PIE Tech

POLLACHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE and Affiliated to Anna University)

sky is the limit

Department of Computer Science and Engineering

Regulation 2021
III Year – V Semester

CCS334-BIG DATA ANALYTICS

UNIT I

Big data:

Big data refers to datasets whose size is beyond the ability of typical database software tool to capture, store, manage and analyze. Big data is data that goes beyond the traditional limits of data along three dimensions: i) Volume, ii) Variety iii) Velocity

Data Volume:

Data Volume can be measured by quality of transactions, events and amount of history. Big Data isn't just a description of raw volume. The real challenge is identifying or developing most cost-effective and reliable methods for extracting value from all the terabytes and petabytes of data now available. That's where Big Data analytics become necessary.

Measuring data volume

Unit of Measure	Approximate Size	Mathematical Representation	Examples
KB = kilobyte	1,000 (10^3 or one thousand) bytes	2^{10} or 1024 bytes	A typical joke = 1KB
MB = megabyte	1,000,000 (10^6 or one million) bytes	2^{20} or 1,048,576 bytes	Complete work of Shakespeare = 5MB
GB = gigabyte	1,000,000,000 (10^9 or one billion) bytes	2^{30} or 1,073,741,824 bytes	Ten yards of books on a shelf = 1GB
TB = terabyte	1,000,000,000,000 (or 10^{12})	2^{40} or 1,099,511,627,776 bytes	All the X-rays for a large hospital = 1TB Tweets; created daily = 12+TB; U.S. Library of Congress = 235TB
PB = petabyte	1,000,000,000,000,000 (or 10^{15})	2^{50} or 1,125,899,906,842,624 bytes	All U.S. academic research libraries = 2PB Data processed in a day by Google = 24PB
EB = exabyte	1,000,000,000,000,000,000 (or 10^{18})	2^{60} or 1,152,921,504,606,846,976 bytes	Total data created in 2006 = 161EB
ZB = zettabyte	1,000,000,000,000,000,000,000 (or 10^{21})	2^{70} or 1,180,591,620,717,411,303,424 bytes	Total amount of global data expected to be 2.7 ZB by end of 2012
YB = yottabyte	1,000,000,000,000,000,000,000,000 (or 10^{24})	2^{80} or 1,208,925,819,614,629,174,706,176	Today, to save all those bytes you need a data center as big as the state of Delaware

Data Variety:

It is the assortment of data. Traditionally data, especially operational data, is ―structured‖ as it is put into a database based on the type of data (i.e., character, numeric, floating point, etc.).

Wide variety of data:

Internet data(Social media ,Social Network-Twitter, Face book), Primary Research (Surveys, Experiences, Observations), Secondary Research (Competitive and Market place data,

Industry reports, Consumer data, Business data), Location data (Mobile device data, Geospatial data), Image data (Video, Satellite image, Surveillance), Supply Chain data (vendor Catalogs, Pricing etc), Device data (Sensor data, RF device, Telemetry)

Structured Data

They have predefined data model and fit into relational database. Especially, operational data is ―structured‖ as it is put into a database based on the type of data (i.e., character, numeric, floating point, etc.)

Semi-structured data

These are data that do not fit into a formal structure of data models. Semi-structured data is often a combination of different types of data that has some pattern or structure that is not as strictly defined as structured data. Semi-structured data contain tags that separate semantic elements which includes the capability to enforce hierarchies within the data.

Unstructured data

Do not have a predefined data model and /or do not fit into a relational database. Oftentimes, text, audio, video, image, geospatial, and Internet data (including click streams and log files) are considered unstructured data.

Data Velocity

Data velocity is about the speed at which data is created, accumulated, ingested, and processed. The increasing pace of the world has put demands on businesses to process information in real-time or with near real-time responses. This may mean that data is processed on the fly or while ―streaming‖ by to make quick, real-time decisions or it may be that monthly batch processes are run inter-day to produce more timely decisions.

Why bother about Unstructured data?

- The amount of data (all data, everywhere) is doubling every two years.
- Our world is becoming more transparent. Everyone is accepting this and people don't mind parting with data that is considered sacred and private.
- Most new data is unstructured. Specifically, unstructured data represents almost 95 percent of new data, while structured data represents only 5 percent.
- Unstructured data tends to grow exponentially, unlike structured data, which tends to grow in a more linear fashion.
- Unstructured data is vastly underutilized.

Need to learn how to:

- Use Big data
- Capitalize new technology capabilities and leverage existing technology assets.
- Enable appropriate organizational change.
- Deliver fast and superior results.

Advantage of Big data Business Models:

Improve Operational Efficiencies	**Increase Revenues**	**Achieve Competitive Differentiation**
Reduce risks and costs	Sell to microtrends	Offer new services
Save time	Enable self service	Seize market share
Lower complexity	Improve customer experience	Incubate new ventures
Enable self service	Detect fraud	

Web Analytics

Web analytics is the measurement, collection, analysis and reporting of web data for purposes of understanding and optimizing web usage. Web analytics is not just a tool for measuring web traffic but can be used as a tool for business and market research, and to assess and improve the effectiveness of a web site. The following are the some of the web analytic metrics: Hit, Page view, Visit / Session, First Visit / First Session, Repeat Visitor, New Visitor, Bounce Rate, Exit Rate, Page Time Viewed / Page Visibility Time / Page View Duration, Session Duration / Visit Duration. Average Page View Duration, and Click path etc.

Why use big data tools to analyse web analytics data?

Web event data is incredibly valuable

- It tells you *how* your customers actually behave (in lots of detail), and how that varies
    - Between different customers
    - For the same customers over time. (Seasonality, progress in customer journey)
    - How behaviour drives value
- It tells you *how* customers engage with you via your website / webapp
    - How that varies by different versions of your product
    - How improvements to your product drive increased customer satisfaction and lifetime value
- It tells you *how* customers and prospective customers engage with your different marketing campaigns and how that drives subsequent behaviour

Deriving value from web analytics data often involves very personalized analytics

- The web is a rich and varied space!
  E.g.
    - Bank
    - Newspaper
    - Social network
    - Analytics application
    - Government organisation (e.g. tax office)
    - Retailer
    - Marketplace
- For each type of business you‟d expect different :
    - Types of events, with different types of associated data
    - Ecosystem of customers / partners with different types of relationships
    - Product development cycle (and approach to product development)
    - Types of business questions / priorities to inform how the data is analysed

Web analytics tools are good at delivering the standard reports that are common across different business types.

- Where does your traffic come from e.g.
    - Sessions by marketing campaign / referrer
    - Sessions by landing page

- Understanding events common across business types (page views, transactions, „goals") e.g.
    - Page views per session
    - Page views per web page
    - Conversion rate by traffic source
    - Transaction value by traffic source
- Capturing contextual data common people browsing the web
    - Timestamps
    - Referer data
    - Web page data (e.g. page title, URL)
    - Browser data (e.g. type, plugins, language)
    - Operating system (e.g. type, timezone)
    - Hardware (e.g. mobile / tablet / desktop, screen resolution, colour depth)
- What is the impact of different ad campaigns and creative on the way users behave, subsequently? What is the return on ad spend?
- How do visitors use social channels (Facebook / Twitter) to interact around video content? How can we predict which content will "go viral"?
- How do updates to our product change the "stickiness" of our service?

Industry Examples of Big data:

I) Digital Marketing

i) Database Marketers Pioneers of Big data.
ii) Big data and New school of marketing.
iii) Cross channel life cycle marketing.
iv) Social and affiliate marketing.
v) Empowering marketing with social intelligence.

II) Financial Services

i) Fraud and Big data
Fraud detection framework.
ii) Risk and Big Data
Credit Risk management
Credit Risk Framework
iii) Big data and Algorithmic trading
Crunching through complex interrelated data
Intraday Risk Analytic a constant flow of big data
Calculating Risk in marketing
Benefits to other industries.

III) Big data and Advances in health care

IV) Pioneering New Frontiers in medicine

V) Advertising and Big data

i) Big data impacts on advertising market
ii) Reach, Resonance and Reaction
iii) Need to act quickly
iv) Real-time optimization
v) Complexity of measurement
vi) Content Delivery
vii) Marketing mixed modeling
viii) The Three Big data Vs in Advertising
ix) Using customer Products as a doorway

Industry Examples of Big Data

I) Digital Marketing

- Introduction
- Database Marketers, Pioneers of Big Data
- Big Data & New School of Marketing
- Cross Channel Life cycle Marketing
- Social and Affiliate Marketing
- Empowering marketing with Social Intelligence

Introduction

Digital marketing encompasses using any sort of online media (profit or non-profit) for driving people to a website, a mobile app etc. and retaining them and interacting with them to understand what consumers really want.

Digital \marketing is easy when consumers interact with corporates primary platform (ie. The one the corporates own) because corporates get good information about them. But corporates get very little information once people start interacting with other platforms (eg. Face book, Twitter, Google +).

One of the problems that have to be dealt with in a digital existence, is that corporate do not have access to data for all consumers (ie. There is very little visibility about people when they interact in social networking sites). So corporates lose control of their ability to access the data they need, in order to make smart, timely decisions.

Big data on the web will completely transform a company's ability to understand the effectiveness of its marketing and the ability to understand how its competitors are behaving. Rather than a few people making big decisions, the organizations will be able to make hundreds and thousands of smart decisions every day.

Database Marketers, Pioneers of Big Data

Database marketing is concerned with building databases containing info about individuals, using that information to better understand those individuals and communicating effectively with some of those individuals to drive business value.

Marketing databases are typically used for

i) Customer acquisition
ii) Retaining and cross-selling to existing customers which reactivates the cycle

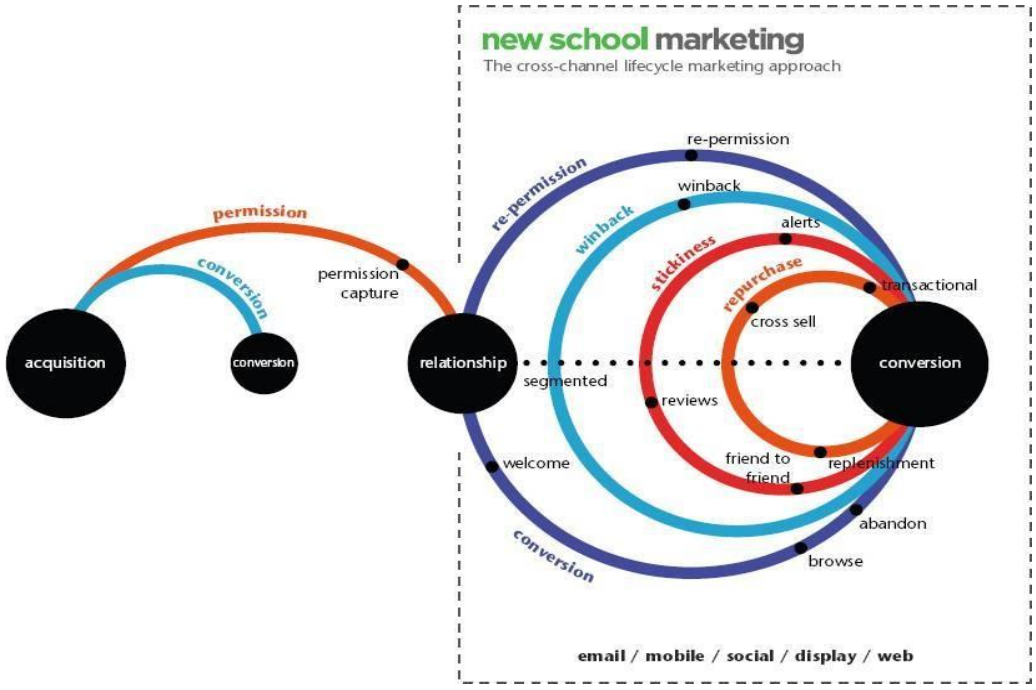
As companies grew and systems proliferated, a situation where there was one system for one product and another for another product etc. was landed up (silos). Then companies began developing technologies to manage and duplicate data from multiple sources. Companies started developing software that could eliminate duplicate customer info (de-duping). This enable them to extract customer information from silos product systems, manage the info into single database, remove all the duplicates and then send direct mail to subsets of the customers in the database. Companies such as Reader's Digest and several other firms were early champions of this new kind of marketing and they used it very effectively. By the 1980's marketers developed the ability to run reports on the info in their databases which gave them better and deeper insights into buying habits and preferences of customers. Telemarketing became popular when marketers figured out how to feed information extracted from customer databases to call centers. In 1990's email entered the picture and marketers saw opportunities to reach customers via Internet and WWW. In the past five years there has been exponential growth in database marketing and the new scale is pushing up against the limits of technology.

Big Data & New School of Marketing

New school marketers deliver what today's consumers want ie. Relevant interactive communication across digital power channels
Digital power channels: email, mobile, social display and web.
Consumers have changed so must marketers

Right approach – Cross Channel Lifecycle Marketing

Cross-Channel Lifecycle Marketing really starts with the capture of customer permission, contact information, and preferences for multiple channels. It also requires marketers to have the right integrated marketing and customer information systems, so that (1) they can have complete understanding of customers through stated preferences and observed behavior at any given time; and (2) they can automate and optimize their programs and processes throughout the customer lifecycle. Once marketers have that, they need a practical framework for planning marketing activities. The various loops that guide marketing strategies and tactics in the Cross-Channel Lifecycle Marketing approach: conversion, repurchase, stickiness, win-back, and re-permission are shown in the following figure.



Social & Affiliate Marketing or Pay for Performance Marketing on the Internet

The concept of affiliate marketing, or pay for performance marketing on the Internet is often credited to William J. Tobin, the founder of PC Flowers & Gifts. Amazon.com launched its own affiliate program in 1996 and middleman affiliate networks like Link-share and Commission Junction emerged preceding the 1990s Internet boom, providing the tools and technology to allow any brand to put affiliate marketing practices to use. Today, most of the major brands have a thriving affiliate program. Today, industry analysts estimate affiliate marketing to be a $3 billion industry. It's an industry that largely goes anonymous. Unlike email and banner advertising, affiliate marketing is a behind the scenes channel most consumers are unaware of.

In 2012, the emergence of the social web brings these concepts together. What only professional affiliate marketers could do prior to Facebook, Twitter, and Tumblr, now any consumer with a mouse can do. Couponmountain.com and other well known affiliate sites generate multimillion dollar yearly revenues for driving transactions for the merchants they promote. The expertise required to build, host, and run a business like Couponmountain.com is no longer needed when a consumer with zero technical or business background can now publish the same content simply by clicking ―Update Status‖ or ―Tweet.‖ The barriers to enter the affiliate marketing industry as an affiliate no longer exist.

Empowering Marketing with Social intelligence

As a result of the growing popularity and use of social media around the world and across nearly every demographic, the amount of user-generated content—or ―big data‖—created is immense, and continues growing exponentially. Millions of status updates, blog posts, photographs, and videos are shared every second. Successful organizations will not only need to identify the information relevant to their company and products—but also be able to dissect it, make sense of it, and respond to it—in real time and on a continuous basis, drawing business intelligence—or insights—that help predict likely future customer behavior. Very intelligent software is required to parse all that social data to define things like the sentiment of a post.

Marketers now have the opportunity to mine social conversations for purchase intent and brand lift through Big Data. So, marketers can communicate with consumers regardless of the channel. Since this data is captured in real-time, Big Data is forcing marketing organizations to quickly optimize media and message. Since this data provides details on all aspects of consumer behavior, companies are eliminating silos within the organization to prescription across channels, across media, and across the path to purchase.

II) Financial Services

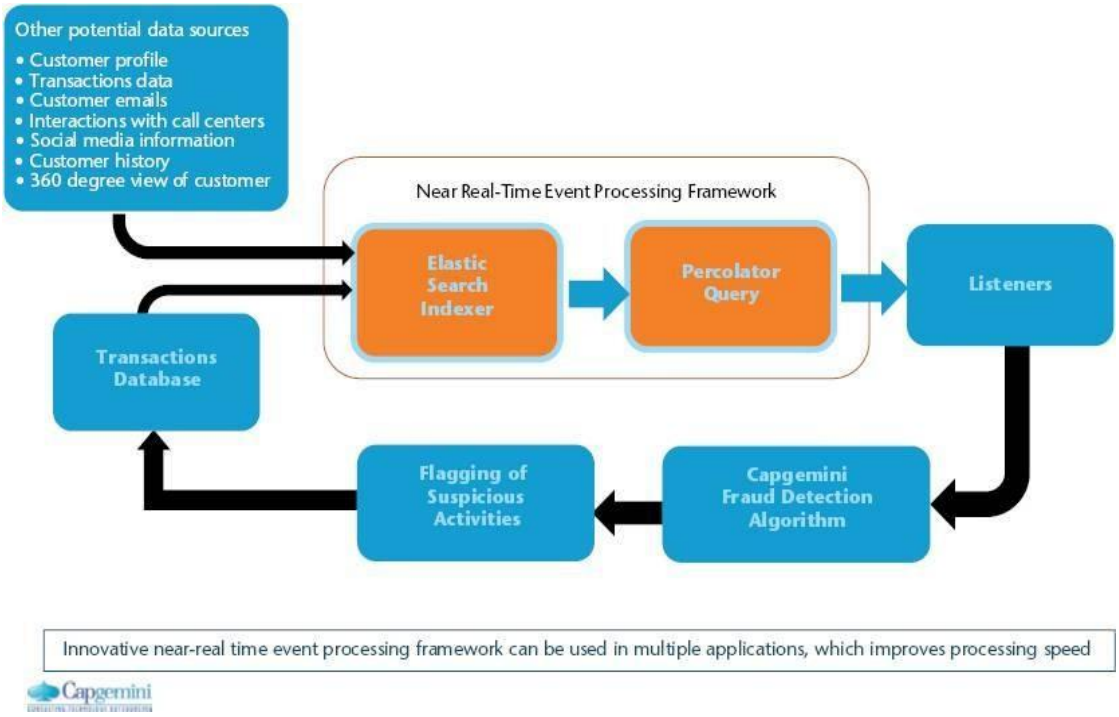
- **Fraud & Big Data**

- Fraud is intentional deception made for personal gain or to damage another individual.
- One of the most common forms of fraudulent activity is credit card fraud.
- Social media and mobile phones are forming new frontiers fraud.
- Capgemini financial services team believes that due to the nature of data streams and processing required BIG Data Technologies provide an optimal technology solution based on the following three Vs :

1. **High volume:** Years of consumer records and transactions (150 billion +
2. records per year).
3. **High velocity:** Dynamic transactions and social media info.
4. **High variety:** Social media plus other unstructured data such as customer E-mails, call center conversations as well as transactional structured data.

Fraud Detection Powered by Near Real-Time Event Processing Framework

- The near real-time event processing framework can be used in multiple applications which improves processing speed.



- This fraud detection system uses an open source search server based on Apache Lucene. It can be used to search all kind of documents at near real-time. The tool is used to index new transactions which are sourced in real-time, which allows analytics to run in a distributed fashion utilizing the data specific to the index. Using this tool, large historical data sets can be used in conjunction with real-time data to identify deviation from typical payment patterns. The big data component allows overall historical patterns to be compared and contrasted and allows the number of attributes and characteristics about consumer behavior to be very wide with little impact on overall performance.

- Percolator performs the function of identifying new transactions that have raised profiles. Percolator can handle both structured and unstructured data. This provides scalability to the event processing framework and allows specific suspicious transactions to be enriched with additional unstructured information (E.g. Phone location/geospatial records, customer travel schedules and so on). This ability to enrich the transaction further can reduce false positives and increase the experience of customer while redirecting fraud efforts to actual instances of suspicious activity.

- Capgemini's fraud Big Data initiative focuses on flagging the suspicious credit card transactions to prevent fraud in near real-time via multi-attribute monitoring. Real-time inputs involving transaction data and customers records are monitored via validity checks and detection rules. Pattern recognition is performed against the data to score and weight individual transactions across each of the rules and scoring dimensions. A cumulative score is then calculated for each transaction record and compared against thresholds to decide if the transaction is suspicious or not.

