**Materi 1 - Big Data A**

**Intro**

Big data adalah asset informasi yang besar secara volume, kecepatan (velocity), variety(keberagaman) yang dapat digunakan untuk membuat suatu keputusan atau mendapatkan wawasan. Juga tentang pemrosesan dari data yang besar.

Karakteristik Big Data

3V dan 5V

3V:

* Volume, ukuran data yang sangat besar, dari data set yang besar atau dari data kecil yang dikumpulan dari waktu ke waktu. Tantangannya adalah bagaimana untuk menyimpan dan menganalisa dari data yang banyak.
* Velocity(kecepatan), seberapa cepat data terakumulasi dan seberapa cepet data diproses dan dianalisa. Contohnya seperti rekomendasi. Memungkinkan untuk mengetahui masalah, tren, bahkan kesempatan dalam hitungan detik.
* Variety(keberagaman), sumber data yang digunakan dari big data memiliki keberagaman (terstruktur, tidak terstruktur dan semi terstruktur)

5V:

* Veracity(kebenaran dan ketepaatan), kepercayaan dari sumber data kita karena data bisa bersifat noise dan tidak benar/ tidak akurat. Karena ketikda akurat maka tidak bisa digunakan (junk in junk out).
* Value(nilai/makna), dari proses Analisa big data apakah hal yg bisa didapatkan apakah wawasan baru atau keuntungan dari individual.

Ekosistem Big Data

1. sumber data
2. pengambilan data, bisa menggunakan spark dan flume
3. penyimpanan data, dibantu dengan HDFS atau cassandra
4. analisis data, Hive, spark, cloudera impara
5. pelaporan data, tableau

Batch Processing(pemrosesan per gelombang)

Data dikumpulkan dulu baru diproses, cocok untuk data set besar dan mesinnya besar. Untuk informasi yang tidak sensitive terhadap waktu.

Stream Processing

Data untuk diproses meskipun data masih berjalan dan diambil per bagian yang diperlukan secepatnya.

**Materi 2 - Big Data A**

**Sumber Data Big Data**

Sumber data merupakan komponen utama dari big data.

Bentuk/tipe dari data:

1. Terstruktur

Panjang dan format tertentu dan memiliki 20% dari total data, contohnya:

database -> table (baris dan kolom), angka (int), nama(varchar)

csv -> spreadsheet/excel disimpan dalam baris kolom

website log -> untuk memprediksi untuk celah keamanan

1. Tidak terstruktur

Tidak memiliki format tertentu, tidak memiliki metadata contohnya foto, radar teks

Diperlukan suatu proses tambahan agar bisa dikenali komputer. Seperti teks yang akan dihitung ada berapa yang muncul dipakai sebagai analisa.

1. Semi terstruktur

Biasanya berkisar hingga 80% yang sangat besar. Dimana tidak memiliki struktur data yang baik namun terdapat metadata. Metadata ini informasi yg terasosiasi akan tercontrol pada schema yg ada pada data dimana mampu menjelaskan dirisendiri. Mempunyai data yg struktur dan ada bagian yg tidak struktur. Seperti format: data pada situs, data media social(XML, JSON dan YAML)

Sumber Data

1. Dari mesin terbuat tanpa campur tangan manusia (sensor suhu, tekanan, jalan, asap/kabut). Tantangannya besar data yg tergenerate.
2. Dari organisasi (basis data, dokumen perusahaan, sheets, dbms) dan memiliki format penyimpanan data berbeda tiap perusahaan. IT tidak sama dengan Bank. Tantangannya yaitu silos atau data yg tersebar dan sulit diambil.
3. Dari manusia(formulir, foto, video, text, status). Tantangannya tipe data yg tidak tentu, yaitu semi terstruktur atau tidak terstruktur dimana kadang harus dilakukan proses untuk analisa.

**Materi 3 - Big Data A**

**Pengambilan Data**

Mempunyai 2 sifat yang akan membedakan dalam pengambilan data:

1. Volume
2. variety

Pengambilan data dari sumber data setelah melakukan penyimpanan ke sistem, dilakukan data preprocessing(cleaning dan transformasi) setelah selesai baru dilakukan penyimpanan data.

ETL, cocok dengan data kuliatas rendah

ELT, terdapat tujuan pemrosesan data yg berbeda, dimana data besar penggunaan ELT lebih cocok dimana memindahkan data langsung ke penyimpanan sebelum melakukan transformasi.

