

BIG DATA DAN HADOOP TRAINING

PENDAHULUAN	4
Google dan Big Data	4
BIG DATA	6
Karakteristik Big Data: Volume, Variety, Velocity	7
TEKNOLOGI BIG DATA GOOGLE	11
Google dan Big Data	11
Google File System	11
Google Mapreduce.....	13
APACHE HADOOP	21
Apache HBase.....	27
Ekosistem Hadoop	30
TUTORIAL HADOOP DAN HBASE	31
Instal Hadoop	31
Prasyarat	31
Installasi Java JDK	32
Hadoop Mode Standalone	33
Hadoop Mode Pseudo-Distributed	36
Install HBase	43
Tutorial HBase	46
HBase Shell.....	46
HBase General Commands	47
Membuat Table	49
Memasukan Data (Create)	50
Menampilkan Data (Read).....	52
Update Data (Update)	52
Menghapus Data (Delete)	53
Import Data TSV (Tab Separated Values)	54
Hadoop MapReduce.....	56
Membuat Program Pertama Hadoop MapReduce.....	57
Apache PIG	65
Apache Pig Architecture	66
Pig Latin Scripts	67

Apache Pig Latin Data Model	67
Installasi Apache Pig.....	70
Membuat Apache Pig Program.....	72
Grunt Apache Pig Shell	73
Apache Pig dengan HBase	75
Apache Hive.....	77
Arsitektur Hive.....	77
Hive Services	79
Installasi Apache Hive.....	81
Perintah Dasar Hive SQL.....	84
Hadoop Cluster.....	87
Prasyarat	87
Architecture Hadoop Cluster.....	87
Konfigurasi Hadoop Cluster.....	88

PENDAHULUAN

Google dan Big Data

Berawal dari keberhasilan perusahaan-perusahaan web service raksasa seperti halnya Google dan Facebook dalam mengelola dan memanfaatkan data tak berstruktur (Unstructured Data) yang berupa Consumer Generated Media (CGM) maupun Click Stream dalam volume yang sangat besar, sebuah konsep yang dikenal dengan istilah Big Data kemudian menjadi pusat perhatian dalam dunia teknologi informasi[1].

Di lain pihak, fakta juga menunjukkan bahwa semakin banyak organisasi-organisasi di dunia, baik itu yang berupa perusahaan-perusahaan swasta maupun instansi-instansi pemerintah, yang mengalami kesulitan dalam mengelola data yang volumenya makin bertumbuh dan jenisnya yang makin kompleks[2]. Mereka harus memanage sekaligus menganalisa data-data tersebut, dan mereka harus menemukan arti atau nilai dari tumpukan data yang terus berkembang dan makin kompleks, yang dikatakan telah melewati batas kemampuan aplikasi pengolah data konvensional untuk memprosesnya. Kondisi data semacam ini juga dikategorikan sebagai Big Data, yang diartikan sebagai sekumpulan data dalam jumlah yang sangat besar yang tantangannya terletak pada bagaimana data-data tersebut mesti disimpan, bagaimana melakukan pencarian dalam tumpukan data-data tersebut, bagaimana membagi-bagikannya, bagaimana memvisualisasikannya, dan bagaimana data-data itu harus dianalisa.

Secara sederhana, Big Data telah didefinisikan sebagai sekumpulan data dengan volume yang sangat besar yang terlalu kompleks itu dapat diproses dengan teknologi pengolahan data konvensional. Saat ini, Big Data juga sering dideskripsikan sebagai data yang memiliki 3 (tiga) karakteristik, yaitu: volume, variety, dan velocity[3] [4].

Demi menjawab tantangan kompleksitas pemberdayaan Big Data, Apache Software Foundation (ASF) kemudian menciptakan Apache Hadoop, sebuah framework Distributed System yang dikembangkan melalui proyek Open Source. Apache Hadoop utamanya terdiri atas Hadoop Distributed File System (HDFS) dan framework Distributed Parallel Processing System yang disebut Hadoop MapReduce[5] [6]. Dalam hal ini, HDFS berfungsi sebagai sistem penyimpanan data secara terdistribusi, sedangkan Hadoop MapReduce berfungsi sebagai framework pemrosesan data secara paralel dan terdistribusi.

Saat ini Apache Hadoop telah digunakan oleh berbagai perusahaan besar seperti halnya Yahoo!, Facebook, E-Bay, IBM, Intel, NEC, NTT, Recruit, Amazon, dan Rakuten. Oleh karena itu, Hadoop, sistem yang dapat dioperasikan pada komputer server biasa, telah dikatakan sebagai penggerak utama dalam perkembangan Big Data. Dengan memberdayakan Hadoop untuk memproses dan menganalisa Big Data, dikatakan bahwa inovasi yang dihasilkan dari pengetahuan baru ataupun pemahaman yang lebih mendalam maupun solusi – solusi baru dalam proses pengambilan keputusan telah dapat direalisasikan.

Tidak hanya di dunia bisnis, di lingkungan akademisi maupun peneliti pun Hadoop telah menjadi pusat perhatian. Shvachko dkk menjelaskan secara rinci tentang pengalamannya menggunakan Hadoop dalam mengelola sekitar 25 petabytes data perusahaan di Yahoo![7]. Data-data tersebut disimpan dan diproses dalam sejumlah cluster yang secara total terdiri atas 25.000 komputer server. Diantara cluster-cluster komputer tersebut, cluster yang paling besar

terdiri atas 3.500 komputer server. Mereka menemukan bahwa dalam hal scalability Hadoop memang memiliki keunggulan yang tak dapat dipungkiri. Selain itu mereka juga menyarankan bahwa mengelola Hadoop dalam beberapa cluster yang lebih kecil akan lebih menguntungkan daripada mengelola Hadoop dalam satu cluster besar.

Berkat keunggulan Hadoop dalam hal scalability, fault tolerance, dan programming model yang sederhana, banyak peneliti di perguruan-perguruan tinggi maupun di perusahaan-perusahaan menggunakan Hadoop untuk melakukan penelitian yang berkaitan dengan pemrosesan data multidimensi dalam jumlah besar. Ariel Cary dari Florida International University, mengajukan solusi yang scalable dalam menyusun indeks R-tree dengan menggunakan MapReduce parallel programming model[8]. Dalam paper-nya, dijelaskan bahwa dengan menggunakan MapReduce, penyusunan indeks R-tree dapat dilakukan dalam waktu yang lebih singkat dan hanya dengan menambahkan jumlah komputer yang dipergunakan dapat meningkatkan scalability hampir secara linier. Nathan Thomas Kerr dari Arizona State University, telah mencoba menguji kemungkinan untuk memproses data Geographic Information System (GIS) dengan menggunakan sistem terdistribusi[9]. Dalam hal ini, ia menggunakan Hadoop dan Message Passing Interface (MPI). Berdasarkan hasil penelitiannya, ia menyimpulkan bahwa jika data GIS yang menjadi objek diproses secara keseluruhan dalam satu kali pemrosesan, maka data GIS dapat diproses dengan sistem terdistribusi dan hanya dengan menerapkan algoritma yang sederhana saja, waktu pemrosesan dapat dipersingkat dengan sangat signifikan. Sementara itu, Zhang dkk juga mengajukan algoritma baru untuk memproses data multidimensi yang berhubungan dengan ruang yang mereka sebut dengan Spatial Join with MapReduce (SJMR)[10]. Mereka telah berhasil membuktikan bahwa SJMR memiliki performa yang lebih unggul daripada algoritma yang telah ada yang dikenal dengan Parallel Partition Based Spatial-Merge join (PPMSM).

MapReduce sebagai Parallel Distributed Processing Framework yang dimiliki Hadoop, memang memiliki keunggulan dalam hal memproses data dalam jumlah besar secara batch processing dengan model pemrograman yang relatif sederhana. Namun demikian, MapReduce dinyatakan tidak cocok untuk memproses data yang sifatnya interaktif atau real time. Untuk mengatasi kekurangan ini, Apache kemudian mengembangkan Apache HBase, perangkat lunak yang memungkinkan pemrosesan Big Data dilakukan secara real time pada HDFS (Hadoop Distributed File System)[11][12]. Seperti halnya Hadoop, HBase juga dikembangkan secara Open Source. Facebook, website SNS (Social Network Service) raksasa yang pada Maret 2013 telah memiliki sekitar 1,1 miliar pengguna aktif per bulannya, menggunakan Apache HBase untuk menyimpan dan memproses data Facebook Message dari para penggunanya.

Seperti halnya Hadoop, HBase juga telah menjadi pusat perhatian di kalangan para akademisi dan peneliti bidang teknologi informasi. Shouji Nishimura, seorang peneliti dari NEC, telah mengembangkan sebuah ekstensi dari HBase yang mampu menerapkan Multi-Dimensional Range Searching secara efektif tanpa mengorbankan fitur scalability yang merupakan salah satu keunggulan HBase. Ekstensi ini disebut dengan MD-HBase[13]. Teknik yang diterapkan dalam MD-HBase adalah dengan memetakan data dua dimensi menjadi data satu dimensi menggunakan metode Space Filling Curve yang dikenal dengan Z-Curve. Selanjutnya diterapkan teknik Space Partition yang dikenal dengan Kd-Tree untuk menyusun indeks dari data dua dimensi yang diproses.

Berdasarkan penjelasan diatas, dapat ditarik suatu kesimpulan bahwa fenomena Big Data telah menjadi satu bidang baru yang tak dapat dipandang sebelah mata meskipun di dalam negeri gemanya belum begitu terdengar. Teknologi dasar untuk memproses dan memberdayakan Big Data seperti halnya Apache Hadoop dan Apache HBase pun telah diadopsi oleh banyak perusahaan bertaraf dunia, juga telah menjadi objek penelitian di kalangan akademisi, dan masih terus dikembangkan oleh Apache Software Foundation.

Demi memperkenalkan konsep Big Data beserta teknologi pendukungnya, dalam panduan ini akan dibahas teori tentang Big Data, kemudian tentang konsep teknologi Big Data yang dikembangkan di Google serta tentang cara kerja Hadoop dan HBase yang dikembangkan oleh Apache Software Foundation. Tidak hanya berhenti pada pemahaman teorinya saja, pada panduan ini juga disertakan tutorial tentang cara setup dan penggunaan Hadoop maupun HBase, baik pada PC desktop Windows maupun Linux yang disertai dengan contoh program aplikasinya.

BIG DATA

Definisi

Kurang lebih sejak tahun 2010, istilah Big Data telah menjadi topik yang dominan dan sangat sering dibahas dalam dunia IT. Banyak pihak yang mungkin heran kenapa topik ini baru menjadi pusat perhatian padahal ledakan informasi telah terjadi secara berkelangsungan sejak dimulainya era informasi. Perkembangan volume dan jenis data yang terus meningkat secara berlipat-lipat dalam dunia maya Internet semenjak kelahirannya adalah fakta yang tak dapat dipungkiri. Mulai data yang hanya berupa teks, gambar atau foto, lalu data yang berupa video hingga data yang berasal dari sistem pengindraan. Lalu kenapa baru sekarang orang ramai-ramai membahas istilah Big Data? Apa sebenarnya Big Data itu?

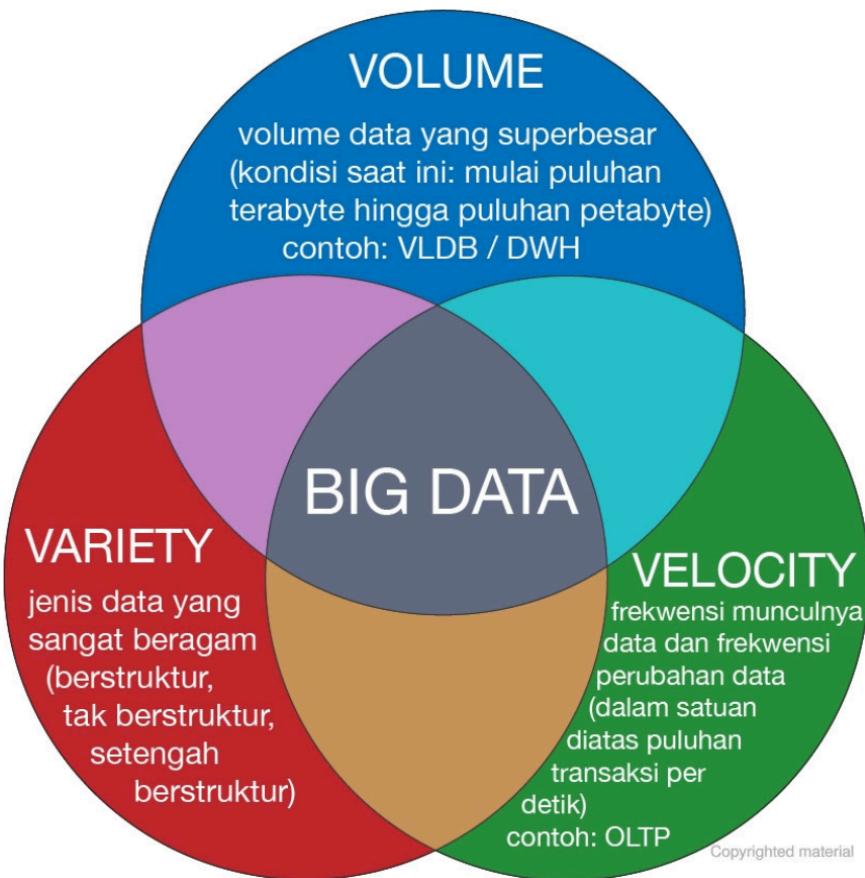
Jika diterjemahkan secara mentah-mentah maka Big Data berarti suatu data dengan kapasitas yang besar. Sejauh ini, perusahaan-perusahaan terkemuka telah memberdayakan infomasi dan data dengan beragam teknologi manajemen data guna menunjang kemajuan bisnisnya. Sebagian besar telah menggunakan tools seperti Data Warehouse (DWH) maupun Business Intelligence (BI) sebagai alat pengolah data yang volumenya terus membesar dan ragamnya yang makin kompleks. Sebagai contoh, saat ini kapasitas DWH (Data Warehouse) yang digunakan oleh perusahaan-perusahaan di Jepang berkisar dalam skala terabyte. Namun, jika misalnya dalam suatu sistem terdapat 1000 terabyte (1 petabyte) data, apakah sistem tersebut bisa disebut Big Data?

Satu lagi, Big Data sering dikaitkan dengan SNS (Social Network Service), contohnya Facebook. Memang benar Facebook telah memiliki lebih dari satu miliar pengguna aktif dan dikatakan bahwa dalam satu hari Facebook memproses sekitar 10 terabyte data. Pada umumnya, SNS seperti Facebook tidak menggunakan RDBMS (Relational DataBase Management System) sebagai software pengolah data, melainkan lebih banyak menggunakan NoSQL. Lalu, apa kita bisa menyebut sistem NoSQL sebagai Big Data?

Dengan mengkombinasikan kedua uraian diatas, dapat ditarik sebuah definisi bahwa Big Data adalah "suatu sistem yang menggunakan NoSQL dalam memproses atau mengolah data yang berukuran sangat besar, misalnya dalam skala petabyte". Apakah definisi ini tepat? Boleh dikatakan masih setengah benar. Definisi tersebut masih belum menggambarkan Big Data secara menyeluruh. Big Data tidak sesederhana itu, Big Data memuat arti yang lebih kompleks sehingga perlu definisi yang sedikit lebih kompleks pula demi mendeskripsikannya secara menyeluruh. Mengapa butuh definisi yang lebih kompleks? Fakta menunjukkan bahwa bukan hanya NoSQL saja yang mampu mengolah data dalam skala raksasa (petabyte). Beberapa perusahaan telah menggunakan RDBMS untuk memberdayakan data dalam kapasitas yang sangat besar. Sebagai contoh, Bank of America memiliki DWH dengan kapasitas lebih dari 1,5 petabyte, Walmart Stores yang bergerak dalam bisnis retail (supermarket) berskala dunia telah mengelola data berkapasitas lebih dari 2,5 petabyte, dan bahkan situs auction (lelang) eBay memiliki DWH yang menyimpan lebih dari 6 petabyte data. Oleh karena itu, hanya karena telah berskala petabyte saja, suatu data belum bisa disebut Big Data. Sekedar referensi,

DWH dengan kapasitas sangat besar seperti beberapa contoh diatas disebut EDW (Enterprise Data Warehouse) dan database yang digunakannya disebut VLDB (Very Large Database).

Memang benar, NoSQL dikenal memiliki potensi dan kapabilitas Scale Up (peningkatan kemampuan mengolah data dengan menambah jumlah server atau storage) yang lebih unggul daripada RDBMS. Tetapi, bukan berarti RDBMS tak diperlukan. NoSQL memang lebih tepat untuk mengolah data yang sifatnya tak berstruktur seperti data teks dan gambar, namun NoSQL kurang tepat bila digunakan untuk mengolah data yang sifatnya berstruktur seperti data-data numerik, juga kurang sesuai untuk memproses data secara lebih detail demi menghasilkan akurasi yang tinggi. Pada kenyataannya, Facebook juga tak hanya menggunakan NoSQL untuk memproses data-datanya, Facebook juga tetap menggunakan RDBMS. Lain kata, penggunaan RDBMS dan NoSQL mesti disesuaikan dengan jenis data yang hendak diproses dan proses macam apa yang dibutuhkan guna mendapat hasil yang optimal.



Karakteristik Big Data: Volume, Variety, Velocity

Kembali ke pertanyaan awal, apakah sebenarnya Big Data itu? Sayang sekali, hingga saat ini masih belum ada definisi baku yang telah disepakati secara umum. Namun demikian, latar belakang dari munculnya istilah ini adalah fakta yang menunjukkan bahwa pertumbuhan data yang terus berlipat ganda dari waktu ke waktu telah melampaui batas kemampuan media penyimpanan maupun sistem database konvensional (RDBMS).

Kemudian, McKinsey Global Institute (MGI), dalam laporannya yang dirilis pada Mei 2011[14], mendefinisikan bahwa Big Data adalah data yang sudah sangat sulit untuk dikoleksi, disimpan, dikelola maupun dianalisa dengan menggunakan sistem database biasa karena volumenya yang terus berlipat. Tentu saja definisi ini masih sangat relatif, tidak

mendeskripsikan secara eksplisit sebesar apa Big Data itu. Ada juga yang mendeskripsikan Big Data sebagai fenomena yang lahir dari meluasnya penggunaan internet dan kemajuan teknologi informasi yang diikuti dengan terjadinya pertumbuhan data yang luar biasa cepat, yang dikenal dengan istilah ledakan informasi (Information Explosion) maupun banjir data (Data Deluge). Hal ini mengakibatkan terbentuknya aliran data yang super besar dan terus-menerus sehingga sangat sulit untuk dikelola, diproses, maupun dianalisa dengan menggunakan teknologi pengolahan data yang selama ini digunakan (RDBMS). Definisi ini dipertegas lagi dengan menyebutkan bahwa Big Data memiliki tiga karakteristik / atribut yang dikenal dengan istilah 3V: Volume, Variety, Velocity.

Ketiga atribut (3V) dari Big Data ini pertama kali dikemukakan oleh Gartner[15] yang kemudian juga digunakan oleh IBM dalam mendefinisikan Big Data. Berikut adalah makna dari ketiga atribut (3V) tersebut:

Volume berkaitan dengan ukuran yang super besar, dalam hal ini kurang lebih sama dengan definisi dari MGI.

Variety menggambarkan tipe atau jenis data yang sangat beragam, yang meliputi berbagai jenis data baik data yang telah berstruktur dalam suatu database maupun data yang tidak terorganisir dalam suatu database seperti halnya data teks pada web pages, data suara, video, click stream, log file dan lain sebagainya.

Velocity dapat diartikan sebagai kecepatan dihasilkannya suatu data dan seberapa cepat data itu harus diproses agar dapat memberikan hasil yang valid. Hal ini juga dapat dimaknai sebagai laju pertumbuhan maupun perubahan suatu data atau frekuensi kemunculan maupun frekuensi perubahannya.

Gambar 1.1 menggambarkan 3 karakteristik Big Data (3V) tersebut. Namun demikian, definisi ini tentu masih sulit untuk dipahami. Oleh karena itu, uraian berikut mencoba memberikan gambaran yang lebih jelas dan nyata berkaitan dengan maksud definisi Big Data tersebut.

Tidak Hanya Soal Ukuran, Tapi Lebih Pada Ragam

Kini jelas bahwa Big Data bukan hanya masalah ukuran yang besar, lebih dari itu yang menjadi ciri khasnya adalah jenis datanya yang sangat beragam dan laju pertumbuhan maupun frekwensi perubahannya yang tinggi. Dalam hal ragam data, Big Data tidak hanya terdiri dari data berstruktur seperti halnya data angka-angka maupun deretan huruf-huruf yang berasal dari sistem database pada umumnya seperti halnya sistem database keuangan, tetapi juga terdiri atas data multimedia seperti data teks, data suara dan video yang dikenal dengan istilah data tak berstruktur. Terlebih lagi, Big Data juga mencakup data setengah berstruktur seperti halnya data e-mail maupun XML. Dalam hal kecepatan pertumbuhan maupun frekwensi perubahannya, Big Data mencakup data-data yang berasal dari berbagai jenis sensor, mesin-mesin, maupun data log komunikasi yang terus menerus mengalir. Bahkan, juga mencakup data-data yang tak hanya data yang berada di internal perusahaan, tetapi juga data-data di luar perusahaan seperti data-data dari Internet yang terus bertumbuh. Misalnya, data-data dari suatu sistem OLTP (Online Transaction Processing) yang sudah menangani ratusan transaksi per detik. Begitu beragamnya jenis data yang dicakup dalam Big Data inilah yang kiranya dapat dijadikan patokan untuk membedakan Big Data dengan sistem manajemen data pada umumnya.

Fokus Pada Trend Per-Individu, Kecepatan Lebih Utama Daripada Ketepatan

Hingga saat ini, pemanfaatan Big Data didominasi oleh perusahaan-perusahaan jasa berbasis Internet seperti halnya Google dan Facebook. Data yang mereka berdayakan pun bukanlah data-data internal perusahaan seperti halnya data-data penjualan maupun data pelanggan,

melainkan lebih menitik beratkan pada pengolahan data-data teks dan gambar yang berada di Internet. Bila kita melihat gaya pemberdayaan data yang dilakukan oleh perusahaan-perusahaan pada umumnya, yang dicari adalah trend yang didapat dari pengolahan data secara keseluruhan. Misalnya, dari data konsumen akan didapat informasi tentang trend konsumen dengan memproses data konsumen secara keseluruhan, bukan memproses data per-konsumen untuk mendapatkan trend per-konsumen. Dilain pihak, perusahaan-perusahaan jasa berbasis Internet yang memanfaatkan Big Data justru memfokuskan pemberdayaan data untuk mendapatkan informasi trend per-konsumen dengan memanfaatkan atribut-atribut yang melekat pada pribadi tiap konsumen.

Sebut saja toko online Amazon yang memanfaatkan informasi maupun atribut yang melekat pada diri per-konsumen, untuk memberikan rekomendasi yang sesuai kepada tiap pelanggan. Satu lagi, pemberdayaan data ala Big Data ini dapat dikatakan lebih berfokus pada kecepatan ketimbang ketepatan[16].

Berdasarkan uraian diatas, dapat ditarik suatu kesimpulan bahwa Big Data itu adalah limpahan data dengan volume dan ragam yang telah melampaui kapasitas sistem manajemen data konvensional, yang terbentuk dari meluasnya penggunaan internet maupun pemanfaatan teknologi informasi yang semakin canggih, dan memiliki tiga atribut : volume, variety, velocity.

Big Data Bisa Apa, Untuk Siapa?

Big Data adalah konsep yang lahir dari dunia teknologi informasi. Ketika kita berdiskusi tentang teknologi informasi, topik pembicaraan pasti tak akan terlepas dari soal software atau hardware atau bahkan mungkin kedua-duanya. Sering berkaitan dengan kecanggihan, performa, maupun hal-hal baru yang ditawarkan oleh suatu software/hardware yang menjadi topik pembicaraan. Demikian juga dengan Big Data, pasti juga berbicara soal software dan hardware.

Namun demikian, aplikasi Big Data tidak hanya berkaitan dengan penggunaan hardware dan software yang berperforma tinggi, tetapi lebih mengacu pada pemberdayaan data sebagai penunjang bisnis. Tidak dapat dipungkiri bahwa dunia bisnis di era informasi ini sangat bergantung pada pelayanan data dan informasi. Oleh karena itu, dunia bisnis adalah salah satu wilayah aplikasi Big Data.

Pertama-tama, ketika kita ditanya tentang perusahaan macam apa saja yang menggunakan Big Data?, maka sebagian besar akan menjawab perusahaan web service. Sebagai contoh, Google menggunakan akumulasi data dalam skala yang sangat besar untuk melakukan bisnis iklan. Selain itu dalam dunia SNS, Facebook juga menggunakan data user-nya yang sangat besar sebagai dasar untuk meningkatkan keuntungan dari iklan, permainan game dan penjualan software. Bahkan EC (E-Commerce / Electronic Commerce) shop / toko komersial elektronik seperti Amazon dan Rakuten pun menggunakan data atribut yang melekat pada tiap user-nya, sejarah pembelian, click stream (perilaku pengunjung dalam situs web) dan data lainnya sebagai bahan untuk membuat rekomendasi yang sesuai dengan karakteristik setiap user-nya. Dengan memberikan informasi berupa rekomendasi tersebut diharapkan dapat memperkuat keinginan para pengunjungnya untuk berbelanja.