Social Network Analysis (SNA)

- This is another approach to solving fraud with Big data.
- SNA views social relationships and makes assumptions.
- SNA could reveal all individuals involved in fraudulent activity from perpetrators to their associates and understand their relationships and behavior to identify a bust out fraud case. Bust out is a hybrid credit and fraud problem and the scheme is typically defined by the following behavior.

    - The account in question is delinquent or charged off.
    - The balance is close to or over the limit.
    - One or more payments have been returned.
    - The customer cannot be located.
    - The above conditions exist with more than one account and/or financial institution.

- There are some Big Data solutions in the market like SAS's SNA solution, which helps institutions and goes beyond individual and account views to analyze all related activities and relationships at a network dimension. The network dimension allows visualization of social networks and helps to see hidden connections and relationships, which could be a group of fraudsters. There are huge amounts of data involved behind the scene, but the key to SNA solutions like SAS's is the visualization techniques for users to easily engage and take action.

➢ Risk and Big data

Types of Risk Management

- Credit risk management
- Market risk management
- Operational risk management (not common as credit & market)

The tactics of risk professionals include

- Avoiding risk
- Reducing the negative effect or probability of risk
- Accepting some or all of the potential consequences in exchange for potential gain.
- Credit risk analytics focus on past credit behaviors' to predict the likelihood that a borrower will default on any type of debt by failing to make payments which they accepted to do. E.g. Is this person likely to default on $300000 mortgage?
- Market risk analytics focus on understanding the likelihood that the value of a portfolio will decrease due to the change in stock prices, interest rates, foreign exchange rates and commodity prices. E.g. should we sell this share if the price drops another 10%?

Credit Risk Management

- Credit risk management focuses on reducing risks to acceptable levels that can boost profitability
- Risk data sources and advanced analytics are instrumental for credit risk management

- Social media and cell phone usage data are opening up new opportunities to analyze customer behavior that can be used for credit decisioning.

Credit Risk Framework

- As Figure illustrates, there are four critical parts of the typical credit risk framework: planning, customer acquisition, account management, and collections. All four parts are handled in unique ways through the use of Big Data.

➢ Big Data and Algorithmic Trading

- Financial institutions particularly investment banks have been at the forefront of applying analytics for risk management, proprietary trading and portfolio management.
- Many investment banks use algorithmic trading a highly sophisticated set of processes in which ―insights‖ are made ―actionable‖ via automated ―decisions‖.
- Algorithmic trading relies on sophisticated mathematics to determine buy and sell orders for equities, commodities, interest rate and foreign exchange rates at blinding speed.
- A key component of algorithmic trading is determining return and risk of each trade and then making a decision to buy or sell.
- Algorithmic trading involves a huge number of transactions with complex interdependent data and every millisecond matters.

Crunching through complex interrelated data

- For market risk, the data explodes very quickly. Today, the portfolios being evaluated are quite large and include multiple financial instruments. Even for a single financial instrument it is possible to have hundreds of thousands of scenarios. Hence the problem of scenario analysis, equity derivatives, different equities, different maturities and different strike prices becomes very complex**.**

Intraday Risk Analytics, a constant flow of Big Data

- Banks have moved from daily evaluation of risks to intraday risk evaluation and management.
- Intraday risk management involves pricing the entire portfolio and calculating the risk limits of each of the counter-parties within the bank's portfolio. The problem gets complex and computationally intensive.

- For a bank to do their basic scenario analysis, it takes a million calculations for determining the value at risk for just one instrument. This must happen fast enough so that risk limits on the entire portfolio can be evaluated several times during the course of the day

Calculating Risk in Marketing

- While risk analytics is used for risk management, banks are using risk predictive analytics for marketing.
- When a bank scores its customers and prospects for credit card solicitations, it will use some risk management tools. This tool not only determines who has a high likelihood of responding to promotional offers but also considers the underlying risk for each of the prospects to whom the solicitations are being sent. Without taking risk profiles of individuals, bank promotion responses can result in customers with a higher risk profile
- One of the challenges for retail banks is to score large number of people for its marketing initiatives. The existing customers have to be scored to determine the borrowers whose probabilities of not paying on their credit card or on the mortgage is rising. Once these potential defaulters are identified steps could be taken to mitigate risk of default.
- The huge volume of population that have to be scored increases the problem. The scores have to be calculated quickly in order to take action promptly whether it is promotion or risk mitigation.

Benefits to other industries from financial services risk experience

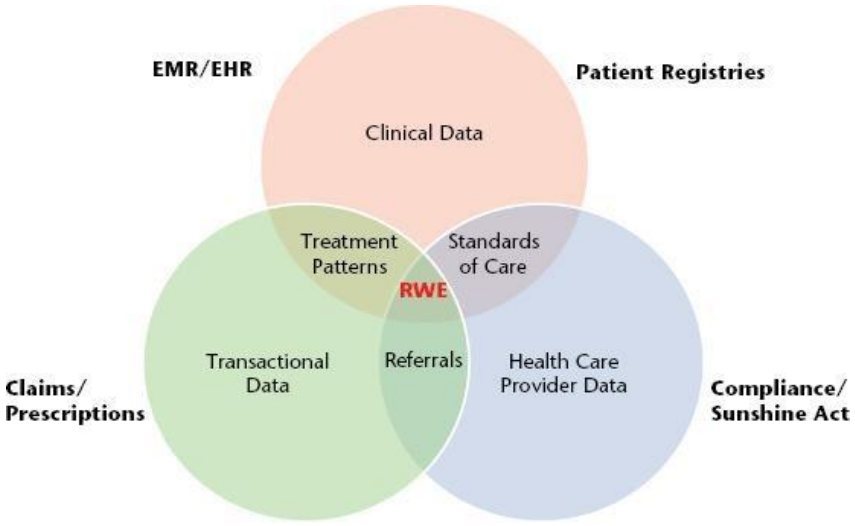
- While adoption of analytics has been slower in industries such as retail, media and telecommunications, momentum is starting to build around Bigdata Analytics.
- Marketing is an area that is mature in terms of adopting analytics. In marketing analytics can be used for marketing campaign management, targeted micromarketing (sending of different offers to different people depending on their likelihood to buy) and market basket analysis which indicates what people buy together.
- In retail, forecasting is key area where analytics is being applied. Customer churn analysis has been used by banks to determine who is likely to cancel their credit card or account. This is the same technique used by telecommunication companies and retailers to determine customer defection. Churn is also a factor used in determining customer lifetime value. Customer lifetime value indicates how much money a firm can make, over the customer with the firm. Companies use their customer lifetime value to segment their customers and determine the customers to focus on.
- The insurance industry uses actuarial models for estimating losses. The emerging trend is to use Monte-Carlo simulations for estimating potential losses. These computationally complex models require large hardware and hence adoption of analytics becomes difficult which has started changing with the advent of parallel and grid computing.
- Another use of Big Data analytics on banks is identifying manipulate behavior or fraudulent activities in real-time so that it can be mitigated or penalized

immediately. For this we have to dig through voluminous transactions and find patterns quickly. A fair trading platform could be created by quickly catching and correcting market manipulating behavior.

III) Big data and Healthcare

- Big data promises enormous revolution in healthcare, with the advancements in everything from the management of chronic disease to the delivery of personalized medicine.
- In addition to saving and improving lives, Big Data has the potential to transform the entire healthcare system by replacing guesswork and intuition with objective data-driven science.
- The healthcare industry now has huge amount of data: from biological data such as gene expression, Special Needs Plans (SNPs), proteomics, metabolomics, and next-generation gene sequence data etc. The exponential growth in data is further accelerated by the digitization of patient level data stored in Electronic Medical Records (EMRs) or Electronic Health Records (EHRs) and Health Information Exchanges (HIEs) enhanced with data from imaging and test results, medical and prescription claims and personal health devices.
- In addition to saving and improving lives, Big Data has the potential to transform the entire health care system by replacing guesswork and intuition with objective, data-driven science (see Figure).

Figure: Data in the World of Healthcare



- The healthcare system is facing severe economic, effectiveness and quality challenges. These factors are forcing transformation in pharmaceutical business model. Hence the healthcare industry is moving from traditional model built on regulatory approval and settling of claims to medical evidence and proving economic effectiveness through improved analytics derived insights. The success of this model depends on the creation of robust analytics capability and harnessing integrated real-world patient level data.

Disruptive analytics

- Data science and disruptive analytics can have immediate beneficial impact on the healthcare systems.
- Data analytics makes it possible to create transparent approach to pharmaceutical decision making based on the aggregation and analysis of healthcare data such as electronic medical records and insurance claims data.
- Creating healthcare analytics framework has significant value for individual stakeholders.
- For providers (physicians), there is an opportunity to build analytics systems for evidence – based medicine(EBM) lifting through clinical and health outcomes data to determine the best clinical protocols that provide the best heath outcomes for patients and create defined standards of care.
- For producers(Pharmaceutical and medical device companies)there is an opportunity to build analytics systems to enable(transactional medicine) integrating externally generated post marketing safety, epidemiology and health outcomes data with internally generated clinical and discovery data (sequencing, expression, biomarkers) to enable improved strategic R&D decision making across the pharmaceutical value chain.
- For payers (ie, insurance companies) there is an opportunity to create analytics systems to enable comparative effectiveness research(CER) that will be used to drive reimbursement by mining large collections of claims, health care records(EMR/EHR), economic and geographic, demographic data sets to determine what treatment and therapies work best for which patients in which context and with what overall economic and outcomes benefit.

A Holistic Value Proposition

- The ability to collect, integrate, analyze and manage data can make health care data such as EHR/EMR, valuable.
- Big data approach to analyze health care data creates methods and platform for analysis of large volumes of disparate kinds of data (Clinical, EMR, Claims, Labs etc.) to better answer questions of outcomes, epidemiology, safety, effectiveness and pharma economic benefit.
- Big data technology and platforms such as Hadoop, R, Open health data etc. help clients create real-world evidence-based approaches to realize solutions for competitive effectiveness research, improve outcomes in complex populations and to improve decision making.

BI is not Data Science

- Traditional Business Intelligence and data warehousing skills do not help in predictive analytics. Like a lawyer who draws a conclusion and then looks for supporting evidence. Traditional BI is declarative and doesn't necessarily require any real domain understanding. Generating automated reports from aging data warehouses that are briefly scanned by senior management does not meet the definition of data science.
- Making data science useful to business is about identifying that question management really tries to answer question.

IV) Pioneering New Frontiers in Medicine

- In Medical Field, Big Data analytics are being used by researches to understand autoimmune disease such as Rheumatoid Arthritis, Diabetes and lupus and neurodegenerative disease such as multi sclerosis Parkinson's and Alzheimer's. In most these cases, the goal is to identify the genetic variations that causes the diseases. The data sets used for such identification contain thousands of genes. For example a research work on the role of environment factors and interactions between environmental factors in multiple sclerosis typically uses data sets that contain 100,000 to 500,000 genetic variations. The algorithms used to identify the interactions between environmental factors and diseases. They also have rapid search techniques built into them and should be able to do statistical analysis and permutation analysis which can be very, very time consuming if not properly done.

Challenges faced by pioneers of quantitative pharmacology

- The data set is very large 1000 by 2000 matrix.
- When an interactive analysis for first order and second order interactions are done each of the 500,000 genetic locations have to be compared to each of all the rest of the 500,000 genetics locations for the first order and this has to be done twice and then 500,000 may reduce to a third for the second order interaction and so on. Basically a second order interaction would be 500,000 squared, a third order would be 500,000 cubed and so on. Such huge computations are made possible in little time with the aid of big data technologies.

V) Advertising and Big Data

Big Data is changing the way advertisers address three related needs.

(i) How much to spend on advertisements.
(ii) How to allocate amount across all the marketing communication touch points.
(iii) How to optimize advertising effectiveness.

Given these needs advertisers need to measure their advertising end to end in terms of Reach, Resonance & Reaction.

Reach, Resonance, and Reaction

Reach: First part of reach is to identify the people who are most volumetrically responsive to their advertising and then answer questions such as what do those people watch? What do they do online? How to develop media plan against intended audience. The second part of reach is delivering advertisements to the right audience. That is, to understand if we are actually reaching our desired audience. If we think about the online world, it's a world where we can deliver 100 million impressions but we never really know for sure who our campaign was actually delivered to. If our intended audience is women aged 18 to 35, out of our 100 million impressions, what percentage of impressions were actually delivered to the intended audience? What was the reach, what was the frequency, what was the delivery against the intended audience?

Resonance: If we know whom we want to reach and we're reaching them efficiently with our media spend, the next question is, are our ads breaking through? Do people know they're from our brand? Are they changing attitudes? Are they making consumers more likely to want to buy our brand? This is what is called ―resonance.‖

Reaction: Advertising must drive a behavioral ―reaction‖ or it isn't really working. We have to measure the actual behavioral impact. Whether we've identified the

highest potential audience, reached them efficiently with our media plan, delivered ads that broke through the clutter and increased their interest in buying our brand—did it actually result in a purchase? Did people actually buy our product or service based on exposure to our advertising?
The three guiding principles to measurement are
1. End to end measurement—reach, resonance and reaction
2. Across platforms (TV, digital, print, mobile, etc.)
3. Measured in real-time (when possible)

The Need to Act Quickly (Real-Time When Possible)

When we start executing a campaign, how do we know on a daily basis whether our advertising campaign is actually being delivered to our intended audience the way it's supposed to?
For example, in digital, ad performance will differ across websites. Certain websites are really good; certain websites are really bad. How do we optimize across sites? - By moving money out of weak performing sites into better performing sites.

Real time optimization:
When new ad campaign is launched, it is good if the ad is break thru and is highly memorable and it is bad news if the consumers think the ad is for the key competitor.

If out of three ads aired, two have high breakthrough but one is weak, the weak performing ad could be quickly taken off air and the media spend can be rotated to the higher performing ads. This will make breakthrough scores go up.

Instead of 30 second ads, a mix of 15s and 30s ads, can be planned, suppose real time data shows that 15s ads work as well as 30s ads. Instead of spending money on 30s ads, all money can be spent on 15-second ads and scores will continue to grow.

The measurement tools and capabilities are enabling real-time optimization on this and so there's a catch-up happening both in terms of advertising systems and processes, but the industry infrastructure must be able to actually enable all of this real-time optimization.

Measurement can be tricky

There are tools that allow us to tag digital advertising through a panel, we can read those people who were exposed to the advertising and those who were not and measure their actual offline purchase behavior.
A company doing this for a large beer client could see that this campaign generated (after the fact) a 20 percent sales increase among consumers exposed versus not exposed to the advertising. The average person would look at that and think that the advertising is working. But when the reach for this particular client was analyzed, their intended audience was males, aged 21–29. Of their 100 million delivered impressions, only about 40 million were actually delivered to males aged 21–29. Sixty million went to someone other than their intended audience; some went to kids (not good for a beer brand); some went to people 65+. It would have been good if instead of 40% of the impressions hitting the intended audience, 70 or 80% of the impressions had hit them. When the 40 percent of impressions that hit the intended audience were analyzed, the reach and frequency of those was 10 percent reach 65 frequency. In

other words, they only hit about 10 percent of their intended audience, but each of these people was bombarded with, on average, 65 ads. That's not quite the optimization one would hope for. There's a lot of science in advertising that shows that by maximizing reach and minimizing frequency, we can get best response. If the measuring of all of this were in real time, the plan could have been quickly adjusted to increase delivery to the intended audience, increase reach, and reduce frequency.

Content Delivery Matters

The ads were on twelve websites: four of the ads didn't perform well in those sites. The other ones were really good. If they had measured that in flight, they could have moved spending out of the bad performing sites, into good performing sites, and further improved results. End-to-end measurement is important. Reach times resonance equals reaction. Measuring the sales impact alone is great, but it's not enough. Sales impact could be great and still be completely non-optimized on the reach and resonance factors that cause the reaction.

Optimization and Marketing Mixed Modeling

Marketing Mixed Modeling (MMM) is a tool that helps advertisers understand the impact of their advertising and other marketing activities on sales results. MMM can generally provide a solid understanding of the relative performance of advertising by medium (e.g., TV, digital, print, etc.), and in some cases can even measure sales performance by creative unit, program genre, website, and so on.

Now, the impact on sales in social media can be measured through market mixed modeling. Market mixed modeling is a way that can take all the different variables in the marketing mix—including paid, owned, and earned media—and uses them as independent variables that regress against sales data and tries to understand the single variable impact of all these different things.

Since these methods are quite advanced, organizations use high-end internal analytic talent and advanced analytics platforms such as SAS or point solutions such as Unica and Omniture. Alternatively, there are several large analytics providers like Mu Sigma that supply it as a software-as-a-service (SaaS).

As the world becomes more digital, the quantity and quality of marketing data is improving, which is leading to more granular and insightful MMM analyses.

The Three Big Data Vs in Advertising

Impact of the three Vs (volume, velocity, and variety) in advertising:

Volume

The volume of information and data that is available to the advertiser has gone up exponentially versus what it was 20 years ago. In the old days, we would copy test our advertising. The agency would build a media plan demographically targeted and we'd execute it. Maybe 6 to 12 months later, we'd try to use whatever sales data we had to try to understand if there was any impact. In today's world, there is hugely more advertising effectiveness data. On TV advertising, we can measure every ad in every TV show every day, across about 70 percent of the viewing audience. We measure clients digital ad performance hourly—by ad, by site, by exposure, and by audience. On a daily or weekly basis, an advertiser can look at their advertising performance.

Velocity
There are already companies that will automate and optimize advertising on the web without any human intervention at all based on click-thru. It's now beginning to happen on metrics like breakthrough, branding, purchase intent etc. This is sometimes called programmatic buying. Literally, we'll have systems in place that will be measuring the impact of the advertising across websites or different placements within websites, figuring out where the advertising is performing best. It will be automated optimization and reallocation happening in real-time. The volume and the velocity of data, the pace at which we can get the data, make decisions and do things about it is dramatically increased.

Variety
Before, we really didn't have a lot of data about how our advertising was performing in market. We have a lot more data and it's a lot more granular. We can look at our brand's overall advertising performance in the market. But we can also decompose it to how much of a performance is because of the creative quality, the media weight, how much is because of the program that the ads sit in. How much is because of the placement: time of day, time of year, pod position, how much is because of cross-platform exposure, how much is because of competitive activity. Then we have the ability to optimize on most of those things—in real time. And now we can also measure earned (social) and owned media. Those are all things that weren't even measured before.

Using Consumer Products as a Doorway

Over the last year, RIM's Research in Motion (RIM), the maker of Blackberry share price has plunged 75 percent. The company once commanded more than half of the American smartphone market. Today it has 10 percent. Companies with deep pockets and resources such as Hewlett Packard (HP) failed to enter the tablet market, while Apple is selling iPads in its sleep.

Apple entered into the mobile and tablet market because of the iPod, which crushed giants like Sony in the MP3 market. For Apple, the market was not just about selling hardware or music on iTunes. It gave them a chance to get as close to a consumer as anyone can possibly get. This close interaction also generated a lot of data that help them expand and capture new customers. Again it's all about the data, analytics, and putting it into action.

Google gives away product that other companies, such as Microsoft, license for the same the reason. It also began playing in the mobile hardware space through the development of the Android platform and the acquisition of Motorola. It's all about gathering consumer data and monetizing the data. With Google Dashboard we can see every search we did, e-mails we sent, IM messages, web-based phone calls, documents we viewed, and so on. This is powerful for marketers.

The online retailer Amazon has created new hardware with Kindle and Barnes and Noble released the Nook. If both companies know every move we make, what we download, what we search for, they can study our behaviors to present new products that they believe will appeal to us. The connection with consumers and more importantly taking action on the derived data is important to win.