Data Ingestion (penyerapan data)

* secara umum sama seperti ELT, ngambil beberapa data dari beberapa sumber, terjadi pembeda dimana penempatan/menyimpan data yang mempunyai kecepatan data yg tinggi.

Pengambilan data secara Real Time Process

* Streaming data, Data streaming adalah data yang dihasilkan secara terus menerus oleh ribuan sumber data, yang biasanya dikirimkan dalam catatan data secara bersamaan, dan dalam ukuran kecil (urutan Kilobyte).

Tantangan:

1. Pada layer storage, membutuhkan proses penyimpnan secara cepat dan tidak membutuhkan uang yg besar dan kemampuan menulis data stream yg besar
2. Pada layer prosessing, untuk pengkonsumsian data dari lapisan penyimpanan dan tanggung jawab pelayanan komputasi dan kemampuan memberitahu apabila ada data yg tidak diperlukan di penyimapan.

Contoh:

Spark Add On, dimana akan memproses data streaming menjadi micro batches.

Pengambilan data secara Batch Processing(latensi tinggi)

* Data crawling, cara untuk mengambil data dari beberapa level sumber untuk diakses. Contoh twitter: menyediakan API untuk kita bisa mendownload tweet tertentu pada halaman home dengan kode ethic dengan ketetapan twitter.
* Langkah:
  + Membuat key dan token
  + Import library untuk koneksi dengan phyton dengan library tweepy
  + Membuat koneksi
  + Mengambil twit dengan kebutuhan
  + Hasilnya disimpan dengan dump dalam bentuk json. (biasanya dalam bentuk array yg dapat dibaca manusia)

**BAB 4**

**RDBMS(SQL)**

Membantu dalam akses penyimpanan data yang berbasis relasional. Ini terdiri dari relations dimana table yang berisi nilai yg setiap barisnya berisi kumpulan dari data dan setiap barisnya adalah entitas dari dunia nyata.

Lalu ada skema yang merupakan hal pertama yang dibuat saat memulai desain. Berisi dari bentuk data dalam bentuk table yang berisi baris dan kolom. Terdapat bentuk normal yang sudah dinormalisasikan yang akan mendapatkan masslah ketika mempunyai table data yang cukup banyak dan melakukan query dari jumlah table yg cukup banyak juga. Nilai nya yg harus dicocokkan merupakan tantangan, jika data tersebar di beberapa fisik harus di join dari server ke server lain.

SQL

* ACID

1. Atomicity, suatu transaksi harus berhasil semua maka bisa dinyatakan berhasil, ketika ada yg gagal maka harus dikembalikan dari awal transaksi.

2. Consistency, data bersifat valid yang diambil dari hasil kondisi yang konsisten

3. Isolation, harus tidak boleh mempengaruhi satu sama lain.

4. Durability, kondisi data tersimpan secara tetap atau persistence setelah adanya transaksi yang berhasil dari yang di commit

Pengambilan data dari RDBMS

1. Menggunakan pyspark

Membuat koneksi JDBC dan database properties dengan parameter yang dibutuhkan. Yaitu URL, Nama database, skema, username, password dan driver. Ketika berhasil maka bisa melakukan query pada database yang sudah dibuat dan memasukan data query kedalam dataframe yang bisa memiliki kolom dan baris. Kemudian diambil manipulasi data dari dataframe tersebut

2. menggunakan Sqoop

Dimana akan melakukan transfer data melalui HDFS dan basis data relasional. Pertama melakukan install dilanjutkan dengan import sqoop untuk menggunakan apache sqoop ini.

**BAB 5**

**NON RELASIONAL DBMS(NO SQL)**

Karakteristik umum:

* Skalabilitas, kemampuan sistem untuk bertambah besar dan berkembang. Menggunakan horizontal scalability, menambah node baru yang tidak menggangu sistem yg berjalan dan membutuhkan OS untuk membantu tanpa interupsi. Menggunakan node yang berbeda juga setiap proses yang berbeda.
* Model data dan data query, tidak menggunakan model data relasional maka dari itu tidak menggunakan baris, kolom, struktur kunci/key. juga tidak membutuhkan skema yang mampu men self-describe data itu sendiri.
* Desain yang persisten, menggunakan memory dan in memory yang menyimpan data di sistem.
* Antar muka, juga bisa memfasilitasi basis data, programmer, manajer bassis data, alat tool untuk visualisasi
* Konsistensi, menggunakan konsep Basically Available Soft State and Eventual Consistency(BASE). Menjamin permintaan mendapat respon tapi hasil dari respon tidak dijamin konsisten.