Kemudian, jika suatu perusahaan web service tidak menggunakan E-Commerce sebagai salah satu cara berbisnis, apakah mereka memiliki hubungan dengan Big Data? Jawabannya, aplikasi Big Data tidak terbatas hanya untuk perusahaan web maupun E-Commerce saja. Sebagai contoh, ada perusahaan telekomunikasi yang menganalisis log komunikasinya untuk mencari tahu provider manakah yang paling banyak menjadi arah tujuan ketika seseorang menelpon dan mengirim surat melalui Email atau SMS. Dengan demikian perusahaan/provider tersebut mampu mencegah bahaya akan pindahnya seorang pelanggan ke provider lain dengan cara memberikan penawaran khusus secara terpisah, maupun melakukan promo baru untuk mengundang temannya agar mau memakai provider yang ia gunakan.

Berdasarkan contoh-contoh diatas, mungkin akan timbul anggapan bahwa perusahaan web dan perusahaan telekomunikasi kan menggunakan data-data dan informasi secara digital, makanya mudah, bagaimanakah dengan perusahaan yang informasi dan datanya tidak berbentuk digital?

Mari kita simak beberapa contoh lainnya, misalnya: saat ini perusahaan-perusahaan asuransi di Jepang telah mengumpulkan informasi tentang perilaku mengemudi kliennya dengan memonitor GPS pada peralatan navigasi digital yang dipasang pada mobil milik setiap kliennya. Dengan mengumpulkan informasi ini, tidak hanya data tentang umur, jarak tempuh, jenis perizinan saja yang didapat, namun dengan mengetahui gaya mengemudi setiap kliennya, perusahaan asuransi tersebut mampu menganalisis resiko setiap kliennya dan memastikan margin harga yang sesuai. Hal ini sama dengan ketika perusahaan kartu kredit menganalisis tempat kliennya menggunakan kartu kredit dan penggunaan kartu kredit dengan smartphone sehingga dapat dideteksi apakah terjadi penyalahgunaan atau tidak.

Lagi-lagi akan timbul suatu pertanyaan: apakah perusahaan selain perusahaan financial dan perusahaan jasa seperti halnya perusahaan yang langsung menjual barang nyata pun dapat menjadi pengguna Big Data? Sebagai mana kita ketahui perusahaan manapun yang terdapat kegiatan penjualan dan memiliki pelanggan maka pasti membutuhkan feedback dan informasi tentang needs dari para pelanggannya. Disinilah analisis Big Data memegang peranan yang menentukan bagi kemajuan perusahaan.

Penggunaan Big Data juga dapat diaplikasikan pada perusahaan infrastruktur maupun industri primer. Sebagai contoh, dengan menggabungkan sensor yang dipasang di jalan raya dengan data yang dikumpulkan dari GPS yang telah terpasang pada mobil para pengguna jalan raya maka dapat diketahui kondisi kepadatan lalu lintas dari tiap ruas jalan yang diinginkan. Bahkan dengan menggabungkannya dengan sistem traffic light, dipastikan akan dapat mempercepat waktu tempuh suatu perjalanan sehingga mampu mengurangi pengeluaran gas CO₂. Dalam industri primer yang dikatakan tertinggal dengan IT sekalipun dapat menjadi lebih efektif dalam segi operasionalnya jika mampu menerapkan analisa Big Data. Sebagai contoh, dengan pemasangan sensor cuaca pada ladang pertanian, para pelaku industri primer dapat menggabungkan data cuaca dengan data penghasilan maupun kualitas produk yang dihasilkan sehingga dapat dicari cara yang lebih efektif agar mampu meningkatkan kualitas produknya untuk mendapatkan keuntungan yang lebih besar.

Hasil yang didapat dari pengaplikasian Big Data merupakan bentuk dari sebuah knowhow dan suatu strategi bisnis yang semestinya menjadi PR yang perlu dipikirkan oleh para pelaku bisnis saat ini.

Dengan demikian, dapat dipahami bahwa Big Data memiliki wilayah aplikasi yang sangat luas. Anggapan yang menyatakan bahwa penggunaan Big Data yang terbatas hanya pada bisnis web service dan SNS saja merupakan pemikiran yang tidak tepat. Penggunaan Big Data mampu membuat kita untuk melihat lagi apa saja data yang kita miliki, apakah ada data yang terlantar atau terabaikan, kemudian kita dapat mengoptimalkan sumber data yang kita miliki. Namun demikian, bukan berarti dengan menggunakan Big Data segalanya akan berjalan dengan baik. Meski telah menganalisis data dan mengetahui barang seperti apa yang mesti dijual sekalipun, jika tidak dapat didistribusikan dengan efektif dan efisien maka akan sama saja menyia-nyiakan modal berharga yang kita punya. Begitu pula apabila biaya menganalisis data jauh melampaui potensi keuntungan yang bisa diperoleh, maka hasil pengolahan data tersebut dapat diibaratkan "besar pasak daripada tiangnya".

TEKNOLOGI BIG DATA GOOGLE

Google dan Big Data

Google berasal dari sebuah proyek penelitian yang dimulai pada Januari 1996 oleh dua mahasiswa pasca sarjana, Larry Page dan Sergey Brin, di Stanford University. Saat ini Google telah berkembang menjadi raksasa Internet yang sangat pintar, yang telah menjadi guru tempat bertanya apa saja di dunia maya Internet. Boleh dikatakan terlalu naif bila ada pengguna Internet yang mengaku tidak tahu Google. Google telah menjadi bagian hidup para penjelajah dunia maya. Menjadi milik semuanya dan memiliki semuanya.

Sejak kelahirannya, Google dibekali dengan teknologi yang disebut PageRank yang telah menjadikannya search engine yang unik karena teknologi tersebut belum pernah dimiliki oleh search engine yang telah ada sebelum Google.

Bukan hanya itu, Google juga telah mampu mengelola, memanfaatkan, dan memberdayakan pertumbuhan data yang membludak pada era ledakan informasi (Information Explosion) abad XXI. Google memiliki teknologi crawler yang mampu mendownload seluruh web page yang ada di dunia maya Internet dan secara terus menerus mengikuti update pada jutaan web page tersebut[17]. Google memiliki teknologi sistem penyimpanan data yang dikenal dengan nama Google File System (GFS) yang menyimpan data secara terdistribusi dalam ribuan komputer[18]. Kapasitas penyimpanan GFS ini bisa terus diperbesar hanya dengan menambah jumlah komputer yang disertakan didalamnya. Google memiliki teknologi pengolahan data yang disebut MapReduce, yang mampu mengolah data yang disimpan dalam GFS secara paralel dan independen menggunakan ratusan hingga ribuan komputer, sehingga mampu mengolah data ukuran raksasa dalam waktu berkali lipat lebih cepat daripada sistem konvensional[19]. Google juga memiliki sistem database yang disebut Bigtable, teknologi yang mampu menyajikan data berukuran raksasa, yang sudah tak memungkinkan lagi untuk ditangani dengan menggunakan sistem database konvensional[20].

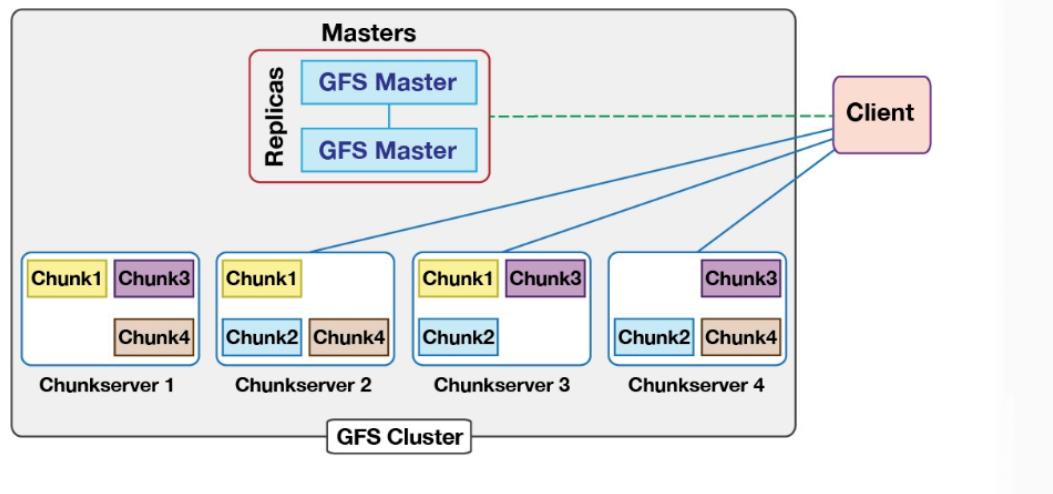
Dari segi teknologi, dipublikasikannya Google Bigtable pada 2006 telah menjadi momen kemunculan dan meluasnya kesadaran akan pentingnya kemampuan untuk memproses Big Data. Berbagai layanan yang disediakan Google, yang melibatkan pengolahan data dalam skala besar termasuk search engine-nya, dapat beroperasi secara optimal berkat adanya Bigtable. Semenjak itu, teknik pengelolaan dan penyimpanan data KVS (Key-Value Store) dan teknik komputasi paralel yang disebut MapReduce mulai menyedot banyak perhatian. Jadi, Google telah mampu melihat potensi pada Big Data, kemudian mengolah dan memberdayakannya jauh sebelum istilah Big Data mulai diperbincangkan. Oleh karena itu, mungkin tak berlebihan jika dikatakan bahwa Google adalah pelopor dalam pemanfaatan dan pemberdayaan Big Data.

Google File System

Google File System (GFS) adalah file system yang diterapkan pada sistem terdistribusi (distributed system). GFS merupakan suatu format file system yang dapat diibaratkan layaknya FAT32 ataupun NTFS yang digunakan untuk menformat media penyimpanan sebelum media tersebut dapat digunakan untuk menyimpan data digital. Bedanya, kalau FAT32 ataupun NTFS biasa digunakan untuk memformat media penyimpanan pada komputer yang berdiri sendiri (standalone), seperti: hard disk drive (HDD), USB flash disk, DVD-R, SD Card dan sebagainya, GFS digunakan untuk memformat media penyimpanan yang berupa sistem terdistribusi yang terdiri dari komputer-komputer yang tergabung dalam suatu cluster. Cluster dalam konteks ini adalah network atau jaringan komputer yang terdiri atas puluhan, ratusan atau bahkan ribuan komputer. Yang namanya network, berarti komputer yang ada didalamnya sudah dalam kondisi saling terkoneksi, bisa saling berkomunikasi, dan bisa saling bertukar data. Dalam hal ini, GFS telah merealisasikan sistem penyimpanan data secara terdistribusi pada komputer-komputer yang tergabung dalam suatu cluster. Berkat kemampuannya yang bisa menyimpan data secara terdistribusi, GFS dapat menampung data super besar yang tidak bisa disimpan dalam suatu HDD paling besar sekalipun.

Arsitektur Google File System

GFS memiliki tiga entitas / unsur utama, yaitu: Master Servers, Chunkservers, dan Clients. Master Servers adalah server sentral yang mengendalikan dan memonitor kondisi GFS secara keseluruhan. Chunkservers adalah unsur yang bertanggung jawab menyimpan / membaca file ke / dari HDD tiap node. Dalam konteks ini, node adalah satu unit komputer yang berada dalam suatu cluster GFS. Chunkservers yang jumlahnya banyak ini berada dibawah kendali Master Servers. Clients adalah aplikasi yang menggunakan GFS untuk menyimpan dan membaca file ke dan dari GFS (Gambar 3.1). Tiap chunk (potongan data) akan direplikasi secara otomatis dan disimpan dalam tiga Chunkservers. Ini berguna sebagai cadangan jika terdapat Chunkservers yang rusak. Kemudian Master Servers akan mengatur distribusi chunks (potongan-potongan data) ke tiap Chunkservers sedemikian rupa sehingga terdistribusi secara merata. Tidak ada yang mendapat chunk berlebih, juga tidak ada yang mendapat terlalu sedikit.



Gambar 3.1 Arsitektur Dasar Google File System

Pada saat suatu aplikasi membaca data, GFS akan memilih Chunkservers yang terdekat, sedangkan pada saat menulis / menyimpan data, GFS akan melakukannya dalam beberapa servers sekaligus. GFS tidak memfasilitasi penyimpanan / penulisan data secara bersamaan, melainkan dilakukan secara atomic. Artinya, selama data masih dalam proses penyimpanan / penulisan oleh suatu aplikasi, maka data tersebut beserta lokasinya dalam memory tidak akan bisa diakses oleh aplikasi lain sampai proses penyimpanan tersebut benar-benar selesai.

Keunggulan Google File System

Salah satu kelebihan GFS adalah GFS dapat dijalankan pada cluster yang terdiri atas komputer-komputer standar yang relatif tidak mahal. Google tidak menjalankan GFS dengan menggunakan komputer-komputer yang berspesifikasi super. Oleh karena itu, kerusakan atau kegagalan pada suatu node adalah hal yang lumrah terjadi dan sudah diprediksi sebelumnya. Untuk mengantisipasi hal ini, GFS didesain untuk tetap dapat berjalan normal walaupun terjadi kegagalan dalam suatu node. Singkatnya, GFS itu tahan banting.

Selain tahan banting, GFS juga scalable. Artinya, GFS bisa di-scale up dan di-scale out. Di-scale up artinya GFS bisa ditingkatkan kapasitasnya dengan meningkatkan spesifikasi komputer-komputer yang membentuk cluster-nya. Sedangkan di-scale out artinya GFS dapat diperbesar kapasitasnya dengan menambah jumlah komputer yang tergabung dalam cluster tempat GFS dijalankan. Kedua proses ini dapat dilakukan tanpa harus men-setup dari awal, tidak perlu setting ulang dan tak perlu melalukan pemindahan data dari sistem lama ke sistem baru yang telah di-upgrade. Beda halnya kalau kita menggunakan sistem konvensional. Pada saat akan mengganti atau meng-upgrade kapasitas ataupun spesifikasi komputer, kita harus melakukan setting ulang. Disamping itu, proses pemindahan data dari sistem lama ke sistem yang baru juga merupakan hal yang mutlak dilakukan.

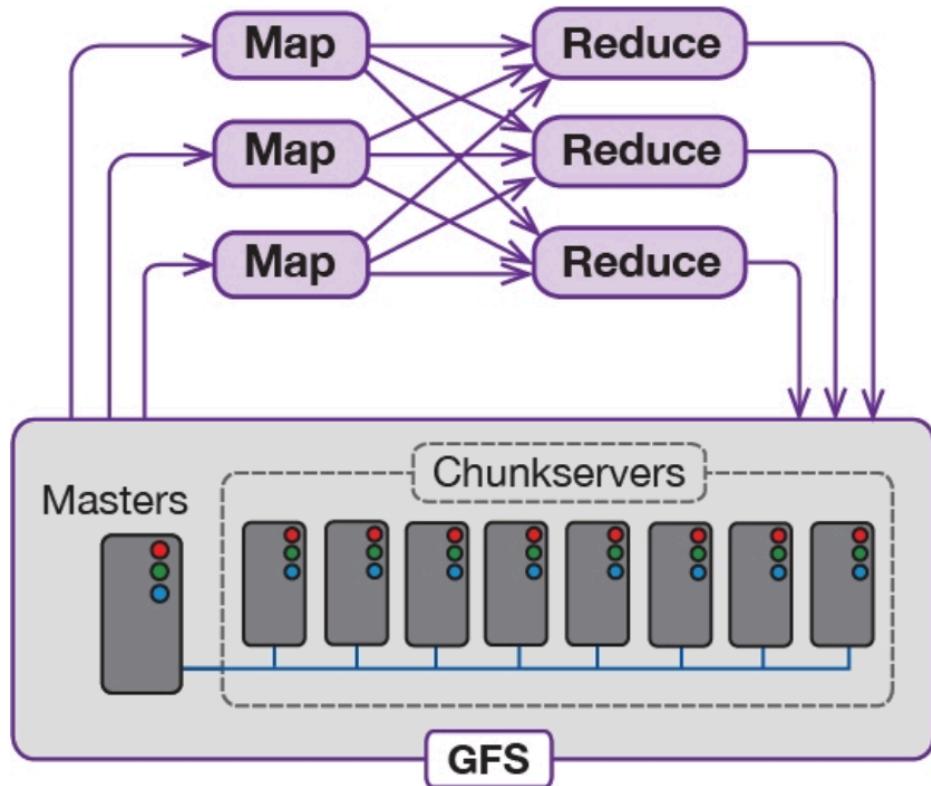
[Google Mapreduce](#)

MapReduce adalah model pemrograman rilisan Google yang ditujukan untuk memproses data berukuran raksasa secara terdistribusi dan paralel dalam cluster yang terdiri atas ribuan komputer. Dalam hal ini, cluster yang digunakan Google adalah cluster yang sudah diformat dengan GFS. Dalam memproses data, secara garis besar MapReduce dapat dibagi dalam dua proses yaitu proses Map dan proses Reduce. Kedua jenis proses ini didistribusikan atau dibagi-bagikan ke setiap komputer yang berada dalam cluster yang bersangkutan dan berjalan secara paralel tanpa saling bergantung satu dengan yang lainnya. Proses Map bertugas untuk mengumpulkan informasi dari potongan-potongan data yang terdistribusi dalam tiap komputer yang membentuk cluster dimana MapReduce dijalankan. Hasilnya diserahkan kepada proses Reduce untuk diproses lebih lanjut sesuai intensi pengguna / client. Hasil proses Reduce merupakan hasil akhir atau output dari MapReduce yang akan dikirim ke client-nya.

Desain yang Sederhana

Menyimak penjelasan diatas, mungkin akan timbul kesan bahwa MapReduce itu terlalu kompleks untuk diaplikasikan karena untuk memproses suatu data, data tersebut harus dipotong-potong kemudian didistribusikan ke tiap komputer yang dilibatkan. Lalu proses Map dan proses Reduce pun harus dibagi-bagikan ke tiap komputer tersebut dan dijalankan secara paralel. Hasil akhirnya pun, juga disimpan secara terdistribusi.

Beruntunglah, MapReduce telah didesain sedemikian rupa untuk dapat diaplikasikan secara sederhana alias simple. Sehingga, untuk menggunakan MapReduce, seorang programer cukup membuat dua program saja, yaitu program yang memuat kalkulasi atau prosedur yang akan dikerjakan oleh proses Map dan proses Reduce. Jadi, tidak perlu harus memikirkan bagaimana memotong-motong data untuk dibagi-bagikan kepada tiap komputer, dan memprosesnya secara paralel kemudian mengumpulkannya kembali. Semua proses ini akan dikerjakan secara otomatis oleh MapReduce yang dijalankan di dalam cluster GFS (Gambar 3.2).



Gambar 3.2 Proses Map dan Reduce pada MapReduce

Oleh karena itu, walaupun MapReduce menerapkan teknologi komputasi terdistribusi dan paralel yang sangat kompleks, dalam pola kerja dan pengoperasiannya ia dikenal sangat sederhana dibanding sistem lain yang menerapkan teknologi serupa.

Model Pemrograman : Fungsi Map dan Fungsi Reduce

Program yang memuat kalkulasi yang akan dilakukan dalam proses Map disebut Fungsi Map, dan yang memuat kalkulasi yang akan dikerjakan oleh proses Reduce disebut Fungsi Reduce. Jadi, seorang programmer yang akan menjalankan MapReduce harus membuat dua program, yaitu: program yang bekerja sebagai Fungsi Map dan program yang bekerja sebagai Fungsi Reduce.

Fungsi Map bertugas untuk membaca input dalam bentuk pasangan Key/Value, lalu menghasilkan output berupa pasangan Key/Value juga. Pasangan Key/Value hasil Fungsi Map ini disebut pasangan Key/Value intermediate. Kemudian, Fungsi Reduce akan membaca pasangan Key/Value intermediate hasil Fungsi Map, dan menggabungkan atau mengelompokkannya berdasarkan Key-nya masing-masing. Dengan kata lain, tiap Value yang memiliki Key yang sama akan digabungkan dalam satu kelompok yang sama. Fungsi Reduce juga menghasilkan output berupa pasangan Key/Value.

Untuk mempermudah pemahaman terhadap cara kerja Fungsi Map dan Fungsi Reduce, mari kita simak satu contoh program sederhana yang diberi nama Wordcount. Sesuai dengan namanya, program ini digunakan untuk menghitung jumlah / frekuensi kemunculan dari tiap kata dalam satu atau sekumpulan file teks dengan mengaplikasikan model pemrograman MapReduce. Dalam program Wordcount, Fungsi Map dan Fungsi Reduce dapat didefinisikan sebagai berikut:

```
map(String key, String value):
//key : nama file teks.
//value: isi file teks tersebut.
for each word W in value:
emitIntermediate(W,"1");
reduce(String key, Iterator values):
//key : sebuah kata.
//values : daftar berisi hasil hitungan.
int result = 0;
for each V in values:
result+= ParseInt(V);
emit(AsString(result));
```

Agar dapat melihat secara lebih nyata tentang bagaimana cara kerja Fungsi Map dan Fungsi Reduce pada program Wordcount ini, mari kita terapkan pada sebuah contoh file teks dengan nama "uang01.txt" dan "uang02.txt" yang isinya seperti berikut:

----- uang01.txt -----
Uang tidak menjamin bahagia.

Uang tidak bisa membeli cinta.

----- uang02.txt -----

Tapi, uang bisa membeli ruang dan waktu untuk bahagia.

Uang bisa membeli ruang dan waktu untuk cinta.