BIG DATA TECHNOLOGY

- Hadoop's Parallel World
  - Hadoop Distributed File System(HDFS)
  - Map Reduce
- Old Vs New Approaches
- Data Discovery
- Open Source Technology for Big Data Analytics
- The cloud and Big Data
- Predictive Analytics
- Software as a Service BI
- Mobile Business Intilligence
- Crowdsourcing Analytics
- Inter and Trans Firewall Analytics
- R & D Approach

- **Hadoop's parallel World(Creators-Doug cutting was at yahoo now at cloudera, Mike Cafarella-teaching at University of Micigan)**
  - Hadoop is an open-source platform for storage and processing of diverse data types that enables data-driven enterprises to rapidly derive value from all their data.

Advantages of Hadoop:

- The scalability and elasticity of free open-source Hadoop running on standard hardware allow organizations to hold onto more data and take advantage of all their data to increase operational efficiency and gain competitive edge. Hadoop supports complex analyses across large collections of data at one tenth the cost of traditional solutions.
- Hadoop handles a variety of workloads, including search, log processing, recommendations systems, data warehousing and video/image analysis.
- Apache Hadoop is an open-source project by the Apache Software foundations. The software was originally developed to capture and analyze the data that they generate. Unlike traditional, structured platforms.
- Hadoop is able to store any kind of data in its native format and perform a wide variety of analyses and transformation on that data.
- Hadoop stores terabytes and even peta bytes of data inexpensively.
- It is robust and reliable and handles hardware and system failures automatically without losing data analyses.
- Hadoop runs on clusters of commodity servers and each of those servers has local CPUs and disk storage.

Components of Hadoop:

The two critical components of Hadoop are

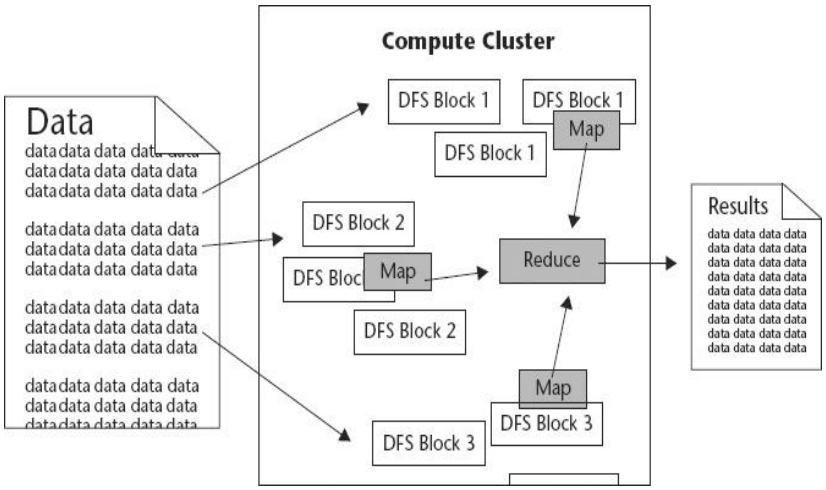
1) The Hadoop Distributed File System (HDFS)

- HDFS is the storage system for a cluster.
- When data lands in the cluster, HDFS breaks it into pieces and distributes those pieces among the different servers participating in the cluster.
- Each server stores just a small fragment of the complete data set and each piece of data is replicated on more than one server.

2) Map Reduce:

- Because Hadoop stores the entire dataset in small pieces across a collection of servers, analytical jobs can be distributed in parallel to each of the servers storing part of the data. Each server evaluates the question against its local fragment simultaneously and reports its result back for collation into a comprehensive answer.
- Map Reduce is the agent that distributes the work and collects the results.
- Both HDFS and Map Reduce are designed to continue to work even if there are failures.
- HDFS continuously monitors the data stored on the cluster. If a server becomes unavailable, a disk drive fails or data is damaged due to hardware or software problems, HDFS automatically restores the data from one of the known good replicas stored elsewhere on the cluster.
- Map Reduce monitors the progress of each of the servers participating in the job, when an analysis job is running. If one of them is slow in returning an answer or fails before completing its work, Map Reduce automatically starts another instance of the task on another server that has a copy of the data.
- Because of the way that HDFS and Map Reduce work, Hadoop provides scalable, reliable and fault-tolerant services for data storage and analysis at very low cost.

Figure-DFS & Map Reduce



➢ Old Vs New approaches to Data Analytics

Old Approach (Database approach)	New Approach (Big data Analytics)
Follows data and analytics technology stack with different layers of cross-communicating data and works on ―scale-up‖ expensive hardware.	Follows data and analytics platform that does all the data processing and analytics in one layer without moving data back and forth on cheap but scalable (―scale-out‖) commodity hardware.
Data is moved to places where they have to be processed.	Data must be processed and converted into usable business intelligence where it sits.
Massive parallel processing was not employed due to hardware and storage limitations.	Hardware and storage is affordable and continuing to get cheaper to enable massive parallel processing.
Due to technological limitations storing, managing and analyzing massive data sets were difficult.	New proprietary technologies and open source inventions enable different approaches that make it easier and more affordable to store, manage & analyze data.
Not able to handle unstructured data.	The variety of data and ability to handle unstructured data is on the rise. Big data approach provides solution to this.

➢ Data Discovery

- Data discovery is the term used to describe the new wave of business intelligence that enables users to explore data, make discoveries and uncover insights in a dynamic and intuitive way versus predefined queries and preconfigured drill-down dashboards. This approach is being followed by many business users due to its freedom and flexibility to view Big Data. There are two software companies that stand out in the crowd by growing their businesses at unprecedented rates in this space: Tableau Software and QlikTech International.
- Both companies' approach to the market is much different than the traditional BI software vendor. They used a sales model referred to as ―land and expand‖. This model was based on the fact that analytics and reporting are produced by the people using the results. The model enabled business people to create their own reports and dashboards.
- The most important characteristic of rapid-fire BI is that business users, not specialized developers, drive the applications. The result is that everyone wins. The IT team can stop the backlog of change requests and instead spend time on strategic IT issues. Users can serve themselves data and reports when needed.
- There is a simple example of powerful visualization. A company uses an interactive dashboard to track the critical metrics driving their business. Every day, the CEO and other executives are plugged in real-time to see how their

markets are performing in terms of sales and profit, what the service quality scores look like against advertising investments, and how products are performing in terms of revenue and profit. Interactivity is key: a click on any filter lets the executive look into specific markets or products. She can click on any data point in any one view to show the related data in the other views. She can look into any unusual pattern or outlier by showing details on demand. Or she can click through the underlying information in a split-second.

- Business intelligence needs to work the way people's minds work. Users need to navigate and interact with data any way they want to—asking and answering questions on their own and in big groups or teams.
- Qliktech has designed in a way for users to leverage direct—and indirect—search. With QlikView search, users type relevant words or phrases in any order and get instant, associative results. With a global search bar, users can search across the entire data set. With search boxes on individual list boxes, users can confine the search to just that field. Users can conduct both direct and indirect searches. For example, if a user wanted to identify a sales rep but couldn't remember the sales rep's name—just details about the person, such as that he sells fish to customers in the Nordic region—the user could search on the sales rep list box for ―Nordic‖ and ―fish‖ to narrow the search results to just the people who meet those criteria.

➢ Open Source Technology for Big Data Analytics

- Open- source software is computer software that is available in source code form under an open- source license that permits users to study, change, improve and distribute the software.
- Hadoop is a open- source project.
- One of the key attributes of open- source projects is that it is not constrained by someone else's predetermined ideas or vision which makes it flexible, extensible and low cost.
- One disadvantage of open- source is that it has to coexist with the proprietary solution for a long time for many reasons. For example getting data from Hadoop to a database required a Hadoop expert in the middle to do the data cleansing and the data type translation. If the data was not 100% (clean which is the case with most circumstances) a developer was needed to get it to a consistent, proper form. Besides wasting the valuable time of that expert, this process meant that business analysts couldn't directly access and analyze data in Hadoop clusters. SQL-H is software that is developed to solve this problem.

➢ The Cloud and Big Data

- Market economics are demanding that capital-intensive infrastructure costs disappear and business challenges are forcing clients to consider newer models. The cloud-deployment model satisfies such needs. With a cloud model, payment is on subscription basis with no capital expense. Typical 30% maintenance fees are not incurred and all the updates on the platform are automatically available.
- The traditional cost of value chains is completely eliminated by massively scalable platforms (such as cloud) where marginal cost to deliver an incremental

product/service is zero. Whether a private hosted model or a publicly shared one, the true value lies in delivering software, data and/or analytics in an ―as a service' model.

- **Predictive Analytics**

  - Enterprises will move from being in reactive positions (Business Intelligence) to forward learning positions (Predictive analysis). Using all the data available i.e traditional internal data sources combined with new rich external data sources will make the predictions more accurate and meaningful. Algorithm trading and supply chain optimizations are two examples where predictive analytics have greatly reduced the friction in business. Predictive analytics proliferate in every facet of our lives both personal and business. Some of the leading trends in business today are

  - Recommendation engines similar to those used in Netflix, Amazon that use past purchases and buying behavior to recommend new purchases.
  - Risk engines for a wide variety of business areas, including market and credit risk, catastrophic risk and portfolio risk.
  - Innovation engines for new product innovation, drug discovery and consumer and fashion trends to predict new product formulations and new purchases.
  - Consumer insight engines that integrate a wide variety of consumer-related information including sentiment, behavior and emotions.
  - Optimization engines that optimize complex interrelated operations and decisions that are too complex to handle.

- **Software as a Service Business Intelligence**

  - The basic principle is to make it easy for companies to gain access to solutions without building and maintaining their own onsite implementation.
  - SaaS is less expensive.
  - The solutions are typically sold by vendors on a subscription or pay-as-you-go basis instead of the more traditional software licensing model with annual maintenance fees.
  - SaaS BI can be a good choice when there is little or no budget money available for buying BI software and related hardware. Because they don't involve upfront purchase costs or additional staffing requirements needed to manage the BI system, total cost of ownership may be lower than that of on-premise software.
  - Another common buying factor for SaaS is the immediate access to talent especially in the world of information management, BI and predictive analytics.
  - Omniture (now owned by adobe) is a successful SaaS BI. Omniture's success was due to its ability to handle big data in the form of web log data.

  Omniture's success was also due to the following reasons:

  - **Scaling the Saas delivery model**
    Omniture was built from the ground up to be SaaS. It made use of concept called magic number that helps analyze and understand SaaS business.
  - **Killer sales organizations**
    Omniture's well known customers like HP, eBay and Gannet were sale organizations.

- **A focus on customer success**
  Unlike traditional enterprise software, with SaaS business, it is easy for customers to leave if they are not satisfied. Today's BI is not designed for the end-user. It must be designed to be more intuitive, easily accessible, real time and meet the expectations of today's customer technology who expect a much more connected experience.

➢ Mobile Business Intelligence

- Simplicity and ease of use had been the major barriers to BI adoption. But mobile devices have made complicated actions to be performed very easily. For example, a young child can use an ipad or iphone easily but not a laptop. This ease of use will drive the wide adoption of mobile BI.
- Multi touch and software oriented devices have brought mobile analytics and intelligence to a much wider audience.
- Ease of mobile application development and development have also contributed to the wide adoption of mobile BI.

Three elements that have impacted the viability of mobile BI are

i) Location-GPS component enables finding location easy.
ii) Transaction can be done through smart phones.
iii) Multimedia functionality allows virtualization.

Three challenges with mobile BI include

i) Managing standards for these devices.
ii) Managing security (always a big challenge).
iii) Managing ―bring your own device‖, where you have devices both owned by the company and devices owned by the individual, both contributing to productivity.

➢ Crowdsourcing Analytics

- Crowdsourcing is the recognition that organizations can't always have the best and brightest internal people to solve all their big problems. By creating an open, competitive environment with clear rules and goals, problems can be solved.
- In October 2006, Netflix an online DVD rental business announced a contest to create a new predictive model for recommending movies based on past user ratings. The grand prize was $1,000,000. Netflix already had an algorithm to solve the problem but thought there was an opportunity to improve the model which would turnout huge revenues.
- Kaggle is an Australian firm that provides innovative solutions for statistical analytics for outsourcing. Kaggle manages competitions among world's best data scientist corporations, governments and research laboratories. Organizations that confront complex statistical challenges describe the problems to kaggle and provide data sets. Kaggle converts the problems and the data into contests that are posted on its website. The contest features cash prizes ranging in values from $100 to $3 million. Kaggle's clients range in size from tiny start-ups to Multinational Corporations such as Ford Motor Company and government agencies such as NASA.
- The idea is that someone comes to Kaggle with a problem, they put it up on their website and then people from all over the world can compete to see who can

produce the best solution. In essence Kaggle has developed an effective global platform for crowdsourcing complex analytic problems.

- There are various types of crowdsourcing such as crowd voting, crowd purchasing, wisdom of crowds, crowd funding and contests.
- Example:
  - ❖ 99designs.com/, does crowdsourcing of graphic design.
  - ❖ Agentanything.com/, posts missions where agents are invited to do various jobs.
  - ❖ 33needs.com/, allows people to contribute to charitable programs to make social impact.

➢ **Inter and Trans-Firewall Analytics**

- Yesterday companies were doing functional silo-based analytics. Today they are doing intra-firewall analytics with data within the firewall. Tomorrow they will be collaborating on insights with other companies to do inter-firewall analytics as well as leveraging the public domain spaces to do trans-firewall analytics (Fig.1).

- As fig.2 depicts, setting up inter-firewall and trans-firewall analytics can add significant value. But this presents some challenges. When information is collected outside the firewall, the information to noise ratio increases putting additional requirements on analytical methods and technology requirements.

- Further, organizations are limited by a fear of collaboration and overreliance on proprietary information. The fear of collaboration is driven by competitive fears, data privacy concerns and proprietary orientations that limit opportunities for cross-organizational learning and innovations. The transition to an inter-firewall and trans-firewall paradigm may not be easy but it continues to grow and will become a key weapon for decisions scientists to drive disruptive value and efficiencies.

Figure 1

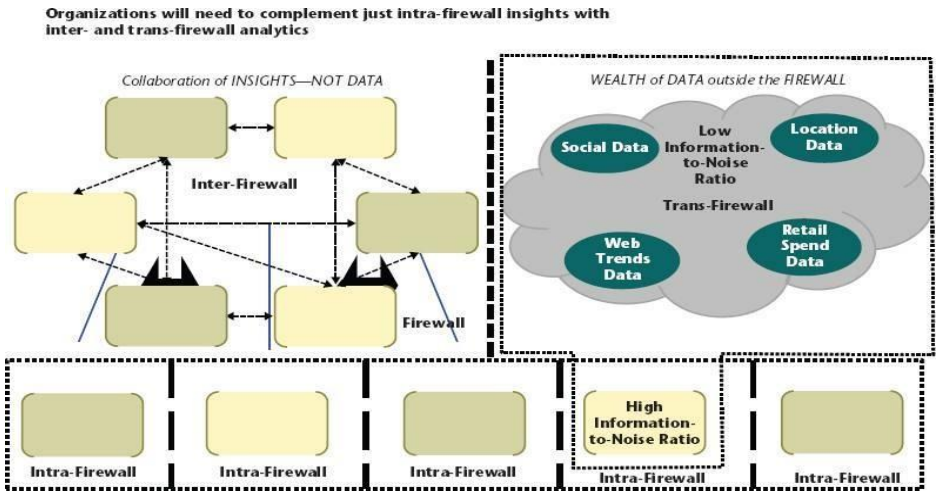
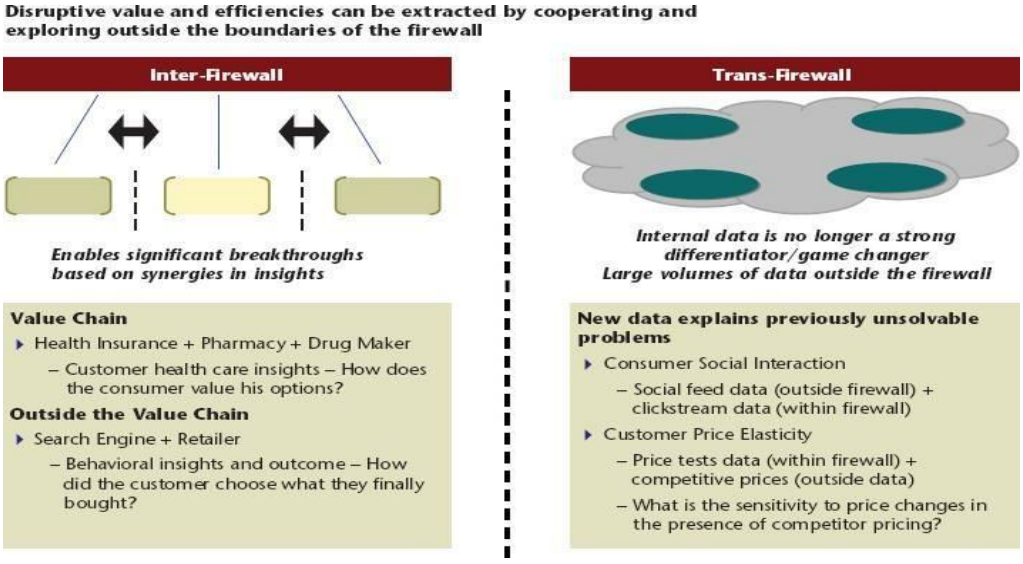


Figure 2



- **R & D Approach helps adapt new technology**

- Business analytics can certainly help a company to embrace innovation and direction by leveraging critical data and acting on the results. For example, a market research executive analyzes customers and market data to anticipate new product features that will attract customers.

- For many reasons, organizations find it hard to make changes after spending many years implementing a data management, BI and analytic stack. So the organizations have to do lot of research and development on the new technologies before completely adopting the technologies to minimize the risk. The two core programs that have to focused by R & D teams are

Program	Goal	Core Elements
Innovation management	Tap into the latent creativity of all Visa employees, providing them with a platform to demonstrate mastery and engage collaboratively with their colleagues.	Employee personal growth
		Employee acquisition and retention
Research and open innovation	Look outside of the company and scan the environment for trends, new technology, and approaches	Innovation
		Competitive advantage

Adding Big Data Technology

The process that enterprises must follow to get started with the big data technology

1. Practical approach – Start with the problem and then find a solution.

2. Opportunistic Approach – Start with the technology and then find a home for it.

For both the approaches the following activities have to be conducted,

(i) **Play** – R & D team members may request to install their lab to get more familiar with the technology.

(ii) **Initial Business review** – Talk with the business owner to validate the applicability and rank, the priorities to ensure that it is worth pursuing.

(iii) **Architecture Review** – Asses the validity of the underlying architecture and ensure that it maps to IT's standards.

(iv) **Pilot use cases** – find the use case to test the technology out.

(v) **Transfer from R & D to Production** – Negotiate internally regarding what it would take to move it from research to production using the following table.

Who needs to be involved in this process?

Operations	Nonfunctional integration	How does this integrate with our existing programs? (Monitoring, security, intrusion, etc.)
Engineering	Readiness	What do we need to do to prepare from a skill and infrastructure perspective to take on this project?
Application development	Functional requirements	How does this new technology map to the business functional requirements?
Business users	Derived value	How can new technology solve business problems I have or anticipate? How can I find new business opportunities?

Organizations may have a lot of smart people, but there are other smart people outside. Organizations need to be exposed to the value they are creating. A systematic program that formalizes relationships with a powerful ecosystem is shown in the following table.

Innovation ecosystem: Leveraging brain power from outside of the organization

Source	**Example**
Academic community	Tap into a major university who did a major study on social network analytics.
Vendors' research arms	Leverage research a vendor completed in their labs demonstrating success leveraging unstructured data.
Research houses	Use research content to support a given hypothesis for a new endeavor.
Government agencies	Discuss fraud strategies with the intelligence community.
Venture capital organizations	Have a venture capital firm review some new trends they are tracking and investing in.
Start-ups	Invite BI and analytic technology start-ups in instead of just sticking with the usual suspects.