Kategori Basis Data Non Relasional:

* Key-value pair, pasangan kunci dan nilai-> key dan value. Simple key digunakan sebagai index yang masing masing key nya memiliki nilai yang disimpan dalam bentuk value.
* Document, type data berupa string dan merupakan nilai dari entitas. Terdapat 2 macam yaitu repository pertama adalah konten dokumen(bekas word, halaman situs) kedua adlaah penyimpanan statis untuk penyimpanan bagian dari dokumen. Struktur dari dokumen beserta bagiannya disediakan oleh JSON. Bisa dibaca dan dilakukan query atau penyaringan data. Mampu melakukan query berdasarkan 1 query atau lebih.
* Columnar, berbasis kolom yang menyimpan data per kolom (kebawah). Biasanya bagus untuk query analitik biasanya bersifat read only.
* Graph, berbentuk graph dari Nodes(entitas) dan Edges(Relasi/hubungan). Untuk mencari informasi dll.
* Spatial, sering digunakan biasanya GPS untuk mencari posisi tertentu dan arah atau bisa juga cuaca untuk badai dll. Berupa objek yang biasanya 2D (panjangxlebar) 3D(p x l x kedalaman) untuk spatial ini mengunakan konsep 2.5D(tambahan informasi dari mapping). Elementnya ada garis, titik dan polygon.

NoSQL untuk Manajemen Big Data

* Evolusi hardware, kemampuan bertambah dan harga semakin murah
* Cloud, membuat banyak node untuk penyimpanan, server, dan pemrosesan
* Kebutuhan aplikasi
* Horizontal partitioning, pembagian data atau membagi load untuk setiap node nya dan jika ditambahkan replikasi ketersediaan data akan lebih baik.
* Akses data yang cepat untuk mencari record untuk individual atau object dari sekian banyaknya data yg ada.

**BAB 6**

**Komputasi Terdistribusi Dalam Big Data**

Distributed Computing merupakan metode untuk menjalankan program pada beberapa komputer di suatu jaringan sehingga bisa dilihat sebagai satu kesatuan sistem jika dilihat dari sisi pengguna. Karena sudah mengerti cara menghubungkan dari banyak komputer.

Hubungan dari big data dimana pada karakteristik big data yang volume dan velocity merupakan dasar yang membantu menjawab dari tantangan dari komputasi terdistribusi.

**Komputasi Terdistribusi**

**Dimana komputasi terdistribusi** adalah bidang dari ilmu komputer yang mempelajari dari sistem terdistribusi sedangkan sistem terdistribusi adalah sistem yang komponennya berada pada jaringan komputer yang bisa berkomunikasi melalui pesan. Memiliki skalabilitas, fault tolerance dan berbagi sumber daya(bukan independent). Solusi yang diberikan yaitu pemrosessan, penyimpanan dan komunikasi (message passing). Ketika ada kegagalan akan bersifat partial meskpun ada kegagalan maka komponen masih ada yang berfungsi dan sistem akan deteksi dan melakukan perbaikan dari kegagalan itu bisa dari recovery atau rollback.

Skalabilitas pada Komputasi Terdistribusi

* Scalling up(vertical), akan menambah kemampuan hardware yang aada(storage/ram) pada satu sistem yang sama. Masalah latency akan lebih kecil karena dalam satu sistem yang sama. Dimana memiliki kekurangan dalam biaya karena menambah hardware mahal dan ada resiko kegagalan lebih tinggi karena sistem dari sistem yang sama.
* Scalling out(horizontal), menambahkan mesin yang performasinya sama atau lebih rendah bisa dalam sistem atau penyimpanannya. Memiliki kelebihan mudah untuk menambahh sumberdaya baik komputasi atau penyimpanan dan skalabilitasnya tidak terbatas karena bisa menambah data nya terus menerus. Untuk voltulens nya lebih baik karena nodeatau server untuk standby dan replikasi dari server atau data server. Lebih mudah untuk recoverynya baik data atau layanannya. Kekurangannya adalah pengaturan untuk platform yg besar, tjika terjadi amsalah akan mengambil waktu yg lama untuk pengecekan node mana yg melakukan kegagalan.

