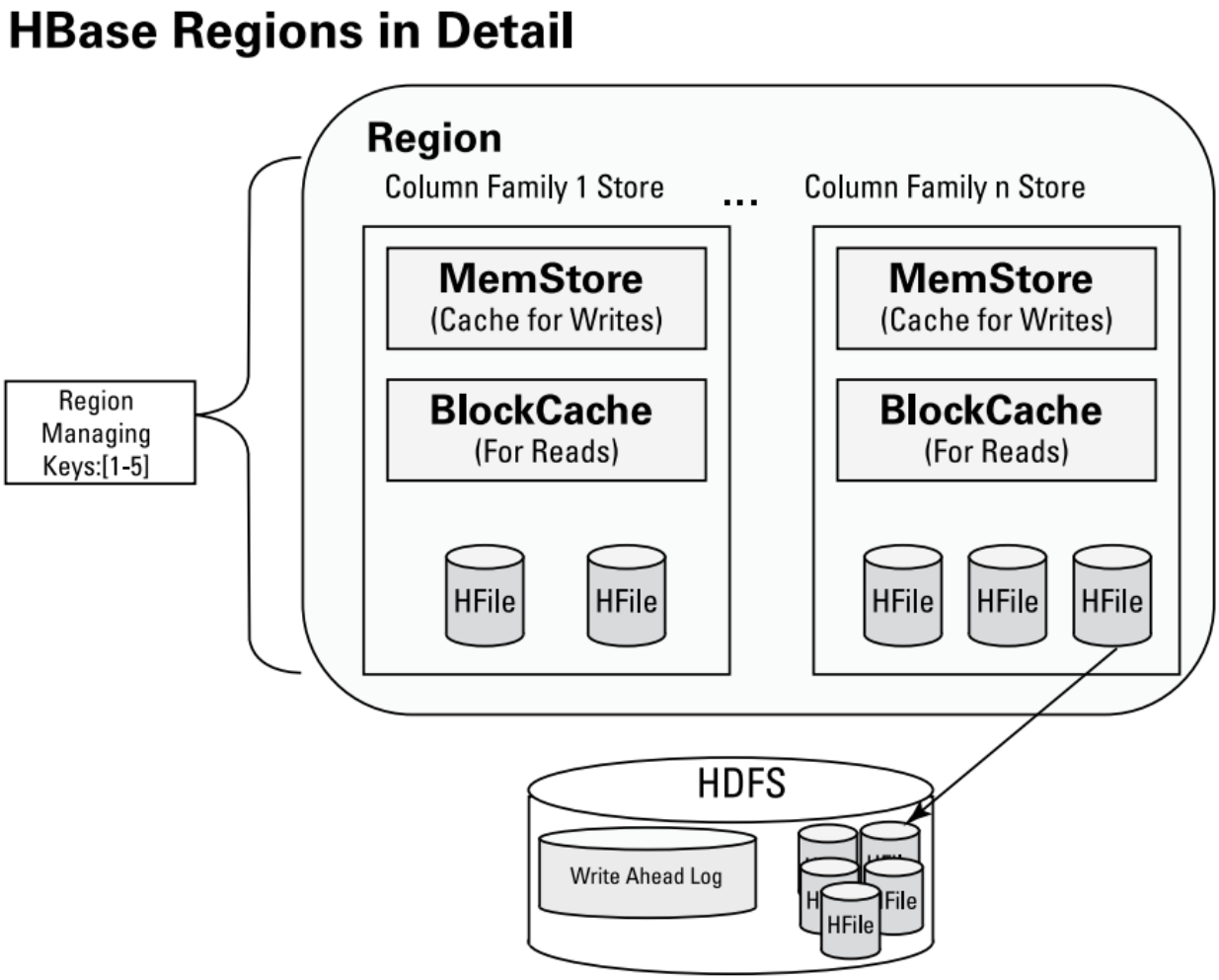
C. Apa yang dimaksud dengan region?