Kita proses dengan program Wordcount

```
map ( "uang01.txt", isi file "uang01.txt" )
```

Hasilnya:

```
( uang, "1" )
( tidak, "1" )
( menjamin, "1" )
( bahagia, "1" )
( uang, "1" )
( tidak, "1" )
( bisa, "1" )
( membeli, "1" )
( cinta, "1" )
```

```
map ( "uang02.txt", isi file "uang02.txt" )
```

Hasilnya:

```
( tapi, "1" )
( uang, "1" )
( bisa, "1" )
( membeli, "1" )
( ruang, "1" )
```

```
( dan, "1" )
( waktu, "1" )
( untuk, "1" )
( bahagia, "1" )
( uang, "1" )
( bisa, "1" )
( membeli, "1" )
( ruang, "1" )
( dan, "1" )
( waktu, "1" )
( untuk, "1" )
( cinta, "1" )
```

```
reduce ( uang, ["1", "1", "1", "1"] )
```

```
    result = 1 + 1 + 1 + 1 = 4
```

```
reduce ( tidak, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce ( menjamin, ["1"] )
```

```
    result = 1
```

```
reduce (bahagia, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce (bisa, ["1", "1", "1"] )
```

```
    result = 1 + 1 + 1 = 3
```

```
reduce (membeli, ["1", "1", "1"] )
```

```
    result = 1 + 1 + 1 = 3
```

```
reduce (cinta, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce ( tapi, ["1"] )
```

```
    result = 1
```

```
reduce (ruang, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce ( dan, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce ( waktu, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

```
reduce (untuk, ["1", "1"] )
```

```
    result = 1 + 1 = 2
```

Hasil Fungsi Reduce:

uang = 4

tidak = 2

menjamin = 1

bahagia = 2

bisa = 3

membeli = 3

cinta = 2

tapi = 1

ruang = 2

dan = 2

waktu = 2

untuk = 2

Demikianlah kita telah dapat menghitung jumlah tiap kata pada file teks "uang01.txt" dan "uang02.txt" secara paralel dengan menggunakan model pemrograman MapReduce dari program aplikasi Wordcount.

Ditujukan untuk Batch Processing

Google menciptakan MapReduce tentu dengan suatu maksud ataupun tujuan untuk dapat menuntaskan suatu pekerjaan tertentu secara efektif dan efisien. Dalam hal ini, MapReduce didesain untuk memproses data-data raksasa secara Batch Processing. Faktanya, MapReduce memang unggul dalam hal pemrosesan data yang sifatnya Batch Processing. Apa itu Batch Processing? Sederhananya, Batch Processing merupakan kebalikan dari Interactive Processing. Dalam Batch Processing, program akan megambil suatu set data sebagai input, memprosesnya, dan memberikan satu set data tertentu sebagai output tanpa intervensi manual dari user. Program berjalan berdasarkan setting yang ditentukan sebelum program dieksekusi. Begitu program dieksekusi, user tidak bisa melakukan intervensi. Program akan berjalan sampai selesai dengan suatu output. Batch Processing biasanya dilakukan untuk memproses data yang tidak berstruktur. Sedangkan pada Interactive Processing, program akan banyak berkomunikasi dengan user pada saat memproses suatu data. User dapat memberikan masukan ataupun koreksi pada saat program sedang berjalan. Di Google, pekerjaan-pekerjaan Batch Processing yang diselesaikan dengan MapReduce diantaranya: sorting data yang berukuran terabyte dan indexing data dalam skala besar. Google tidak memaksakan MapReduce untuk memproses data secara interaktif karena Google tahu tidak akan dapat diselesaikan secara efektif dan efisien.

Google Bigtable

Untuk melengkapi keberadaan MapReduce, Google telah menciptakan Bigtable untuk memproses data berskala besar yang harus dilakukan secara interaktif. Google Bigtable adalah sistem penyimpanan data terdistribusi yang ditujukan untuk mengelola data berstruktur dan didesain sebagai sistem yang handal untuk memproses data dalam skala petabytes dan dalam ribuan mesin (komputer). Dari definisi ini, kita bisa membayangkan bahwa Bigtable memang table yang benar-benar besar. Tabel yang berisi data dalam skala petabytes, bisa berukuran beberapa petabytes, bisa puluhan, ratusan bahkan ribuan petabytes. Data-data tersebut disimpan secara terdistribusi dalam ribuan komputer yang dapat diakses dan dikelola melalui Bigtable. Dalam hal kemampuan mengelola data, Bigtable dinyatakan sebagai sistem yang sangat fleksibel. Bigtable mampu memproses data mulai data yang berukuran kecil seperti halnya URL, data berukuran sedang seperti halnya web pages, hingga data berukuran besar berupa photo atau citra satelit. Bigtable bisa menangani pemrosesan data mentah 'gelondongan' yang berorientasi pada hasil akhir dan perlu diproses dalam jangka waktu tertentu (batch processing) maupun pemrosesan data secara real time yang menuntut hasil seketika (Interactive Processing). Google menggunakan Bigtable dalam lebih dari 60 produk dan proyeknya termasuk Google web indexing, Google Analytics, Google Finance, Orkut, Personalized Search, Writely dan Google Earth.

Apa Bedanya dengan Relational Database?

Kalau disederhanakan, Bigtable itu dapat dianggap sebagai suatu sistem manajemen database yang mampu menangani data dalam ukuran yang sangat besar yang sudah tak mampu lagi diatasi oleh sistem database yang telah ada sebelumnya (Relational Database Management System). Namun demikian, ada beberapa perbedaan mencolok antara Bigtable dan Relational Database Management System (RDBMS), diantaranya dalam hal struktur tabel dan bahasa pengoperasian.

Struktur tabel dalam Bigtable lebih kompleks dan fleksibel daripada tabel dalam RDBMS. Tabel dalam Bigtable tidak hanya terdiri atas baris dan kolom, tapi juga memiliki timestamps. Berkat adanya timestamps ini, tiap cell dalam suatu tabel Bigtable dapat memuat suatu data dalam beberapa versi. Jadi, dalam hal ini tabel dalam Bigtable terdiri atas tiga dimensi, yang bila digambarkan dengan sumbu x, y, dan z, x adalah baris, y adalah kolom dan z adalah timestamps. Tidak hanya itu, struktur kolom dalam Bigtable juga tidak sesederhana struktur kolom dalam tabel RDBMS. Kolom dalam Bigtable terdiri atas Column Families yang masing-masing terdiri atas beberapa hingga ribuan atau bahkan jutaan Column Keys.

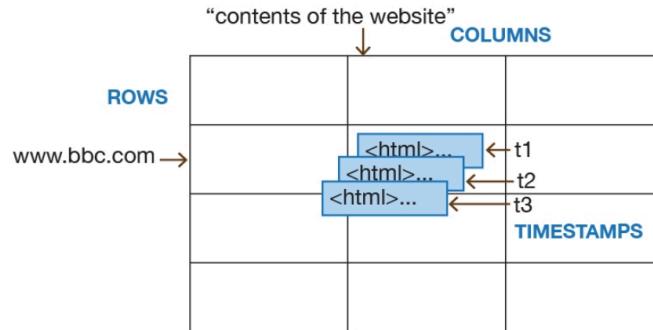
Dalam hal pengoperasiannya untuk memproses suatu data, Bigtable tidak menyediakan bahasa pengoperasian tersendiri namun praktis layaknya SQL. Bigtable harus dioperasikan dengan menggunakan bahasa pemrograman biasa seperti halnya C++ ataupun Java. Namun demikian, Google telah menyediakan library tersendiri bagi para programmer yang akan membuat aplikasi yang menggunakan Bigtable. Melalui aplikasi yang merupakan Client dari Bigtable inilah Bigtable dapat dioperasikan untuk memproses data.

Perbedaan-perbedaan lainnya tentu masih banyak, namun disini kita tidak akan membahas perbedaan tersendiri bagi para programmer yang akan membuat aplikasi yang menggunakan Bigtable. Melalui aplikasi yang merupakan Client dari Bigtable inilah Bigtable dapat dioperasikan untuk memproses data.

Perbedaan-perbedaan lainnya tentu masih banyak, namun disini kita tidak akan membahas perbedaan maupun persamaan kedua sistem tersebut secara lebih detail, juga tidak akan membandingkannya dari sudut pandang manapun. Kita akan fokus membahas spesifikasi Bigtable saja.

Model Data dalam Bigtable

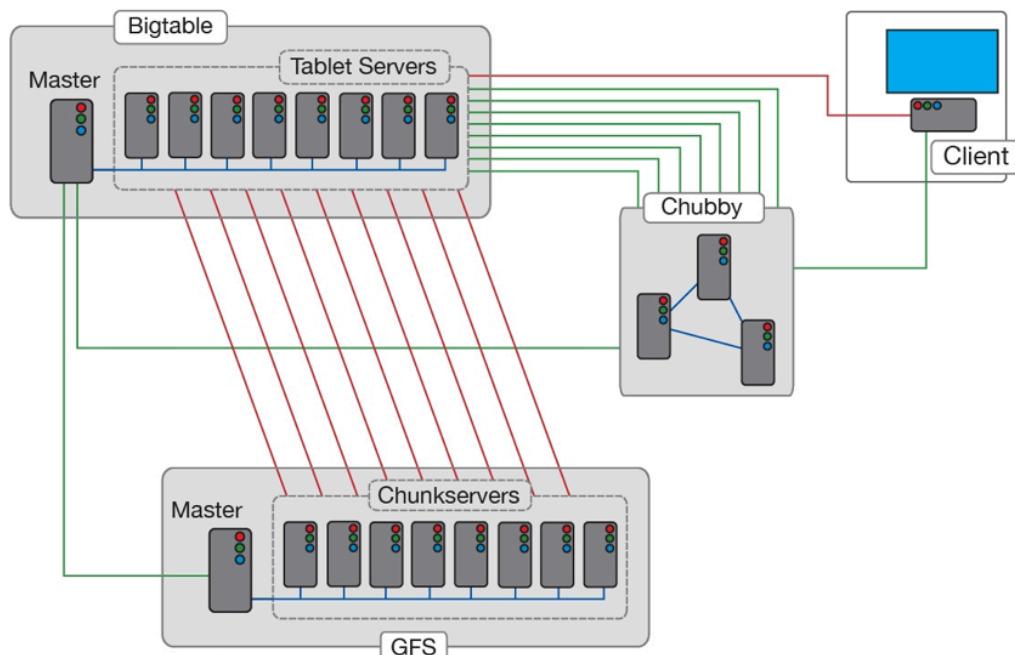
Telah disebutkan bahwa tabel dalam Bigtable terdiri atas baris, kolom, dan timestamps (Gambar 3.3). Baris dalam Bigtable dapat dianalogikan seperti halnya baris dalam tabel pada umumnya. Tidak berbeda dengan baris pada tabel RDB (Relational Database), tiap baris dalam Bigtable juga memiliki kunci yang disebut Row Key dan baris-baris tersebut disusun berurutan berdasar Row Keys ini. Kemudian, baris-baris yang berada dalam suatu bentangan Row Keys dikelompokkan dalam satu group disebut Tablet. Tiap tabel dalam Bigtable akan dibelah menjadi sejumlah Tablet yang berukuran antara 100 – 200 MB. Pada awalnya, tiap tabel hanya terdiri atas satu Tablet. Seiring dengan pertambahan jumlah barisnya (karena bertambahnya data yang dimuat) tabel akan secara otomatis di-split menjadi Tablet-tablet yang berukuran 100 hingga 200 MB tersebut. Tablet-tablet inilah yang kemudian dikelola secara terdistribusi, dibagi-bagikan ke tiap Tablet Servers. Kolom dalam Bigtable terdiri atas sejumlah Column Families yang masing-masing terdiri atas sejumlah Column Keys. Biasanya, data dalam satu tipe disimpan dalam satu Column Family. Kemudian Column Family ini dibagi-bagi dalam sejumlah Column Keys yang masing-masing menyimpan data yang berbeda (unik). Timestamps dalam Bigtable memungkinkan tiap cell memiliki beberapa versi data dari data yang sama yang tentunya dibedakan berdasarkan tanda waktu. Bigtable menentukan timestamps dalam microsecond, namun demikian timestamps juga dapat ditentukan sendiri oleh aplikasi client yang menggunakan Bigtable. Data yang telah memiliki timestamps ini disimpan dengan urutan menurun sehingga data yang paling baru akan berada paling atas (dapat diakses paling dahulu). Jadi, untuk mendapatkan suatu data yang berada dalam suatu cell Bigtable, kita harus menentukan Row Key, Column Key, dan Timestamps dari data tersebut.



Gambar 3.3 Model Data pada Tabel Bigtable

Arsitektur Implementasi Bigtable

Sebagai sistem manajemen data terdistribusi, Bigtable diimplementasikan pada cluster GFS. Dalam implementasinya tersebut Bigtable terdiri atas tiga komponen utama, yaitu: satu Master, sejumlah Tablet Servers, dan satu Library yang terhubung dengan tiap Client. Jumlah Tablet Servers dapat ditambah atau dikurangi secara dinamis menyesuaikan dengan besar kecilnya beban kerja. Ilustrasi arsitektur implementasi Bigtable dapat disimak pada Gambar 3.4. Garis-garis hijau menunjukkan jalur koordinasi, sedangkan garis-garis merah menunjukkan jalur aliran data read / write. Kemudian garis-garis biru menggambarkan network dari sejumlah komputer yang berada dalam satu cluster.



Gambar 3.4 Arsitektur Implementasi Bigtable

Master bertanggung jawab mendistribusikan Tablet kepada tiap Tablet Server, memonitor perubahan jumlah Tablet Servers, menyeimbangkan beban kerja tiap Tablet Server dengan meratakan pendistribusian Tablet, membersihkan file-file sampah pada GFS, dan menangani perubahan skema pada Bigtable seperti pembentukan tabel maupun Column Family baru. Singkat kata, Master bertanggungjawab mengontrol kerja Bigtable secara keseluruhan. Peranan Master pada Bigtable hampir sama dengan peranan Master pada GFS. Bedanya, Master pada Bigtable tidak berhubungan langsung dengan Client sedangkan Master pada GFS

berinteraksi langsung dengan Client. Tiap Tablet Server bertanggungjawab menangani satu set Tablets yang terdiri atas 10 – 1000 Tablets. Tablet Servers bertugas menangani permintaan read/write pada tiap Tablets yang menjadi tanggung jawabnya dan juga bertugas membelah Tablet yang sudah terlalu besar menjadi Tablet-tablet yang lebih kecil. Client pada Bigtable adalah program aplikasi yang mengoperasikan Bigtable itu sendiri. Seperti telah disebutkan sebelumnya, Client tidak berhubungan langsung dengan Master. Untuk mengetahui lokasi suatu Tablet, Client juga tidak bergantung pada Master. Client berkomunikasi langsung dengan Tablet Servers untuk read/write data. Dengan demikian, beban Master menjadi tidak begitu berat meskipun satu cluster Bigtable dapat terdiri atas ribuan komputer.

Kaitan Bigtable dengan Sistem Lain

Bigtable bukanlah sistem yang berdiri sendiri, melainkan adalah sistem yang terkait dengan teknologi Google lainnya, yaitu: GFS dan Chubby seperti yang diilustrasikan pada Gambar 3.4. Untuk menyimpan data dan log, Bigtable bergantung pada GFS. Semua data yang ditangani ataupun diproses oleh Bigtable termasuk juga log data Bigtable itu sendiri disimpan di cluster GFS. Jadi, andaikan terdapat Tablet Server yang rusak atau mati, data yang menjadi tanggung jawabnya tidak akan hilang karena sudah tersimpan dalam cluster GFS. Bila itu terjadi, Bigtable pun bisa dengan segera membentuk Tablet Server baru yang identik dengan Tablet Server yang telah mati dengan menggunakan informasi pada log data yang disimpan di cluster GFS. Chubby sebagai teknologi pendukung Bigtable berperan sebagai administrator yang memegang informasi-informasi mendasar mengenai Bigtable secara keseluruhan. Tablet Servers secara rutin berkomunikasi dengan Chubby untuk mengetahui kondisinya pada saat itu. Master juga secara rutin berkomunikasi dengan Chubby untuk mengetahui kondisi para Tablet Servers. Berdasarkan informasi ini, Master akan mengatur pendistribusian Tablet-tablet kepada para Tablet Servers.

Kesimpulan

Berdasarkan uraian diatas dapat disimpulkan bahwa ternyata Google telah mampu memberdayakan potensi Big Data jauh sebelum istilah Big Data tersebut menjadi perhatian dan bahan pembicaraan dalam dunia Teknologi Informasi. Google telah berhasil menciptakan dan mengembangkan teknologi Big Data yang telah terbukti handal dan mumpuni yang telah membuatnya sukses mendominasi dunia maya Internet. Teknologi tersebut diantaranya adalah GFS, MapReduce, dan Bigtable. Masing-masing telah didesain dan dikembangkan untuk menjalankan fungsi tertentu, yaitu: GFS sebagai sistem media penyimpanan terdistribusi, MapReduce untuk memproses data tak berstruktur yang sifatnya Batch Processing secara terdistribusi dan paralel, dan Bigtable sebagai sistem manajemen data terdistribusi untuk memproses data berstruktur yang sifatnya Interactive Processing. Dalam hal ini, MapReduce dan Bigtable dua-duanya beroperasi dalam cluster GFS.

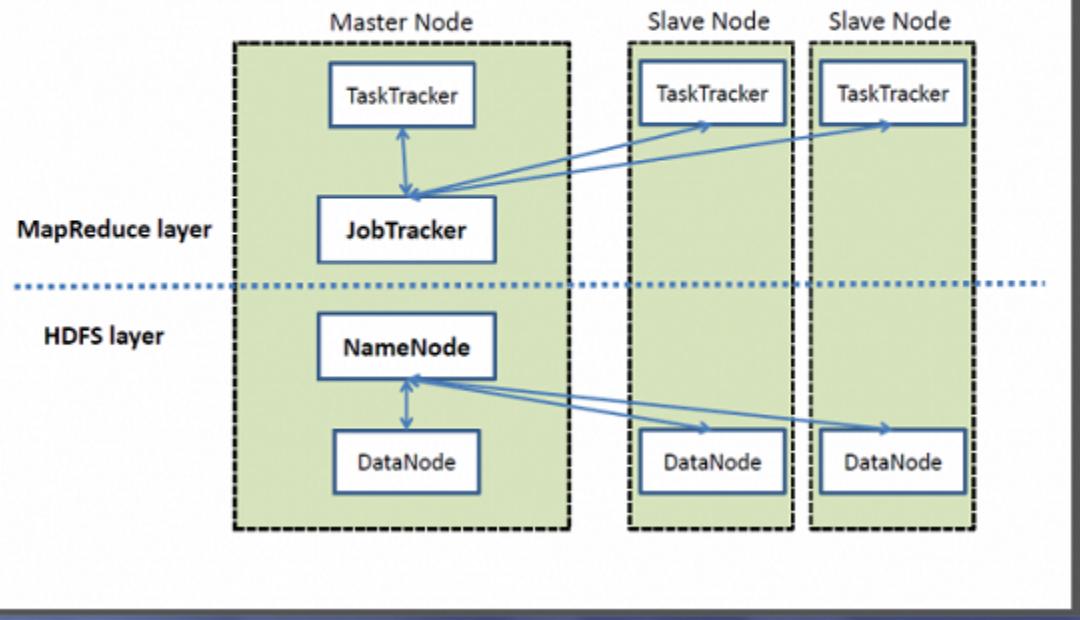
APACHE HADOOP

Pada bab sebelumnya telah dapat disimak bahwa keberhasilan Google dalam memberdayakan Big Data telah menjadi salah satu keunggulan strategis yang menjadikan Google berjaya sebagai raksasa Internet serta berhasil menjadi bagian yang tak terpisahkan dari sebagian besar pengguna Internet di dunia. Dalam hal ini, Google memiliki teknologi tersendiri untuk memproses Big Data, yaitu GFS, Google MapReduce, dan Google Bigtable. Google juga telah mempublikasikan konsep dan spesifikasi teknis dari ketiga teknologi Big Data tersebut. Sayangnya, Google tidak pernah mempublikasikan teknologi Big Data ciptaannya dalam bentuk framework ataupun software yang telah siap digunakan untuk memproses Big Data.

Namun demikian, Apache Software Foundation (ASF), yang juga melihat peluang besar pada fenomena Big Data, telah berhasil mengimplementasikan konsep dan spesifikasi teknis dari GFS, MapReduce dan Bigtable ke dalam bentuk framework yang dikembangkan dengan bahasa Java. Kemudian, pengembangan framework tersebut dikelola dalam bentuk proyek Open Source sehingga dapat diakses dan digunakan oleh semua orang, entah sebagai individu maupun sebagai organisasi / instansi. Pertama-tama, Apache mengembangkan sistem terdistribusi yang diberi nama Hadoop, yang juga dikenal sebagai Apache Hadoop.

Hadoop adalah sistem terdistribusi open source yang menerapkan programming model sederhana yang ditujukan untuk memproses data berukuran raksasa dalam suatu cluster komputer. Hadoop terdiri atas tiga komponen pokok, yaitu Hadoop Common, HDFS (Hadoop Distributed File System), dan Hadoop MapReduce. Namun, sejak dirilisnya Hadoop 2, selain tiga komponen pokok tersebut, Hadoop juga dilengkapi dengan job scheduling and cluster management framework yang disebut dengan nama YARN (Gambar 4.1). Dilihat dari fungsinya, YARN juga dianggap sebagai Operating System (OS)-nya Hadoop[21].

High Level Architecture of Hadoop



HDFS

HDFS (Hadoop Distributed File System), sesuai dengan namanya, adalah sistem file terdistribusi pada Hadoop, yang berfungsi untuk menangani penyimpanan data berukuran raksasa pada sistem terdistribusi Hadoop. Yang dimaksud dengan sistem file terdistribusi (distributed file system) adalah file system yang menyimpan data tidak dalam satu hard disk drive (HDD) atau media penyimpanan lainnya, tetapi data dipecah-pecah dan disimpan tersebar dalam suatu cluster yang terdiri atas sejumlah komputer, bisa hanya puluhan, ratusan, atau bahkan ribuan komputer. Cluster komputer yang telah diformat dengan HDFS disebut cluster Hadoop. Kemudian, yang dimaksud dengan data berukuran raksasa belum ada definisi yang baku. Namun demikian, kita dapat mengambil cluster Hadoop yang dioperasikan Yahoo! sebagai contoh. Yahoo! memiliki cluster Hadoop yang terdiri atas 3500 komputer dan menangani data sebesar 25 petabytes (1 petabyte = 1000 terabytes, 1 terabyte = 1000 gigabytes)[0]. Data sebesar ini dapat digolongkan sebagai data berukuran raksasa dan secara volume sudah dapat dikategorikan sebagai Big Data.

Jika disederhanakan, HDFS dapat dianggap sebagai medan dikembangkan berdasarkan konsep dan spesifikasi teknis dari GFS yang telah dipublikasikan oleh Google. Dalam HDFS, suatu file yang berukuran raksasa akan dipecah-pecah menjadi block-block dengan ukuran tertentu, kemudian tiap block akan direplikasi menjadi 3 (atau sesuai yang dikehendaki user) yang selanjutnya tiap replikasi tersebut disimpan secara terpisah dalam beberapa computer penyusun cluster Hadoop. Ukuran block dapat disesuaikan dengan keperluan user, tetapi secara default ukurannya adalah 128 megabytes.

Arsitektur HDFS

Sebagai sistem terdisitribusi, HDFS terdiri atas 1 Namenode yang bertindak sebagai master dan sejumlah Datanode yang ditugaskan sebagai slaves. Namenode adalah komputer yang bertugas menyimpan informasi tentang distribusi blok-blok data pada cluster Hadoop, dia menyimpan pada cluster Hadoop. HDFS didesain dan mengatur pendistribusian blok-blok data tersebut di dalam cluster Hadoop secara keseluruhan. Sedangkan Datanode adalah komputer yang bertugas menyimpan potongan-potongan data yang didistribusikan kepadanya

dan secara berkala melaporkan kondisi dirinya kepada Namenode. Laporan dari Datanode kepada Namenode tentang kondisi diri Datanode tersebut disebut dengan heart beat, yang secara default dilakukan dengan interval 3 detik. Berdasarkan heart beat ini lah Namenode dapat mengetahui dan menguasai kondisi cluster secara keseluruhan, dan sebagai jawaban atas heart beat ini, Namenode akan mengirimkan perintah tertentu kepada Datanode yang bersangkutan. Jika dalam 10 detik Namenode tidak menerima laporan heart beat, maka Datanode yang bersangkutan akan dikeluarkan dari cluster Hadoop. Membaca dan Menyimpan Data pada HDFS

Untuk menyimpan suatu data ke HDFS, pertama-tama komputer klien (Hadoop Client) akan menghubungi Namenode untuk mendapatkan alamat (address) dari Datanode yang dapat melayani permintaan penyimpanan data. Setelah itu, komputer pengguna akan secara langsung menyimpan datanya ke Datanode yang alamatnya telah didapatkan tersebut. Pada saat itu, data yang disimpan akan secara otomatis dipecah-pecah menjadi blok-blok data dengan ukuran yang telah ditentukan dan juga akan direplikasi menjadi 3 atau sesuai jumlah yang ditentukan sebelumnya. Bila Datanode telah berhasil menyimpan potongan-potongan data tersebut, maka Datanode akan melaporkan kepada Namenode bahwa penyimpanan data telah sukses atau telah berjalan normal.

Kemudian, untuk membaca data dari HDFS, pertama-tama komputer klien akan menyampaikan permintaan membaca data kepada Namenode untuk mendapatkan nama dan alamat Datanode yang menyimpan data yang diinginkan. Setelah itu, komputer klien akan langsung mengakses Datanode yang bersangkutan untuk mendapatkan data yang dikehendaki tersebut.

Hadoop MapReduce

Hadoop MapReduce dikembangkan berdasarkan konsep dan spesifikasi teknis Google MapReduce yang diimplementasikan dengan menggunakan bahasa program Java. Oleh karena itu, Hadoop MapReduce memiliki programming model yang identik dengan Google MapReduce. Demikian juga dengan arsitekturnya, Hadoop MapReduce memiliki arsitektur yang ekuivalen dengan Google MapReduce. Perbedaannya adalah Google MapReduce dijalankan dalam cluster GFS, sedangkan Hadoop MapReduce dioperasikan dalam cluster Hadoop yang tentunya sudah dalam format HDFS.

Arsitektur Hadoop MapReduce

Hadoop MapReduce adalah framework yang ditujukan untuk memproses data berukuran raksasa secara terdistribusi dan paralel dalam cluster Hadoop. Dilihat dari sisi arsitekturnya, Hadoop MapReduce terdiri atas 1 komputer yang berfungsi sebagai JobTracker dan sejumlah komputer yang berfungsi sebagai TaskTracker. Hubungan kerja antara JobTracker dan TaskTracker adalah layaknya hubungan kerja master dan slaves. Sebagai master, JobTracker memegang informasi tentang pemrosesan blok-blok data pada HDFS secara paralel, JobTracker tahu pasti komputer mana yang sedang memproses tiap-tiap blok data tersebut. JobTracker mengawasi komputer-komputer yang sedang memproses blok-blok data secara paralel dan mengendalikannya. Sedangkan TaskTracker, sebagai slave, adalah komputer pekerja yang bertugas memproses blok-blok data yang menjadi tanggung jawabnya. Dalam konteks ini, Task adalah process yang berjalan pada tiap komputer dalam memproses tiap blok data. Dalam waktu tertentu, TaskTracker akan mengatur dan mengendalikan sejumlah Task dan secara berkala memberikan laporan kepada JobTracker. Laporan berkala dari TaskTracker kepada JobTracker ini disebut heartbeat. Berdasarkan heartbeat ini lah JobTracker dapat mengetahui kondisi TaskTracker dan mengendalikannya.

Pada cluster Hadoop, keberadaan Namenode dan JobTracker tidak harus dalam satu komputer yang sama, sedangkan Datanode dan TaskTracker dibuat berada dalam satu komputer yang sama untuk mempertahankan data locality.

Model Pemrograman Hadoop MapReduce

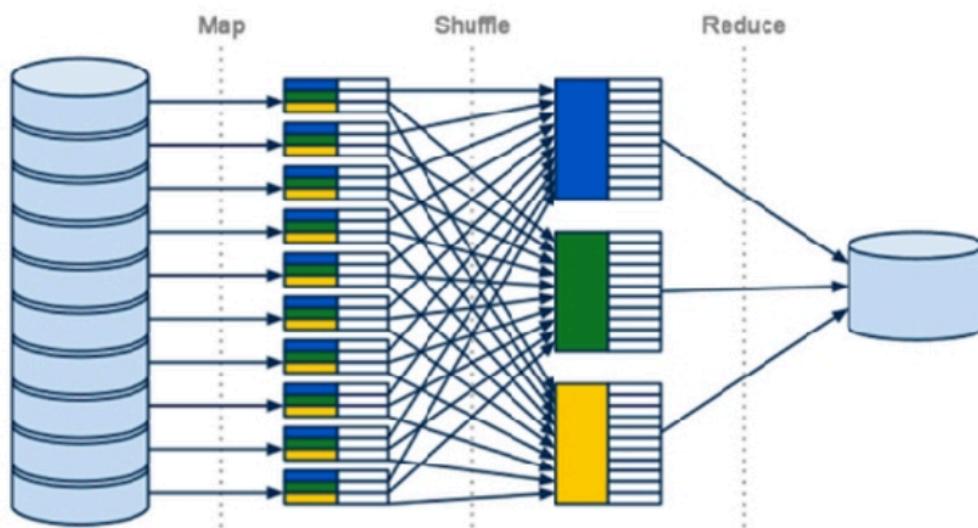
Sebagaimana telah disebutkan diatas, Hadoop MapReduce memiliki programming model yang identik dengan Google MapReduce, yaitu: terdiri atas fase Map dan fase Reduce.

Sebenarnya, diantara fase Map dan fase Reduce terdapat fase Shuffle. Namun demikian, fase Shuffle ini tidak bersifat mutlak, bisa ada atau ditidakan bergantung dari bagaimana suatu data akan diproses. Pada fase Map, Mapper akan memproses blok-blok data dan menghasilkan intermediate key/value pair (pasangan key/value intermediate) sebagai output. Kemudian, pada fase Shuffle, terdapat Comparator yang bertugas menyortir (mengurutkan intermediate key/value pair) berdasarkan key-nya, dan Partitioner yang bertugas untuk menentukan ke Reducer nomor berapa tiap intermediate key/value pair yang diterimanya mesti diserahkan. Selanjutnya, pada fase Reduce, value dari tiap-tiap key dari data yang telah disortir/diurutkan tersebut akan dibaca secara berurutan untuk diproses sesuai dengan yang diprogramkan. Hasil dari proses ini akan disimpan di HDFS (Gambar 4.2).

Mencari Mutual Friends Ala Facebook dengan MapReduce

Pada bab sebelumnya telah dibahas tentang program Wordcount sebagai satu contoh program aplikasi sederhana yang mengaplikasikan programming model MapReduce untuk menghitung jumlah tiap kata yang terdapat dalam satu atau sejumlah file teks. Pada bab ini kita akan membahas satu contoh lagi yang sedikit lebih

kompleks, yaitu: program aplikasi MapReduce untuk mencari jumlah Mutual Friends dari network/jaringan pertemanan ala Facebook[22]. Bagi para pengguna Facebook, pastinya sudah sangat familiar dengan istilah Mutual Friends ini. Lalu, bagaimana Facebook dapat menghitung dan meng-update jumlah Mutual Friends dari tiap usernya yang jumlahnya sudah lebih dari satu miliar dan masih terus bertambah itu? Disini akan kita lihat bagaimana MapReduce dapat memberikan solusi yang cukup sederhana namun cepat dan akurat.



Gambar 4.2 Model Pemrosesan Data pada MapReduce (developers.google.com)

Facebook memiliki daftar pertemanan (list of friends), dan perlu dicatat bahwa friends atau pertemanan dalam Facebook bersifat dua arah : jika aku adalah temanmu maka kamu adalah temanku juga. Facebook juga memiliki server penyimpanan data yang sangat besar dan melayani ratusan juta permintaan akses setiap harinya. Berkaitan

dengan hal ini, Facebook telah memutuskan untuk melakukan komputasi pendahuluan / pre-computation (jika memungkinkan) untuk mempersiapkan jawaban / responses terhadap permintaan akses yang jumlahnya ratusan juta tersebut. Hal ini diharapkan dapat memberikan respon yang lebih cepat terhadap suatu permintaan daripada melakukan komputasi setiap kali

permintaan akses itu datang. Salah satu permintaan penghitungan yang sangat umum adalah Mutual Friends. Sebagai contoh, saat saya melihat profile Facebook milik Henny, saya dapat mengetahui berapa jumlah Mutual Friends antara saya dan Henny : "Kamu dan Henny memiliki 54 teman yang sama". Kemudian saya juga akan dapat mengetahui siapa saja yang termasuk dalam 54 orang tersebut. Jumlah teman yang sama atau Mutual Friends ini tentu saja tidak setiap saat berubah, jadi akan sangat mubazir jika Facebook harus selalu menghitung kembali setiap kali saya melihat profile-nya Henny.

Dalam kasus ini, kita akan menggunakan MapReduce untuk menghitung jumlah teman yang sama atau Mutual Friends dari setiap pengguna Facebook sekali sehari dan menyimpan hasilnya dalam bentuk Key Value Store (Apache HBase). Dengan demikian, setiap kali ada seorang pengguna yang melihat profile pengguna yang lain, maka jumlah teman yang sama atau Mutual Friends dari kedua orang ini dapat langsung diketahui dari database Key Value Store tanpa menghitungnya lagi.

Bagaimana algoritmanya? Kita dapat menyimpan daftar pertemanan dari seluruh user Facebook dalam bentuk pasangan key dan value dengan format seperti berikut:

Key->Value : Orang->[Daftar Teman]

Sebagai contoh, daftar pertemanan tersebut akan :

terlihat seperti berikut

Annya -> [Brahm, Chieya, Dipa]
Brahm -> [Annya, Chieya, Dipa, Elsa]
Chieya -> [Annya, Brahm, Dipa, Elsa]
Dipa -> [Annya, Brahm, Chieya, Elsa]
Elsa -> [Brahm, Chieya, Dipa]

Setiap baris, yang merupakan pasangan key->value, akan menjadi argument dari sebuah Mapper dalam MapReduce. Mapper tersebut akan menghasilkan satu pasang key->value dari setiap teman dalam "Daftar Teman". Yang menjadi key adalah seorang teman dari "Daftar Teman" dan "Orang". Sedangkan yang menjadi value adalah "Daftar Teman". Kemudian, key akan diurut berdasarkan urutan alfabet (abjad).

Setelah proses Mapper selesai, maka daftar pertemanan diatas akan menjadi seperti berikut (key diurut berdasar abjad sehingga semua pasangan pertemanan dari key yang sama akan diproses lanjut oleh Reducer yang sama pula dalam MapReduce) :

For map(Annya -> [Brahm, Chieya, Dipa]), hasilnya :

(Annya Brahm) -> [Brahm, Chieya, Dipa]
(Annya Chieya) -> [Brahm, Chieya, Dipa]
(Annya Dipa) -> [Brahm, Chieya, Dipa]

For map(Brahm -> [Annya, Chieya, Dipa, Elsa])

hasilnya : (pada bagian key dapat dilihat Annya ditulis lebih dahulu daripada Brahm)

(Annya Brahm) -> [Annya, Chieya, Dipa, Elsa]
(Brahm Chieya) -> [Annya, Chieya, Dipa, Elsa]
(Brahm Dipa) -> [Annya, Chieya, Dipa, Elsa]
(Brahm Elsa) -> [Annya, Chieya, Dipa, Elsa]

For map(Chieya -> [Annya, Brahm, Dipa, Elsa]), hasilnya:

(Annya Chieya) -> [Annya, Brahm, Dipa, Elsa]
(Brahm Chieya) -> [Annya, Brahm, Dipa, Elsa]
(Chieya Dipa) -> [Annya, Brahm, Dipa, Elsa]

(Chieya Elsa) -> [Annya, Brahm, Dipa, Elsa]

For map(Dipa -> [Annya, Brahm, Chieya, Elsa]), hasilnya:

(Annya Dipa) -> [Annya, Brahm, Chieya, Elsa]
(Brahm Dipa) -> [Annya, Brahm, Chieya, Elsa]
(Chieya Dipa) -> [Annya, Brahm, Chieya, Elsa]
(Dipa Elsa) -> [Annya, Brahm, Chieya, Elsa]

For map(Elsa -> [Brahm, Chieya, Dipa]), hasilnya:

(Brahm Elsa) -> [Brahm, Chieya, Dipa]
(Chieya Elsa) -> [Brahm, Chieya, Dipa]
(Dipa Elsa) -> [Brahm, Chieya, Dipa]

Sebelum pasangan key->value ini diproses lanjut oleh tiap Reducer-nya, kita akan mengelompokkannya berdasarkan key-nya masing-masing seperti berikut:

(Annya Brahm) -> [Annya, Chieya, Dipa, Elsa] [Brahm, Chieya, Dipa]
(Annya Chieya) -> [Annya, Brahm, Dipa, Elsa] [Brahm, Chieya, Dipa]
(Annya Dipa) -> [Annya, Brahm, Chieya, Elsa] [Brahm, Chieya, Dipa]
(Brahm Chieya) -> [Annya, Brahm, Dipa, Elsa] [Annya, Chieya, Dipa, Elsa]
(Brahm Dipa) -> [Annya, Brahm, Chieya, Elsa] [Annya, Chieya, Dipa, Elsa]
(Brahm Elsa) -> [Annya, Chieya, Dipa, Elsa] [Brahm, Chieya, Dipa]
(Chieya Dipa) -> [Annya, Brahm, Chieya, Elsa] [Annya, Brahm, Dipa, Elsa]
(Chieya Elsa) -> [Annya, Brahm, Dipa, Elsa] [Brahm, Chieya, Dipa]
(Dipa Elsa) -> [Annya, Brahm, Chieya, Elsa] [Brahm, Chieya, Dipa]

Tiap baris, yang merupakan pasangan key->[list of value] akan menjadi argument dari sebuah Reducer. Kemudian reduce function akan mengambil perpotongan / interseksi dari value yang terdapat dalam [list of value]. Output dari reduce function adalah key dan value (yang tak lain adalah hasil dari interseksi tersebut) :

key -> [hasil interseksi]. Sebagai contoh, reduce((Annya Brahm)
-> [Annya, Chieya, Dipa, Elsa] [Brahm, Chieya, Dipa]) menghasilkan (Annya Brahm)
-> [Chieya Dipa] yang berarti Annya dan Brahm memiliki 2 teman yang sama (Mutual Friends)
yaitu Chieya dan Dipa.

Hasil dari seluruh proses Reducer adalah sebagai berikut:

(Annya Brahm) -> [Chieya Dipa]
(Annya Chieya) -> [Brahm Dipa]
(Annya Dipa) -> [Brahm Chieya]
(Brahm Chieya) -> [Annya Dipa Elsa]
(Brahm Dipa) -> [Annya Chieya Elsa]
(Brahm Elsa) -> [Chieya Dipa]
(Chieya Dipa) -> [Annya Brahm Elsa]
(Chieya Elsa) -> [Brahm Dipa]
(Dipa Elsa) -> [Brahm Chieya]

Nah, sekarang ketika Dipa melihat profile Brahm, kita dapat dengan cepat menunjuk pada key (Brahm Dipa) dan melihat bahwa mereka memiliki 3 teman yang sama (Mutual Friends) yaitu [Annya Chieya Elsa].

Apache HBase

Seperti halnya Google MapReduce, Hadoop MapReduce terbukti unggul dalam memproses data berukuran rakasa yang sifatnya batch processing, tetapi memiliki kelemahan dalam memproses data yang sifatnya interactive / real time processing. Dengan alasan ini, ASF kemudian mengembangkan Apache HBase berdasarkan konsep dan spesifikasi teknis dari Google Bigtable yang diimplementasikan dalam bahasa program Java. Dengan HBase, Big Data dapat diproses secara real time / interactive di dalam HDFS.

Apache HBase adalah database Hadoop, yaitu database terdistribusi yang berorientasi pada kolom, bersifat scalable, dan ditujukan untuk menyimpan Big Data. Oleh ASF, Apache HBase dikembangkan dan dipublikasikan dalam proyek sebuah open source.

HBase VS RDBMS

Relational Database Management System (RDBMS) seperti halnya MySQL maupun PostgreSQL, yang sudah sangat dikenal dan biasa digunakan oleh berbagai kalangan, pada awalnya tidaklah didesain untuk menangani data super besar dan dalam sistem terdistribusi yang memperkerjakan puluhan hingga ribuan komputer. RDBMS memang memiliki fungsi sharding atau partisi data pada saat volume data yang dikelola sudah tak bisa lagi ditangani dalam satu mesin (komputer). Namun demikian, fungsi sharding ini adalah fungsi tambahan yang cukup kompleks baik dalam proses instal maupun pemeliharaannya. Sedangkan, HBase sedari awal memang didesain untuk dapat mengelola data berukuran super besar dalam suatu sistem terdistribusi dan memiliki fungsi sharding original bawaan yang dapat bekerja secara otomatis maupun manual. HBase memiliki karakteristik fault tolerance atau mampu menjamin keutuhan data meskipun terjadi kegagalan pada beberapa komputer yang diperjerjakannya. HBase juga mampu menangani input data yang terjadi secara terus-menerus dari ribuan user yang selama ini menjadi bottle neck pada sistem database konvensional.

Struktur Data pada HBase

Dalam HBase, data dikelola dalam bentuk key value store (KVS), yakni: data tersimpan dengan format pasangan key dan value (key/value pair). Key dalam HBase terdiri atas row key (key pada baris), column key (key pada kolom), dan time stamp, sedangkan value-nya adalah data yang berkoresponden dengan key tersebut. Row key adalah primary key (key utama). Column key terdiri atas column family dan qualifier. Data yang disimpan sebagai value akan diurut berdasarkan row key, column key, dan time stamp.

KEY				VALUE
row key	column family	qualifier	timestamp	value
cargo.001	Geohash	aaa0123xyz	10:30:25	cargo.001 AIS broadcast data
cargo.001	Geohash	aaa0123xyz	10:30:45	cargo.001 AIS broadcast data
cargo.001	Geohash	aab0123xyz	09:30:30	cargo.002 AIS broadcast data
cargo.001	Geohash	aac0123xyz	11:30:15	tanker.001 AIS broadcast data
cargo.002	Geohash	aaa0123xyz	10:45:25	tanker.001 AIS broadcast data
Region 2	tanker.001	Geohash	aaa0123xyz	10:55:20 cargo.001 AIS broadcast data
	tanker.001	Geohash	aaa0123xyz	11:30:40 cargo.001 AIS broadcast data
	tanker.002	Geohash	aaa0123xyz	09:30:30 cargo.002 AIS broadcast data
	tanker.002	Geohash	aab0123xyz	11:30:15 tanker.001 AIS broadcast data
	tanker.003	Geohash	aab0123xyz	10:45:25 tanker.001 AIS broadcast data

Gambar 4.3 Struktur data pada HTable

Sebagai sistem yang mengelola data dalam bentuk KVS, HBase menyimpan dan menampilkan data secara terstruktur dalam bentuk tabel yang disebut HTable. Baris pada HTable dinyatakan dengan row key, sedangkan kolomnya dinyatakan dengan column key. Data atau value pada HTable dimuat dalam cell yang lokasinya ditunjukkan oleh baris dan kolom. Data dalam cell tersebut akan diurut berdasarkan time stamp. Dalam hal ini, cell dengan versi terbaru akan berada pada urutan paling atas. Seiring dengan bertambahnya data yang disimpan HTable akan terus bertambah besar, dan ketika ukuran HTable telah melampaui ukuran maksimal yang telah ditentukan sebelumnya, maka HTable akan terbelah menjadi dua Region dengan ukuran yang sama besar (Gambar 4.3). Secara fisik, data pada HBase akan dipartisi berdasarkan column family dan Region-nya masing-masing yang kemudian disimpan pada Datanode-datanode HDFS.

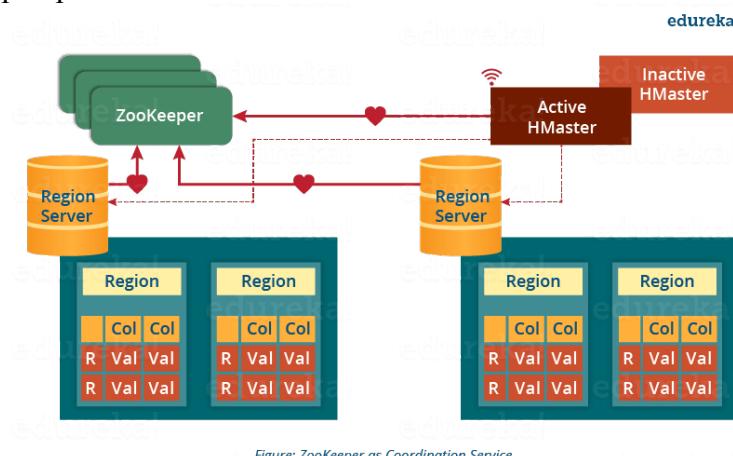


Figure: ZooKeeper as Coordination Service

Gambar 4.4 Arsitektur HBase

Arsitektur HBase

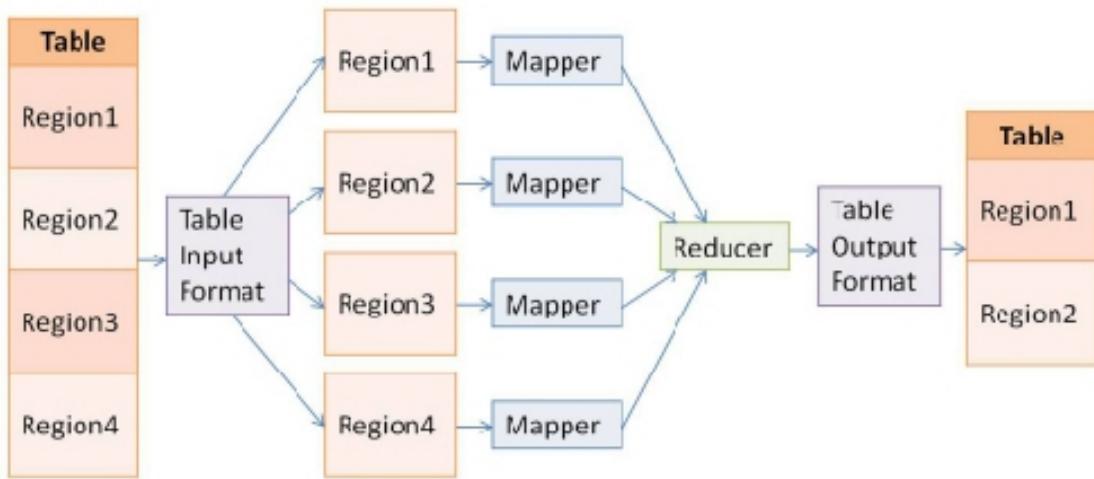
HBase terbentuk oleh 4 komponen penyusunnya, yaitu: HMaster, RegionServer, ZooKeeper, dan HDFS (Fungsi dari masing-masing komponen penyusunnya tersebut adalah sebagai berikut:

1. HMaster:
 - a. Menghidupkan HBase setelah berhasil menjalin koneksi dengan HDFS.
 - b. Mendistribusikan Region kepada RegionServer-RegionServer yang terdaftar serta memperbaiki atau memulihkan RegionServer yang mengalami kerusakan atau kegagalan.
2. RegionServer:
 - a. Menyimpan dan memanage Region yang didistribusikan kepadanya.
 - b. Membelah tiap Region yang telah tumbuh membesar melampaui ukuran maksimalnya, dan melaporkan Region-Region baru hasil pembelahan kepada HMaster.
 - c. Merespon atau menangani permintaan klien yang hendak membaca maupun menyimpan data di HBase.
3. ZooKeeper:
 - a. Menyimpan informasi yang memuat lokasi Root Catalog HBase dan alamat HMaster.
 - b. Mengawasi kondisi tiap RegionServer dan senantiasa menyediakan informasi tentang kondisi terkini seluruh RegionServer kepada HMaster. Berdasarkan informasi inilah HMaster akan mendistribusikan Region kepada para RegionServer dan memulihkan kondisi RegionServer yang mengalami kerusakan atau kegagalan.
4. HDFS: Tempat menyimpan data dari klien dan data log HBase.

Untuk mengakses data pada HBase saat pertama kalinya, klien harus menghubungi ZooKeeper terlebih dahulu untuk dapat menjalin koneksi dengan RegionServer. Kemudian, untuk akses selanjutnya, klien dapat secara langsung menjalin koneksi dengan RegionServer berdasarkan cache yang dimilikinya. Saat koneksi dengan RegionServer telah terjalin, klien dapat mengirim permintaan untuk membaca maupun menyimpan data di HBase.

Kompatibilitas HBase dengan MapReduce

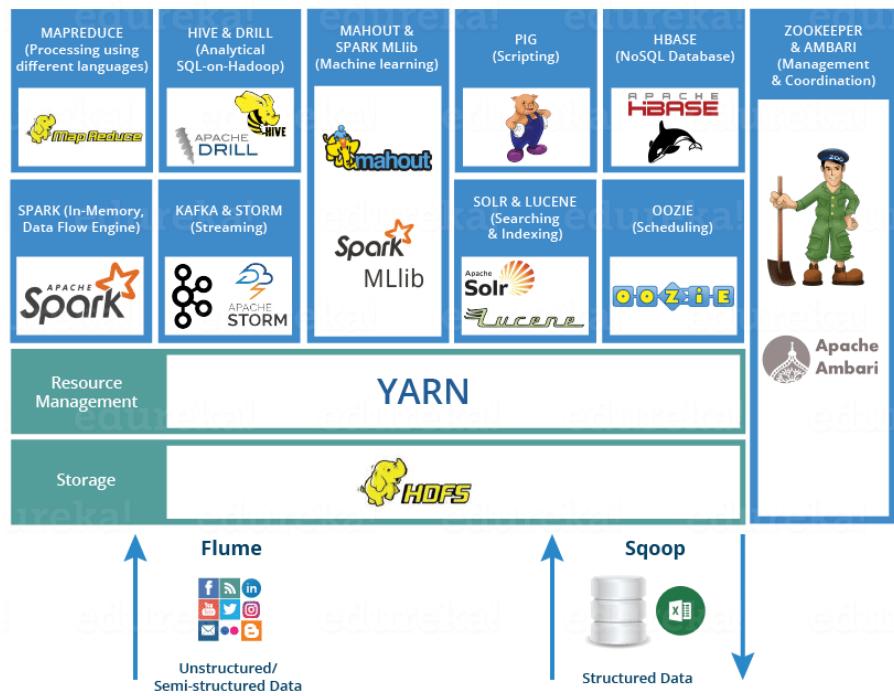
Telah disebutkan sebelumnya bahwa HBase dikembangkan untuk memungkinkan Big Data dapat diproses secara interaktif / real time dalam suatu sistem terdistribusi, khususnya dalam HDFS. Namun demikian, HBase tidak terbatas hanya untuk memproses data secara interaktif saja, HBase dapat menggerakkan MapReduce untuk memproses data secara batch processing. Dalam hal ini, MapReduce akan mengambil input data dari HBase dalam bentuk TableInputFormat, memproses data tersebut sesuai yang diprogramkan, kemudian memberikan output data kepada HBase dalam bentuk TableOutputFormat (Gambar 4.5).



Gambar 4.5 Kompatibilitas HBase dengan MapReduce

Ekosistem Hadoop

Pada awal pengembangannya, Hadoop memang hanya terdiri atas dua framework, yaitu: HDFS sebagai sistem file terdisitribusi untuk menyimpan data dengan ukuran raksasa dan MapReduce untuk memproses data raksasa tersebut secara terdisitribusi dan paralel. Namun demikian, saat ini Hadoop telah berkembang menjadi pondasi bagi framework-framework lain yang memiliki kemampuan untuk memproses data secara terdisitribusi seperti halnya HBase. Apalagi semenjak dirilisnya Hadoop 2, yaitu Hadoop yang telah dilengkapi dengan YARN yang berfungsi sebagai Operating System pada HDFS, Hadoop menjadi makin fleksibel dalam mewadahi aplikasi-aplikasi lain yang membutuhkan suatu sistem terdisitribusi sebagai pondasinya. Oleh karena itu, kini Hadoop telah dianggap sebagai sebuah ekosistem bagi framework maupun aplikasi lain yang didesain untuk menjalankan fungsi tertentu di dalam suatu sistem terdisitribusi (Gambar 4.6).



TUTORIAL HADOOP DAN HBASE

Instal Hadoop

Sebagai sebuah sistem terdistribusi, sudah tentu diperlukan sejumlah komputer untuk menjalankan Hadoop. Namun demikian, Apache Hadoop dapat dijalankan dalam tiga mode, yaitu: mode standalone, mode pseudo-distributed, dan mode fully distributed.

1. Mode Standalone adalah menjalankan Hadoop dalam satu PC Windows, Linux, maupun Mac OSX dengan menggunakan Java Eclipse IDE.
2. Mode Pseudo-Distributed adalah menginstal dan menjalankan Hadoop pada platform Linux namun di-setting seolah-olah dijalankan dalam beberapa komputer.
3. Mode Fully Distributed adalah menginstal dan menjalankan Hadoop pada sejumlah komputer server dengan OS Linux (Linux Cluster).

Berikut akan dijelaskan langkah-langkah menginstal dan menjalankan Hadoop dalam tiap mode tersebut diatas serta menjalankan aplikasi Wordcount sebagai Hello World!-nya MapReduce.

Hadoop didistribusikan dalam dua bentuk, yaitu dalam bentuk source dan dalam bentuk binary. Distribusi yang berupa source setelah diunduh harus dikompilasi terlebih dahulu. Distribusi binary setelah diunduh dapat langsung digunakan, dengan beberapa tambahan konfigurasi

Prasyarat

Pada tutorial ini berikut prasyarat untuk melakukan installasi Hadoop mode standalone :

1. Operating System : Linux Debian 9 (**stretch**)
2. Java JDK

Installasi Java JDK

1. Login kedalam kedalam *shell* linux
2. Update Repository debian, dengan cara sebagai berikut
 - a. Buka File /etc/apt/sources.list