Basis Data NoSQL Terdistribusi

Teorema CAP memiliki 3 sifat yaitu:

* Consistency, menjelaskan tampilan data yang sama untuk semua node dalam sistem yg terdistribusi. Dimana data akan tersebar secara stimultan ke setiap node.
* Availablity, ketersediaan dimana sistem menjawab semua request walaupun gagal operasinya adalah read or write.
* Partition Tolerance, kemampuan 2 segment dari partisi untuk melanjutkan keberadaan data walaupun terdapat kegagalan dalam jaraingan. Ketika memiliki partiion tolerance lebih tinggi, maka bassis data akan tersedia untuk pengguna sehingga akan lebih tidak konsisten.

Penyimpanan terdistribusi

* Dibutuhkan cara untuk menyimpan data yg tersebar di beberapa node dan memiliki mekanisme sycronisasi dan koordinasi antar klaster node dan nodenya. Mekanisme ini disebut dengan penyimpanan terdistribusi. Memiliki sistem berkas terdistribusi. Contohnya HDFS, GFD dan Amazon S3.

Pemrosesan data Terdistribusi

* Dimana tugas akan dibagi prosesnya ke beberapa sub tugas pada node yang berbeda. Dimana sudah tersegmentasi dan ada pada local.

Parallel Processing

* Dua mekanisme:
* Data parallelism bisa membagi data menjadi data yg lebih kecil dan melakukan proses yang sama untuk setiap bagian kecil tadi.
* Control parallelism(task paralellism)

Berbeda dalam instruksinya yang berbeda meskipun waktunya bersamaan

Multiprocessing

* Menggunakan 2 prosesor atau lebih sehingga meningkatkan keluaran dan pekerjaan akan lebih cepet. Tipenya asymetrik dimana setiap processor akan memiliki tugas tertentu dan ada prossor sebagai pos mengatur sistem dan prosessor lain menunggu tugas lain. Tipe Symetric setiap prossesor melakukan tugas dan tidak ada hubungan seperti post.

Multicore

* Satu chip mempunyai banyak core yang membuat sistem lebih efisien. Dimungkinkan komunikasi lebih cepat.

Multihreading

Thread, adalah enggunaan terkecil dari CPU dan sequence dari instruksi pemrograman bisa dijadwalkan secara independ. Proses bisa memiliki single atau multithread.

**BAB 6**

**Pengantar Hadoop**

HDFS(Hadoop distributed filesystem)

Sejarah Hadoop

* 2002, mesin pencarian situs namun biaya tidak memungkinkan nama projectnya yaitu apache nutch
* 2003, google distributed file system yang mampu menyeleseaikan masalah dari menyimpan data yang besar hasil dari proses crawling dan indexing dari situs/web
* 2006, yahoo menggunakan Hadoop muncul dan konsep map reduce. Hadoop berasal dari nama dari gajah kuning mainan anaknya dari dop.
* 2008, menjadi proyek utama pada apache dan mengalahkan super computing untuk sorting data 1tb.
* 2011, Hadoop 1.0 muncul dan terdapat keamanan dan basis data Hbase dan pada 2012 sebagai manajemen mengeluarkan versi fionce.
* Akhir 2017, versi Hadoop akhir yaitu 3.0 terdapat proyek lain mendukung Hadoop. Pada sampai saat ini masih berlaku.

Hadoop adalah framework yang mempunyai skalabilitas yang mampu menyimpan data sangat besar dan melakukan konsep komputasi map reduces serta sistem berkas komputasi pada node komputasi. Pada map reduce sendiri yaitu membagi bagian kecil dan dieksekusi oleh node.

Kapan penggunaan Hadoop:

* Ukuran data sangat besar dari banyak sumber dan berbagai bentuk, pengunaan HDFS sangat cocok untuk data yang sangat besar yang nantinya akan diproses dalam ukuran kecil sehingga ukuran besar akan banyak.
* Karena Hadoop lebih sering untuk batch prosessing maka akan lebih cocok untuk Analisa yang tidak diperlukan dalam waktu dekat.
* Penyimpanan Hadoop juga cocok untuk penyimpanan dalam jangka Panjang. Juga dapat menambahkan node untuk menambah penyimpanan kita.
* Melakukan integrasi dengan berbagai alat untuk Analisa.