UNIT II INTRODUCTION TO NOSQL

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. Unlike traditional relational databases that use tables with pre- de跨椀ned schemas to store data, NoSQL databases use 跨黭exible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

The term NoSQL originally referred to "non-SQL" or "non-relational" databases, but the term has since evolved to mean "not only SQL," as NoSQL databases have expanded to include a wide range of different database architectures and data models.

Types of NoSQL database

There are multiple types of NoSQL databases. 4 of the most common NoSQL databases are:

1. **Document databases:** A collection of documents, where each document is JSON or JSON-like format. Each document contains pairs of 跨椀elds and values. The primary storage is in the storage layer and we cache it out to memory. **Examples** – MongoDB, CouchDB, Cloudant
2. **Key-value stores:** Key-value stores; similar to Python dictionaries. Query either by using the key or search through the entire database. The key-value stores tend to be used in memory and use a backing store behind it. **Examples** – Memcached, Redis, Coherence
3. **Wide column databases:** Similar to relational database tables; the difference is the storage on the backend is different. We can put SQL on top of a wide column database, which makes it very similar to querying a relational database **Examples** – Hbase, Big Table, Accumulo

Graph databases: Stores data as nodes (vertices) and relationships (edges). Vertices typically store object information while edges represent the relationships between nodes. We can have a SQL-like query language in our graph databases. **Examples**
– Amazon Neptune, Neo4j

- NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages. A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines **has** and 跨椀ner control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve.

- NoSQL databases, also known as "not only SQL" databases, are a new type of database management system that has**,** gained popularity in recent years. Unlike traditional relational databases, NoSQL databases are designed to handle large amounts of unstructured or semi-structured data, and they can accommodate dynamic changes to the data model. This makes NoSQL databases a good 跨椀t for modern web applications, real-time analytics, and big data processing.

- Data structures used by NoSQL databases are sometimes also viewed as more 昀敿exible than relational database tables. Many NoSQL stores compromise consistency in favor of availability, speed,, and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases.

- Most NoSQL stores lack true ACID(Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic,

  Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.

- Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also, **has** some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.

- One simple example of a NoSQL database is a document database. In a document database, data is stored in documents rather than tables. Each document can contain a different set of 昀椀elds, making it easy to accommodate changing data requirements

- For example, “Take, for instance, a database that holds data regarding employees.”. In a relational database, this information might be stored in tables, with one table for employee information and another table for department information. In a document database, each employee would be stored as a separate document, with all of their information contained within the document.

- NoSQL databases are a relatively new type of database management system that has **a** gained popularity in recent years due to their scalability and 昀敿exibility. They are designed to handle large amounts of unstructured or semi-structured data and can handle dynamic changes to the data model. This makes NoSQL databases a good 昀椀t for modern web applications, real-time analytics, and big data processing.

Key Features of NoSQL:

1. **Dynamic schema:** NoSQL databases do not have a 昀椀xed schema and can accommodate changing data structures without the need for migrations or schema alterations.

2. **Horizontal scalability:** NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well- suited for handling large amounts of data and high levels of tra昀케c.

3. **Document-based:** Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in a **scales** semi-structured format, such as JSON or BSON.

4. **Key-value-based:** Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key- value pairs.
5. **Column-based:** Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.

AGGREGATE DATA MODELS

Aggregate means a collection of objects that are treated as a unit. In NoSQL Databases, an aggregate is a collection of data that interact as a unit. Moreover, these units of data or aggregates of data form the boundaries for the ACID operations.

Aggregate Data Models in NoSQL make it easier for the Databases to manage data storage over the clusters as the aggregate data or unit can now reside on any of the machines. Whenever data is retrieved from the Database all the data comes along with the Aggregate Data Models in NoSQL.

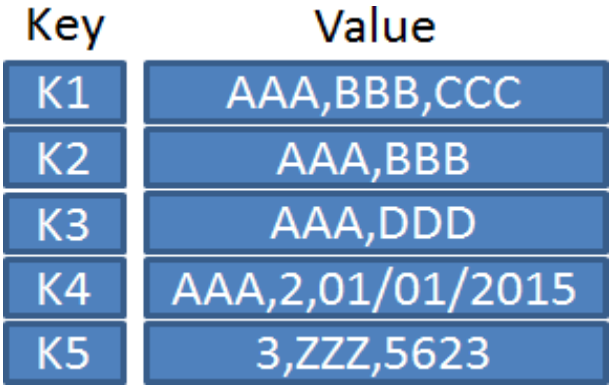
Aggregate Data Models in NoSQL don't support ACID transactions and sacri的椀ce one of the ACID properties. With the help of Aggregate Data Models in NoSQL, you can easily perform OLAP operations on the Database.

You can achieve high e的刁ciency of the Aggregate Data Models in the NoSQL Database if the data transactions and interactions take place within the same aggregate.

Types of Aggregate Data Models in NoSQL Databases

The Aggregate Data Models in NoSQL are majorly classi的椀ed into 4 Data Models listed below:

Key-Value Model



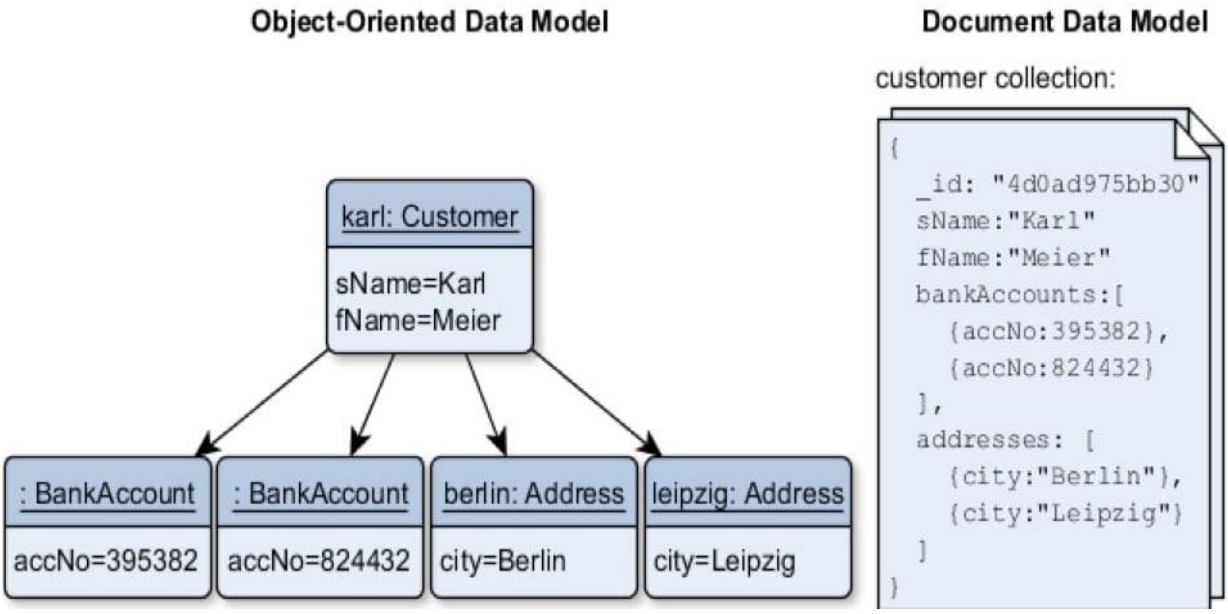
The Key-Value Data Model contains the key or an ID used to access or fetch the data of the aggregates corresponding to the key. In this Aggregate Data Models in NoSQL, the data of the aggregates are secure and encrypted and can be decrypted with a Key.

Use Cases:

- These Aggregate Data Models in NoSQL Database are used for storing the user session data.

- Key Value-based Data Models are used for maintaining schema-less user pro的椀les.
- It is used for storing user preferences and shopping cart data.

Document Model

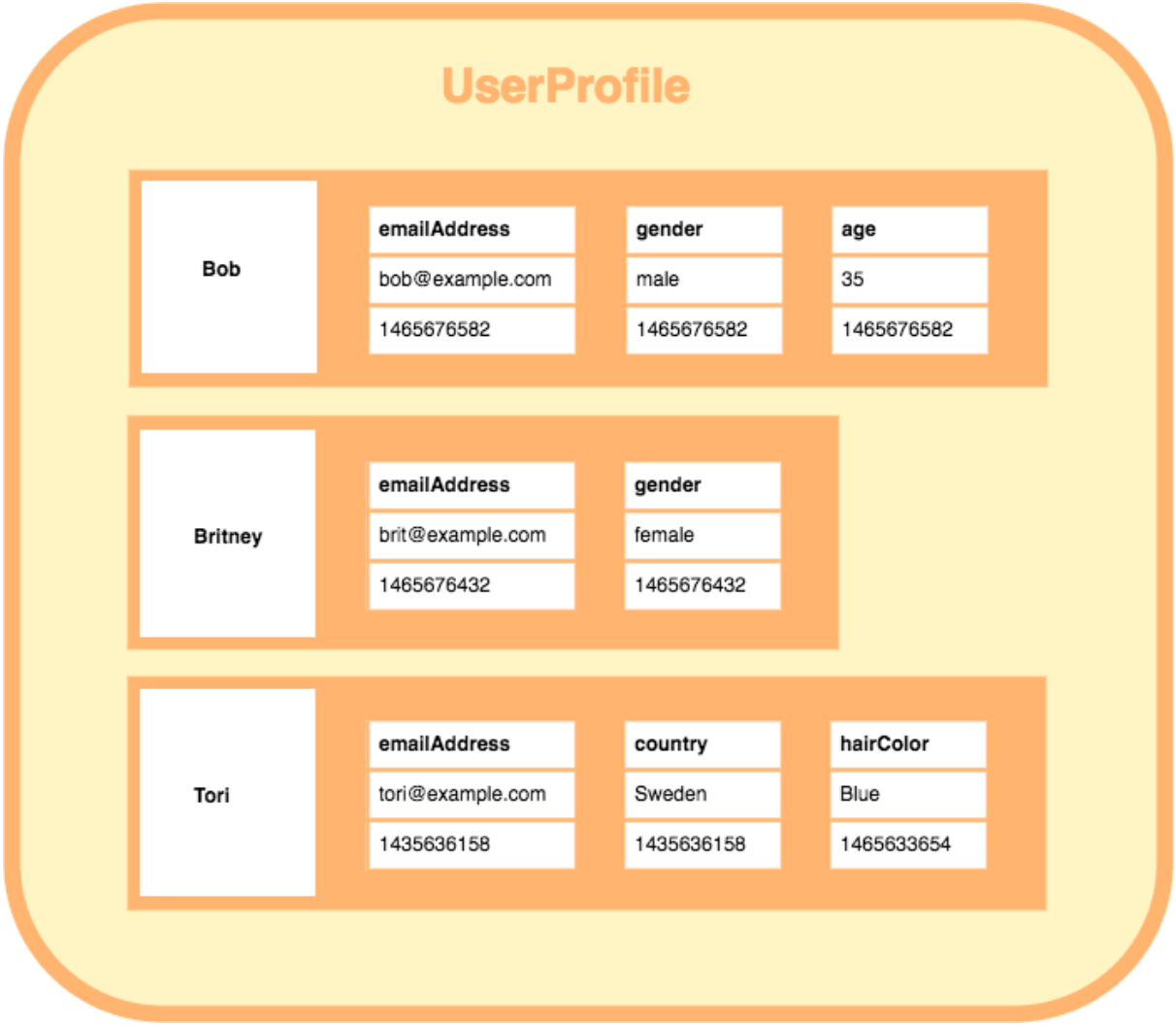


The Document Data Model allows access to the parts of aggregates. In this Aggregate Data Models in NoSQL, the data can be accessed in an in的龘exible way. The Database stores and retrieves documents, which can be XML, JSON, BSON, etc. There are some restrictions on data structure and data types of the data aggregates that are to be used in this Aggregate Data Models in NoSQL Database.

Use Cases:

- Document Data Models are widely used in E-Commerce platforms
- It is used for storing data from content management systems.
- Document Data Models are well suited for Blogging and Analytics platforms.

Column Family Model

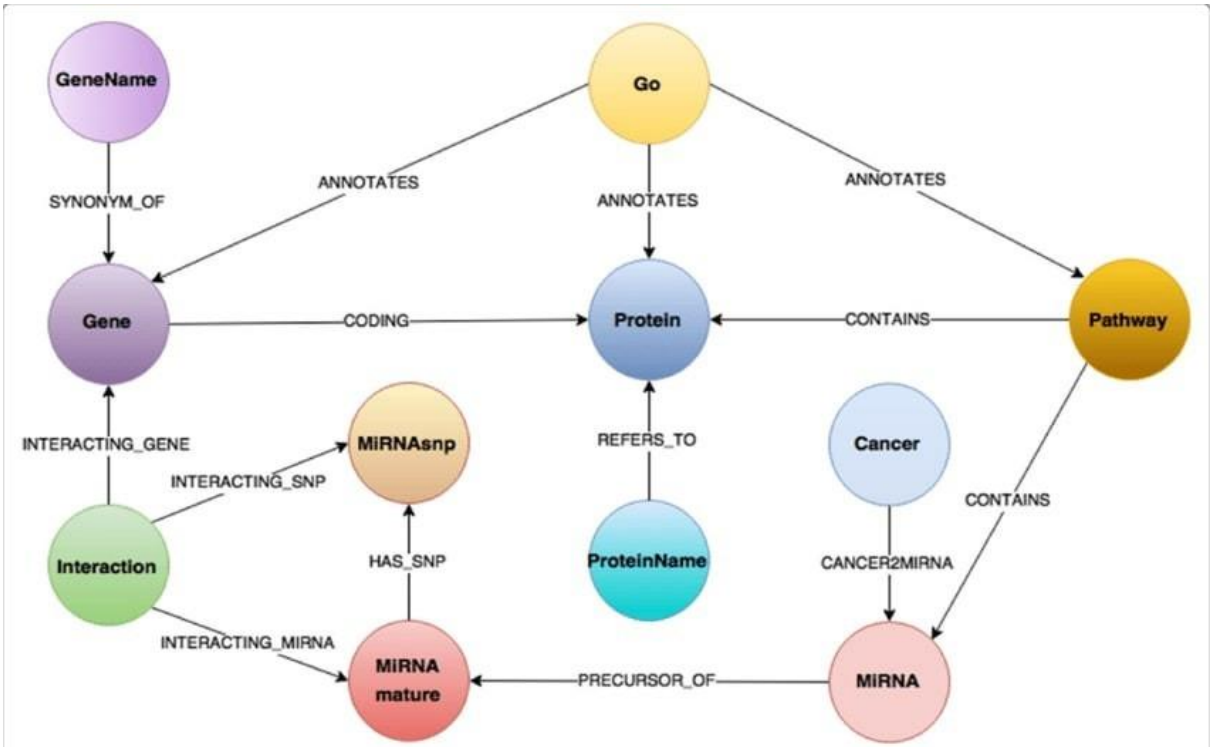


Column family is an Aggregate Data Models in NoSQL Database usually with big-table style Data Models that are referred to as column stores. It is also called a two-level map as it offers a two-level aggregate structure. In this Aggregate Data Models in NoSQL, the 肭椀rst level of the Column family contains the keys that act as a row identi肭椀er that is used to select the aggregate data. Whereas the second level values are referred to as columns.

Use Cases:

- Column Family Data Models are used in systems that maintain counters.
- These Aggregate Data Models in NoSQL are used for services that have expiring usage.
- It is used in systems that have heavy write requests.

Graph-Based Model



Graph-based data models store data in nodes that are connected by edges. These Aggregate Data Models in NoSQL are widely used for storing the huge volumes of complex aggregates and multidimensional data having many interconnections between them.

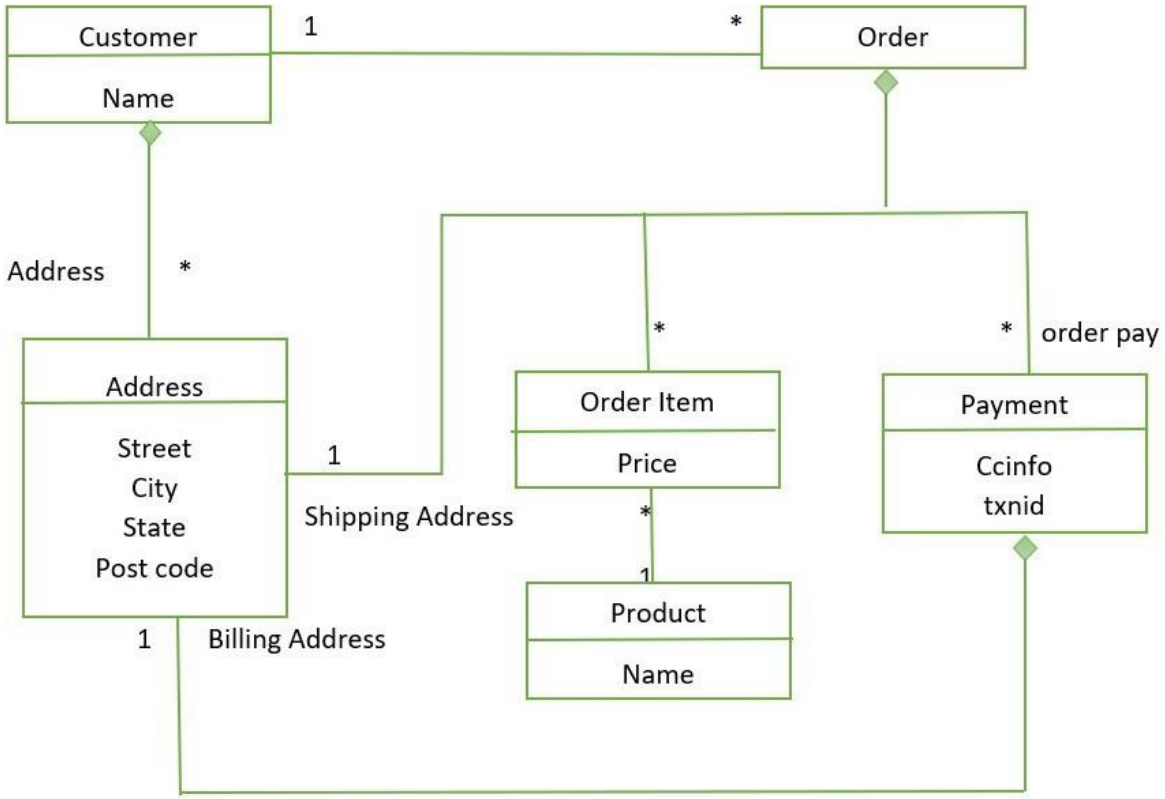
Use Cases:

- Graph-based Data Models are used in social networking sites to store interconnections.
- It is used in fraud detection systems.
- This Data Model is also widely used in Networks and IT operations.

Steps to Build Aggregate Data Models in NoSQL Databases

Now that you have a brief knowledge of Aggregate Data Models in NoSQL Database. In this section, you will go through an example to understand how to design Aggregate Data Models in NoSQL. For this, a Data Model of an E-Commerce website will be used to explain Aggregate Data Models in NoSQL.

This example of the E-Commerce Data Model has two main aggregates – customer and order. The customer contains data related to billing addresses while the order aggregate consists of ordered items, shipping addresses, and payments. The payment also contains the billing address.



Here in the diagram have two Aggregate:

- Customer and Orders link between them represent an aggregate.
- The diamond shows how data 跨椀t into the aggregate structure.
- Customer contains a list of billing address
- Payment also contains the billing address
- The address appears three times and it is copied each time
- The domain is 跨椀t where we don't want to change shipping and billing address.

If you notice a single logical address record appears 3 times in the data, but its value is copied each time wherever used. The whole address can be copied into an aggregate as needed. There is no pre-de跨椀ned format to draw the aggregate boundaries. It solely depends on whether you want to manipulate the data as per your requirements.