```
#vi /etc/apt/sources.list
```

- b. Update isi file dengan repository di bawah

```
deb http://kartolo.sby.datautama.net.id/debian/ stretch main  
deb-src http://kartolo.sby.datautama.net.id/debian/ stretch main
```

- c. Simpan file
- d. Jalankan apt-get update

```
#apt-get update
```

3. Install OpenJDK

```
# apt install default-jdk
```

4. Check Java Version

```
# javac -version
```

5. Setting Java Environment
Edit file /etc/environment
1. Tambahkan parameter berikut pada baris terakhir

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-i386/jre/"
```

2. Reload Environment

```
#source /etc/environment
```

3. Verifikasi Java Home

```
#echo $JAVA_HOME
```

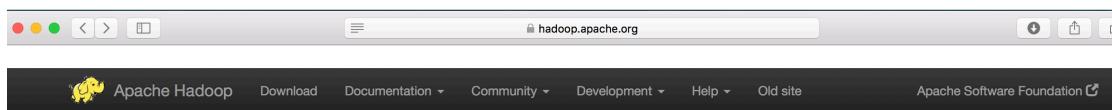
Hadoop Mode Standalone

Hadoop Mode Standalone dapat dijalankan pada PC Windows, Linux maupun Mac OSX yang tentunya sudah terinstal Java Virtual Machine. Penggunaan Hadoop mode ini ditujukan untuk pengembangan aplikasi MapReduce (MapReduce Application Development) dan testing program dengan data berukuran kecil.