Di HBase, sebuah tabel tersebar di sejumlah RegionServer dan juga terdiri dari region-region individu[5]. Saat tabel dibagi, pembagian tersebut menjadi region. Region menyimpan rentang pasangan kunci-nilai, dan setiap RegionServer mengelola sejumlah region yang dapat dikonfigurasi[5].

Tabel secara otomatis dipartisi secara horizontal oleh HBase menjadi beberapa region. Setiap region terdiri dari subset baris tabel. Sebuah region ditandai oleh tabel tempat ia berada, baris pertamanya, inklusif, dan baris terakhir, eksklusif[6]. Awalnya, sebuah tabel terdiri dari satu region, tetapi seiring bertambahnya ukuran region, setelah melewati ambang batas ukuran yang dapat dikonfigurasi, ia akan terbagi di batas baris menjadi dua region baru dengan ukuran yang kurang lebih sama[6]. Sampai pemecahan pertama ini terjadi, semua pemuatan akan dilakukan terhadap server tunggal yang menjadi tuan rumah region asli[6].

Seiring bertambahnya tabel, jumlah regionnya juga bertambah. Region adalah unit yang didistribusikan ke seluruh kluster HBase[6]. Dengan cara ini, sebuah tabel yang terlalu besar untuk satu server dapat ditangani oleh kluster server, dengan setiap node menjadi tuan rumah subset dari

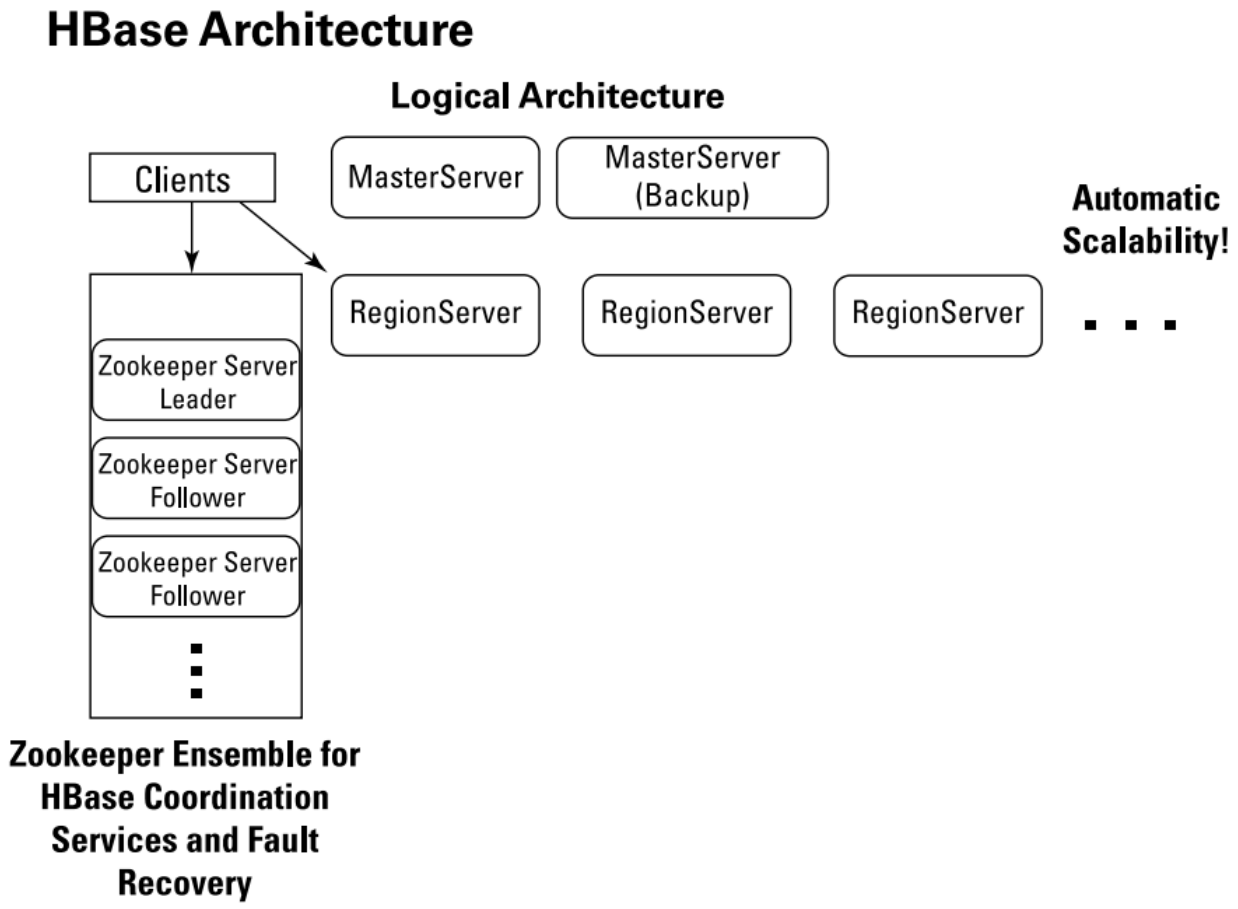
total region tabel[6]. Ini juga merupakan cara di mana beban pada tabel didistribusikan[6]. Set region yang diurutkan secara online mencakup total konten tabel[6].