The Data Model for customer and order would look like this.

```
// in customers
{
"customer": {
"id": 1,
"name": "Martin",
"billingAddress": [{"city": "Chicago"}],
"orders": [
 {
 "id":99,
```

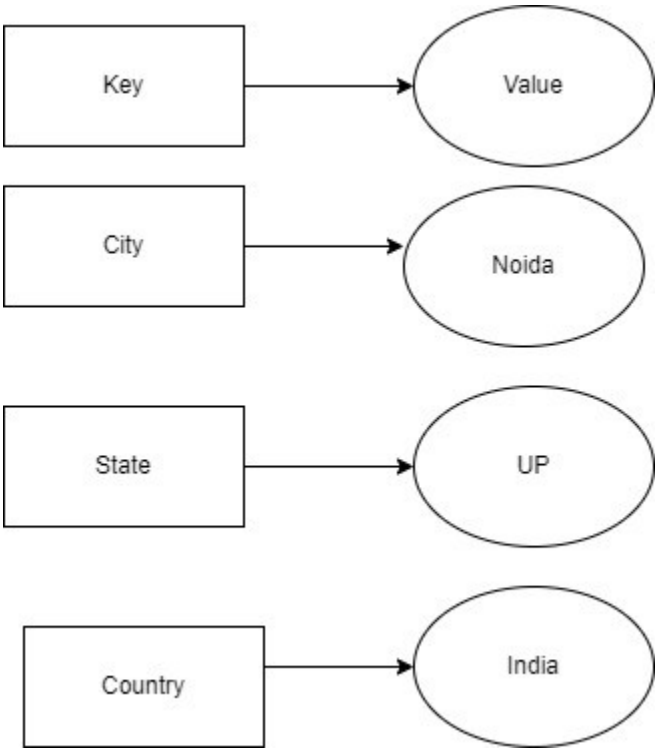
```
"customerId":1,
"orderItems":[
{
"productId":27, "price":
32.45,
"productName": "NoSQL Distilled"
}
],
"shippingAddress":[{"city":"Chicago"}]

"orderPayment":[
{
"ccinfo":"1000-1000-1000-1000",
"txnId":"abelif879rft", "billingAddress":
{"city": "Chicago"}
}],
}]
}
}
```

In these Aggregate Data Models in NoSQL, if you want to access a customer along with all customer's orders at once. Then designing a single aggregate is preferable. But if you want to access a single order at a time, then you should have separate aggregates for each order. It is very content-speci的椀c.

KEY-VALUE DATA MODEL

A key-value data model or database is also referred to as a key-value store. It is a non-relational type of database. In this, an associative array is used as a basic database in which an individual key is linked with just one value in a collection. For the values, keys are special identi的椀ers. Any kind of entity can be valued. The collection of key-value pairs stored on separate records is called key-value databases and they do not have an already de的椀ned structure.



How do key-value databases work?

A number of easy strings or even a complicated entity are referred to as a value that is associated with a key by a key-value database, which is utilized to monitor the entity. Like in many programming paradigms, a key-value database resembles a map object or array, or dictionary, however, which is put away in a tenacious manner and controlled by a DBMS.

An e昀케cient and compact structure of the index is used by the key-value store to have the option to rapidly and dependably 昀椀nd value using its key. For example, Redis is a key-value store used to tracklists, maps, heaps, and primitive types (which are simple data structures) in a constant database. Redis can uncover a very basic point of interaction to query and manipulate value types, just by supporting a predetermined number of value types, and when arranged, is prepared to do high throughput.

When to use a key-value database:

Here are a few situations in which you can use a key-value database: -

- User session attributes in an online app like 昀椀nance or gaming, which is referred to as real-time random data access.
- Caching mechanism for repeatedly accessing data or key-based design.
- The application is developed on queries that are based on keys.

Features:

- One of the most un-complex kinds of NoSQL data models.

- For storing, getting, and removing data, key-value databases utilize simple functions.
- Querying language is not present in key-value databases.
- Built-in redundancy makes this database more reliable. Some

examples of key-value databases:

Here are some popular key-value databases which are widely used:

- Couchbase: It permits SQL-style querying and searching for text.
- Amazon DynamoDB: The key-value database which is mostly used is Amazon DynamoDB as it is a trusted database used by a large number of users. It can easily handle a large number of requests every day and it also provides various security options.
- Riak: It is the database used to develop applications.
- Aerospike: It is an open-source and real-time database working with billions of exchanges.
- Berkeley DB: It is a high-performance and open-source database providing scalability.

DOCUMENT DATA MODEL

A Document Data Model is a lot different than other data models because it stores data in JSON, BSON, or XML documents. in this data model, we can move documents under one document and apart from this, any particular elements can be indexed to run queries faster. Often documents are stored and retrieved in such a way that it becomes close to the data objects which are used in many applications which means very less translations are required to use data in applications. JSON is a native language that is often used to store and query data too.

So in the document data model, each document has a key-value pair below is an example for the same.

```
{
"Name" : "Yashodhra",
"Address" : "Near Patel Nagar",
"Email" : "yahoo123@yahoo.com",
"Contact" : "12345"
}
```

Working of Document Data Model:

This is a data model which works as a semi-structured data model in which the records and data associated with them are stored in a single document which means this data model is not completely unstructured. The main thing is that data here is stored in a document.

Features:

- Document Type Model: As we all know data is stored in documents rather than tables or graphs, so it becomes easy to map things in many programming languages.
- Flexible Schema: Overall schema is very much 昀氀exible to support this statement one must know that not all documents in a collection need to have the same 昀椀elds.
- Distributed and Resilient: Document data models are very much dispersed which is the reason behind horizontal scaling and distribution of data.
- Manageable Query Language: These data models are the ones in which query language allows the developers to perform CRUD (Create Read Update Destroy) operations on the data model.

Examples of Document Data Models :

- Amazon DocumentDB
- MongoDB
- Cosmos DB
- ArangoDB
- Couchbase Server
- CouchDB

Applications of Document Data Model :

- Content Management: These data models are very much used in creating various video streaming platforms, blogs, and similar services Because each is stored as a single document and the database here is much easier to maintain as the service evolves over time.
- Book Database: These are very much useful in making book databases because as we know this data model lets us nest.
- Catalog: When it comes to storing and reading catalog 昀椀les these data models are very much used because it has a fast reading ability if incase Catalogs have thousands of attributes stored.
- Analytics Platform: These data models are very much used in the Analytics Platform.

6. **Distributed and high availability:** NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
7. **Flexibility:** NoSQL databases allow developers to store and retrieve data in a 的數exible and dynamic manner, with support for multiple data types and changing data structures.
8. **Performance:** NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

Advantages of NoSQL: There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

1. **Schema Agnostic:** NoSQL Databases do not require any speci的椀c schema or s storage structure than traditional RDBMS.
2. **Flexibility:** NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accommodate dynamic changes to the data model. This makes NoSQL databases a good 的椀t for applications that need to handle changing data requirements.
3. **High availability:** The auto**,** replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.
4. **Scalability:** NoSQL databases are highly scalable, which means that they can handle large amounts of data and tra的케c with ease. This makes them a good 的椀t for applications that need to handle large amounts of data or tra的케c
5. **Performance:** NoSQL databases are designed to handle large amounts of data and tra的케c, which means that they can offer improved performance compared to traditional relational databases.
6. **Cost-effectiveness:** NoSQL databases are often more cost- effective than traditional relational databases, as they are typically less complex and do not require expensive hardware or software.
7. **Agility:** Ideal for agile development.

Disadvantages of NoSQL: NoSQL has the following disadvantages.

1. **Lack of standardization:** There are many different types of NoSQL databases, each with its own unique strengths and weaknesses. This lack of standardization can make it di的케cult to choose the right database for a speci的椀c application
2. **Lack of ACID compliance:** NoSQL databases are not fully ACID-compliant, which means that they do not guarantee the consistency,

integrity, and durability of data. This can be a drawback for applications that require strong data consistency guarantees.

3. **Narrow focus:** NoSQL databases have a very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the 뎡eld of Transaction Management than NoSQL.
4. **Open-source:** NoSQL is an open-source database. There is no reliable standard for NoSQL yet. In other words, two database systems are likely to be unequal.
5. **Lack of support for complex queries:** NoSQL databases are not designed to handle complex queries, which means that they are not a good 뎡t for applications that require complex data analysis or reporting.
6. **Lack of maturity:** NoSQL databases are relatively new and lack the maturity of traditional relational databases. This can make them less reliable and less secure than traditional databases.
7. **Management challenge:** The purpose of big data tools is to make the management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than in a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
8. **GUI is not available:** GUI mode tools to access the database are not 뎡뀷exibly available in the market.
9. **Backup:** Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
10. **Large document size:** Some database systems like MongoDB and CouchDB store data in JSON format. This means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts since they increase the document size.

GRAPH DATABASES

A graph database is a type of NoSQL database that is designed to handle data with complex relationships and interconnections. In a graph database, data is stored as nodes and edges, where nodes represent entities and edges represent the relationships between those entities.

1. Graph databases are particularly well-suited for applications that require deep and complex queries, such as social networks, recommendation engines, and fraud detection systems. They can also be used for other types of applications, such as supply chain management, network and infrastructure management, and bioinformatics.
2. One of the main advantages of graph databases is their ability to handle and represent relationships between entities. This is because the relationships between entities are as important as the entities themselves, and often cannot be easily represented in a traditional relational database.
3. Another advantage of graph database is their 뎡뀷exibility. Graph databases can handle data with changing structures and can be adapted to new use cases

without requiring signi卟cant changes to the database schema. This makes them particularly useful for applications with rapidly changing data structures or complex data requirements.

4. However, graph databases may not be suitable for all applications. For example, they may not be the best choice for applications that require simple queries or that deal primarily with data that can be easily represented in a traditional relational database. Additionally, graph databases may require more specialized knowledge and expertise to use effectively.

SCHEMALESS DATABASES

Traditional relational databases are well-de卟ned, using a schema to describe every functional element, including tables, rows views, indexes, and relationships. By exerting a high degree of control, the database administrator can improve performance and prevent capture of low- quality, incomplete, or malformed data. In a SQL database, the schema is enforced by the Relational Database Management System (RDBMS) whenever data is written to disk.

But in order to work, data needs to be heavily formatted and shaped to 卟t into the table structure. This means sacri卟cing any unde卟ned details during the save, or storing valuable information outside the database entirely.

A schemaless database, like MongoDB, does not have these up-front constraints, mapping to a more ‘natural’ database. Even when sitting on top of a data lake, each document is created with a partial schema to aid retrieval. Any formal schema is applied in the code of your applications; this layer of abstraction protects the raw data in the NoSQL database and allows for rapid transformation as your needs change.

Any data, formatted or not, can be stored in a non-tabular NoSQL type of database. At the same time, using the right tools in the form of a schemaless database can unlock the value of all of your structured and unstructured data types.

How does a schemaless database work?

In schemaless databases, information is stored in JSON-style documents which can have varying sets of 昀椀elds with different data types for each昀椀eld. So, a collection could look like this:

```
{
   name : “Joe”, age : 30, interests : ‘football’ }
{
   name : “Kate”, age : 25
   }
```

As you can see, the data itself normally has a fairly consistent structure. With the schemaless MongoDB database, there is some additional structure — the system namespace contains an explicit list of collections and indexes. Collections may be implicitly or explicitly created — indexes must be explicitly declared.

What are the benefits of using a schemaless database?

- Greater flexibility over data types

By operating without a schema, schemaless databases can store, retrieve, and query any data type — perfect for big data analytics and similar operations that are powered by unstructured data. Relational databases apply rigid schema rules to data, limiting what can be stored.

- No pre-defined database schemas

The lack of schema means that your NoSQL database can accept any data type — including those that you do not yet use. This future-proofs your database, allowing it to grow and change as your data-driven operations change and mature.

- No data truncation

A schemaless database makes almost no changes to your data; each item is saved in its own document with a partial schema, leaving the raw information untouched. This means that every detail is always available and nothing is stripped to match the current schema. This is particularly valuable if your analytics needs to change at some point in the future.

- Suitable for real-time analytics functions

With the ability to process unstructured data, applications built on NoSQL databases are better able to process real-time data, such as readings and measurements from IoT sensors. Schemaless databases are also ideal for use with machine learning and artificial intelligence operations, helping to accelerate automated actions in your business.

- Enhanced scalability and flexxibility

With NoSQL, you can use whichever data model is best suited to the job. Graph databases allow you to view relationships between data points, or you can use traditional wide table views with an exceptionally large

number of columns. You can query, report, and model information however you choose. And as your requirements grow, you can keep adding nodes to increase capacity and power.

When a record is saved to a relational database, anything (particularly metadata) that does not match the schema is truncated or removed. Deleted at write, these details cannot be recovered at a later point in time.

CASSANDRA

Apache Cassandra is highly scalable, high performance, distributed NoSQL database. Cassandra is designed to handle huge amount of data across many commodity servers, providing high availability without a single point of failure.

Cassandra has a distributed architecture which is capable to handle a huge amount of data. Data is placed on different machines with more than one replication factor to attain a high availability without a single point of failure.

Cassandra is a NoSQL database

NoSQL database is Non-relational database. It is also called Not Only SQL. It is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

Reasons behind its popularity

Cassandra is an Apache product. It is an open source, distributed and decentralized/distributed storage system (database). It is used to manage very large amounts of structured data spread out across the world. It provides high availability with no single point of failure.

Important Points of Cassandra

- Cassandra is a column-oriented database.
- Cassandra is scalable, consistent, and fault-tolerant.
- Cassandra's distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.
- Cassandra is created at Facebook. It is totally different from relational database management systems.
- Cassandra follows a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.

- Cassandra is being used by some of the biggest companies like Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, , and more.

History of Cassandra

Cassandra was initially developed at Facebook by two Indians Avinash Lakshman (one of the authors of Amazon's Dynamo) and Prashant Malik. It was developed to power the Facebook inbox search feature.

The following points specify the most important happenings in Cassandra history:

- Cassandra was developed at Facebook by Avinash Lakshman and Prashant Malik.
- It was developed for Facebook inbox search feature.
- It was open sourced by Facebook in July 2008.
- It was accepted by Apache Incubator in March 2009.
- Cassandra is a top level project of Apache since February 2010.
- The latest version of Apache Cassandra is 3.2.1.

Features of Cassandra

There are a lot of outstanding technical features which makes Cassandra very popular. Following is a list of some popular features of Cassandra:

High Scalability

Cassandra is highly scalable which facilitates you to add more hardware to attach more customers and more data as per requirement.

Rigid Architecture

Cassandra has not a single point of failure and it is continuously available for business-critical applications that cannot afford a failure.

Fast Linear-scale Performance

Cassandra is linearly scalable. It increases your throughput because it facilitates you to increase the number of nodes in the cluster. Therefore it maintains a quick response time.

Fault tolerant

Cassandra is fault tolerant. Suppose, there are 4 nodes in a cluster, here each node has a copy of same data. If one node is no longer serving then other three nodes can served as per request.

Flexible Data Storage

Cassandra supports all possible data formats like structured, semi- structured, and unstructured. It facilitates you to make changes to your data structures according to your need.

Easy Data Distribution

Data distribution in Cassandra is very easy because it provides the flexibility to distribute data where you need by replicating data across multiple data centers.

Transaction Support

Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).

Fast writes

Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

Cassandra Architecture

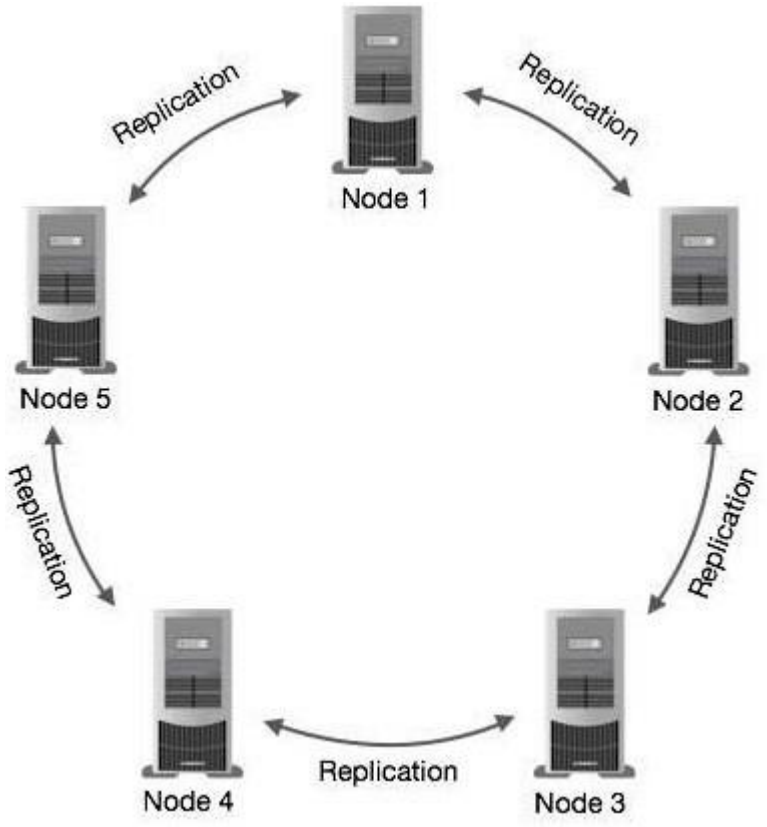
Cassandra was designed to handle big data workloads across multiple nodes without a single point of failure. It has a peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

- In Cassandra, each node is independent and at the same time interconnected to other nodes. All the nodes in a cluster play the same role.
- Every node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- In the case of failure of one node, Read/Write requests can be served from other nodes in the network.

Data Replication in Cassandra

In Cassandra, nodes in a cluster act as replicas for a given piece of data. If some of the nodes are responded with an out-of-date value, Cassandra will return the most recent value to the client. After returning the most recent value, Cassandra performs a read repair in the background to update the stale values.

See the following image to understand the schematic view of how Cassandra uses data replication among the nodes in a cluster to ensure no single point of failure.



Components of Cassandra

The main components of Cassandra are:

- **Node:** A Cassandra node is a place where data is stored.
- **Data center:** Data center is a collection of related nodes.
- **Cluster:** A cluster is a component which contains one or more data centers.
- **Commit log:** In Cassandra, the commit log is a crash-recovery mechanism. Every write operation is written to the commit log.
- **Mem-table:** A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem- tables.
- **SSTable:** It is a disk to which the data from the mem- table when its contents reach a threshold value.
- **Bloom:** These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is

a special kind of cache. Bloom are accessed after every query.

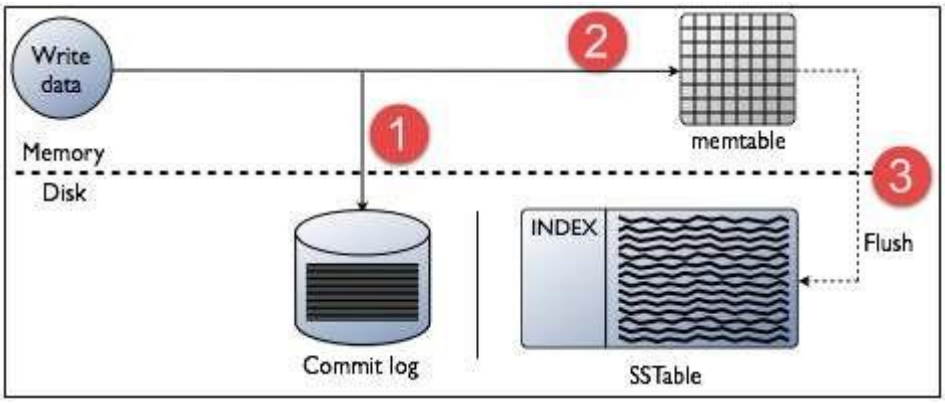
Cassandra Query Language

Cassandra Query Language (CQL) is used to access Cassandra through its nodes. CQL treats the database (Keyspace) as a container of tables. Programmers use cqlsh: a prompt to work with CQL or separate application language drivers.

The client can approach any of the nodes for their read-write operations. That node (coordinator) plays a proxy between the client and the nodes holding the data.

Write Operations

Every write activity of nodes is captured by the commit logs written in the nodes. Later the data will be captured and stored in the mem-table. Whenever the mem-table is full, data will be written into the SStable. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.