Installasi Hadoop

Pada panduan ini menggunakan Hadoop versi 3.1.1 dengan binary version installation.

1. File bisa didownload pada <https://hadoop.apache.org/releases.html>



The screenshot shows the Apache Hadoop website's download section. At the top, there's a navigation bar with links for 'Download', 'Documentation', 'Community', 'Development', 'Help', and 'Old site'. Below the navigation bar, there's a sub-navigation menu for 'Apache Software Foundation' with links for 'ASF Home', 'ASF Governance', 'ASF Events', 'ASF Archives', 'ASF Artifacts', and 'ASF License Agreements'. The main content area is titled 'Download' and contains a table of available Hadoop releases. The table has columns for 'Version', 'Release date', 'Source download', 'Binary download', and 'Release notes'. The table lists versions 2.8.5, 3.1.1, 2.7.7, 3.0.3, and 2.9.1, each with their respective release dates and download links.

Version	Release date	Source download	Binary download	Release notes
2.8.5	2018 Sep 15	source (checksum signature)	binary (checksum signature)	Announcement
3.1.1	2018 Aug 8	source (checksum signature)	binary (checksum signature)	Announcement
2.7.7	2018 May 31	source (checksum signature)	binary (checksum signature)	Announcement
3.0.3	2018 May 31	source (checksum signature)	binary (checksum signature)	Announcement
2.9.1	2018 May 3	source (checksum signature)	binary (checksum signature)	Announcement

To verify Hadoop releases using GPG:

1. Download the release hadoop-X.Y.Z-src.tar.gz from a [mirror site](#).
2. Download the signature file hadoop-X.Y.Z-src.tar.gz.asc from [Apache](#).
3. Download the [Hadoop KEYS](#) file.
4. gpg --import KEYS
5. gpg --verify hadoop-X.Y.Z-src.tar.gz.asc

2. Membuat working directory Hadoop

```
#mkdir -p /opt/hadoop/src  
#cd /opt/hadoop/src
```

3. Download file menggunakan perintah *wget* pada linux

```
#wget http://kambing.ui.ac.id/apache/hadoop/common/hadoop-  
3.1.1/hadoop-3.1.1.tar.gz
```

4. Extract hadoop file

```
#tar -zxf hadoop-3.1.1.tar.gz
```

5. Pindahkan directory hadoop kedalam directory /usr/local

```
#mv hadoop-3.1.1 /usr/local/hadoop
```

6. Menjalankan Hadoop sebagai user bisa (non-root)

```
$/usr/local/hadoop/bin/hadoop
```

```
[deje@debian:/opt/hadoop/src$ /usr/local/hadoop/bin/hadoop
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
or   hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
      where CLASSNAME is a user-provided Java class

OPTIONS is none or any of:

--config dir          Hadoop config directory
--debug               turn on shell script debug mode
--help                usage information
buildpaths           attempt to add class files from build tree
hostnames list[,of,host,names] hosts to use in slave mode
hosts filename        list of hosts to use in slave mode
loglevel level        set the log4j level for this command
workers              turn on worker mode

SUBCOMMAND is one of:

      Admin Commands:

daemonlog    get/set the log level for each daemon

      Client Commands:

archive       create a Hadoop archive
checknative   check native Hadoop and compression libraries availability
classpath     prints the class path needed to get the Hadoop jar and the required libraries
```

MapReduce

Untuk memastikan bahwa hadoop berfungsi dengan baik dengan menjalankan contoh program MapReduce yang disertakan pada saat installasi. Untuk melakukannya, buat direktori yang *input* di direktori home Anda dan salin file konfigurasi Hadoop ke dalamnya untuk menggunakan file-file tersebut sebagai data kami.

1. Buat directory dengan nama *input* pada *home directory*

```
$mkdir ~/input
```

2. Copy seluruh file *.xml yang terletak pada directory hadoop kedalam directory *input*

```
$cp /usr/local/hadoop/etc/hadoop/*.xml ~/input
```

Selanjutnya, kita akan menjalankan program MapReduce `hadoop-mapreduce-example`, arsip Java dengan beberapa opsi. Pada panduan ini akan menjalankan program `grep`, salah satu dari banyak contoh yang termasuk dalam `hadoop-mapreduce-examples`, diikuti oleh direktori `input`, masukan dan direktori output `grep_example`. Program `grep` MapReduce akan menghitung kecocokan kata harfiah atau ekspresi reguler. Akhirnya, disini akan menggunakan ekspresi reguler yang diizinkan `[.]*` Untuk menemukan kemunculan kata yang diizinkan di dalam atau di akhir kalimat deklaratif. Ekspresi bersifat case-sensitive, jadi pada panduan ini tidak akan menemukan kata itu jika huruf kapital dikapitalisasi pada awal kalimat.

3. Menjalankan Program MapReduce dengan opsi `grep` untuk mencari berapa banyak kata `allowed`

```
$/usr/local/hadoop/bin/hadoop jar  
/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-  
examples-3.1.1.jar grep ~/input ~/grep_example 'allowed[.]*' 
```

Jika tidak ada error maka akan terbentuk direktori `grep_example` yang berisi hasil dari program mapreduce

```
Combine input records=0  
Combine output records=0  
Reduce input groups=2  
Reduce shuffle bytes=43  
Reduce input records=2  
Reduce output records=2  
Spilled Records=4  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=0  
Total committed heap usage (bytes)=673710080  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=147  
File Output Format Counters  
Bytes Written=34
```

deje@debian:~\$

4. Melihat hasil program

```
$ cat grep_example/*
```

```
[deje@debian:~$ cat grep_example/*
19      allowed.
1      allowed
deje@debian:~$
```

5. Jika sudah muncul, maka hadoop dengan mode stand alone sudah bisa digunakan.

Hadoop Mode Pseudo-Distributed

Berikutnya, akan dijelaskan langkah-langkah untuk menginstal Hadoop Mode Pseudo-Distributed pada platform Linux dan menjalankan contoh aplikasi Wordcount. Mode pseudo-distributed ini ditujukan untuk testing program MapReduce yang telah dibuat, dikembangkan, dan ditest dengan menggunakan Hadoop.

1. Instalasi dan Konfigurasi SSH

SSH digunakan untuk mengakses node-node Hadoop pada mode multinode maupun single node pseudo distributed, atau sebagai remote access terhadap Hadoop. Apabila SSH belum terpasang, pada terminal dimasukkan perintah berikut untuk memasang SSH.

```
#apt-get install openssh-server sudo maven build-essential
```

Setelah SSH dipasang, kemudian perlu dilakukan generate sebuah kunci rsa khusus untuk pengguna *hduser*. Tujuannya adalah agar akun *hduser* mempunyai otoritas untuk melakukan remote access terhadap localhost (mode single node pseudo-distributed) maupun pada node-node Hadoop (mode multinode). Masukkan terlebih dahulu masuk ke akun *hduser* dan buat kunci ssh dengan menggunakan perintah berikut.

```
#adduser hduser
```

Masuk sebagai user *hduser*, selanjutnya generate ssh key

```
#su - hduser
$ssh-keygen
```

```
[hduser@debian:~$ ssh-keygen
Generating public/private rsa key pair.
[Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:8/hUkp/ly7Rm26tHp5JHRGTo7fJxZ+05o6Lp/AxZy/I hduser@debian
The key's randomart image is:
+---[RSA 2048]---+
|      oo   |
|      ...   |
|      . o    |
|      .. o   |
| S o..o. .  |
| +++o++.*   |
| .+ooo=++*  |
| .o*.oo*0.  |
| .=+E.B0o=  |
+---[SHA256]---+
hduser@debian:~$
```

Pada prompt yang muncul, tekan enter agar nama file kunci tetap id_rsa. Kemudian, atur passphrase kosong, agar tidak perlu memasukkan password setiap melakukan SSH. Tekan enter untuk membiarkan passphrase tetap kosong. Kemudian masukkan kunci publik untuk SSH ke daftar kunci yang terautorisasi menggunakan perintah berikut.

```
$cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

Perintah berikut digunakan untuk memberikan hak akses ke direktori .ssh dan file kunci yang terautorisasi.

```
$chmod 700 ~/.ssh/
$chmod 600 ~/.ssh/authorized_keys
[hduser@debian:~$ chmod 700 ~/.ssh/
[hduser@debian:~$ chmod 600 ~/.ssh/authorized_keys
hduser@debian:~$
```

Kemudian, akun pengguna hduser tersebut dimasukkan ke dalam grup sudoers. Caranya adalah dengan mengedit file sudoers dengan menggunakan text editor vi. Pada terminal, dimasukkan perintah visudo

```
#visudo
```

Kemudian akan muncul text editor vi yang membuka file sudoers. Tambahkan sebaris konfigurasi berikut

```
hduser ALL=(ALL)    ALL
```

2. Memperbarui Environment Variable (.bashrc)

Pada file .bashrc, ditambahkan beberapa baris untuk menambahkan path ke direktori Hadoop dan Java. Penambahan path ini digunakan untuk memudahkan ketika kita melakukan perintah terkait Hadoop dan Java di terminal. Buka file .bashrc menggunakan perintah berikut sebagai user *hduser*.

```
$nano ~/.bashrc
```

Pada file .bashrc yang dibuka dengan, tambahkan di environment variable untuk path ke direktori hadoop dan direktori Java. Berikut adalah baris yang ditambahkan dalam .bashrc.

```
HADOOP_HOME=/home/hduser/hadoop
```

Selanjutnya reload environment dengan menggunakan perintah sebagai berikut

```
$source ~/.bashrc
```

3. Mematikan IPv6

Konfigurasi Hadoop yang berhubungan dengan jaringan akan mengikat ke alamat IPv6 pada 0.0.0.0. Hal tersebut akan menimbulkan permasalahan tersendiri. Oleh karena itu, IPv6 sebaiknya dinonaktifkan.

Buka dan edit file sysctl.conf yang ada di direktori /etc dengan menggunakan text editor.

```
#nano /etc/sysctl.conf
```

Setelah file sysctl.conf dibuka menggunakan text editor, tambahkan beberapa baris konfigurasi berikut. Kemudian simpan dan tutup text editor.

```
net.ipv6.conf.all.disable_ipv6 = 1  
net.ipv6.conf.default.disable_ipv6 = 1  
net.ipv6.conf.lo.disable_ipv6 = 1
```

Kemudian, perbarui pengaturan dengan perintah berikut.

```
# sysctl -p
```

4. Instalasi Hadoop (sebagai user *hduser*)

4. Mengunduh dan Membuat Direktori Hadoop

Distribusi binary setelah diunduh dapat langsung digunakan, dengan beberapa tambahan konfigurasi

Untuk mengunduh Hadoop binary distribution, sebagai hduser, pada terminal masukkan perintah berikut.

```
wget  
http://kambing.ui.ac.id/apache/hadoop/common/hadoop-  
3.1.1/hadoop-3.1.1.tar.gz
```

Hadoop yang telah diunduh dalam bentuk tarball diekstrak ke direktori /home/hduser/hadoop. Berikut adalah perintahnya.

```
$mkdir hadoop  
$tar -zxvf hadoop-3.1.1.tar.gz -C hadoop
```

5. Konfigurasi Hadoop

Sebelum Hadoop dapat dijalankan, terlebih dahulu harus dilakukan beberapa konfigurasi untuk menyesuaikan HDFS dan MapReduce framework yang digunakan. Semua konfigurasi Hadoop terletak di direktori /home/hduser/hadoop/etc/hadoop.

5. Buka file *hadoop-env.sh* dengan text editor.

```
$cd $HADOOP_HOME/etc/hadoop  
$sudo nano hadoop-env.sh
```

Pada file *hadoop-env* tersebut, set variabel *JAVA_HOME* menjadi lokasi instalasi Java sebagai berikut.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386/jre/
```

6. Buka core-site.xml dengan text editor

```
$sudo nano core-site.xml
```

isikan pada tag <configuration>...</configuration> dengan konfigurasi berikut

```
<property>  
    <name>fs.defaultFS</name>  
    <value>hdfs://localhost:9000</value>  
</property>
```

7. Selanjutnya, buka hdfs-site.xml dengan text editor

```
$sudo nano hdfs-site.xml
```

isikan tag <configuration></configuration> dengan konfigurasi berikut. Konfigurasi ini digunakan untuk mendefinisikan banyaknya replikasi data pada HDFS.

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
```

8. Selanjutnya adalah konfigurasi MapReduce framework yang digunakan pada file mapred-site.xml

```
$sudo nano mapred-site.xml
```

isikan tag <configuration></configuration>

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
```

9. Konfigurasi selanjutnya adalah konfigurasi YARN. Langkahnya adalah dengan membuka yarn-site.xml

```
$sudo nano yarn-site.xml
```

isikan konfigurasi dalam tag <configuration>...</configuration> seperti berikut. Hasil konfigurasi ini diperlihatkan pada gambar dibawah ini.

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
```

6. Format Hadoop Distributed File System via NameNode

Setelah Hadoop selesai dikonfigurasi, kemudian dilakukan format pada Hadoop Distributed File System (HDFS). Hadoop Distributed File System (HDFS) tersebut perlu di-format sebelum Hadoop dijalankan untuk pertama kalinya.

Format dilakukan melalui NameNode. Untuk mengakses NameNode, harus dilakukan remote access ke localhost dengan SSH menggunakan perintah berikut ini. Perintah

pada baris ketiga merupakan perintah untuk format NameNode yang dilakukan dengan mengeksekusi file hdfs pada direktori bin.

```
$ssh hduser@localhost  
$cd $HADOOP_HOME  
$bin/hdfs namenode -format
```

7. Menjalankan dan Menghentikan Hadoop

Mode menjalankan Hadoop dengan menggunakan HDFS dan faktor replikasi satu biasa disebut sebagai mode pseudo-distributed. Perintah yang digunakan untuk menjalankan Hadoop adalah start-dfs.sh dan start-yarn.sh. Perintah start-dfs.sh akan mengaktifkan NameNode, DataNode, dan Secondary NameNode, sedangkan start-yarn.sh akan mengaktifkan ResourceManager dan NodeManager. Kita bisa menggunakan perintah jps untuk mendapatkan informasi mengenai proses- proses Hadoop yang berjalan

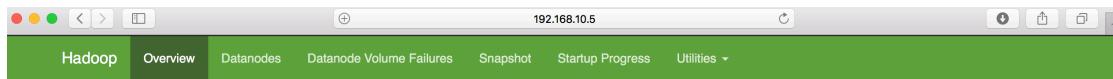
```
$sbin/start-dfs.sh  
$sbin/start-yarn.sh
```

8. Cek Status Hadoop Process

Digunakan untuk mengecek process dfs adan yarn yang berjalan

```
$jps
```

Antarmuka web Hadoop dapat digunakan untuk memonitor jalannya node-node dan job-job Hadoop yang sedang berjalan. Web Hadoop dapat diakses di alamat <http://localhost:9870> untuk NameNode dan <http://localhost:8088> untuk Resource Manager. Antarmuka web untuk keduanya diperlihatkan pada gambar dibawah ini.



Overview 'localhost:9000' (active)

Started:	Thu Oct 11 22:43:37 +0700 2018
Version:	3.1.1, r2b9a8c1d3a2caf1e733d57f346af3ff0d5ba529c
Compiled:	Thu Aug 02 11:26:00 +0700 2018 by leftnoteeasy from branch-3.1.1
Cluster ID:	CID-331d6480-d948-4b4a-8df6-d5318b152be0
Block Pool ID:	BP-2032600311-127.0.1.1-1539272005611

Summary

Security is off.
 Safemode is off.
 1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
 Heap Memory used 42.32 MB of 177.25 MB Heap Memory. Max Heap Memory is 442.75 MB.
 Non Heap Memory used 28.8 MB of 29.56 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue
No data available in table															

Showing 0 to 0 of 0 entries

Setelah kita menjalankan service HDFS, kita bisa membuat direktori pada HDFS. Direktori pada HDFS ini digunakan untuk menyimpan data yang akan diolah menggunakan Hadoop. Pembuatan direktori pada HDFS menggunakan perintah sebagai berikut

```
 ${HADOOP_HOME}/bin/hdfs dfs -mkdir /user
 ${HADOOP_HOME}/bin/hdfs dfs -mkdir /user/hduser
```

Direktori-direktori pada HDFS dapat dicek melalui antarmuka web Hadoop untuk NameNode Manager, yaitu pada tab Utilities, lalu pilih Browse File System, seperti diperlihatkan pada gambar dibawah ini

The screenshot shows the Hadoop Web UI's 'Browse Directory' page. At the top, there's a green header bar with links for Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. Below the header is a search bar with placeholder 'Search:' and a file browser interface with icons for folder, file, and refresh. The main content area has a table header with columns: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. A single entry is listed: 'drwxr-xr-x' for permission, 'hduser' for owner, 'supergroup' for group, '0 B' for size, 'Oct 11 22:56' for last modified, '0' for replication, '0 B' for block size, and 'user' for name. Below the table, it says 'Showing 1 to 1 of 1 entries'. At the bottom right, there are buttons for Previous, Next, and a page number '1'.

Install HBase

Terdapat tiga mode installasi pada HBase yaitu *Standalone mode*, *Pseudo Distributed mode* dan *Fully Distributed mode*. Pada panduan disini akan di install menggunakan mode *Presudo Distributed* yang akan mensimulasikan distribusi data menggunakan single computer.

1. Mengunduh Apache Hbase

- Distribusi binary setelah diunduh dapat langsung digunakan, dengan beberapa tambahan konfigurasi , perintah dijalankan sebagai user *hduser*

```
$wget  
http://kambing.ui.ac.id/apache/hbase/2.1.0/hbase-  
2.1.0-bin.tar.gz
```

- HBase yang telah diunduh dalam bentuk tarball diekstrak dan dimasukan kedalam direktori /home/hduser/hbase. Berikut adalah perintahnya.

```
$tar -zxvf hbase-2.1.0-bin.tar.gz  
$mv hbase-2.1.0 /home/hduser/hbase
```

- Setting JAVA_HOME

```
$nano /home/hduser/hbase/conf/hbase-env.sh
```

Pada parameter JAVA_HOME , masukan path dari sebagai berikut

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-i386/jre/
```

Lalu Save

- Setting PATH environment

```
$nano ~/.profile
```

tambahkan parameter sebagai berikut, lalu save

```
export HBASE_HOME=/home/hduser/hbase  
export PATH=$PATH:$HBASE_HOME/bin
```

- Load environment baru

```
$source ~/.profile
```

2. Konfigurasi Apache Hbase

File konfigurasi apache hbase bernama hbase-site.xml

- Buka file hbase-site.xml , dan tambahkan parameter sebagai berikut

```
<configuration>  
  <property>  
    <name>hbase.rootdir</name>  
    <value>hdfs://localhost:9000/hbase</value>  
  </property>  
  
  <property>  
    <name>hbase.zookeeper.property.dataDir</name>  
    <value>/home/hduser/zookeeper</value>  
  </property>  
  
  <property>  
    <name>hbase.cluster.distributed</name>  
    <value>true</value>  
  </property>  
</configuration>
```

3. Menjalankan Layanan Apache Hbase

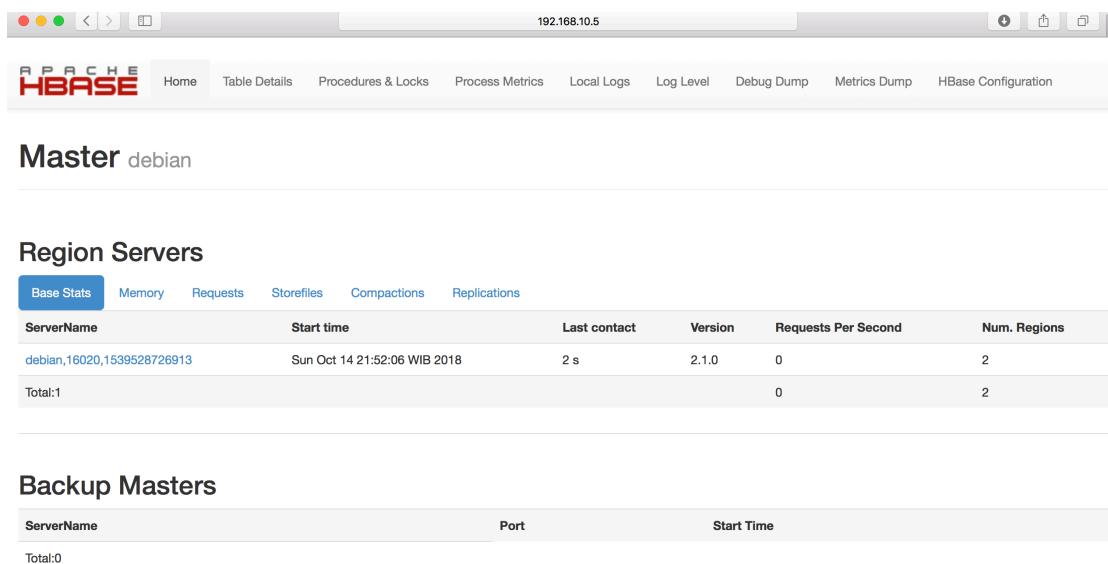
```
$/home/hduser/hbase/bin/start-hbase.sh
```

4. Masuk kedalam HBase shell

```
$hbase shell
```

5. HBase Master Status

Untuk mengecek status layanan HBase Master bisa melalui browser dengan alamat
<http://localhost:16010/master-status>



The screenshot shows a web browser window displaying the Apache HBase master status. The title bar says "192.168.10.5". The main content area has a header "Master debian". Below it, there's a section titled "Region Servers" with a table showing one server entry:

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
debian,16020,1539528726913	Sun Oct 14 21:52:06 WIB 2018	2 s	2.1.0	0	2
Total:1				0	2

Below this, there's a section titled "Backup Masters" with a table showing no entries:

ServerName	Port	Start Time
Total:0		

Tutorial HBase

HBase Shell

HBase berisi shell yang digunakan untuk berkomunikasi dengan HBase. HBase menggunakan Sistem File Hadoop untuk menyimpan datanya. HBase mempunyai *master server* dan *region server*. Penyimpanan data akan dalam bentuk region (tabel). *region* ini akan dibagi dan disimpan di *region server*.

Master Server mengelola *region server* ini dan semua tugas ini dilakukan pada HDFS. Pada panduan di bawah ini adalah beberapa perintah yang didukung oleh HBase Shell.

Perintah Umum (*General Commands*)

Perintah	Fungsi
<i>status</i>	Memberikan status HBase, misalnya, jumlah server
<i>version</i>	Menyediakan informasi versi HBase yang digunakan
<i>table help</i>	Memberikan bantuan untuk perintah referensi-tabel.
<i>whoami</i>	Memberikan informasi tentang penggunaannya

Perintah Data Definition Language (DDL)

Berikut adalah perintah yang beroperasi pada tabel di HBase.

Perintah	Fungsi
<i>create</i>	Membuat table
<i>list</i>	Menampilkan list daftar <i>table</i>
<i>disable</i>	Disable sebuah table
<i>is_disabled</i>	Melakukan verifikasi apakah sebuah table terdisabled
<i>enable</i>	Enables sebuah table.
<i>is_enabled</i>	Melakukan verifikasi apakah sebuah table enable
<i>describe</i>	Menyediakan deskripsi dari table
<i>alter</i>	Alters sebuah table.
<i>exists</i>	Melakukan verifikasi apakah sebuah table ada

drop	Menghapus table dari HBase
drop_all	Menghapus semua table dari HBase
Java Admin API	Semua perintah di atas, Java menyediakan API Admin untuk mencapai fungsi DDL melalui pemrograman. Di bawah paket org.apache.hadoop.hbase.client, HBaseAdmin dan HTableDescriptor adalah dua kelas penting dalam paket ini yang menyediakan fungsionalitas DDL.