Jangan gunakan Hadoop sebagai:

* Tidak bisa digunakan untuk Analisa real time analytics maka diperlukan spark untuk menjawab tantangan tersebut. Penyimpanan akan menggunakan HDFS dan pemrosesan dilakukan oleh SPARK.
* Hadoop bukan sebagai pengganti infrastruktur yang ada, bukan sebagai penganti basis data yang dipunyai, dan lebih tepat sebagai pelengkap infrastruktur yang dimiliki.
* HDFS menggunakan metadata menyimpan informasi terkait data yang disimpannya. Ketika kita memiliki dataset yang kecil dalam jumlah banyak, maka lebih baik dijadikan satu menjadi dataset yang besar dan diakses menggunakan Hadoop. HDFS juga tidak mendukung pembacaan secara acak dan penulisan secara parallel karena HDFS akan membaca secara page.

Komponen Hadoop:

Mempunyai beberapa komponen yang saling terhubung satu sama lain yang membentuk suatu ekosistem. Komponen bisa berupa proyek yang mendukung Hadoop.

1. Penyimpanan, mengunakan HDFS dimana sistem berkas terdistribusi ini dapat menyebar ke beberapa node sekaligus, dimana setiap node memiliki sistem operasi lokalnya sendiri.
2. Sebagai penjadwalan pekerjaan/manejemen cluster sumber daya, kita dapat menggunakan YARN (Yet Another Resource Negotiater).
3. Pemrosesan akan menggunakan MapReduce, dimana terhubung dengan YARN dan HDFS. Untuk inmemory prosesing atau proses dalam memory bisa menggunakan Spark.
4. Untuk SQL dan datawarehouse pada Hadoop bisa menggunakan HIVE dan juga PIG dimana pig framework untuk komputasi pararel dan juga scripting. Dimana hive dan pig terhubung dengan MapReduce turun sampai point 1.
5. Lalu ada komponen terakhir yaitu, zookeeper / ambari dimana sebagai manajemen dan koordinasi dari komponen lainnya. Lalu untuk pemindahan data bisa dibantu dengan flume, sqoop. Untuk flume digunakan ketika data tidak terstruktur maupun semi terstruktur. Sedangkan sqoop untuk data terstruktur.

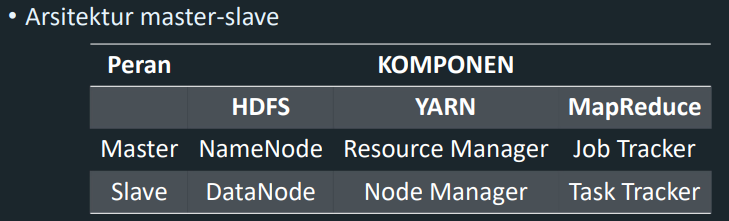
Arsitektur Hadoop

* Hadoop cluster, merupakan 1 kelompok yang terdiri dari beberapa node yang saling terhubung bisa dari LAN/computer yang disetiap nodenya memiliki sumberdaya sendiri seperti memory prosesor dan storage.

Pastinya setiap node memiliki Hadoop yang sudah terinstall dan node bisa memiliki peran master / slave. Dimana master tugasnya melakukan control terhadap penyimpanan dan prosesan. Dimana slave berfungsi menyimpan dan melakukan prosesan data.

Namun jika hanya ada satu node dalam satu cluster maka disebut single node, maka peran master/slave ini aka nada 1 komputer dimana untuk Latihan/fase pengembangan. Untuk multi node biasanya digunakan skala entreprise.

Nama:



* HDFS, berfungsi sebagai penyimpanan data big data yang terdistribusi dalam cluster. Dengan mekanisme begini biaya akan kecil, dapat diandalkan, mampu mengatasi kegagalan hardware dan mengani dataset yang besar.

Memliki directory untuk menyimpan data Hadoop dan agar dapat berjalan harus diformat sesuai kebutuhan Hadoop. Dimana path mirip dengan UNIX atau bersifat hirarki juga untuk permission dan kepemilikan dari berkas mirip dengan UNIX, perbedaannya adalah tidak adanya konsep current directory dan berkas tidak dapat dimodifikasi maka jika salah harus hapus dan download ulang.