Read Operations

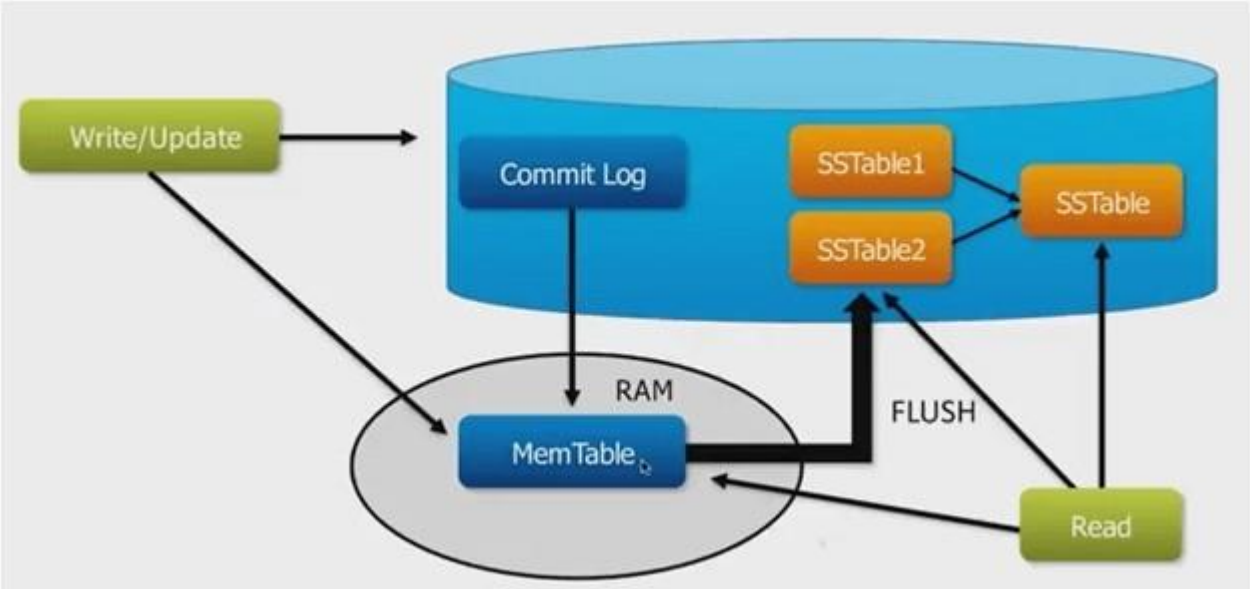
In Read operations, Cassandra gets values from the mem-table and checks the bloom the appropriate SSTable which contains the required data.

There are three types of read request that is sent to replicas by coordinators.

- Direct request
- Digest request
- Read repair request

The coordinator sends direct request to one of the replicas. After that, the coordinator sends the digest request to the number of replicas speci的碗ed by the consistency level and checks if the returned data is an updated data.

After that, the coordinator sends digest request to all the remaining replicas. If any node gives out of date value, a background read repair request will update that data. This process is called read repair mechanism.



Use Cases/ Applications of Cassandra

Cassandra can be used for different type of applications. Following is a list of use cases where Cassandra should be preferred:

Messaging

Cassandra is a great database which can handle a big amount of data. So it is preferred for the companies that provide Mobile phones and messaging services. These companies have a huge amount of data, so Cassandra is best for them.

Handle high speed Applications

Cassandra can handle the high speed data so it is a great database for the applications where data is coming at very high speed from different devices or sensors.

Product Catalogs and retail apps

Cassandra is used by many retailers for durable shopping cart protection and fast product catalog input and output.

Social Media Analytics and recommendation engine

Cassandra is a great database for many online companies and social media providers for analysis and recommendation to their customers.

Cassandra Data Model

Data model in Cassandra is totally different from normally we see in RDBMS. Let's see how Cassandra stores its data.

Cassandra database is distributed over several machines that are operated together. The outermost container is known as the Cluster which contains different nodes. Every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

Keyspace

Keyspace is the outermost container for data in Cassandra. Following are the basic attributes of Keyspace in Cassandra:

- **Replication factor:** It the number of machine in the cluster that will receive copies of the same data.
- **Replica placement Strategy:** It is a strategy which species how to place replicas in the ring. There are three types of strategies such as:

1) Simple strategy (rack-aware strategy)
2) old network topology strategy (rack-aware strategy)

   network topology strategy (datacenter-shared strategy)

Cassandra doesn't support JOINS, GROUP BY, OR clause, aggregation etc. So you have to store data in a way that it should be retrieved whenever you want.

Cassandra is optimized for high write performances so you should

maximize your writes for better read performance and data availability. There is a tradeoff between data write and data read. So, optimize you data read performance by maximizing the number of data writes

Cassandra doesn't support JOINS, GROUP BY, OR clause, aggregation etc. So you have to store data in a way that it should be retrieved whenever you want.

Cassandra is optimized for high write performances so you should
maximize your writes for better read performance and data availability. There is a tradeoff between data write and data read. So, optimize you data read performance by maximizing the number of data writes

UNIT III MapReduce

- **Job** – A Job in the context of HadoopMapReduce is the unit of work to be performed as requested by the client / user. The information associated with the Job includes the data to be processed (input data), MapReduce logic / program / algorithm, and any other relevant configuration information necessary to execute the Job.
- **Task** – HadoopMapReduce divides a Job into multiple sub-jobs known as Tasks. These tasks can be run independent of each other on various nodes across the cluster. There are primarily two types of Tasks – Map Tasks and Reduce Tasks.
- **Job Tracker**– Just like the storage (HDFS), the computation (MapReduce) also works in a master-slave / master-worker fashion. A Job Tracker node acts as the Master and is responsible for scheduling / executing Tasks on appropriate nodes, coordinating the execution of tasks, sending the information for the execution of tasks, getting the results back after the execution of each task, re-executing the failed Tasks, and monitors / maintains the overall progress of the Job. Since a Job consists of multiple Tasks, a Job's progress depends on the status / progress of Tasks associated with it. There is only one Job Tracker node per Hadoop Cluster.
- **TaskTracker** – A TaskTracker node acts as the Slave and is responsible for executing a Task assigned to it by the JobTracker. There is no restriction on the number of TaskTracker nodes that can exist in a Hadoop Cluster. TaskTracker receives the information necessary for execution of a Task from JobTracker, Executes the Task, and Sends the Results back to Job Tracker.
- **Map()** – Map Task in MapReduce is performed using the Map() function. This part of the MapReduce is responsible for processing one or more chunks of data and producing the output results.
- **Reduce()** – The next part / component / stage of the MapReduce programming model is the Reduce() function. This part of the MapReduce is responsible for consolidating the results produced by each of the Map() functions/tasks.
- **Data Locality** – MapReduce tries to place the data and the compute as close as possible. First, it tries to put the compute on the same node where data resides, if that cannot be done (due to reasons like compute on that node is down, compute on that

node is performing some other computation, etc.), then it tries to put the compute on the node nearest to the respective data node(s) which contains the data to be processed. This feature of MapReduce is “Data Locality”.

The following diagram shows the logical flow of a MapReduce programming model.

MapReduce Work Flow

The stages depicted above are

- **Input:** This is the input data / file to be processed.
- **Split:** Hadoop splits the incoming data into smaller pieces called “splits”.
- **Map:** In this step, MapReduce processes each split according to the logic defined in map() function. Each mapper works on each split at a time. Each mapper is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
- **Combine:** This is an optional step and is used to improve the performance by reducing the amount of data transferred across the network. Combiner is the same as the reduce step and is used for aggregating the output of the map() function before it is passed to the subsequent steps.
- **Shuffle & Sort:** In this step, outputs from all the mappers is shuffled, sorted to put them in order, and grouped before sending them to the next step.
- **Reduce:** This step is used to aggregate the outputs of mappers using the reduce() function. Output of reducer is sent to the next and final step. Each reducer is treated as a task and multiple tasks are executed across different TaskTrackers and coordinated by the JobTracker.
- **Output:** Finally the output of reduce step is written to a file in HDFS.

Game Example

Say you are processing a large amount of data and trying to find out what percentage of your user base where talking about games. First, we will identify the keywords which we

are going to map from the data to conclude that it's something related to games. Next, we will write a mapping function to identify such patterns in our data. For example, the keywords can be Gold medals, Bronze medals, Silver medals, Olympic football, basketball, cricket, etc.

Let us take the following chunks in a big data set and see how to process it.

"Hi, how are you"

"We love football"

"He is an awesome football player"

"Merry Christmas"

"Olympics will be held in China"

"Records broken today in Olympics"

"Yes, we won 2 Gold medals"

"He qualified for Olympics"

Mapping Phase – So our map phase of our algorithm will be as

1. Declare a function "Map"
2. Loop: For each words equal to "football"
3. Increment counter
4. Return key value "football"=>counter

In the same way, we can define n number of mapping functions for mapping various words: "Olympics", "Gold Medals", "cricket", etc.

Reducing Phase – The reducing function will accept the input from all these mappers in form of key value pair and then processing it. So, input to the reduce function will look like the following:

reduce ("football"=>2)

reduce (“Olympics”=>3)

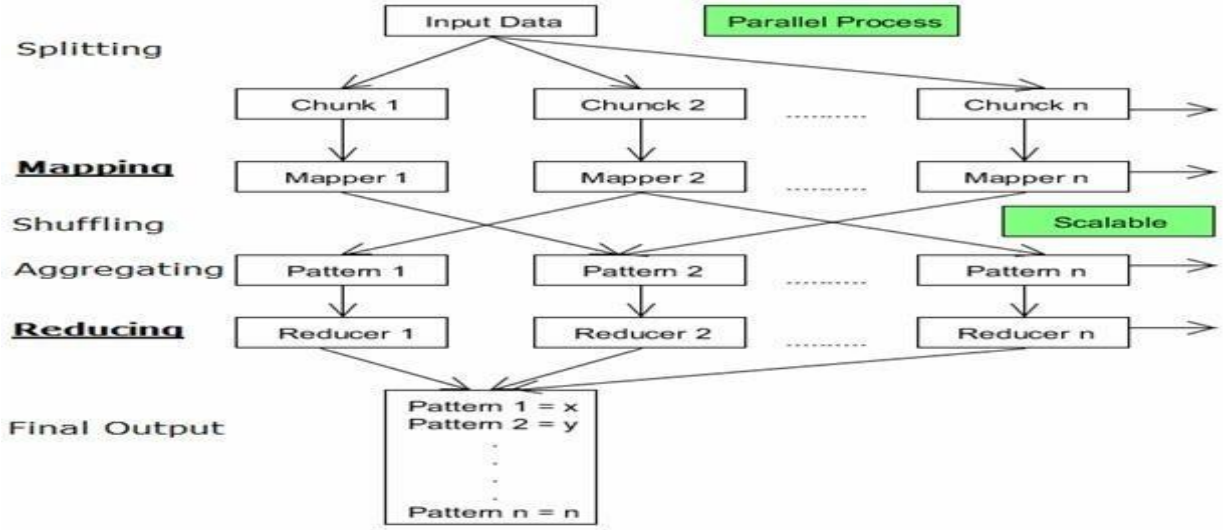
Our algorithm will continue with the following steps

5. Declare a function reduce to accept the values from map function.
6. Where for each key-value pair, add value to counter.
7. Return “games”=> counter.

At the end, we will get the output like “games”=>5.

Now, getting into a big picture we can write n number of mapper functions here. Let us say that you want to know who all where wishing each other. In this case you will write a mapping function to map the words like “Wishing”, “Wish”, “Happy”, “Merry” and then will write a corresponding reducer function.

Here you will need one function for shuffling which will distinguish between the “games” and “wishing” keys returned by mappers and will send it to the respective reducer function. Similarly you may need a function for splitting initially to give inputs to the mapper functions in form of chunks. The following diagram summarizes the flow of Map reduce algorithm:



In the above map reduce flow

- The input data can be divided into n number of chunks depending upon the amount of data and processing capacity of individual unit.
- Next, it is passed to the mapper functions. Please note that all the chunks are processed simultaneously at the same time, which embraces the parallel processing of data.
- After that, shuffling happens which leads to aggregation of similar patterns.
- Finally, reducers combine them all to get a consolidated output as per the logic.
- This algorithm embraces scalability as depending on the size of the input data, we can keep increasing the number of the parallel processing units.

Unit Tests with MR Unit

HadoopMapReduce jobs have a unique code architecture that follows a specific template with specific constructs.

This architecture raises interesting issues when doing test-driven development (TDD) and writing unit test

With MRUnit, you can craft test input, push it through your mapper and/or reducer, and verify its output all in a JUnit test.

As do other JUnit tests, this allows you to debug your code using the JUnit test as a driver.
A map/reduce pair can be tested using MRUnit'sMapReduceDriver. , a combiner can be tested using MapReduceDriver as well.
A PipelineMapReduceDriver allows you to test a workflow of map/reduce jobs. Currently, partitioner's do not have a test driver under MRUnit.

MRUnit allows you to do TDD(Test Driven Development) and write lightweight unit tests which accommodate Hadoop's specific architecture and constructs.

Example: We're processing road surface data used to create maps. The input contains both linear surfaces and intersections. The mapper takes a collection of these mixed surfaces as input, discards anything that isn't a linear road surface, i.e., intersections, and then processes each road surface and writes it out to HDFS. We can keep count and eventually print out how many non-road surfaces are inputs. For debugging purposes, we can additionally print out how many road surfaces were processed.

Anatomy of a MapReduce Job Run

You can run a MapReduce job with a single method call: submit () on a Job object (you can also call waitForCompletion(), which submits the job if it hasn't been submitted already, then waits for it to finish). This method call conceals a great deal of processing behind the scenes. This section uncovers the steps Hadoop takes to run a job.
The whole process is illustrated in Figure 7-1. At the highest level, there are five independent entities:

- The client, which submits the MapReduce job.
- The YARN resource manager, which coordinates the allocation of compute resources On the cluster.
- The YARN node managers, which launch and monitor the compute containers on Machines in the cluster.
- The MapReduce application master, which coordinates the tasks running the MapReduce job. The application master and the MapReduce tasks run in containers That are scheduled by the resource manager and managed by the node managers.

- The distributed filesystem, which is used for sharing job files between the other entities.
- He distributed filesystem ,which is used for sharing job files between the other entities.

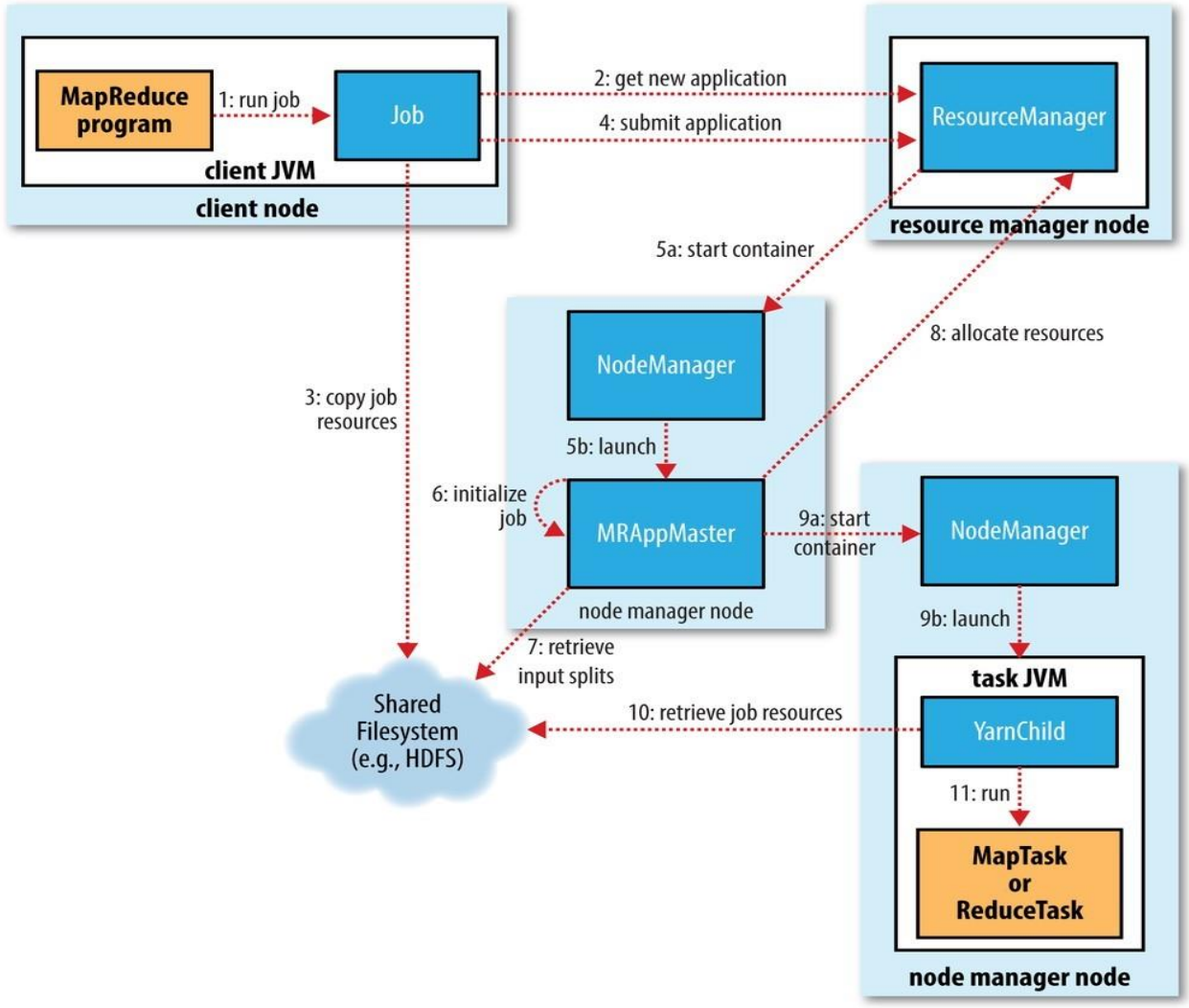


Figure 7-1. How Hadoop runs a MapReduce job

Classic MapReduce

A job run in classic MapReduce is illustrated in Figure 6-1. At the highest level, **there are four independent entities**:

• The client, which submits the MapReduce job.

• The jobtracker, which coordinates the job run. The jobtracker is a Java application whose

main class is JobTracker.

- The tasktrackers, which run the tasks that the job has been split into. Tasktrackers are Java applications whose main class is TaskTracker.
- The distributed filesystem, which is used for sharing job files between the other entities.

Job Initialization:

When the JobTracker receives a call to its submitJob() method, it puts it into an internal queue from where the job scheduler will pick it up and initialize it. Initialization involves creating an object to represent the job being run.

To create the list of tasks to run, the job scheduler first retrieves the input splits computed by the client from the shared filesystem. It then creates one map task for each split.

Task Assignment:

Tasktrackers run a simple loop that periodically sends heartbeat method calls to the jobtracker. Heartbeats tell the jobtracker that a tasktracker is alive As a part of the heartbeat, a tasktracker will indicate whether it is ready to run a new task, and if it is, the jobtracker will llocate it a task, which it communicates to the tasktracker using the heartbeat return value.

Task Execution:

Now that the tasktracker has been assigned a task, the next step is for it to run the task. First, it localizes the job JAR by copying it from the shared filesystem to the tasktracker's filesystem. It also copies any files needed from the distributed cache by the application to the local disk. TaskRunner launches a new Java Virtual Machine to run each task in.

Progress and Status Updates:

MapReduce jobs are long-running batch jobs, taking anything from minutes to hours to run. Because this is a significant length of time, it's important for the user to get feedback on how the job is progressing. A job and each of its tasks have a *status.*When a task is running, it keeps track of its *progress*, that is, the proportion of the task completed.

Job Completion

When the jobtracker receives a notification that the last task for a job is complete (this will be the special job cleanup task), it changes the status for the job to "successful."