Perintah Data Manipulation Language (DML)

Berikut adalah perintah yang beroperasi untuk mengelola data di HBase.

Perintah	Fungsi
put	Menempatkan nilai sel pada kolom yang ditentukan dalam baris tertentu dalam tabel tertentu.
get	Mengambil isi baris atau sel.
delete	Menghapus nilai sel dalam tabel.
deleteall	Menghapus semua sel di baris tertentu.
scan	Memindai dan mengembalikan data tabel.
count	Menghitung dan mengembalikan jumlah baris dalam sebuah tabel.
truncate	Menonaktifkan, menghapus, dan membuat ulang tabel yang ditentukan.
Java Admin API	Semua perintah di atas DML, Java menyediakan API klien untuk mencapai fungsi DML, operasi CRUD (Create Retrieve Update Delete) dan lainnya melalui pemrograman, di bawah paket org.apache.hadoop.hbase.client. HTable Put and Get adalah kelas-kelas penting dalam paket ini.

HBase General Commands

Sebelum menjalankan HBase shell terlebih dahulu masuk kedalam hbase shell dengan

perintah sebagai berikut :

```
$hbase shell
```

```
[hduser@debian:~$ hbase shell
2018-10-17 05:40:14,250 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop
classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.1.0, re1673bb0bbfea21d6e5dba73e013b09b8b49b89b, Tue Jul 10 17:26:48 CST 2018
Took 0.0018 seconds
hbase(main):001:0> ]
```

- **status**

Perintah ini menghasilkan status sistem termasuk rincian server yang berjalan pada sistem. Sintaksnya adalah sebagai berikut:

```
hbase(main):002:0>status
```

```
[hbase(main):001:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
Took 0.8049 seconds
hbase(main):002:0> ]
```

- **version**

Perintah ini menampilkan versi HBase yang digunakan di sistem Anda. Sintaksnya adalah sebagai berikut:

```
hbase(main):002:0>version
```

```
[hbase(main):002:0> version
2.1.0, re1673bb0bbfea21d6e5dba73e013b09b8b49b89b, Tue Jul 10 17:26:48 CST 2018
Took 0.0002 seconds
hbase(main):003:0> ]
```

- **table_help**

Perintah ini dapat membantu Anda apa dan bagaimana menggunakan perintah yang direferensikan oleh tabel. Diberikan di bawah ini sintaks untuk menggunakan perintah ini.

```
hbase(main):002:0>table_help
```

```
|hbase(main):003:0> table_help  
Help for table-reference commands.  
  
You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.  
See the standard help information for how to use each of these commands.  
  
However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.  
For instance, you can get create a table and keep around a reference to it via:  
  
    hbase> t = create 't', 'cf'  
  
Or, if you have already created the table, you can get a reference to it:  
  
    hbase> t = get_table 't'
```

- **whoami**

Perintah ini menampilkan rincian pengguna HBase. Jika Anda menjalankan perintah ini, mengembalikan pengguna HBase saat ini seperti yang ditunjukkan di bawah ini..

```
hbase(main):004:0>whoami
```

```
hbase(main):004:0> whoami  
hduser (auth:SIMPLE)  
groups: hduser  
Took 0.1059 seconds  
hbase(main):005:0>
```

Membuat Table

Anda dapat membuat tabel menggunakan perintah *create*, di sini Anda harus menentukan nama tabel dan nama *column family*. Sintaks untuk membuat tabel di shell HBase ditunjukkan di bawah ini.

```
create '<table name>', '<column family>'
```

Contoh di berikut adalah sebuah skema sederhana dari table *karyawan*, dan mempunyai dua column family bernama “*personal data*” dan “*profesional data*”

Row key	Personal data	Professional data

- Membuat table dengan perintah sebagai berikut

```
create 'karyawan', 'personal data', 'professional data'
```

```
[hbase(main):005:0> create 'karyawan', 'personal data', 'professional data'
Created table karyawan
Took 2.9399 seconds
=> Hbase::Table - karyawan
hbase(main):006:0>
```

- Verifikasi hasil menggunakan perintah *list*

```
list
```

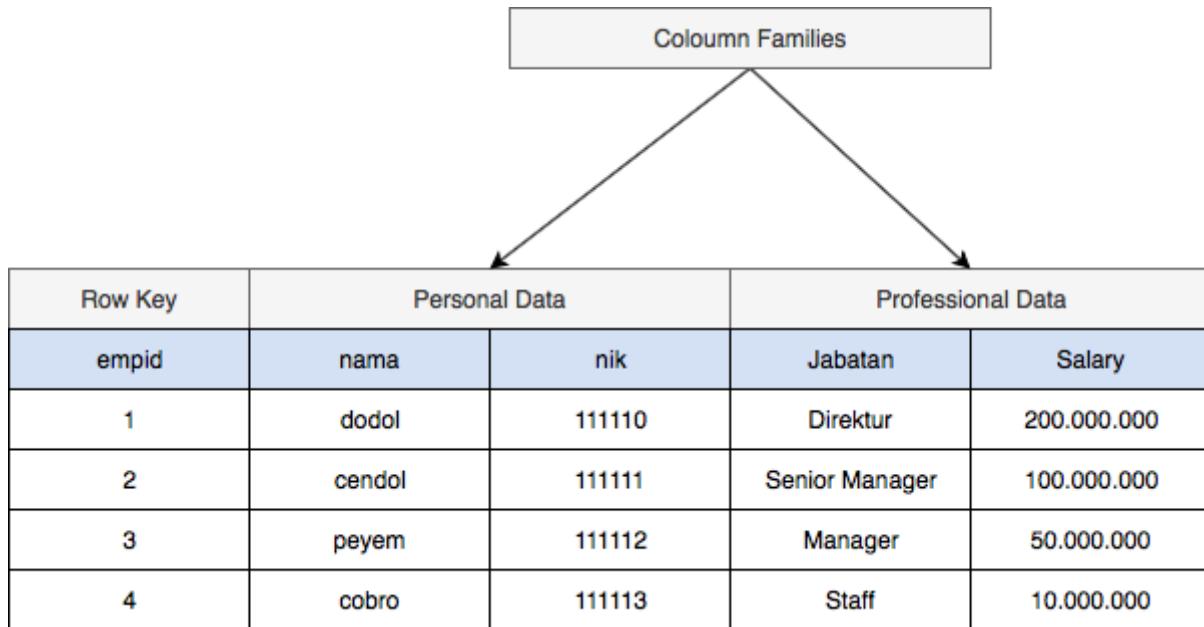
```
[hbase(main):006:0> list
TABLE
karyawan
1 row(s)
Took 0.0673 seconds
=> ["karyawan"]
hbase(main):007:0>
```

Memasukan Data (Create)

Pada bagian ini akan di jelaskan cara membuat data dalam tabel HBase. Untuk membuat data dalam tabel HBase, perintah dan metode berikut digunakan:

- **put** command
- **add()** menggunakan metode dari *Put* class
- **put()** menggunakan metode dari *HTable* class

Sebagai sebuah contoh berikut struktur data yang akan dibuat dalam table karyawan didalam HBase



- Menggunakan perintah put untuk melakukan insert data kedalam row

```
put '<table name>', 'row1', '<colfamily:colname>', '<value>'
```

- Berikut adalah perintah untuk melakukan insert data pada baris pertama

```
put 'karyawan','1','personal data:nama','dodol'
put 'karyawan','1','personal data:nik','111110'
put 'karyawan','1','professional data:jabatan','Direktur'
put 'karyawan','1','professional data:salary','200000000'
```

- Selanjutnya melihat data pada table karyawan menggunakan perintah scan

```
scan 'karyawan'
```

```
[hbase(main):020:0> scan 'karyawan'
ROW                                         COLUMN+CELL
1                                           column=personal data:nama, timestamp=1539732153247, value=dodol
1                                           column=personal data:nik, timestamp=1539732187475, value=111110
1                                           column=professional data:jabatan, timestamp=1539732374408, value=Direktur
1                                           column=professional data:salary, timestamp=1539732407324, value=200000000
2                                           column=personal data:nama, timestamp=1539732641000, value=cendol
2                                           column=personal data:nik, timestamp=1539732628961, value=111111
2                                           column=professional data:jabatan, timestamp=1539732672814, value=Senior Manager
2                                           column=professional data:salary, timestamp=1539732684362, value=100000000
2 row(s)
Took 0.0084 seconds
hbase(main):021:0> ]
```

Menampilkan Data (Read)

Untuk membaca data pada table di HBase menggunakan perintah *get* , dengan syntak sebagai berikut :

```
get '<table name>', 'row1'
```

- Menampilkan data pada baris pertama

```
get 'karyawan', '1'
```

```
[hbase(main):021:0> get 'karyawan', '1'
COLUMN                                CELL
personal data:nama                      timestamp=1539732153247, value=dodol
personal data:nik                        timestamp=1539732187475, value=111110
professional data:jabatan               timestamp=1539732374408, value=Direktur
professional data:salary                 timestamp=1539732407324, value=200000000
1 row(s)
Took 0.0223 seconds
hbase(main):022:0>
```

- Membaca spesifik column dengan syntak sebagai berikut

```
get 'table name', 'rowid', {COLUMN => 'column family:column name'}
```

```
[hbase(main):025:0> get 'karyawan', '1', {COLUMN => 'personal data:nama'}
COLUMN                                CELL
personal data:nama                      timestamp=1539732153247, value=dodol
1 row(s)
Took 0.0052 seconds
hbase(main):026:0>
```

Update Data (Update)

Untuk update data pada table di HBase menggunakan perintah *put* , dengan syntak sebagai berikut :

```
put 'table name','row ','Column family:column name','new value'
```

- Pada panduan ini saya akan mengganti data salary pada row1 menjadi 250000000
Data sebelum di ganti

```
[hbase(main):021:0> get 'karyawan','1'
COLUMN                                CELL
personal data:nama                      timestamp=1539732153247, value=dodol
personal data:nik                       timestamp=1539732187475, value=111110
professional data:jabatan               timestamp=1539732374408, value=Direktur
professional data:salary                 timestamp=1539732407324, value=200000000
1 row(s)
Took 0.0223 seconds
hbase(main):022:0>
```

```
put 'karyawan','1','professional data:salary','250000000'
```

```
[hbase(main):027:0> put 'karyawan','1','professional data:salary','250000000'
Took 0.0076 seconds
[hbase(main):028:0> get 'karyawan', '1'
COLUMN                                CELL
personal data:nama                      timestamp=1539732153247, value=dodol
personal data:nik                       timestamp=1539732187475, value=111110
professional data:jabatan               timestamp=1539732374408, value=Direktur
professional data:salary                 timestamp=1539733718187, value=250000000
1 row(s)
Took 0.0065 seconds
hbase(main):029:0>
```

Menghapus Data (Delete)

Untuk update data pada table di HBase menggunakan perintah *delete* , dengan syntax sebagai berikut :

```
delete '<table name>', '<row>', '<column name >', '<time stamp>'
```

- Sebagai contoh disini akan menghapus data pada spesifik cell di column salary baris kedua
Sebelum cell di hapus

```
[hbase(main):029:0> get 'karyawan', '2'
COLUMN                                CELL
personal data:nama                      timestamp=1539732641000, value=cendol
personal data:nik                       timestamp=1539732628961, value=111111
professional data:jabatan              timestamp=1539732672814, value=Senior Manager
professional data:salary                timestamp=1539732684362, value=100000000
1 row(s)
Took 0.0094 seconds
hbase(main):030:0>
```

```
delete 'karyawan','2','professional data:salary' >
```

```
[hbase(main):030:0> delete 'karyawan', '2', 'professional data:salary'
Took 0.0951 seconds
[hbase(main):031:0> get 'karyawan', '2'
COLUMN                                     CELL
personal data:nama                         timestamp=1539732641000, value=cendol
personal data:nik                          timestamp=1539732628961, value=111111
professional data:jabatan                 timestamp=1539732672814, value=Senior Manager
1 row(s)
Took 0.0096 seconds
hbase(main):032:0>
```

Import Data TSV (Tab Separated Values)

Sebelum melakukan import data, pertama kita harus membuat sebuah table untuk menampung hasil data yang akan di import, column name disesuaikan dengan jumlah field yang berada pada file yang akan di import

- Membuat table

```
create 'sales_transaction',{NAME => 'transaction_data'}
```

- Import Data kedalam table sales_transaction

```
$hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator='-' -Dimporttsv.columns='HBASE_ROW_KEY,transaction_data:Transaction_date,transaction_data:Product,transaction_data:Price,transaction_data:Payment_Type,transaction_data:Name,transaction_data:City,transaction_data:State,transaction_data:Country,transaction_data:Account_Created,transaction_data>Last_Login,transaction_data:Latitude,transaction_data:Longitude' sales_country /home/hduser/latihan/MapReduce/inputMapReduce/SalesJan2009.csv
```

- Check hasil import data , masuk kedalam hbase shell

```
scan 'sales_country'
```

```
GC time elapsed (ms)=12
Total committed heap usage (bytes)=32440320
ImportTsv
    Bad Lines=0
    File Input Format Counters
        Bytes Read=123637
    File Output Format Counters
        Bytes Written=0
[hduser@debian:~]$ hbase shell
2018-10-17 07:35:33,323 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-
applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.1.0, re1673bb0bbfea21d6e5dba73e013b09b8b49b89b, Tue Jul 10 17:26:48 CST 2018
Took 0.0019 seconds
[hbase(main):001:0> scan 'sales_country'
ROW                                     COLUMN+CELL
1/1/09 10:06                           column=transaction_data:Account_Created, timestamp=1539736517947, value=1/25/09 11:37
1/1/09 10:06                           column=transaction_data:City, timestamp=1539736517947, value=Bayern
1/1/09 10:06                           column=transaction_data:Country, timestamp=1539736517947, value=1/1/09 9:13
1/1/09 10:06                           column=transaction_data:Last_Login, timestamp=1539736517947, value=48.15
```

Hadoop MapReduce

Beberapa poin dari defisi dari hadoop MapReduce :

- Hadoop MapReduce adalah algoritma atau proses komputasi pada sistem terdistribusi
- Ditulis di atas bahasa Java
- Mempunyai 2 proses utama, yaitu map dan reduce
- Mampu untuk memproses secara pararel data yang sangat besar, karena komputasi bisa berjalan di puluhan, ratusan, atau bahkan ribuan node

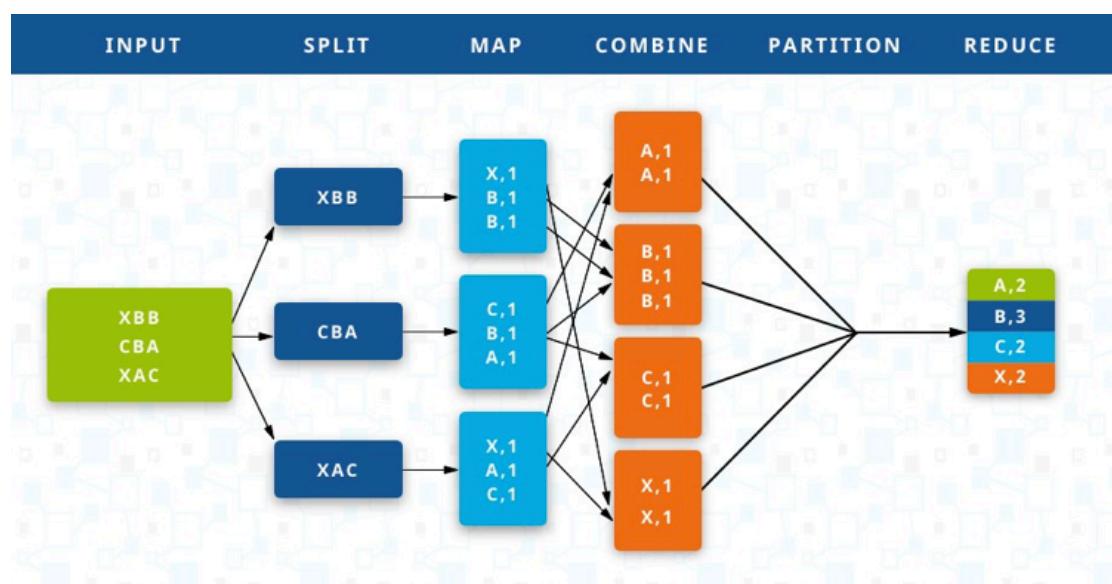
Map mengambil satu set data dan mengubahnya menjadi satu set data, di mana unsur-unsur individu dipecah menjadi tupel. Kemudian, Reduce mengambil output dari hasil proses map sebagai masukan dan menggabungkan/mengelompokkan tupel data ke satu set data yang lebih kecil dari tupel. Sebagai urutan pada penamaan MapReduce menyiratkan, proses reduce selalu dilakukan setelah proses map. Tupel di sini berarti pasangan antara key dan value-nya, dapat digambarkan dengan simbol "(k1, v1)"

Tahapan proses MapReduce

MapReduce terdiri atas tiga tahap, yaitu tahap map, tahap shuffle, dan terakhir tahap reduce. Untuk tahapan shuffle dan reduce digabungkan kedalam satu tahap besaran-nya yaitu tahap reduce.

1. Tahap map, memproses data inputan yang umumnya berupa file yang tersimpan dalam HDFS (dapat dibaca di Sistem file terdistribusi), inputan tersebut kemudian diubah menjadi tuple yaitu pasangan antara key dan value-nya.
2. Tahap reduce, memproses data inputan dari hasil proses map, yang kemudian dilakukan tahap shuffle dan reduce yang hasil data set baru-nya disimpan di HDFS kembali.

Berikut ini ilustrasi untuk mendapatkan gambaran tentang proses map dan reduce.



Membuat Program Pertama Hadoop MapReduce

Setelah selesai melakukan installasi Hadoop , kali ini kita akan mulai membuat sebuah program kecil dengan menggunakan JAVA untuk memproses sebuah data dengan memanggil fungsi MapReduce Hadoop.

Prasyarat

Berikut adalah prasyarat untuk dapat menjalankan program yang akan dibuat menggunakan environment sebagai berikut :

- Operating system : Debian 9.0 dengan OpenJDK Versi 1.8.0.91
- Hadoop Versi : 3.1.1
- HADOOP HOME : /home/hduser/hadoop

Studi Kasus

Berikut kasus yang harus diselesaikan menggunakan hadoop MapReduce

1. Cari Jumlah Penjualan Product yang terjual pada masing-masing negara

Sumber data bernama *SalesJan2009.csv*

Transaction_date	Product	Price	Payment_Type	Name	City	State	Country
1/2/09 6:17	Product1	1200	Mastercard	carolina	Basildon	England	United Kingdom
1/2/09 4:53	Product1	1200	Visa	Betina	Parkville	MO	United States
1/2/09 13:08	Product1	1200	Mastercard	Federica e Andrea	Astoria	OR	United States
1/3/09 14:44	Product1	1200	Visa	Gouya	Echuca	Victoria	Australia
1/4/09 12:56	Product2	3600	Visa	Gerd W	Cahaba Heights	AL	United States
1/4/09 13:19	Product1	1200	Visa	LAURENCE	Mickleton	NJ	United States
1/4/09 20:11	Product1	1200	Mastercard	Fleur	Peoria	IL	United States
1/2/09 20:09	Product1	1200	Mastercard	adam	Martin	TN	United States

Panduan

1. Login kedalam system hadoop sebagai user hduser

```
#su – hduser
```

2. Buat Directory latihan/MapReduce , lalu masuk kedalam directory latihan/MapReduce

```
$mkdir -p latihan/MapReduce  
$cd latihan/MapReduce
```

```
[hduser@debian:~$ mkdir -p latihan/MapReduce  
[hduser@debian:~$ cd latihan/MapReduce/  
[hduser@debian:~/latihan/MapReduce$ ls  
[hduser@debian:~/latihan/MapReduce$ pwd  
/home/hduser/latihan/MapReduce  
hduser@debian:~/latihan/MapReduce$ ]
```

3. Download file SalesJan2009.csv pada <https://github.com/deje212/hadoop>

```
$wget  
https://raw.githubusercontent.com/deje212/hadoop/master/SalesJan2009.csv
```

```
[hduser@debian:~/latihan/MapReduce$ wget https://raw.githubusercontent.com/deje212/hadoop/master/SalesJan2009.csv  
--2018-10-16 05:55:15-- https://raw.githubusercontent.com/deje212/hadoop/master/SalesJan2009.csv  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.8.133  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.8.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 123637 (121K) [text/plain]  
Saving to: 'SalesJan2009.csv'  
  
SalesJan2009.csv 100%[=====] 120.74K --.-KB/s in 0.08s  
2018-10-16 05:55:16 (1.55 MB/s) - 'SalesJan2009.csv' saved [123637/123637]  
[hduser@debian:~/latihan/MapReduce$ ls -l  
total 124  
-rw-r--r-- 1 hduser hduser 123637 Oct 16 05:55 SalesJan2009.csv  
hduser@debian:~/latihan/MapReduce$ ]
```

4. Setting Permission directory /home/hduser/latihan/MapReduce

```
$chmod +x /home/hduser/latihan/MapReduce
```

5. Program aplikasi terdiri dari 3 file bernama SalesMapper.java, SalesCountryReducer.java dan SalesCountryDriver.java
6. Buat File SalesMapper.java

```
$nano SalesMapper.java
```

Masukan isi sebagai berikut lalu simpan

```
package SalesCountry;  
  
import java.io.IOException;  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.Text;
```

```

import org.apache.hadoop.mapred.*;

public class SalesMapper extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleCountryData = valueString.split(",");
        output.collect(new Text(SingleCountryData[7]), one);
    }
}

```

7. Buat file SalesCountryReducer.java

```
$nano SalesCountryReducer.java
```

```

package SalesCountry;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SalesCountryReducer extends MapReduceBase implements
Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        Text key = t_key;
        int frequencyForCountry = 0;
        while (values.hasNext()) {
            // replace type of value with the actual type of our value
            IntWritable value = (IntWritable) values.next();
            frequencyForCountry += value.get();

        }
    }
}

```

```
        output.collect(key, new IntWritable(frequencyForCountry));
    }
}
```

8. Buat file SalesCountryDriver.java

```
$nano SalesCountryDriver.java
```

```
package SalesCountry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class SalesCountryDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        // Create a configuration object for the job
        JobConf job_conf = new JobConf(SalesCountryDriver.class);

        // Set a name of the Job
        job_conf.setJobName("SalePerCountry");

        // Specify data type of output key and value
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);

        // Specify names of Mapper and Reducer Class
        job_conf.setMapperClass(SalesCountry.SalesMapper.class);
        job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

        // Specify formats of the data type of Input and output
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);

        // Set input and output directories using command line arguments,
        //arg[0] = name of input directory on HDFS, and arg[1] = name of
        output directory to be created to store the output file.
```

```
FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

my_client.setConf(job_conf);
try {
    // Run the job
    JobClient.runJob(job_conf);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

9. Files di atas juga bisa didownload pada <https://github.com/deje212/hadoop>

10. Setting Permission Semua file *.java

```
$chmod +x *.java
```

11. Lihat list file yang telah dibuat

```
$ls -l
```

```
hduser@debian:~/latihan/MapReduce$ ls -l
total 136
-rwxr-xr-x 1 hduser hduser 1368 Oct 16 06:09 SalesCountryDriver.java
-rwxr-xr-x 1 hduser hduser 750 Oct 16 06:09 SalesCountryReducer.java
-rw-r--r-- 1 hduser hduser 123637 Oct 16 05:55 SalesJan2009.csv
-rwxr-xr-x 1 hduser hduser 660 Oct 16 06:09 SalesMapper.java
hduser@debian:~/latihan/MapReduce$
```

12. Export CLASSPATH , disini kita akan mengatur environment CLASSPATH yang akan digunakan untuk melakukan compile file java

```
$export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-client-core-
3.1.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-client-common-
3.1.1.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-
3.1.1.jar:/home/hduser/latihan/MapReduce/SalesCountry/*:$HADOO
P_HOME/lib/*"
```

```
[hduser@debian:~/latihan/MapReduce$ export CLASSPATH="$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.1.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common-3.1.1.jar:$HADOOP_HOME/share/hadoop/common/hadoop-common-3.1.1.jar:/home/hduser/latihan/MapReduce/SalesCountry/*:$HADOOP_HOME/lib/*"
[hduser@debian:~/latihan/MapReduce$ echo $CLASSPATH
/home/hduser/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.1.1.jar:/home/hduser/hadoop/share/hadoop/common/hadoop-common-3.1.1.jar:/home/hduser/latihan/MapReduce/SalesCountry/*
*:~/home/hduser/hadoop/lib/*
hduser@debian:~/latihan/MapReduce$ ]
```

13. Compile Java files

```
$javac -d . SalesMapper.java SalesCountryReducer.java  
SalesCountryDriver.java
```

14. Jika tidak ada error maka file hasil compile ada pada directory *SalesCountry* sesuai dengan nama package

```
[hduser@debian:~/latihan/MapReduce$ javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java
[hduser@debian:~/latihan/MapReduce$ ls -l
total 140
drwxr-xr-x 2 hduser hduser 4096 Oct 16 06:14 SalesCountry
-rw-r--r-- 1 hduser hduser 1368 Oct 16 06:09 SalesCountryDriver.java
-rw-r--r-- 1 hduser hduser 750 Oct 16 06:09 SalesCountryReducer.java
-rw-r--r-- 1 hduser hduser 123637 Oct 16 05:55 SalesJan2009.csv
-rw-r--r-- 1 hduser hduser 660 Oct 16 06:09 SalesMapper.java
[hduser@debian:~/latihan/MapReduce$ ls SalesCountry
SalesCountryDriver.class SalesCountryReducer.class SalesMapper.class
hduser@debian:~/latihan/MapReduce$ ls SalesCountry
```

15. Buat file Manifest.txt