Untuk pembacaan berkas dari HDFS diperlukan alat sendiri dengan bantuan dari utilitas dari Hadoop.

Menggunakan arsitektur master dan slave. Ada server yang berperan sebagai master yaitu NameNode yang melakukan pengaturan filesystem namespace dan melakukan pengaturan akses ke berkas yang dilakukan penggunakan dan isi namenode adalah metadata dari semua isi dari semua berkas.

Lalu ada Block, dimana satuan penyimpanan terkecil dari komputer, pada Hadoop bisa sebesar 128mb atau 156mb. Missal jika ada 300mb berkas dengan block 128mb maka akan dibutuhkan 3 block dengan 128 128 44 block. Namenode adalah demon (background proses) yang bertanggung jawab membuka, menutup, mengganti nama berkas dan directory juga tracking dari pemteaan block ke datanode.Datanode adalah demon (background proses) yang berlajan pada node slide yang bertanggung jawab menyimpan data yang sebenarnya. Semisal memiliki 5 node, 1 master dan 4 slave/ 1 namenode dan 4 datanode dimana akan melakukan pembacaan/penulisan berdasarkan sistem berkas client, juga melakukan pembuatan dan penghapusan dan duplikasi block sesuai instruksi namenode yang berjalan pada LINUX

Dibuat dengan Bahasa java.

Untuk file system namespacenya mendukung, traditional hierarchy file organization dimana pengguna dapat memuat directory dan menyimpan data didalamnya.

**YARN**

YARN adalah teknologi Apache Hadoop dan merupakan singkatan dari Yet Another Resource Negotiator digunakan pada data berskala besar, sistem operasi yang didistribusikan untuk aplikasi big data. Teknologi ini dirancang untuk manajemen cluster dan merupakan salah satu fitur utama dalam generasi kedua Hadoop, open source Apache Software Foundation kerangka kerja pemrosesan terdistribusi.

YARN sebagai arsitektur baru dari Apache hadoop 2 membagi dua fungsi utama dari JobTracker/TaskTracker pada MapReduce menjadi beberapa entitas yang terpisah.

* Global ResourceManager di node master, yang berfungsi mengatur semua resource yang digunakan aplikasi dalam sistem.
* ApplicationMaster di setiap aplikasi, yang berfungsi untuk negosiasi resource dengan ResourceManager dan kemudian bekerja sama dengan NodeManager untuk mengeksekusi dan memonitor tasks
* NodeManager di Agen-Framework setiap node slave, yang bertanggung jawab terhadap Container, dengan memantau penggunaan resource/sumber daya dari container(cpu, memori, disk, jaringan ) dan melaporkannya pada ResourceManager
* Container di setiap aplikasi yang jalan di NodeManager, sebagai wadah penyimpanan data/file

**Kegunaan YARN**

YARN didesain untuk mengatasi beberapa limitasi dari MapReduce pada Framework Hadoop versi sebelumnya, berikut beberapa limitasi dari MapReduce berdasarkan pada gambar arsitektur di atas.

1. Skalabilitas terbatas dikarenakan JobTracker hanya ada pada node master yang menjalan semua proses baik pengelolan resource, proses schedulling, maupun monitoring-nya. Dengan adanya single Jobtracker ini skalabilitas-nya menjadi terbatas. Maksimum cluster adalah 4,000 Nodes Maksimum task secara konkuren adalah 40,000 Sinkronisasi yang berat pada JobTracker

2. Ada isu terkait availability, karena single JobTracker, jika JobTracker gagal maka seluruh job mesti di-restart/re-submit.

3. Pemanfaatan resource yang rendah, karena mengalami kesulitan partisi resource ke dalam slot map dan reduce.

4. Isu kompatibalitas, karena hanya support aplikasi dengan paradigma MapReduce Hanya support MapReduce Aplikasi bersifat iterasi 10x lebih lambat dengan MapReduce Tidak handal untuk aplikasi pemrosesan Grafik.