Gambar 2: HBase regions in detail[5]

Pada gambar 2 menjelaskan bahwa region memisahkan data ke dalam Column Family dan menyimpan data di HDFS menggunakan objek Hfile[5].

D. Apa saja pekerjaan spesifik untuk Hmaster, ZooKeeper dan RegionServer?



Gambar 3: The HBase architecture[5]

Apabila dilihat pada gambar 3 bahwa klien tidak menunjuk ke MasterServer (Hmaster), tetapi malah menunjuk ke kluster Zookeeper dan RegionServer[5]. Tidak terdapat jalur untuk menyimpan data dan akses pada Masterserver – karena itu adalah pekerjaan dari kluster Zookeeper dan RegionServers[5]. Tetapi MasterServer ada berfungsi untuk[5]:

- ✓ Memantau RegionServers di kluster HBase: MasterServer menjaga daftar RegionServers aktif di kluster HBase.
- ✓ Menangani operasi metadata: Ketika tabel dibuat atau atributnya diubah (pengaturan kompresi, pengaturan cache, versioning, dan lainnya) MasterServer menangani operasi dan menyimpan metadata yang diperlukan.
- ✓ Menetapkan region: MasterServer menetapkan region ke RegionServers.
- ✓ Mengelola failover RegionServer: Seperti halnya dengan kluster terdistribusi, Anda berharap kegagalan node tidak terjadi dan Anda merencanakannya. Ketika server region gagal, Zookeeper memberi tahu MasterServer sehingga operasi failover dan pemulihan dapat dimulai. Kami membahas topik ini lebih detail di bagian selanjutnya "Zookeeper dan keandalan HBase."

- ✓ Mengawasi penyeimbangan beban region di semua RegionServers yang tersedia: Anda mungkin ingat bahwa tabel terdiri dari region yang didistribusikan secara merata di semua RegionServers yang tersedia. Ini adalah pekerjaan dari thread balancer (atau tugas, jika Anda mau) yang MasterServer aktifkan secara periodik.
- ✓ Mengelola (dan membersihkan) tabel katalog: Dua tabel katalog kunci - berlabel ROOT- dan .META - digunakan oleh sistem HBase untuk membantu klien menemukan pasangan kunci-nilai tertentu di sistem.
    - Tabel -ROOT- melacak lokasi tabel .META di kluster.
    - Tabel .META melacak di mana setiap region berada di kluster.
- ✓ MasterServer menyediakan manajemen tabel kritis ini atas nama sistem HBase secara keseluruhan.
- ✓ Membersihkan WAL: MasterServer berinteraksi dengan WAL selama failover RegionServer dan secara berkala membersihkan log.
- ✓ Menyediakan kerangka kerja coprocessor untuk mengamati operasi master: Ini adalah istilah baru lainnya untuk glosarium HBase Anda yang berkembang. Coprocessors berjalan dalam konteks MasterServer atau RegionServers. Misalnya, coprocessor pengamat MasterServer memungkinkan Anda untuk mengubah atau memperluas fungsionalitas normal server ketika operasi seperti pembuatan tabel atau penghapusan tabel terjadi. Seringkali coprocessors digunakan untuk mengelola indeks tabel untuk aplikasi HBase tingkat lanjut.