```
$nano Manifest.txt
```

Masukan isi sebagai berikut

```
Main-Class: SalesCountry.SalesCountryDriver ↵-enter
```

16. Membuat Jar file , jika sudah berhasil maka akan terbentuk file **ProductSalePerCountry.jar**

```
$jar cfm ProductSalePerCountry.jar Manifest.txt  
SalesCountry/*.class
```

```
[hduser@debian:~/latihan/MapReduce$ jar cfm ProductSalePerCountry.jar Manifest.txt SalesCountry/*.class
[hduser@debian:~/latihan/MapReduce$ ls -l
total 148
-rw-r--r-- 1 hduser hduser 45 Oct 16 06:17 Manifest.txt
-rw-r--r-- 1 hduser hduser 2970 Oct 16 06:18 ProductSalePerCountry.jar
drwxr-xr-x 2 hduser hduser 4096 Oct 16 06:14 SalesCountry
-rw-r--r-- 1 hduser hduser 1368 Oct 16 06:09 SalesCountryDriver.java
-rw-r--r-- 1 hduser hduser 750 Oct 16 06:09 SalesCountryReducer.java
-rw-r--r-- 1 hduser hduser 123637 Oct 16 05:55 SalesJan2009.csv
-rw-r--r-- 1 hduser hduser 660 Oct 16 06:09 SalesMapper.java
hduser@debian:~/latihan/MapReduce$ ]
```

17. Edit file etc/hadoop/mapred-site.xml dengan tambahkan konfigurasi sebagai berikut sebelum </configuration>

```
<property>
<name>yarn.app.mapreduce.am.env</name>
<value>HADOOP_MAPRED_HOME=/home/hduser/hadoop</value>
</property>
<property>
<name>mapreduce.map.env</name>
<value>HADOOP_MAPRED_HOME=/home/hduser/hadoop</value>
</property>
<property>
<name>mapreduce.reduce.env</name>
<value>HADOOP_MAPRED_HOME=/home/hduser/hadoop</value>
</property>
```

18. Start Hadoop

```
$HADOOP_HOME/sbin/start-dfs.sh
```

```
$HADOOP_HOME/sbin/start-yarn.sh
```

19. Copy the File SalesJan2009.csv kedalam /home/hduser/latihan/MapReduce/input

```
$mkdir /home/hduser/latihan/MapReduce/inputMapReduce
```

```
$cp SalesJan2019.csv
/home/hduser/latihan/MapReduce/inputMapReduce/
```

20. Import Directory inputMapReduce/ kedalam format HDFS

```
$HADOOP_HOME/bin/hdfs          dfs      -copyFromLocal
/home/hduser/latihan/MapReduce/inputMapReduce /
```

```
hduser@debian:~/Latihan/MapReduce$ $HADOOP_HOME/bin/hdfs dfs -copyFromLocal /home/hduser/latihan/MapReduce/inputMapReduce /
2018-10-16 06:27:54,285 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
hduser@debian:~/latihan/MapReduce$
```

21. Verifikasi apakah directory sudah berhasil masuk kedalam format HDFS

```
$HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce
```

```
[hduser@debian:~/latihan/MapReduce$ $HADOOP_HOME/bin/hdfs dfs -ls /inputMapReduce
2018-10-16 06:30:25,615 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
Found 1 items
-rw-r--r-- 1 hduser supergroup 123637 2018-10-16 06:30 /inputMapReduce/SalesJan2009.csv
hduser@debian:~/latihan/MapReduce$ ]
```

22. Menjalankan Aplikasi

```
$HADOOP_HOME/bin/hadoop jar ProductSalePerCountry.jar
/inputMapReduce /mapreduce_output_sales
```

23. Jika tidak ada error maka akan terbentuk folder mapreduce_output_sales yang berisi hasil dari program yang dijalankan

```
$HADOOP_HOME/bin/hdfs dfs -ls /mapreduce_output_sales
```

```
[hduser@debian:~/latihan/MapReduce$ $HADOOP_HOME/bin/hdfs dfs -ls /mapreduce_output_sales
2018-10-16 06:33:42,926 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
Found 2 items
-rw-r--r-- 1 hduser supergroup 0 2018-10-16 06:33 /mapreduce_output_sales/_SUCCESS
-rw-r--r-- 1 hduser supergroup 661 2018-10-16 06:33 /mapreduce_output_sales/part-00000
hduser@debian:~/Latihan/MapReduce$ ]
```

24. Melihat hasil program

```
$HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_sales/part-
00000
```

```
[hduser@debian:~/latihan/MapReduce$ $HADOOP_HOME/bin/hdfs dfs -cat /mapreduce_output_sales/part-00000
2018-10-16 06:35:42,739 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform
where applicable
Argentina      1
Australia     38
Austria       7
Bahrain        1
Belgium        8
Bermuda        1
Brazil         5
Bulgaria      1
CO             1
Canada        76
Cayman Isls   1
China          1
Costa Rica    1
Country        1
Czech Republic 3
Denmark       15
Dominican Republic 1
Finland       2
France        27
Germany      25
Greece        1
Guatemala    1
Hong Kong     1
Hungary       3
Iceland       1
India          2
Ireland       49
Israel         1
Italy          15
Japan          2
Jersey         1
Kuwait         1
Latvia        1
```

Apache PIG

Dalam kerangka Map Reduce, program perlu diterjemahkan ke dalam serangkaian tahapan Map Reduce .Namun, ini bukan model pemrograman yang akrab dengan analis data. Jadi, untuk menjembatani masalah tersebut, sebuah ecosystem yang disebut PIG dibangun di atas Hadoop.

Apache Pig adalah tool yang awalnya dikembangkan dan digunakan oleh Yahoo. Yahoo kemudian merilis Apache Pig untuk Apache sehingga bisa dikembangkan dan digunakan secara luas. Apache Pig lebih cocok digunakan untuk proses ETL (Extract-transform-load). Apache Pig terdiri dari dua jenis komponen. Pig Latin dan Pig Runtime. Pig Latin adalah bahasa yang digunakan di Pig untuk membuat Map Reduce. Pig Latin mengubah syntax low-level dari Map Reduce sehingga menjadi bahasa yang mudah dimengerti. Pig Runtime mengubah dan menjalankan script dari Pig Latin menjadi Map Reduce di Hadoop.

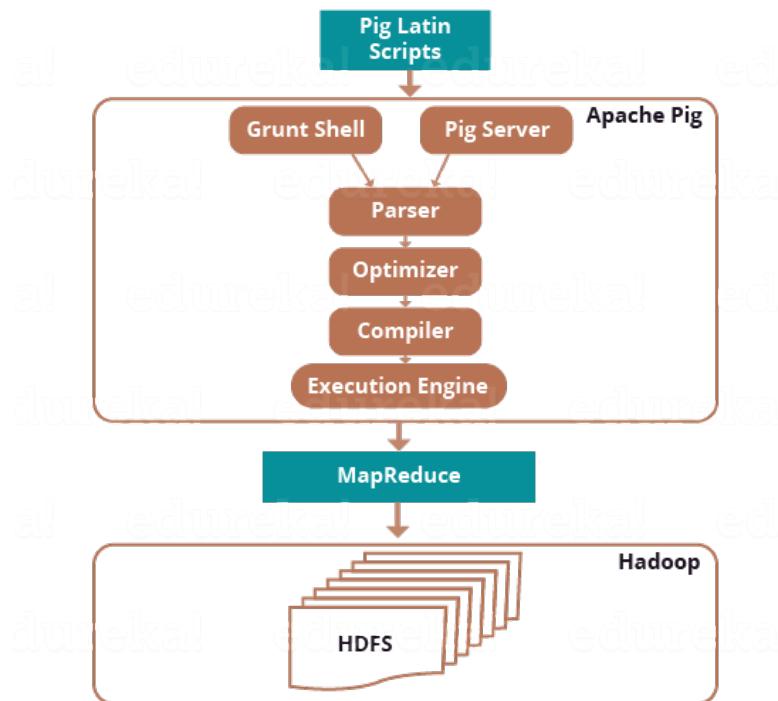
PIG memungkinkan pengguna untuk lebih fokus dalam menganalisis bulk data dan menghabiskan lebih sedikit waktu dalam menulis program Map-Reduce.

Bahasa pemrograman PIG dirancang untuk bekerja pada setiap jenis data. Itu sebabnya namanya, PIG!



Apache Pig Architecture

Untuk menulis sebuah script Pig, kita menggunakan Pig Latin language dan untuk mengeksekusinya, kita membutuhkan pig shell. Arsitektur Apache Pig ditunjukkan pada gambar di bawah ini.



Pig Latin Scripts

Awalnya seperti yang diilustrasikan pada gambar di atas, pengguna mengirimkan script Pig ke environment Apache Pig untuk di eksekusi .

Ada tiga cara untuk menjalankan script Pig:

- Grunt Shell: Ini adalah shell interaktif Pig yang disediakan untuk mengeksekusi semua Script Pig.
- Script file : Tulis semua perintah Pig dalam file script dan jalankan file script Pig. Ini dijalankan oleh Pig Server.
- Embedded Script: Jika beberapa fungsi tidak tersedia di operator bawaan, kita dapat secara terprogram membuat Fungsi yang Ditetapkan Pengguna untuk membawa fungsi tersebut menggunakan bahasa lain seperti Java, Python, Ruby, dll. Dan menyisipkan dalam file Pig Latin Script. Kemudian, jalankan file script tersebut.

Apache Pig Latin Data Model

Model data memungkinkan Pig Latin untuk menangani semua jenis data. Pig Latin dapat menangani kedua tipe atom data seperti int, float, long, double dll. Dan tipe data kompleks seperti tuple, bag dan map. Saya akan menjelaskannya satu per satu.

Gambar di bawah ini menunjukkan tipe data dan kelas terkait yang dapat digunakan untuk menerapkannya:

Pig Data Type	Implementing Class
Bag	org.apache.pig.data.DataBag
Tuple	org.apache.pig.data.Tuple
Map	java.util.Map<Object, Object>
Integer	java.lang.Integer
Long	java.lang.Long
Float	java.lang.Float
Double	java.lang.Double
Chararray	java.lang.String
Bytearray	byte[]

Jenis Data Atom / Skalar

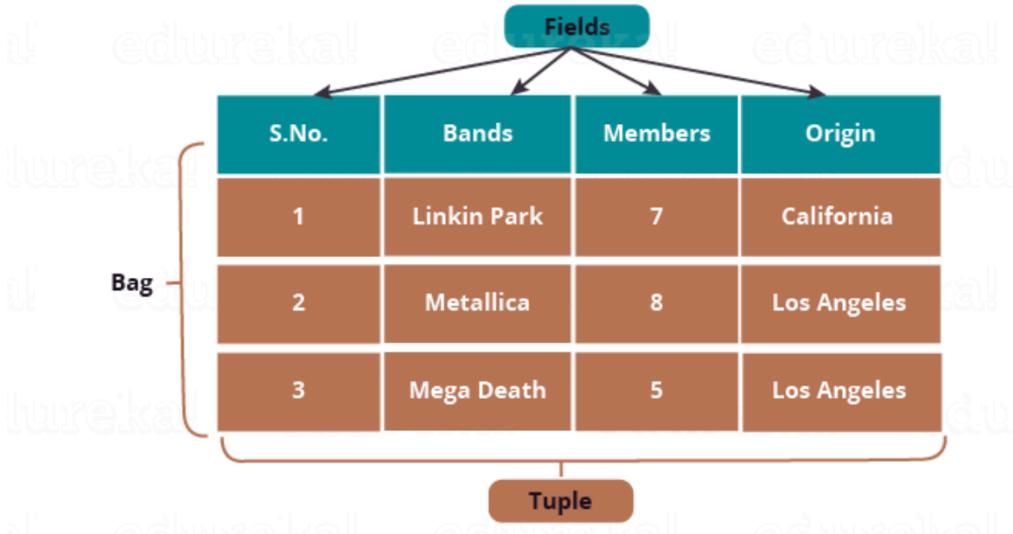
Jenis data atom atau skalar adalah tipe data dasar yang digunakan dalam semua bahasa seperti *string*, *int*, *float*, *long*, *double*, *char []*, *byte []*. Ini juga disebut tipe data primitif. Nilai setiap sel dalam bidang (kolom) adalah tipe data atom seperti yang ditunjukkan pada gambar di bawah ini.

Untuk bidang, indeks posisi yang dihasilkan oleh sistem secara otomatis (juga dikenal sebagai notasi posisional), yang diwakili oleh '\$' dan mulai dari \$ 0, dan tumbuh \$ 1, \$ 2, seterusnya ...

Dibandingkan dengan gambar di bawah

\$0 = S.No.,
\$1 = Band,
\$2 = member,
\$3 = origin.

Jenis data skalar adalah - '1', 'Linkin Park', '7', 'California', dll



Jenis Complex Data

- **Tuple**

Tuple adalah kumpulan field terurut yang mungkin berisi tipe data berbeda untuk setiap field.

Tuple diwakili oleh simbol '()'.

- **Bag**

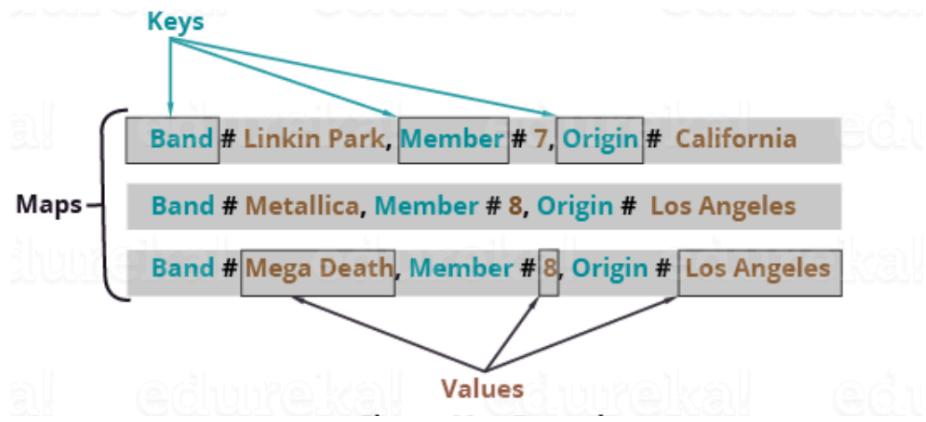
Bag adalah kumpulan satu set tuple dan tuple ini merupakan bagian dari baris atau seluruh baris tabel. Sebuah bag dapat berisi duplikat tuple , dan itu tidak wajib nilainya unik.

Bag diwakili oleh simbol '{}'.

- **Map**

Map adalah pasangan nilai-kunci yang digunakan untuk merepresentasikan elemen data. Kunci harus berupa chararray [] dan harus unik seperti nama kolom, sehingga dapat diindeks dan nilai yang terkait dengannya dapat diakses berdasarkan kunci. Nilai dapat berupa tipe data apa pun.

Map diwakili oleh simbol '[]' dan nilai kunci dipisahkan oleh simbol '#', seperti yang Anda lihat pada gambar di bawah.



Installasi Apache Pig

Berikut adalah langkah-langkah untuk installasi Apache Pig, menggunakan binary dengan versi Apache Pig 0.17.0

1. Download file , disini working directory berada pada /home/hduser/

```
$wget http://kambing.ui.ac.id/apache/pig/latest/pig-0.17.0.tar.gz
```

2. Extract File dan ganti nama folder menjadi pig

```
$tar -zxvf pig-0.17.0.tar.gz
$mv pig-0.17.0 pig
```

3. Extract File dan ganti nama folder menjadi pig

```
$tar -zxvf pig-0.17.0.tar.gz
$mv pig-0.17.0 pig
```

4. Setting Environment , edit file .bashrc dan masukan nilai sebagai berikut

```
$nano ~/.bashrc

# Set PIG_HOME
export HADOOP_CONF_DIR=/home/hduser/hadoop/etc/hadoop
export PIG_HOME=/home/hduser/pig
export PATH=$PATH:/home/hduser/pig/bin
export PIG_CLASSPATH=$HADOOP_CONF_DIR
```

```
# Set PIG_HOME
export HADOOP_CONF_DIR=/home/hduser/hadoop/etc/hadoop
export PIG_HOME=/home/hduser/pig
export PATH=$PATH:/home/hduser/pig/bin
export PIG_CLASSPATH=$HADOOP_CONF_DIR
hduser@debian:~$
```

5. Reload .bashrc

```
$source ~/.bashrc
```

6. Memulai menjalankan pig grunt shell

```
$pig
```

```
hduser@debian:~$ pig
2018-10-21 08:23:28,623 INFO [main] pig.ExecTypeProvider: Trying ExecType : LOCAL
2018-10-21 08:23:28,624 INFO [main] pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2018-10-21 08:23:28,624 INFO [main] pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2018-10-21 08:23:28,654 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2018-10-21 08:23:28,654 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hduser/pig_1540085008653.log
2018-10-21 08:23:28,666 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /home/hduser/.pigbootup not found
2018-10-21 08:23:30,419 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2018-10-21 08:23:30,469 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2018-10-21 08:23:30,469 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000
2018-10-21 08:23:32,719 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-c944f756-5413-407d-bf0b-857736b5acff
2018-10-21 08:23:32,719 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt>
```

Terdapat dua mode untuk mengeksekusi perintah pada Apache Pig di antaranya :

- Mode MapReduce - Ini adalah mode default, yang membutuhkan akses ke kluster Hadoop dan instalasi HDFS. Karena, ini adalah mode default, tidak perlu menentukan -x flag (Anda dapat mengeksekusi babi ATAU mapreduce babi-x). Input dan output dalam mode ini ada pada HDFS.
- Mode Lokal - Dengan akses ke satu mesin, semua file diinstal dan dijalankan menggunakan host lokal dan sistem file. Di sini, mode lokal ditentukan menggunakan '-x flag' (lokal babi -x). Input dan output dalam mode ini ada pada sistem file lokal.

Membuat Apache Pig Program

Disini kita akan mulai membuat program/script sederhana menggunakan Apache Pig dalam mode Map Reduce.

1. Buat Directory latihan/Pig , lalu masuk kedalam directory latihan/Pig

```
$mkdir -p latihan/Pig  
$cd latihan/Pig
```

```
[hduser@debian:~$ mkdir -p latihan/Pig  
[hduser@debian:~$ cd latihan/Pig  
[hduser@debian:~/latihan/Pig$ ls  
hduser@debian:~/latihan/Pig$ █
```

2. Buat file inputKaryawan.txt dan isikan nilai sebagai berikut

```
$nano inputKaryawan.txt  
dodol,111110,Direktur,200000000  
cendol,111111,Senior Manager,150000000  
peyem,111112,Manager,50000000  
cobro,111113,Staff,10000000  
misro,111115,Staff,11000000
```

3. Buat folder /latihan dalam format HDFS

```
$hadoop fs -mkdir /latihan
```

4. Copy file inputKaryawan.txt kedalam folder /latihan dalam format HDFS

```
$hadoop fs -copyFromLocal  
/home/hduser/latihan/Pig/inputKaryawan.txt /latihan
```

5. Check file inputKaryawan.txt

```
$hadoop fs -ls /latihan
```

```
[hduser@debian:~/latihan/Pig$ hadoop fs -ls /latihan  
2018-10-21 08:47:19,346 WARN util.NativeCodeLoader: Unable to load native-hadoop library for  
Found 1 items  
-rw-r--r-- 1 hduser supergroup 157 2018-10-21 08:46 /latihan/inputKaryawan.txt  
hduser@debian:~/latihan/Pig$ █
```

- Buat file dengan nama infoKaryawan.pig pada folder latihan/Pig

```
$nano ~/latihan/Pig/infoKaryawan.pig
```

- Isi file sebagai berikut

```
A = LOAD '/latihan/inputKaryawan.txt' using PigStorage
(',) as (Nama: chararray, NIK: chararray , Jabatan:
chararray, Salary: int);

B = FOREACH A generate Nama, NIK, Jabatan,Salary;

DUMP B;
```

- menjalankan script/program Pig

```
$pig ~/latihan/Pig/infoKaryawan.pig
```

```
2018-10-21 09:16:46,571 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2018-10-21 09:16:46,587 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-metrics-publisher.enabled is dep
se yarn.system-metrics-publisher.enabled
2018-10-21 09:16:46,603 [main] INFO org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2018-10-21 09:16:46,621 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2018-10-21 09:16:46,622 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(dodol,11110,Direktur,20000000)
(cendol,11111,Senior Manager,15000000)
(peyem,11112,Manager,5000000)
(cobro,11113,Staff,1000000)
(misro,11115,Staff,1100000)
2018-10-21 09:16:46,728 [main] INFO org.apache.pig.Main - Pig script completed in 2 minutes, 57 seconds and 674 milliseconds (177674 ms)
hduser@debian:~/latihan/Pig$
```

Grunt Apache Pig Shell

Setelah membuat script/program kecil , kita juga bisa menjalankan script/program tersebut melalui grunt shell

- Studi Kasus :
 - Menampilkan jumlah jabatan (COUNT)

- Masuk kedalam grunt shell dengan menjalankan perintah pig

```
$pig
```

```
hduser@debian:~/latihan/Pig$ pig
2018-10-21 09:55:38,422 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2018-10-21 09:55:38,442 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2018-10-21 09:55:38,443 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2018-10-21 09:55:38,514 [main] INFO org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15:41:58
2018-10-21 09:55:38,514 [main] INFO org.apache.pig.Main - Logging error messages to: /home/hduser/latihan/Pig/pig_1540090538513.log
2018-10-21 09:55:38,580 [main] INFO org.apache.pig.impl.Utils - Default bootup file /home/hduser/.pigbootup not found
2018-10-21 09:55:39,516 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform...
re applicable
2018-10-21 09:55:39,855 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use
2018-10-21 09:55:39,855 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system
2018-10-21 09:55:41,484 [main] INFO org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-0df715e4-a13a-4ce0-9a03-a2f4
2018-10-21 09:55:41,484 [main] WARN org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set to false
grunt> ■
```

2. Load data inputKaryawan.txt

```
karyawanTable = LOAD '/latihan/inputKaryawan.txt' USING
PigStorage (',') as (Nama: chararray, NIK: chararray, Jabatan:
chararray, Salary: chararray);
```

3. Group data by Jabatan

```
GroupByJabatan = GROUP karyawanTable BY Jabatan;
[grunt> GroupByJabatan = GROUP karyawanTable BY Jabatan;
grunt> ■
```

4. Count data Jabatan

```
CountJabatan = FOREACH GroupByJabatan GENERATE
group,COUNT(karyawanTable);
```

5. Menjalankan CountJabatan

```
dump CountJabatan;
```

```
(Staff,2)
(Manager,1)
(Direktur,1)
(Senior Manager,1)
grunt> ■
```

6. Fungsi FILTER , digunakan untuk mencari sesuai dengan string/variable yang diberikan

```
cari_ = FILTER karyawanTable BY Jabatan == 'Staff';
dump cari_
(cobro,111113,Staff,10000000)
(misro,111115,Staff,11000000)
grunt>
```

Apache Pig dengan HBase

Selain data dari TSV Pig juga bisa digabungkan untuk mengambil data dari HBase, pada bahasan ini kita akan mengambil data pada HBase menggunakan script/program Pig.

1. Disini akan membuat table baru bernama info_pangan dan import data dari file “info-harga-pangan-januari-2018-edited_clean.csv”

```
$/home/hduser/hbase/bin/hbase shell
hbase(main):008:0> create 'info_pangan','data_pangan'
hbase(main):010:0> describe 'info_pangan'
```

```
[hbase(main):010:0> describe 'info_pangan'
Table info_pangan is ENABLED
info_pangan
COLUMN FAMILIES DESCRIPTION
{NAME => 'data_pangan', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'true', _ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
1 row(s)
Took 0.0187 seconds
hbase(main):011:0>
```

2. Import data TSV dari file “info-harga-pangan-januari-2018-edited_clean.csv”

```
$/home/hduser/hbase/bin/hbase
org.apache.hadoop.hbase.mapreduce.ImportTsv -
Dimporttsv.separator=',' -
Dimporttsv.columns='HBASE_ROW_KEY,data_pangan:tanggal,
data_pangan:nama_pasar,
data_pangan:komoditas,data_pangan:harga' info_pangan
/home/hduser/latihan/input/info-harga-pangan-januari-2018-edited_clean.csv
```

```
-----  
File System Counters  
    FILE: Number of bytes read=40385300  
    FILE: Number of bytes written=38062810  
    FILE: Number of read operations=0  
    FILE: Number of large read operations=0  
    FILE: Number of write operations=0  
Map-Reduce Framework  
    Map input records=50468  
    Map output records=50468  
    Input split bytes=144  
    Spilled Records=0  
    Failed Shuffles=0  
    Merged Map outputs=0  
    GC time elapsed (ms)=157  
    Total committed heap usage (bytes)=30261248  
ImportTsv  
    Bad Lines=0  
    File Input Format Counters  
        Bytes Read=2982260  
    File Output Format Counters  
        Bytes Written=0  
hduser@hadoop:~$
```

3. Load data HBase menggunakan Apache Pig