Jadi dapat disimpulkan kegunaan dari Hadoop YARN adalah :

* Skalabilitas yang jauh lebih tinggi
* Availability yang lebih baik
* Penggunaan resource lebih optimal
* Mampu support bermacam jenis paradigma dari aplikasi, tidak hanya MapReduce

Map Reduce

Map Reduce adalah sebuah teknologi yang digunakan pada saat pemrosesan data dalam jumlah besar. 2 kata utama pada module ini adalah map dan reduce. map berfungsi untuk memilah data dan melakukan maping terhadap data tersebut. sedangkan reduce digunakan untuk mengurangi data yang tidak digunakan.

•YARN

-Yet Another Resource Negotiator adalah komponen yang terhubung dengan penyimpanan HDFS

-Sistem Manajemen Sumber Daya Hadoop yaitu dengan berinteraksi dengan aplikasi dan melakukan penjadwalan sumberdaya untuk digunakan

-Dikenalkan pada Hadoop versi 2.0 untuk meningkatkan implementasi MapReduce dan Mendukung distributed computing

-Cara Kerja Yarn adalah

 dengan menyediakan API Untuk request dan Untuk working pada sumberdaya yang ada pada cluster

 digunakan dengan cara hack level API yang disediakan framework distributed computing

• Cara Kerja YARN pada Hadoop 3

- 1 Resources manager percluster

 pada node dengan peran Master akan ada resources Manager yang mempunyai otoritas dalam pembagian sumber daya

 Resources Manager juga bertugas untuk menerima permintaan pemrosesan dan memberikan kepada node yang berperan menjadi slave untuk dilakukan pemrosesan sebenarnya. tugas dari resources manager akan juga melakukan optimasi dari penggunaan sumberdaya cluster

 akan ada satu resources Manager per cluster yang bertugas untuk melakukan penjadwalan yang dilakukan oleh komponen scheduler sedangkan komponen Application Manager bertanggungjawab untuk menerima pekerjaan yang diberikan dan melakukan negosiasi sumberdaya.





• Arsitektur Map Reduce

1. Job tracker berperan sebagai master

 Job tracker menjadi satu node dengan name node

 Job tracker membagi tugas dengan task tracker dengan menggunakan data local

Tugas :

- Menegelola job dari mapreduce

- Menerima tugas dari client

- Melakukan pengawasan tugas yang gagal dan dilakukan penjadwalan ulang

Jumlah : 1 per cluster

2. Task tracker berperan sebagai slave

 Task tracker menjadi satu node dengan datanode. Tugas task tracker akan lebih cepat karena membaca data pada node yang sama

Tugas :

- Melakukan tugas map atau reduce

- Komunikasi dengan job tracker

- Membacak block dari datanode

Jumlah : n buah per cluster

•MapReduce

- Mapreduce adalah framework pemrograman dimana melihat data sebagai kesatuan dari beberapa bagian dan dapat melakukan pemrosesan parallel didalam banyak node

- Langkah : Map  (shuffle and sort)  reduce

- Map digunakan untuk memecah data, melakukan penyaringan atau perubahan bentuk data. Hasilnya menjadi input reduce

- Shuffle and sort bertugas untuk mengatur output map dan input reduce

-Kelebihan :

1. tahan terhadap kegagalan
2. sederhana
3. fleksibel dimana mempunyai kemampuan analitik dan digunakan berbagai tipe data
4. skalabilitas

•Contoh Map reduce : Word Count

-Langkah 0 : menyimpan berkas pada hdfs

-Langkah 1 : map  dimana membuat pasangan key-value untuk setiap blok

-Langkah 2 : shuffle and sort akan mengelompokkan hasil dari map berdasarkan keynya

-Langkah 3 : reduce yaitu melakukan agregasi

**BAB 9**

**Spark**

Spark adalah projek apache dan merupakan framework komputasi pada in memory cluster untuk memproses dan Analisa data sangat besar.

2009, membuat ersebarkley pada IMB lab kemudian menjadi opensource pada 2010. Pada 2013 proyek diberikan kepada apache menjadi apache2.0. Kemudian 2020 menjadi versi ke 3.0.

Spark solusi dari kelemahan Hadoop.