YARN

Yet Another Resource Manager takes programming to the next level beyond Java , and makes it interactive to let another application Hbase, Spark etc. to work on it.Different Yarn applications can co-exist on the same cluster so MapReduce, Hbase, Spark all can run at the same time bringing great benefits for manageability and cluster utilization.

Components Of YARN

- **Client:** For submitting MapReduce jobs.
- **Resource Manager:** To manage the use of resources across the cluster
- **Node Manager:**For launching and monitoring the computer containers on machines in the cluster.
- **Map Reduce Application Master:** Checks tasks running the MapReduce job. The application master and the MapReduce tasks run in containers that are scheduled by the resource manager, and managed by the node managers.

Jobtracker&Tasktrackerwere were used in previous version of Hadoop, which were responsible for handling resources and checking progress management. However, Hadoop 2.0 has Resource manager and NodeManager to overcome the shortfall of Jobtracker&Tasktracker.

Benefits of YARN

- **Scalability:** Map Reduce 1 hits ascalability bottleneck at 4000 nodes and 40000 task, but Yarn is designed for 10,000 nodes and 1 lakh tasks.
- **Utiliazation:** Node Manager manages a pool of resources, rather than a fixed number of the designated slots thus increasing the utilization.
- **Multitenancy:** Different version of MapReduce can run on YARN, which makes the process of upgrading MapReduce more manageable.

Sort and Shuffle

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

MapReduce Types

Mapping is the core technique of processing a list of data elements that come in pairs of keys and values. The map function applies to individual elements defined as key-value pairs of a list and produces a new list. The general idea of map and reduce function of Hadoop can be illustrated as follows:

```
map: (K1, V1) -> list (K2, V2)
reduce: (K2, list(V2)) -> list (K3, V3)
```

The input parameters of the key and value pair, represented by K1 and V1 respectively, are different from the output pair type: K2 and V2. The reduce function accepts the same format output by the map, but the type of output again of the reduce operation is different: K3 and V3. The Java API for this is as follows:

```
public interface Mapper<K1, V1, K2, V2> extends JobConfigurable,Closeable
 {
void map(K1 key, V1 value, OutputCollector<K2, V2> output, Reporter reporter) throws
IOException;
}
public interface Reducer<K2, V2, K3, V3> extends JobConfigurable,Closeable
{
void reduce(K2 key, Iterator<V2> values,
OutputCollector<K3, V3> output, Reporter reporter)throws
IOException;
}
```

The *OutputCollector* is the generalized interface of the Map-Reduce framework to facilitate collection of data output either by the *Mapper* or the *Reducer*. These outputs are nothing but intermediate output of the job. Therefore, they must be parameterized with their types.
The *Reporter* facilitates the Map-Reduce application to report progress andupdatecountrsand status information. If, however, the combine function is used, it has

```
map: (K1, V1) -> list (K2, V2)
combine: (K2, list(V2)) -> list (K2, V2)
reduce: (K2, list(V2)) -> list (K3, V3)
```

the same form as the reduce function and the output is fed to the reduce function. This may be illustrated as follows

Note that the combine and reduce functions use the same type, except in the variable names where K3 is K2 and V3 is V2.
The partition function operates on the intermediate key-value types. It controls the partitioning of the keys of the intermediate map outputs. The key derives the partition using a typical hash function. The total number of partitions is the same as the number of reduce tasks for the job. The partition is determined only by the key ignoring the value.

```
public interface Partitioner<K2, V2> extends JobConfigurable {

intgetPartition(K2 key, V2 value, intnumberOfPartition);

}
```

UNIT IV BASICS OF HADOOP

4.1 BASICS OF HADOOP - DATA FORMAT

Hadoop is an open-source framework for processing, storing, and analyzing large volumes of data in a distributed computing environment. It provides a reliable, scalable, and distributed computing system for big data.

Key Components:

- **Hadoop Distributed File System (HDFS):** HDFS is the storage system of Hadoop, designed to store very large files across multiple machines.
- **MapReduce:** MapReduce is a programming model for processing and generating large datasets that can be parallelized across a distributed cluster of computers.
- **YARN (Yet Another Resource Negotiator):** YARN is the resource management layer of Hadoop, responsible for managing and monitoring resources in a cluster.

Advantages of Hadoop:

- **Scalability:** Hadoop can handle and process vast amounts of data by distributing it across a cluster of machines.
- **Fault Tolerance:** Hadoop is fault-tolerant, meaning it can recover from failures, ensuring that data processing is not disrupted.
- **Cost-Effective:** It allows businesses to store and process large datasets cost-effectively, as it can run on commodity hardware.

Here are some basics:

1. **Data Storage in Hadoop**:
    - Hadoop uses the Hadoop Distributed File System (HDFS) to store data across multiple machines in a distributed fashion. Data is divided into blocks (typically 128 MB or 256 MB in size), and each block is replicated across several nodes in the cluster for fault tolerance.
2. **Data Formats**:
    - Hadoop can work with various data formats, but some common ones include:
        - **Text**: Data is stored in plain text files, such as CSV or TSV.
        - **SequenceFile**: A binary file format optimized for Hadoop, suitable for storing key-value pairs.
        - **Avro**: A data serialization system that supports schema evolution. It's often used for complex data structures.

    - **Parquet**: A columnar storage format that is highly optimized for analytics workloads. It's efficient for both reading and writing.
3. **Data Ingestion**:
    - Before analyzing data, you need to ingest it into Hadoop. You can use tools like Apache Flume, Apache Sqoop, or simply copy data into HDFS using Hadoop commands.
4. **Data Processing**:
    - Hadoop primarily processes data using a batch processing model. It uses a programming model called MapReduce to distribute the processing tasks across the cluster. You write MapReduce jobs to specify how data should be processed.
    - In addition to MapReduce, Hadoop ecosystem also includes higher-level processing frameworks like Apache Spark, Apache Hive, and Apache Pig, which provide more user-friendly abstractions for data analysis.
5. **Data Analysis**:
    - Once data is processed, you can analyze it to gain insights. This may involve running SQL-like queries (with Hive), machine learning algorithms (with Mahout or Spark MLlib), or custom data processing logic.
6. **Data Output**:
    - After analysis, you can store the results back into Hadoop, or you can export them to other systems for reporting or further analysis.
7. **Data Compression**:
    - Hadoop allows data compression to reduce storage requirements and improve processing speed. Common compression formats include Gzip, Snappy, and LZO.
8. **Data Schema**:
    - When working with structured data, it's important to define a schema. Some formats like Avro and Parquet have built-in schema support. In other cases, you may need to maintain the schema separately.
9. **Data Partitioning and Shuffling**:
    - During data processing, Hadoop can partition data into smaller chunks and shuffle it across nodes to optimize the processing pipeline.
10. **Data Security and Access Control**:
    - Hadoop provides security mechanisms to control access to data and cluster resources. This includes authentication, authorization, and encryption.

4.1.1. Step-by-step installation of Hadoop on a single Ubuntu machine.

Installing Hadoop on a single-node cluster is a common way to set up Hadoop for learning and development purposes. In this guide, I'll walk you through the step-by-step installation of Hadoop on a single Ubuntu machine.

Prerequisites:

- A clean installation of Ubuntu.
- Java installed on your system.

Let's proceed with the installation:

Step 1: Download Hadoop

1. Visit the Apache Hadoop website (https://hadoop.apache.org) and choose the Hadoop version you want to install. Replace `X.Y.Z` with the version number you choose.
2. Download the Hadoop distribution using `wget` or your web browser. For example:

```
bash
wget        https://archive.apache.org/dist/hadoop/common/hadoop-X.Y.Z/hadoop-X.Y.Z.tar.gz
```

Step 2: Extract Hadoop 3. Extract the downloaded Hadoop tarball to your desired directory (e.g., `/usr/local/`):

```
bash
sudo tar -xzvf hadoop-X.Y.Z.tar.gz -C /usr/local/
```

Step 3: Configure Environment Variables 4. Edit your `~/.bashrc` file to set up environment variables. Replace `X.Y.Z` with your Hadoop version:

```
bash
export HADOOP_HOME=/usr/local/hadoop-X.Y.Z
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

Apply these changes to your current shell:

```
bash
source ~/.bashrc
```

Step 4: Edit Hadoop Configuration Files 5. Navigate to the Hadoop configuration directory:

```
bash
cd $HADOOP_HOME/etc/hadoop
```

6. Edit the `hadoop-env.sh` file to specify the Java home directory. Add the following line to the file, pointing to your Java installation:

```
bash
export JAVA_HOME=/usr/lib/jvm/default-java
```

7. Configure Hadoop's core-site.xml by editing it and adding the following XML snippet. This sets the Hadoop Distributed File System (HDFS) data directory:

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

8. Configure Hadoop's hdfs-site.xml by editing it and adding the following XML snippet. This sets the HDFS data and metadata directories:

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

<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///usr/local/hadoop-X.Y.Z/data/namenode</value>
</property>

<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///usr/local/hadoop-X.Y.Z/data/datanode</value>
</property>
```

Step 5: Format the HDFS Filesystem 9. Before starting Hadoop services, you need to format the HDFS filesystem. Run the following command:

```
bash
hdfs namenode -format
```

Step 6: Start Hadoop Services 10. Start the Hadoop services using the following command:

```
bash
start-all.sh
```

Step 7: Verify Hadoop Installation 11. Check the running Hadoop processes using the `jps` command:

```
bash
jps
```

You should see a list of Java processes running, including `NameNode`, `DataNode`, `ResourceManager`, and `NodeManager`.

Step 8: Access Hadoop Web UI 12. Open a web browser and access the Hadoop Web UI at http://localhost:50070/ (for HDFS) and http://localhost:8088/ (for YARN ResourceManager).

You have successfully installed Hadoop on a single-node cluster. You can now use it for learning and experimenting with Hadoop and MapReduce.

4.2 DATA FORMAT - ANALYZING DATA WITH HADOOP

Data Formats in Hadoop:

- **Text Files:** Simple plain text files, where each line represents a record.

- **Sequence Files:** Binary files containing serialized key/value pairs.
- **Avro:** A data serialization system that provides rich data structures in a compact binary format.
- **Parquet:** A columnar storage file format optimized for use with Hadoop.

Analyzing Data with Hadoop:

- **MapReduce Programming Model:** Data analysis tasks in Hadoop are accomplished using the MapReduce programming model, where data is processed in two stages: the Map stage processes and sorts the data, and the Reduce stage performs summary operations.
- **Hive:** Hive is a data warehousing and SQL-like query language for Hadoop. It allows users to query and manage large datasets stored in Hadoop HDFS.
- **Pig:** Pig is a high-level platform and scripting language built on top of Hadoop, used for creating MapReduce programs for data analysis.

Analyzing data with Hadoop involves understanding the data format and structure, as well as using appropriate tools and techniques for processing and deriving insights from the data. Here are some key considerations when it comes to data format and analysis with Hadoop:

1. Data Format:

- **Structured Data**: If your data is structured, meaning it follows a fixed schema, you can use formats like Avro, Parquet, or ORC. These columnar storage formats are efficient for large-scale data analysis and support schema evolution.
- **Semi-Structured Data**: Data in JSON or XML format falls into this category. Hadoop can handle semi-structured data, and tools like Hive and Pig can help you query and process it effectively.
- **Unstructured Data**: Text data, log files, and other unstructured data can be processed using Hadoop as well. However, processing unstructured data often requires more complex parsing and natural language processing (NLP) techniques.

2. Data Ingestion:

- Before you can analyze data with Hadoop, you need to ingest it into the Hadoop Distributed File System (HDFS) or another storage system compatible with Hadoop. Tools like Apache Flume or Apache Sqoop can help with data ingestion.

3. Data Processing:

- Hadoop primarily uses the MapReduce framework for batch data processing. You write MapReduce jobs to specify how data should be processed. However, there are also high-level processing frameworks like Apache Spark and Apache Flink that provide more user-friendly abstractions and real-time processing capabilities.

4. Data Analysis:

- For SQL-like querying of structured data, you can use Apache Hive, which provides a SQL interface to Hadoop. Hive queries get translated into MapReduce or Tez jobs.

- For SQL-like querying of structured data, you can use Apache Hive, which provides a SQL interface to Hadoop. Hive queries get translated into MapReduce or Tez jobs.
- Apache Pig is a scripting language specifically designed for data processing in Hadoop. It's useful for ETL (Extract, Transform, Load) tasks.
- For advanced analytics and machine learning, you can use Apache Spark, which provides MLlib for machine learning tasks, and GraphX for graph processing.

5. Data Storage and Compression:

- Hadoop provides various storage formats optimized for analytics (e.g., Parquet, ORC) and supports data compression to reduce storage requirements and improve processing speed.

6. Data Partitioning and Shuffling:

- Hadoop can automatically partition data into smaller chunks and shuffle it across nodes to optimize the processing pipeline.

7. Data Security and Access Control:

- Hadoop offers mechanisms for securing data and controlling access through authentication, authorization, and encryption.

8. Data Visualization:

- To make sense of the analyzed data, you can use data visualization tools like Apache Zeppelin or integrate Hadoop with business intelligence tools like Tableau or Power BI.

9. Performance Tuning:

- Hadoop cluster performance can be optimized through configuration settings and resource allocation. Understanding how to fine-tune these parameters is essential for efficient data analysis.

10. Monitoring and Maintenance:

- Regularly monitor the health and performance of your Hadoop cluster using tools like Ambari or Cloudera Manager. Perform routine maintenance tasks to ensure smooth operation.

Analyzing data with Hadoop involves a combination of selecting the right data format, processing tools, and techniques to derive meaningful insights from your data. Depending on your specific use case, you may need to choose different formats and tools to suit your needs.

4.3 SCALING OUT

Scaling out is a fundamental concept in distributed computing and is one of the key benefits of using Hadoop for big data analysis. Here are some important points related to scaling out in Hadoop:

- **Horizontal Scalability**: Hadoop is designed for horizontal scalability, which means that you can expand the cluster by adding more commodity hardware machines to it. This allows you to accommodate larger datasets and perform more extensive data processing.

- **Data Distribution**: Hadoop's HDFS distributes data across multiple nodes in the cluster. When you scale out by adding more nodes, data is automatically distributed across these new machines. This distributed data storage ensures fault tolerance and high availability.

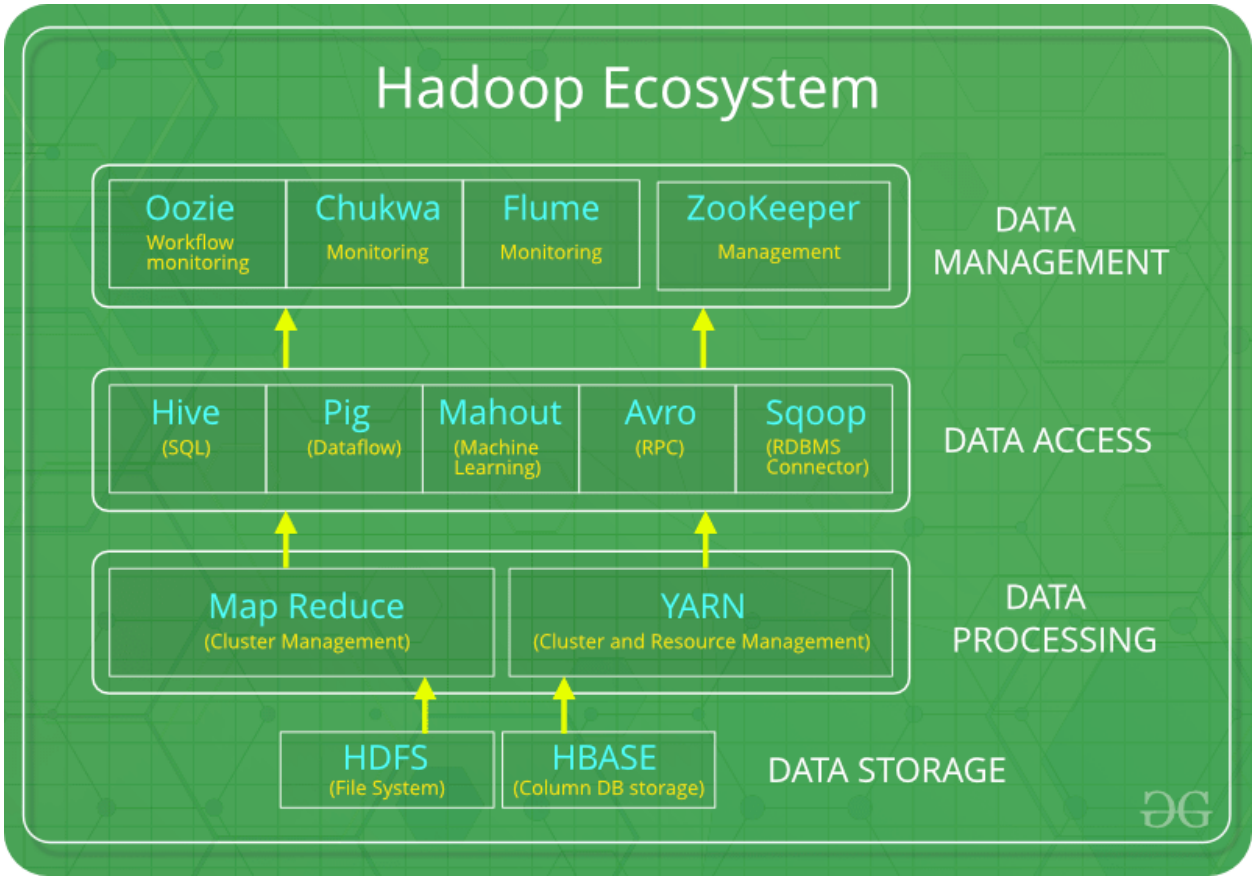
- **Processing Power**: Scaling out also means increasing the processing power of the cluster. You can run more MapReduce tasks and analyze data in parallel across multiple nodes, which can significantly speed up data processing.

- **Elasticity**: Hadoop clusters can be designed to be elastic, meaning you can dynamically add or remove nodes based on workload requirements. This is particularly useful in cloud-based Hadoop deployments where you pay for resources based on actual usage.

- **Balancing Resources**: When scaling out, it's important to consider resource management and cluster balancing. Tools like Hadoop YARN (Yet Another Resource Negotiator) help allocate and manage cluster resources efficiently.

Scaling Hadoop:

- **Horizontal Scaling:** Hadoop clusters can scale horizontally by adding more machines to the existing cluster. This approach improves processing power and storage capacity.
- **Vertical Scaling:** Vertical scaling involves adding more resources (CPU, RAM) to existing nodes in the cluster. However, there are limits to vertical scaling, and horizontal scaling is preferred for handling larger workloads.
- **Cluster Management Tools:** Tools like Apache Ambari and Cloudera Manager help in managing and scaling Hadoop clusters efficiently.
- **Data Partitioning:** Proper data partitioning strategies ensure that data is distributed evenly across the cluster, enabling efficient processing.



4.4. HADOOP STREAMING

What is Hadoop Streaming? Hadoop Streaming is a utility that comes with Hadoop distribution. It allows you to create and run MapReduce jobs with any executable or script as the mapper and/or the reducer. This means you can use any programming language that can read from standard input and write to standard output for your MapReduce tasks.

How Hadoop Streaming Works:

1. **Input:** Hadoop Streaming reads input from HDFS or any other file system and provides it to the mapper as lines of text.

2. **Mapper:** You can use any script or executable as a mapper. Hadoop Streaming feeds the input lines to the mapper's standard input.

3. **Shuffling and Sorting:** The output from the mapper is sorted and partitioned by the Hadoop framework.

4. **Reducer:** Similarly, you can use any script or executable as a reducer. The reducer reads sorted input lines from its standard input and produces output, which is written to HDFS or any other file system.

5. **Output:** The final output is stored in HDFS or the specified output directory.

Advantages of Hadoop Streaming:

- **Language Flexibility:** It allows developers to use languages like Python, Perl, Ruby, etc., for writing MapReduce programs, extending Hadoop's usability beyond Java developers.