Dalam HBase Architechture terdapat ZooKeeper. Zookeeper adalah kluster terdistribusi dari server yang secara kolektif menyediakan layanan koordinasi dan sinkronisasi yang dapat diandalkan untuk aplikasi terkluster[5]. Seperti di HBase, kluster Zookeeper biasanya berjalan pada server x86 komoditas berbiaya rendah. Setiap server x86 individu menjalankan satu proses perangkat lunak Zookeeper (selanjutnya disebut sebagai server Zookeeper), dengan satu server Zookeeper dipilih oleh ansambel sebagai pemimpin dan sisanya adalah pengikut[5]. Tugas spesifik dari ZooKeeper diantaranya:

- ✓ Koordinasi dan Sinkronisasi: ZooKeeper menyediakan layanan koordinasi dan sinkronisasi yang dapat diandalkan melalui apa yang disebutnya znodes. Znodes disajikan sebagai pohon direktori dan menyerupai nama jalur file yang akan Anda lihat di sistem file Unix.
- ✓ Penyimpanan Data: Meskipun znodes memang menyimpan data, jumlahnya tidak banyak - saat ini kurang dari 1 MB secara default. ZooKeeper menyimpan znodes di memori dan znodes berbasis memori ini memberikan akses klien yang cepat untuk koordinasi, status, dan fungsi penting lainnya yang diperlukan oleh aplikasi terdistribusi seperti HBase.
- ✓ Penanganan Pembacaan dan Penulisan: Setiap server ZooKeeper dapat menangani pembacaan dari klien, termasuk pemimpin, tetapi hanya pemimpin yang mengeluarkan penulisan znode atomik - penulisan yang sepenuhnya berhasil atau sepenuhnya gagal.
- ✓ Replikasi Data: ZooKeeper mereplikasi znodes di seluruh ansambel sehingga jika server gagal, data znode masih tersedia selama kuorum mayoritas server masih berjalan dan beroperasi.

- ✓ Sinkronisasi: ZooKeeper menyediakan perintah sinkronisasi. Klien yang tidak dapat mentolerir kurangnya sinkronisasi sementara ini dalam kluster ZooKeeper mungkin memutuskan untuk mengeluarkan perintah sinkronisasi sebelum membaca znodes.
- ✓ Keamanan: Znodes memiliki daftar kontrol akses (ACL) yang terkait dengannya untuk keamanan, dan mendukung versi, timestamp dan pemberitahuan ke klien saat berubah.

Selain itu, dalam HBase Architechture terdapat RegionServers yang terhubung dengan klien. RegionServers adalah proses perangkat lunak (sering disebut daemon) yang Anda aktifkan untuk menyimpan dan mengambil data di HBase[5]. Dalam lingkungan produksi, setiap RegionServer ditempatkan pada node komputasi yang didedikasikan sendiri[5]. RegionServer memiliki tugas spesifik diantaranya[5]:

- ✓ Penyimpanan dan Pengambilan Data: RegionServer bertugas menyimpan dan mengambil data di HBase. Setiap RegionServer biasanya ditempatkan pada node komputasi yang didedikasikan sendiri dalam lingkungan produksi.
- ✓ Pembagian Tabel Otomatis (Auto-sharding): Ketika tabel tumbuh melampaui batas yang dapat dikonfigurasi, sistem HBase secara otomatis membagi tabel dan mendistribusikan beban ke RegionServer lain. Proses ini sering disebut sebagai auto-sharding.
- ✓ Penskalaan Otomatis: HBase secara otomatis menskalakan seiring Anda menambahkan data ke sistem. Ini adalah manfaat besar dibandingkan dengan sebagian besar sistem manajemen database, yang memerlukan intervensi manual untuk menskalakan sistem secara keseluruhan melampaui satu server.
- ✓ Manajemen Beban: Anda mungkin memiliki sejumlah tabel besar atau kecil dan Anda akan ingin HBase memanfaatkan semua RegionServers yang tersedia saat mengelola data Anda. Dengan banyak klien yang mengakses sistem HBase Anda, Anda akan ingin menggunakan banyak RegionServers untuk memenuhi permintaan. HBase menangani semua kekhawatiran ini untuk Anda dan menskalakan secara otomatis dalam hal kapasitas penyimpanan dan daya komputasi.