```
$/home/hduser/pig/bin/pig  
grunt>info_pangan = LOAD 'hbase://info_pangan' USING  
org.apache.pig.backend.hadoop.hbase.HBaseStorage (  
'data_pangan:tanggal,data_pangan:nama_pasar,data_pangan:komoditas,data_pangan:harga','-loadKey true')  
AS (tanggal,nama_pasar,komoditas,harga);
```

```
[grunt> info_pangan = LOAD 'hbase://info_pangan' USING org.apache.pig.backend.hadoop.hbase.HBaseStorage (  
>>  
>> 'data_pangan:tanggal,data_pangan:nama_pasar,data_pangan:komoditas,data_pangan:harga','-loadKey true')  
>> AS (tanggal,nama_pasar,komoditas,harga);
```

Apache Hive

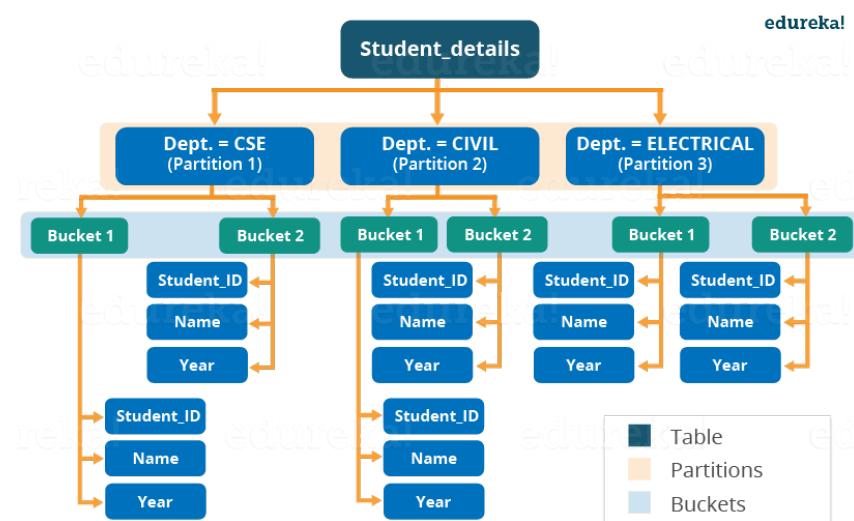
Apache Hive adalah proyek open source yang dijalankan oleh relawan di Apache Software Foundation. Hive merupakan suatu infrastruktur datawarehousing untuk Hadoop. Fungsi utama dari Hive adalah untuk menyediakan data summarization, query dan analisis. Hive juga mendukung analisis dataset berukuran besar yang tersimpan di HDFS Hadoop dan juga pada filesystem Amazon S3. Hive mendukung perintah yang mirip dengan SQL seperti akses ke data terstruktur yang dikenal dengan istilah Hive-QL (atau HQL) serta analisis data yang besar dengan bantuan MapReduce. Hive tidak dibangun untuk mendapatkan respon cepat terhadap query namun dibuat terutama untuk aplikasi data mining. Aplikasi data mining dapat memakan waktu beberapa menit hingga beberapa jam untuk menganalisis data dan Hive terutama digunakan di sana.

Arsitektur Hive

Ada 3 format data yang digunakan dalam Hive yaitu:

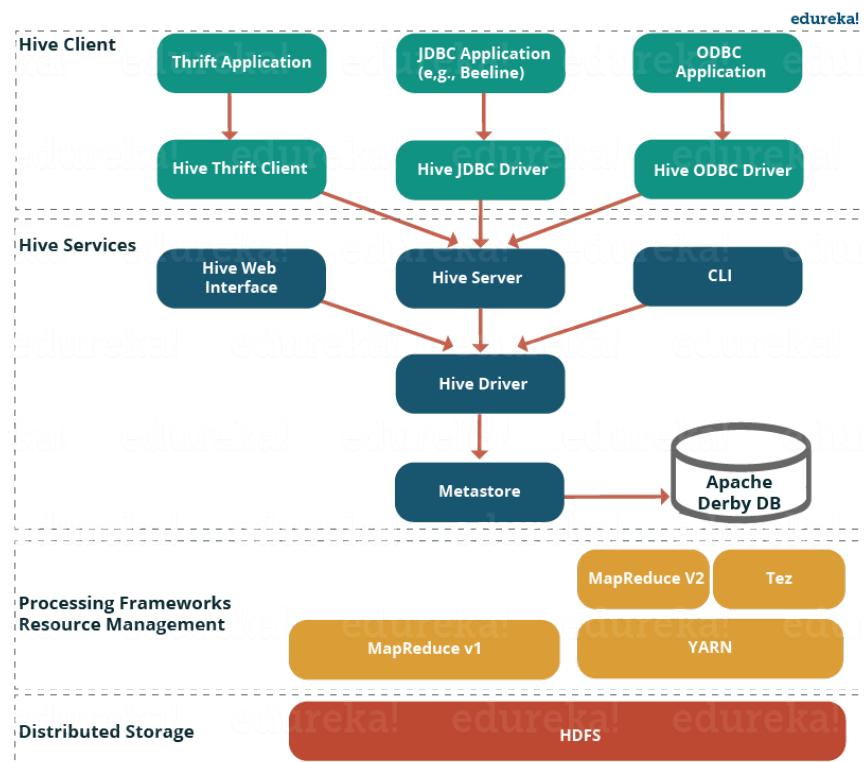
- Tabel: Tabel dalam Hive sangat mirip dengan tabel RDBMS yang juga berisi baris dan tabel. Hive hanya dilapisi dengan Hadoop File System (HDFS), oleh karena itu tabel dipetakan secara langsung ke direktori filesystem. Ini juga mendukung tabel yang tersimpan dalam sistem file asli lainnya.
- Partisi: tabel Hive bisa memiliki lebih dari satu partisi. Mereka dipetakan ke subdirektori dan sistem file juga.
- Bucket: Dalam data Hive dapat dibagi menjadi bucket. Bucket disimpan sebagai file dalam partisi pada sistem file yang mendasarinya.

Hive juga memiliki metastore yang menyimpan semua metadata. Ini adalah database relasional yang berisi berbagai informasi yang berkaitan dengan Skema Hive (jenis kolom, pemilik, data nilai kunci, statistik, dll.). Kita bisa menggunakan database MySQL disini.



Arsitektur Hive dapat dikategorikan ke dalam komponen-komponen berikut:

- Hive Clients: Hive mendukung aplikasi yang ditulis dalam banyak bahasa seperti Java, C ++, Python dll. Menggunakan JDBC, driver Thrift dan ODBC. Oleh karena itu seseorang dapat selalu menulis aplikasi klien sarang yang ditulis dalam bahasa pilihan mereka.
- Hive Services: Apache Hive menyediakan berbagai layanan seperti CLI, Antarmuka Web, dll. Untuk melakukan kueri. Kami akan mengeksplorasi masing-masing dari mereka segera di blog tutorial Hive ini.
- Processing Frameworks Resources Management: Secara internal, Hive menggunakan kerangka Hadoop MapReduce sebagai mesin de facto untuk mengeksekusi kueri. Kerangka MapReduce Hadoop adalah topik yang terpisah dalam dirinya sendiri dan oleh karena itu, tidak dibahas di sini.
- Distributed Storage : Saat Hive dipasang di atas Hadoop, ia menggunakan HDFS yang mendasari untuk penyimpanan terdistribusi. Anda dapat merujuk ke blog HDFS untuk mempelajari lebih lanjut tentang itu.



Hive Client:

Apache Hive mendukung berbagai jenis aplikasi klien untuk melakukan query pada Hive. Klien-klien ini dapat dikategorikan ke dalam tiga jenis:

- Thrift Client : Sebagai server Hive didasarkan pada Apache Thrift, dapat melayani permintaan dari semua bahasa pemrograman yang mendukung Thrift.
- JDBC Clients: Hive memungkinkan aplikasi Java untuk terhubung dengannya menggunakan driver JDBC yang didefinisikan di kelas org.apache.hadoop.hive.jdbc.HiveDriver.
- ODBC Client : The Hive ODBC Driver memungkinkan aplikasi yang mendukung protokol ODBC untuk terhubung ke Hive. (Seperti driver JDBC, driver ODBC menggunakan Thrift untuk berkomunikasi dengan server Hive.)

Hive-QL (HQL)

Hive-QL (HQL)Hive-QL atau Hive Query Language adalah suatu bahasa query yang mirip dengan SQL yang digunakan dalam Hive. Berikut ini adalah beberapa hal yang bisa dilakukan HQL dengan mudah.

- Membuat dan mengelola tabel dan partisi
- Mendukung berbagai Operator Relasional, Aritmatika dan Logika
- Mengevaluasi fungsi
- Mengunduh isi tabel ke direktori lokal atau hasil query ke direktori HDFS

Hive Services

Hive menyediakan banyak layanan seperti yang ditunjukkan pada gambar di atas. Mari kita lihat masing-masing:

- Hive CLI (Command Line Interface): Ini adalah shell default yang disediakan oleh Hive di mana Anda dapat mengeksekusi pertanyaan dan perintah Hive Anda secara langsung.
- Apache Hive Web Interfaces: Selain antarmuka baris perintah, Hive juga menyediakan GUI berbasis web untuk mengeksekusi pertanyaan dan perintah Hive.
- Hive Server: Hive server dibangun di atas Apache Thrift dan oleh karena itu, juga disebut sebagai Thrift Server yang memungkinkan klien yang berbeda untuk mengirimkan permintaan ke Hive dan mengambil hasil akhir.
- Apache Hive Driver: Bertanggung jawab untuk menerima pertanyaan yang diajukan melalui CLI, antarmuka web, Thrift, ODBC atau JDBC antarmuka oleh klien. Kemudian,

driver melewati permintaan ke kompilator di mana parsing, pengecekan jenis dan analisis semantik berlangsung dengan bantuan skema hadir di metastore. Pada langkah berikutnya, rencana logis yang dioptimalkan dihasilkan dalam bentuk DAG (Directed Acyclic Graph) dari tugas-tugas pengurangan peta dan tugas-tugas HDFS. Akhirnya, mesin eksekusi menjalankan tugas-tugas ini dalam urutan ketergantungan mereka, menggunakan Hadoop.

- Metastore: Anda dapat menganggap metastore sebagai repositori pusat untuk menyimpan semua informasi metadata Hive.

Metadata Hive mencakup berbagai jenis informasi seperti struktur tabel dan partisi bersama dengan kolom, tipe kolom, serializer dan deserializer yang diperlukan untuk operasi Baca / Tulis pada data yang ada dalam HDFS.

Metastore terdiri dari dua unit dasar:

- Layanan yang menyediakan akses metastore ke layanan Hive lainnya.
- Penyimpanan disk untuk metadata yang terpisah dari penyimpanan HDFS.

Penggunaan Hive:

1. Penyimpanan didistribusikan Hive Apache.
2. Hive menyediakan alat untuk mengaktifkan ekstrak data mudah / mengubah / memuat (ETL)
3. Hive menyediakan struktur pada berbagai format data.
4. Dengan menggunakan Hive, kita dapat mengakses file yang disimpan di Hadoop Distributed File System (HDFS digunakan untuk meng-query dan mengelola dataset besar yang berada di dalamnya) atau dalam sistem penyimpanan data lain seperti Apache HBase.

Keterbatasan Hive:

1. Hive tidak dirancang untuk pemrosesan transaksi Online (OLTP), itu hanya digunakan untuk Pengolahan Analitik Online.
2. Hive mendukung overwriting atau apprehending data, tetapi tidak memperbarui dan menghapus.
3. Di Hive, sub query tidak didukung.

Installasi Apache Hive

Berikut adalah langkah-langkah untuk installasi Apache Hive, menggunakan binary dengan versi Apache Hive 3.1.0

1. Download file , disini working directory berada pada /home/hduser/

```
$wget http://kambing.ui.ac.id/apache/hive/hive-3.1.0/apache-hive-3.1.0-bin.tar.gz
```

2. Extract File dan ganti nama folder menjadi hive

```
$tar -zxvf apache-hive-3.1.0-bin.tar.gz  
$mv apache-hive-3.1.0-bin hive
```

3. Setting Environment , edit file .bashrc dan masukan nilai sebagai berikut

```
$nano ~/.bashrc  
  
# Set HIVE ENV  
export HIVE_HOME=/home/hduser/hive  
export PATH=$PATH:/home/hduser/hive/bin
```

4. Reload Environment

```
$source ~/.bashrc
```

5. Cek versi hive

```
$hive --version
```

```
hduser@hadoop:~/src$ hive --version
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hduser/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hduser/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive 3.1.0
Git git://vgargwork.local/Users/vgarg/repos/hive.apache.master -r bcc7d93
Compiled by vgarg on Mon Jul 23 16:02:03 PDT 2018
From source with checksum 147d8070369f8c672407753089777fd1
hduser@hadoop:~/src$
```

6. Buat Directory hive dalam format HDFS yang akan digunakan untuk menyimpan data yang berkaitan dengan Hive

```
$hadoop fs -mkdir /hive/warehouse
$hadoop fs -mkdir /tmp
```

7. Konfigurasi Hive Enviroment

```
$nano $HIVE_HOME/conf/hive-env.sh
```

8. Masukan Konfigurasi sebagai berikut

```
export HADOOP_HOME=/home/hduser/hadoop
export HIVE_CONF_DIR=/home/hduser/hive/conf
```

9. Konfigurasi Hive Site file 'hive-site.xml'

```
cp $HIVE_HOME/conf/hive-default.xml.template
$HIVE_HOME/conf/hive-site.xml
```

10. Edit Konfigurasi sebagai berikut

```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/hive/warehouse</value>
  <description>location of default database for the
warehouse</description>
</property>

<property>
```

```

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/home/hduser/hive/metastore_db;create=true</value>
<description>
  JDBC connect string for a JDBC metastore.
  To use SSL to encrypt/authenticate the connection,
  provide database-specific SSL flag in the connection URL.
  For example, jdbc:postgresql://myhost/db?ssl=true
  for postgres database.
</description>
</property>
<property>
  <name>hive.exec.scratchdir</name>
  <value>/tmp/hive</value>
  <description>Scratch space for Hive
  jobs</description>
</property>

```

11. Inisiasi Derby Database

```
$schematool -initSchema -dbType derby
```

```
[hduser@hadoop:~/hive$ schematool -initSchema -dbType derby
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hduser/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hduser/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:      jdbc:derby:;databaseName=/home/hduser/hive/metastore_db;create=true
Metastore Connection Driver :   org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:     APP
Starting metastore schema initialization to 3.1.0
Initialization script hive-schema-3.1.0.derby.sql
```

12. Menjalankan Hive

```
$hive
```

```
[hduser@hadoop:~/hive/conf$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hduser/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hduser/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 7d768d58-b764-4d7d-a471-34af7d6cf4f8

Logging initialized using configuration in jar:file:/home/hduser/hive/lib/hive-common-3.1.0.jar!/hive-log4j2.properties Async:
true
OpenJDK Server VM warning: You have loaded library /home/hduser/hadoop/lib/native/libhadoop.so.1.0.0 which might have disabled
stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link it with '-z noexecstack'.
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
(i.e. spark, tez) or using Hive 1.X releases.
hive> ]
```

Perintah Dasar Hive SQL

Berikut adalah perintah dalam Hive SQL

Data Definition Language (DDL)

Pernyataan DDL digunakan untuk membuat dan memodifikasi tabel dan objek lain dalam database , fungsi tersebut antara lain :CREATE,DROP,TRUNCATE,ALTER,SHOW,DESCRIBE

- CREATE
Membuat database

```
[hive> create database data_pangan;
```

```
[hive> create database data_pangan;
OK
Time taken: 1.129 seconds
hive> ]
```

- SHOW
Melihat daftar database yang ada pada Hive

```
[hive> show databases;
```

```
[hive> show databases;
OK
data_pangan
default
Time taken: 0.076 seconds, Fetched: 2 row(s)
hive> ]
```

- USE

Masuk kedalam database yang akan digunakan

```
hive>use data_pangan;
```

- CREATE

Digunakan untuk membuat table pada database

```
hive>CREATE TABLE info_pangan(id INT,tanggal  
STRING,nama_pasar STRING,komoditas STRING,harga  
DOUBLE) row format delimited fields terminated by ','  
STORED AS textfile;
```

```
[hive> CREATE table info_pangan(id INT,tanggal STRING,nama_pasar STRING,komoditas STRING,harga DOUBLE) row format delimited field  
s terminated by ',' stored as textfile;  
OK  
Time taken: 1.47 seconds  
hive> ]
```

- DESCRIBE

Digunakan untuk melihat schema dari table pada database

```
hive>describe info_pangan;
```

```
[hive> describe info_pangan;  
OK  
id int  
tanggal string  
nama_pasar string  
komoditas string  
harga double  
Time taken: 0.818 seconds, Fetched: 5 row(s)  
hive> ]
```

Data Manipulation Language (DML)

Pernyataan DML digunakan untuk mengambil, menyimpan, memodifikasi, menghapus, menyisipkan dan memperbarui data dalam database.fungsi tersebut antara lain : LOAD, INSERT,SELECT Statements.

- LOAD

Digunakan untuk memindahkan data kedalam format table pada Hive Format

```
LOAD data <LOCAL> inpath <file path> into table  
[tablename]
```

Load table data pangan dari file hdfs yang terletak pada /latihan/info-harga-pangan-januari-2018-edited_clean.csv

```
Hive>LOAD DATA INPATH '/latihan/info-harga-pangan-januari-2018-edited_clean.csv' OVERWRITE INTO TABLE info_pangan;
```

```
hive> LOAD DATA INPATH '/latihan/info-harga-pangan-januari-2018-edited_clean.csv' OVERWRITE INTO TABLE info_pangan;
Loading data to table default.info_pangan
OK
Time taken: 2.527 seconds
hive>
```

- **SELECT**

Digunakan untuk menampilkan data pada database Hive

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
      FROM table_reference
      [WHERE where_condition]
      [GROUP BY col_list]
      [ORDER BY col_list]
      [CLUSTER BY col_list
        | [DISTRIBUTE BY col_list] [SORT BY col_list]
      ]
      [LIMIT [offset,] rows]
```

Select ALL

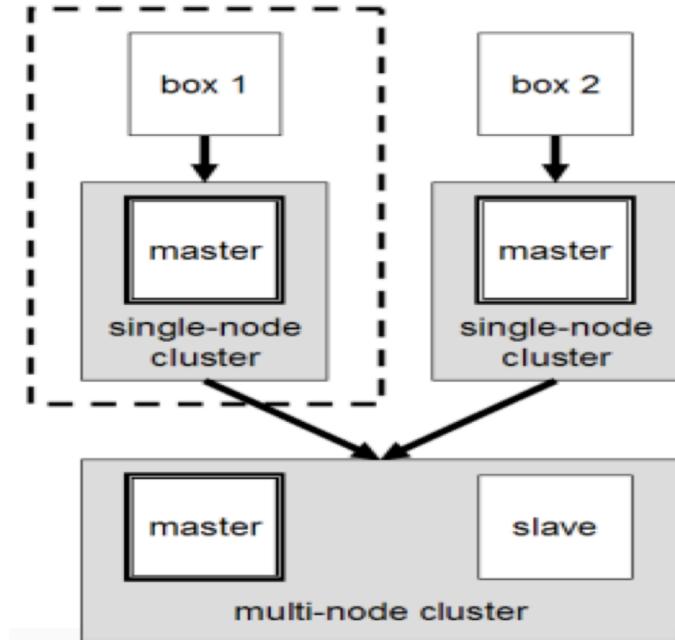
```
Hive>SELECT * FROM info_pangan;
```

Select DISTINCT

```
Hive>SELECT DISTINCT nama_pasar FROM info_pangan;
```

Hadoop Cluster

Pada panduan ini digunakan untuk konfigurasi Hadoop menggunakan multi-node cluster”.



Prasyarat

1. Masing-masing node bisa terhubung satu dengan lainnya
2. Hadoop sudah terinstall pada /home/hduser/hadoop

Architecture Hadoop Cluster

Sebelum mengonfigurasi master node dan slave, penting untuk memahami berbagai komponen dari kelompok Hadoop.

Master Node menyimpan informasi tentang sistem file terdistribusi, seperti tabel inode pada filesystem ext3, dan menjadwalkan alokasi sumber daya. node-master akan menangani peran ini dalam panduan ini, dan menjalankan dua layanan:

- NameNode
- ResourceManager

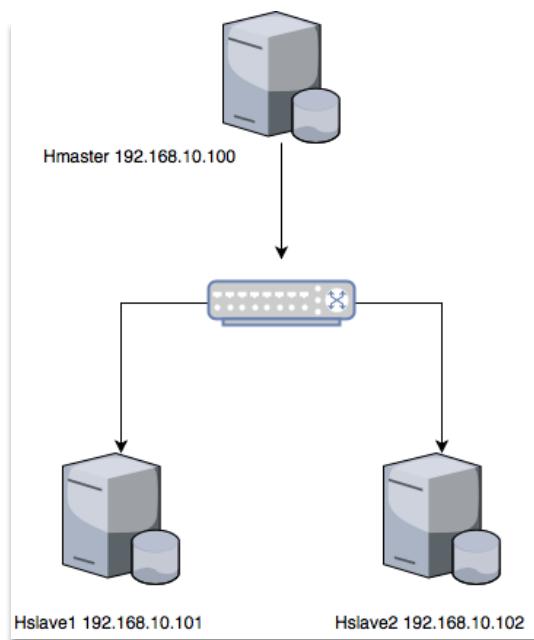
Slave Node menyimpan data secara actual dan menyediakan kekuatan pemrosesan untuk menjalankan sebuah pekerjaan .Pada panduan ini akan menjadi node1 dan node2, slave node menjalankan dua layanan antara lain:

- DataNode
- NodeManager

Konfigurasi Hadoop Cluster

Berikut adalah konfigurasi pada hadoop cluster sesuai dengan gambar di bawah :

- Hmaster : 192.168.10.100
- Hslave1 : 192.168.10.101
- Hslave2 : 192.168.10.102



1. Konfigurasi hosts file yang akan digunakan sebagai pengenal hostname

```
$nano /etc/hosts
```

Dengan isi sebagai berikut

```
192.168.10.100 Hmaster
192.168.10.101 Hslave1
192.168.10.102 Hslave2
```

2. Distribusi SSH Authentication Key-pairs

Hadoop cluster menggunakan ssh untuk melakukan sinkronisasi data, untuk itu masing-masing host baik master maupun slave harus bisa terhubung menggunakan ssh.

Disini kita akan melakukan distribusi ssh key ke masing-masing host/node

- Login kedalam **Hmaster** sebagai user hduser, lalu jalankan perintah ssh-copy-id

```
$ssh-copy-id ~/.ssh/id_rsa.pub hduser@Hmaster  
$ssh-copy-id ~/.ssh/id_rsa.pub hduser@Hslave1  
$ssh-copy-id ~/.ssh/id_rsa.pub hduser@Hslave2
```

- Konfigurasi NameNode

\$HADOOP_HOME/etc/hadoop/core-site.xml

```
$nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

Menjadi

```
<configuration>  
  <property>  
    <name>fs.default.name</name>  
    <value>hdfs://Hmaster:9000</value>  
  </property>  
</configuration>
```

- Konfigurasi HDFS Site \$HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
$nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Menjadi

```
<configuration>  
  <property>  
    <name>dfs.namenode.name.dir</name>  
    <value>/home/hduser/data/nameNode</value>  
  </property>  
  
  <property>  
    <name>dfs.datanode.data.dir</name>  
    <value>/home/hduser/data/dataNode</value>  
  </property>
```

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
</configuration>
```

- Konfigurasi YARN \$HADOOP_HOME/etc/hadoop/yarn-site.xml

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

```
<configuration>
    <property>
        <name>yarn.acl.enable</name>
        <value>0</value>
    </property>

    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>Hmaster</value>
    </property>

    <property>
        <name>yarn.nodemanager.aux-
services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

- Format HDFS

```
nano $HADOOP_HOME/bin/hdfs namenode -format
```

- Login kedalam **Hslave1** sebagai user hduser, ulangi sampai **Hslave2**

- Edit file slaves \$HADOOP_HOME/etc/hadoop/slaves

```
nano $HADOOP_HOME/etc/hadoop/slaves
```

Dengan isi

```
Hslave1  
Hslave2
```

- Edit file YARN \$HADOOP_HOME/etc/hadoop/yarn-site.xml

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

Dengan isi

```
<property>  
    <name>yarn.nodemanager.resource.memory-  
mb</name>  
    <value>1536</value>  
</property>  
  
<property>  
    <name>yarn.scheduler.maximum-allocation-  
mb</name>  
    <value>1536</value>  
</property>  
  
<property>  
    <name>yarn.scheduler.minimum-allocation-  
mb</name>  
    <value>128</value>  
</property>  
  
<property>  
    <name>yarn.nodemanager.vmem-check-  
enabled</name>  
    <value>false</value>  
</property>
```

- Edit file MAPR \$HADOOP_HOME/etc/hadoop/mapred-site.xml

```
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

Dengan isi

```
<property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
        <value>512</value>
</property>

<property>
    <name>mapreduce.map.memory.mb</name>
        <value>256</value>
</property>

<property>
    <name>mapreduce.reduce.memory.mb</name>
        <value>256</value>
</property>
```

- Format HDFS

```
nano $HADOOP_HOME/bin/hdfs namenode -format
```