1. Pada kecepatan pemrosesan, spark bisa 100x lebih cepat dari mapReduce pada data yang sama dengan syarat data cukup dalam memory, namun jika memory tidak cukup maka akan 10x lebih cepat dan datanya disimpan dalam disk. Hal ini dapat terjadi karena spark membolehkan komputasi dalam cluster bersifat inmemory. Sehingga pemrosesan dan data bisa tersimpan di memory.
2. Spark sudah mengeksekusikan dengan implementasi mesin eksekusi lebih canggih, ketika data bersifat kecil maka percepatan tidak terlalu beda tetapi data sangat besar maka waktu pemrosesan akan terlihat mencolok.
3. Pada Pipeline, Hadoop tidak mendukung konsep pipeline dimana data akan diproses menjadi suatu output dan output nanti akan menjadi input dari proses berikutnya. Dimana Hadoop menggunakan konsep map dan reduce dimana tidak semuanya bisa diimplementasikan dan kadang tidak tepat sebagai pipeline bigdata. Ketika ada penggabungan data maka akan menyebabkan kompleksitas pada Hadoop, begitu juga dengan filtering atau penyaringan data.
4. Maka spark disini memiliki karakteristik dimana memudahkan penggunaan karena spark memiliki model pemrograman yang lebih sederhana daripada yang disediakan oleh mapReduce. Selain itu API dari spark menyediakan lebih banyak operator daripada mapReduce, maka akan muncul kesulitan implementasikan algoritma untuk mapReduce. Spark juga bersifat interative dimana bisa menggunakan batch prosesing dan stream prosesing.
5. Besar kode dari Spark lebih kecil dari Hadoop yaitu 50% nya saja dan sudah ada library.

Arsitektur Spark

Dimana akan terdapat peran dan tugasnya.

* Trunc worker pada worker node, dimana akan menyediakan sumber daya seperti cpu, memory, penyimpanan untuk aplikasi spark. Spark akan berjalan sebagai proses terdistribusi pada sebuah kluster yang terdiri dari beberapa nodes.
* Cluster manager, dimana bertanggung jawab untuk mendapatkan sumber daya mengerjakan suatu tugas selain itu melakukan sumber daya terhadap cluster yang terdiri dari beberapa node worker. Spark mendukung 3 cluster manager yaitu standalone, missels dan yarn. Ketika menggunakan yarn maka dapat menjalakan Hadoop dan spark pada worker node yang sama secara bersamaan.
* Kemudian ada driver program, yaitu aplikasi menggunakan spark sebagai library. Sebuah driver program bisa menjalankan job sejumlah satu atau lebih pada cluster spark.
* Kemudian aka nada process executor pada sebuah worker node untuk proses jvm (java virtual machine) untuk mengeksekusi kode yang berjalan secara cuncurent di beberapa thread dan dapat melakukan cache data didalam memory dan disk, ketika spark selesai atau terminasi maka executor akan terminasi juga.
* Lalu ada Task, terdapat 2 task pada suatu executor dalam worker node yang terhubung dengan driver program. Dimana spark akan membuat beberapa task per-partisi data dan executor akan menjalan 1 atau lebih task sehingga jumlah pararellism akan ditentukan oleh jumlah partisi.

Konsep RDD(Resilient Distributed Dataset)

Merupakan kumpulan dari elemen data yang terpartisi yang dapat dioperasikan secara parallel dan merupakan mekanisme abstraksi data utama pada spark.

Karakteristik pada RDD:

1. Immutable, sekali data terbuat maka tidak dapat dimodifikasi, maka ketika RDD melakukan operasi maka akan menghasilkan RDD yang baru.
2. Terpartisi, data direpresentasikan oleh RDD akan dibagi menjadi beberapa partisi dimana partisi akan didistribusikan kedalam beberapa node. Secara default partisi akan berada dalam ram, hal ini terjadi karena Class RDD menyediakan API yang membolehkan komputasi cluster dalam memory.
3. In Memory, Spark membolehkan RDD berada dalam memory dengan seperti itu maka operasi akan lebih cepat jika dibandingkan dengan RDD yang disimpan tidak pada memory atau dengan kata lain tidak dilakukan cache.

Terdapat Konsep Lazy Operations

Dimana eksekusi tidak akan dimulai ketika suatu aksi telah di trigerred. Pembuatan RDD dan Transformasi juga dalam konsep ini. Spark akan melakukan track atau pencatatan dari transformasi yang dilakukan kepada sebuah RDD. Sehingga evaluasi data tidak akan diload sampai diperlukan sehingga tidak terjadi overhead untuk kalkulasi serta menghemat perpindahan data antara driver dan kluster.