E. Bagaimana Hbase melakukan pembacaan dan penulisan?

HBase melakukan pembacaan dan penulisan dengan cara berikut[5]:

Pembacaan[5]:

1. Data dibaca dalam blok dari HDFS dan disimpan di BlockCache, sebuah cache baca.
2. Bacaan berikutnya untuk data - atau data yang disimpan di dekatnya - akan dibaca dari RAM bukan dari disk, yang meningkatkan kinerja keseluruhan.

Penulisan[5]:

1. Ketika Anda menulis atau memodifikasi data di HBase, data pertama kali dipersistenkan ke Write Ahead Log (WAL), yang disimpan di HDFS.
2. Kemudian data ditulis ke cache MemStore.

3. Pada interval yang dapat dikonfigurasi, pasangan kunci-nilai yang disimpan di MemStore ditulis ke HFiles di HDFS dan setelah itu entri WAL dihapus.
4. Jika terjadi kegagalan setelah penulisan WAL awal tetapi sebelum penulisan MemStore akhir ke disk, WAL dapat diputar ulang untuk menghindari kehilangan data.

Selain itu, HBase dirancang untuk mengosongkan data keluarga kolom yang disimpan di MemStore ke satu HFile per flush[5]. Kemudian pada interval yang dapat dikonfigurasi HFiles digabungkan menjadi HFiles yang lebih besar. Ini adalah bagian dari operasi kompaksi kritis di HBase[5]. Harap dicatat bahwa semua ini adalah bagian dari cara HBase menangani kegagalan node dan memastikan keandalan dan kinerja sistem[5].

F. Kapan Anda membutuhkan HBase selain RDBMS?

HBase digunakan ketika:

- Memiliki dataset berukuran besar (jutaan atau miliaran baris dan kolom) dan memerlukan akses baca atau tulis yang cepat, acak, dan real-time terhadap data tersebut.
- Dataset didistribusikan di berbagai kluster dan kita memerlukan skalabilitas tinggi untuk menangani data.
- Data dikumpulkan dari berbagai sumber data dan berbentuk semi terstruktur atau tidak terstruktur atau kombinasi dari semuanya.
- Menyimpan data berorientasi kolom.
- Memiliki banyak versi dataset yang harus disimpan semuanya.

Referensi

1. Microsoft.com. (2024). Data non-relasional dan NoSQL. Diakses dari Situs web Microsoft: https://learn.microsoft.com/id-id/azure/architecture/data-guide/big-data/non-relational-data.
2. MongDB.com. (2024). What Is a Non-Relational Database?. Diakses dari Situs web MongoDB: https://www.mongodb.com/databases/non-relational#:~:text=Non-relational%20databases%20are%20sometimes%20referred%20to%20as%20%E2%80%9CNoSQL%2C%E2%80%9D,flexible%20than%20the%20traditional%2C%20SQL-based%2C%20relational%20database%20structures.
3. Geeksforgeeks.org. (2024). What is a Columnar Database?. Diakses dari Situs web Geekforgeeks: https://www.geeksforgeeks.org/what-is-a-columnar-database/?ref=ml_lbp.
4. George, L. (2015). Hbase: The Definitive Guide Second Edition. Sebastopol: O'Reilly Media, Inc.
5. deRoos, D., Zikopoulos, P., Brown, B., Coss, R., Melnyk, R. (2014). Hadoop for Dummies. New Jersey: John Wiley & Sons, Inc.
6. White, T. (2012). Hadoop: The Definitive Guide, Third Edition. Sebastopol: O'Reilly Media, Inc.