- **Rapid Prototyping:** Developers can quickly prototype and test algorithms without the need to compile and package Java code.

4.5 HADOOP PIPES

What is Hadoop Pipes? Hadoop Pipes is a C++ API to implement Hadoop MapReduce applications. It enables the use of C++ to write MapReduce programs, allowing developers proficient in C++ to leverage Hadoop's capabilities.

How Hadoop Pipes Works:

1. **Mapper and Reducer:** Developers write the mapper and reducer functions in C++.

2. **Input:** Hadoop Pipes reads input from HDFS or other file systems and provides it to the mapper as key-value pairs.

3. **Map and Reduce Operations:** The developer specifies the map and reduce functions, defining the logic for processing the input key-value pairs.

4. **Output:** The final output is written to HDFS or another file system.

Advantages of Hadoop Pipes:

- **Performance:** Programs written in C++ can sometimes be more performant due to the lower-level memory management and execution speed of C++.

- **C++ Libraries:** Developers can leverage existing C++ libraries and codebases, making it easier to integrate with other systems and tools.

Both Hadoop Streaming and Hadoop Pipes provide flexibility in terms of programming languages, enabling a broader range of developers to work with Hadoop and leverage its powerful data processing capabilities.

4.6 DESIGN OF HADOOP DISTRIBUTED FILE SYSTEM (HDFS):

Architecture: Hadoop Distributed File System (HDFS) is designed to store very large files across multiple machines in a reliable and fault-tolerant manner. Its architecture consists of the following components:

1. **NameNode:** The NameNode is the master server that manages the namespace and regulates access to files by clients. It stores metadata about the files and directories, such as the file structure tree and the mapping of file blocks to DataNodes.

2. **DataNode:** DataNodes are responsible for storing the actual data. They store data in the form of blocks and send periodic heartbeats and block reports to the NameNode to confirm that they are functioning correctly.

3. **Block:** HDFS divides large files into fixed-size blocks (typically 128 MB or 256 MB). These blocks are stored across the DataNodes in the cluster.

4. **Secondary NameNode (Deprecated Term):** The Secondary NameNode is not a backup or failover NameNode. Its primary function is to periodically merge the namespace image and edit log files, reducing the load on the primary NameNode.

Replication: HDFS replicates each block multiple times (usually three) and places these replicas on different DataNodes across the cluster. Replication ensures fault tolerance. If a DataNode or block becomes unavailable, the system can continue to function using the remaining replicas.

4.7 HDFS CONCEPTS:

1. Block: Blocks are the fundamental units of data storage in HDFS. Each file is broken down into blocks, and these blocks are distributed across the cluster's DataNodes.

2. Namespace: HDFS organizes files and directories in a hierarchical namespace. The NameNode manages this namespace and regulates access to files by clients.

3. Replication: As mentioned earlier, HDFS replicates blocks for fault tolerance. The default replication factor is 3, but it can be configured based on the cluster's requirements.

4. Fault Tolerance: HDFS achieves fault tolerance by replicating data blocks across multiple nodes. If a DataNode or block becomes unavailable due to hardware failure or other issues, the system can continue to operate using the replicated blocks.

5. High Write Throughput: HDFS is optimized for high throughput of data, making it suitable for applications with large datasets. It achieves this through the parallelism of writing and reading data across multiple nodes.

6. Scalability: HDFS is designed to scale horizontally by adding more nodes to the cluster. This scalability allows Hadoop clusters to handle large and growing amounts of data.

7. Data Integrity: HDFS ensures data integrity by storing checksums of data with each block. This checksum is verified by clients and DataNodes to ensure that data is not corrupted during storage or transmission.

4.8 JAVA INTERFACE IN HADOOP:

Hadoop provides Java APIs that developers can use to interact with the Hadoop ecosystem. The Java interface in Hadoop includes various classes and interfaces that allow developers to create MapReduce jobs, configure Hadoop clusters, and manipulate data stored in HDFS. Here's a brief overview of key components in the Java interface:

1. **`org.apache.hadoop.mapreduce` Package:**
   - `Mapper`: Interface for the mapper task in a MapReduce job.
   - `Reducer`: Interface for the reducer task in a MapReduce job.
   - `Job`: Represents a MapReduce job configuration.
   - `InputFormat`: Specifies the input format of the job.
   - `OutputFormat`: Specifies the output format of the job.
   - `Configuration`: Represents Hadoop configuration properties.
2. **`org.apache.hadoop.fs` Package:**
   - `FileSystem`: Interface representing a file system in Hadoop (HDFS, local file system, etc.).
   - `Path`: Represents a file or directory path in Hadoop.
3. **`org.apache.hadoop.io` Package:**
   - `Writable`: Interface for custom Hadoop data types.
   - `WritableComparable`: Interface for custom data types that are comparable and writable.

Developers use these interfaces and classes to create custom MapReduce jobs, configure input and output formats, and interact with HDFS. They can implement the `Mapper` and `Reducer` interfaces to define their own map and reduce logic for processing data.

4.9 DATA FLOW IN HADOOP:

Data flow in Hadoop refers to the movement of data between different stages of a MapReduce job or between Hadoop components. Here's how data flows in a typical Hadoop MapReduce job:

1. **Input Phase:**
   - Input data is read from one or more sources, such as HDFS files, HBase tables, or other data storage systems.
   - Input data is divided into input splits, which are processed by individual mapper tasks.
2. **Map Phase:**
   - Mapper tasks process the input splits and produce intermediate key-value pairs.
   - The intermediate data is partitioned, sorted, and grouped by key before being sent to the reducers.

3. **Shuffle and Sort Phase:**
    - Intermediate data from all mappers is shuffled and sorted based on keys.
    - Data with the same key is grouped together, and each group of data is sent to a specific reducer.
4. **Reduce Phase:**
    - Reducer tasks receive sorted and grouped intermediate data.
    - Reducers process the data and produce the final output key-value pairs, which are typically written to HDFS or another storage system.
5. **Output Phase:**
    - The final output is stored in HDFS or another output location specified by the user.
    - Users can access the output data for further analysis or processing.

4.10 HADOOP I/O - DATA INTEGRITY:

Ensuring data integrity is crucial in any distributed storage and processing system like Hadoop. Hadoop provides several mechanisms to maintain data integrity:

1. **Replication:**
    - HDFS stores multiple replicas of each block across different nodes. If a replica is corrupted, Hadoop can use one of the other replicas to recover the lost data.
2. **Checksums:**
    - HDFS uses checksums to validate the integrity of data blocks. Each block is associated with a checksum, which is verified by both the client reading the data and the DataNode storing the data. If a block's checksum doesn't match the expected value, Hadoop knows the data is corrupted and can request it from another node.
3. **Write Pipelining:**
    - HDFS pipelines the data through several nodes during the writing process. Each node in the pipeline verifies the checksums before passing the data to the next node. If a node detects corruption, it can request the block from another replica.
4. **Error Detection and Self-healing:**
    - Hadoop can detect corrupted blocks and automatically replace them with healthy replicas from other nodes, ensuring the integrity of the stored data.

4.11 COMPRESSION AND SERIALIZATION IN HADOOP:

1. **Compression:**
    - Hadoop supports various compression algorithms like Gzip, Snappy, and LZO. Compressing data before storing it in HDFS can significantly reduce storage requirements and improve the efficiency of data processing. You can specify the compression codec when writing data to HDFS or when configuring MapReduce jobs.

    Example of specifying a compression codec in a MapReduce job:

```
java
conf.set("mapreduce.map.output.compress", "true");
conf.set("mapreduce.map.output.compress.codec",
"org.apache.hadoop.io.compress.SnappyCodec");
```

- **Serialization:**

- Hadoop uses its own serialization framework called Writable to serialize data efficiently. Writable data types are Java objects optimized for Hadoop's data transfer. You can also use Avro or Protocol Buffers for serialization. These serialization formats are more efficient than Java's default serialization mechanism, especially in the context of large-scale data processing.

Example of using Avro for serialization:

```
java
   // Writing Avro data to HDFS
   DatumWriter<YourAvroRecord> datumWriter = new
   SpecificDatumWriter<>(YourAvroRecord.class);
   DataFileWriter<YourAvroRecord> dataFileWriter = new
   DataFileWriter<>(datumWriter);
   dataFileWriter.create(yourAvroRecord.getSchema(), new File("output.avro"));
   dataFileWriter.append(yourAvroRecord);
   dataFileWriter.close();
```

By utilizing these mechanisms, Hadoop ensures that data integrity is maintained during storage and processing. Additionally, compression and efficient serialization techniques optimize storage and data transfer, contributing to the overall performance of Hadoop applications.

4.12 AVRO - FILE-BASED DATA STRUCTURES:

Apache Avro is a data serialization framework that provides efficient data interchange in Hadoop. It enables the serialization of data structures in a language-independent way, making it ideal for data stored in files. Avro uses JSON for defining data types and protocols, allowing data to be self-describing and allowing complex data structures.

Key Concepts:

1. **Schema Definition:** Avro uses JSON to define schemas. Schemas define the data structure, including types and their relationships. For example, you can define records, enums, arrays, and more in Avro schemas.

```
json
     "type": "record",
     "name": "User",
     "fields": [
       {"name": "name", "type": "string"},
       {"name": "age", "type": "int"},
       {"name": "address", "type": "string"}
```

}
{

1.
2. **Serialization:** Avro encodes data using the defined schema, producing compact binary files. Avro data is self-describing, meaning that the schema is embedded in the data itself.
3. **Deserialization:** Avro can deserialize the data back into its original format using the schema information contained within the data.
4. **Code Generation:** Avro can generate code in various programming languages from a schema. This generated code helps in working with Avro data in a type-safe manner.

Avro is widely used in the Hadoop ecosystem due to its efficiency, schema evolution capabilities, and language independence, making it a popular choice for serializing data in Hadoop applications.

4.13 CASSANDRA - HADOOP INTEGRATION:

Apache Cassandra is a highly scalable, distributed NoSQL database that can handle large amounts of data across many commodity servers. Integrating Cassandra with Hadoop provides the ability to combine the advantages of a powerful database system with the extensive data processing capabilities of the Hadoop ecosystem.

Integration Strategies:

1. **Cassandra Hadoop Connector:** Cassandra provides a Hadoop integration tool called the Cassandra Hadoop Connector. It allows MapReduce jobs to read and write data to and from Cassandra.
2. **Cassandra as a Source or Sink:** Cassandra can act as a data source or sink for Apache Hadoop and Apache Spark jobs. You can configure Hadoop or Spark to read data from Cassandra tables or write results back to Cassandra.
3. **Cassandra Input/Output Formats:** Cassandra supports Hadoop Input/Output formats, allowing MapReduce jobs to directly read from and write to Cassandra tables.

Benefits of Integration:

- **Data Processing:** You can perform complex data processing tasks on data stored in Cassandra using Hadoop's distributed processing capabilities.
- **Data Aggregation:** Aggregate data from multiple Cassandra nodes using Hadoop's parallel processing, enabling large-scale data analysis.
- **Data Export and Import:** Use Hadoop to export data from Cassandra for backup or analytical purposes. Similarly, you can import data into Cassandra after processing it using Hadoop.

Integrating Cassandra and Hadoop allows businesses to leverage the best of both worlds: Cassandra's real-time, high-performance database capabilities and Hadoop's extensive data processing and analytics features. This integration enables robust, large-scale data applications for a variety of use cases.

UNIT V

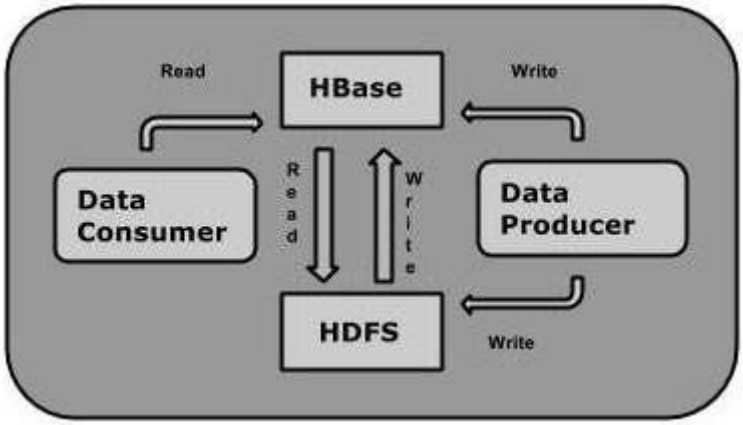
What is HBase?

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.

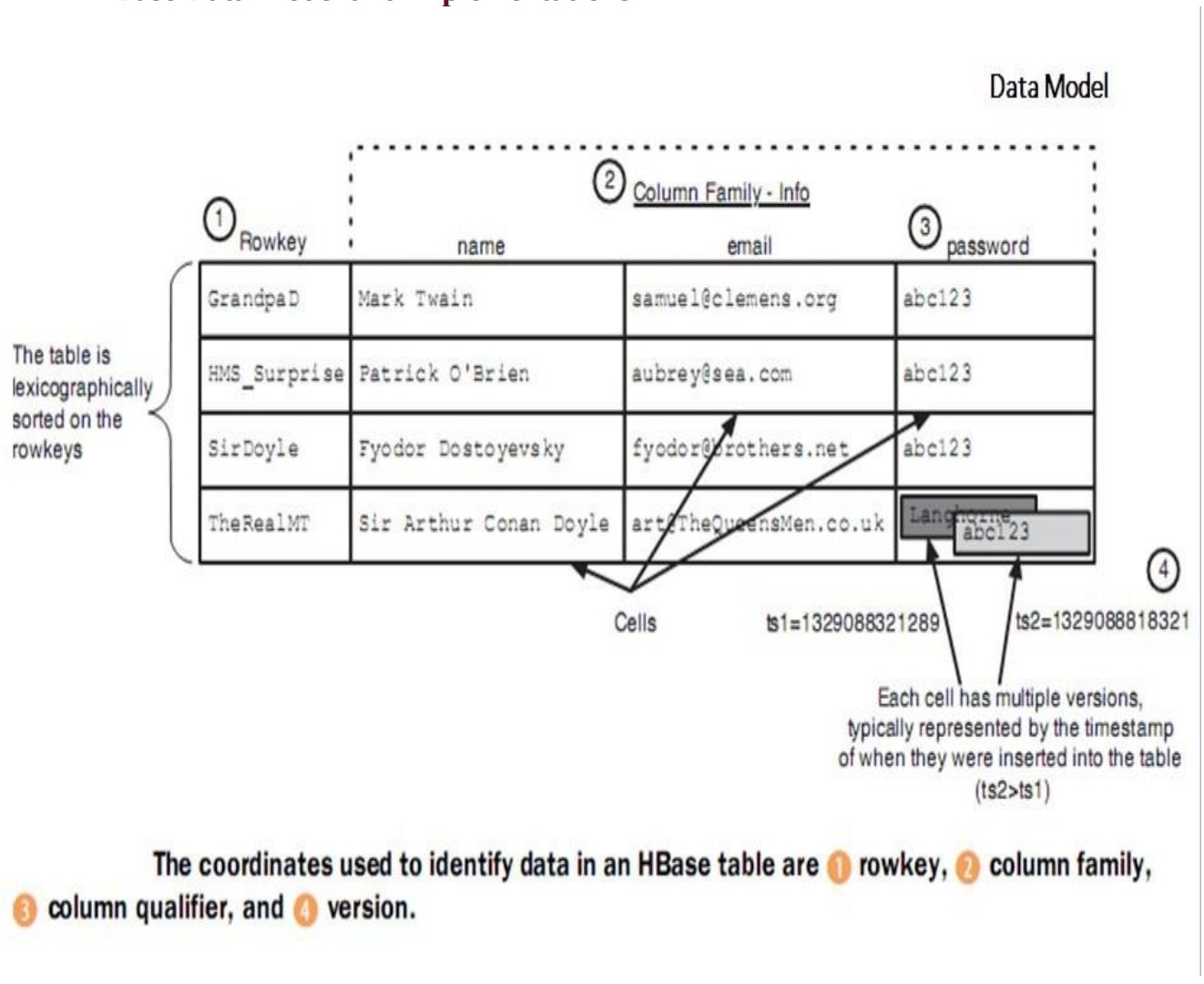


Features of Hbase

- Horizontally scalable: You can add any number of columns anytime.
- Automatic Failover: Automatic failover is a resource that allows a system administrator to automatically switch data handling to a standby system in the event of system compromise
- Integrations with Map/Reduce framework: Al the commands and java codes internally implement Map/ Reduce to do the task and it is built over Hadoop Distributed File System.
- sparse, distributed, persistent, multidimensional sorted map, which is indexed by rowkey, column key,and timestamp.
- Often referred as a key value store or column family-oriented database, or storing versioned maps of maps.

- fundamentally, it's a platform for storing and retrieving data withrandom access.
- It doesn't care about datatypes(storing an integer in one row and astring in another for the same column).
- It doesn't enforce relationships within your data.
- It is designed to run on a cluster of computers, built using commodity hardware.
-

HBase Data Model and implementations



HBase Client API?

Basically, to perform CRUD operations on HBase tables we use **Java** client API for HBase. Since HBase has a Java Native API and it is written in Java thus it offers programmatic access to DML (Data Manipulation Language).

i. Class HBase Configuration

This class adds HBase configuration files to a Configuration. It belongs to the **org.apache.hadoop.hbase** package.

ii. Method

static org.apache.hadoop.conf.Configuration create()
To create a Configuration with HBase resources, we use this method.

Class HTable in HBase Client API

An HBase internal class which represents an HBase table is HTable. Basically, to communicate with a single HBase table, we use this implementation of a table. It belongs to the **org.apache.hadoop.hbase.client** class.

a. Constructors

i. HTable()

ii. HTable(TableName tableName, ClusterConnection connection, ExecutorService pool)
We can create an object to access an HBase table, by using this constructor.

b. Methods

i. void close()
Basically, to release all the resources of the HTable, we use this method.

ii. void delete(Delete delete)
The method "void delete(Delete delete)" helps to delete the specified cells/row.

iii. boolean exists(Get get)
As specified by Get, it is possible to test the existence of columns in the table, with this method.

iv. Result get(Get get)
This method retrieves certain cells from a given row.

v. org.apache.hadoop.conf.Configuration getConfiguration()
It returns the Configuration object used by this instance.

vi. TableName getName()
This method returns the table name instance of this table.

vii. HTableDescriptor getTableDescriptor()
It returns the table descriptor for this table.

viii. byte[] getTableName()
This method returns the name of this table.

ix. void put(Put put)
We can insert data into the table, by using this method.

Pig

Apache Pig is a high-level data 跴斀ow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File

System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

Features of Apache Pig

Let's see the various uses of Pig technology.

1) Ease of programming

Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.

2) Optimization opportunities

It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than e□ciency.

3) Extensibility

A user-de□ned function is written in which the user can write their logic to execute over the data set.

4) Flexible

It can easily handle structured as well as unstructured data.

5) In-built operators

It contains various type of operators such as sort, □lter and joins.

Differences between Apache MapReduce and PIG

Apache MapReduce	Apache PIG
It is a low-level data processing tool.	It is a high-level data □ow tool.
Here, it is required to develop complex programs using Java or Python.	It is not required to develop complex programs.
It is di□cult to perform data operations in MapReduce.	It provides built-in operators to performdata operations like union, sorting and

	ordering.
It doesn't allow nested data types.	It provides nested data types like tuple, bag,and map.

Advantages of Apache Pig

- Less code - The Pig consumes less line of code to perform anyoperation.
- Reusability - The Pig code is flexible enough to reuse again.
- Nested data types - The Pig provides a useful concept of nested datatypes like tuple, bag, and map.

Pig's Data Model

Before we take a look at the operators that Pig Latin provides, we first need tounderstand Pig's data model. This includes Pig's data types, how it handles concepts such as missing data, and how you can describe your data to Pig.

Types

Pig's data types can be divided into two categories: *scalar* types, whichcontain a single value, and *complex* types, which contain other types.

Scalar Type

Pig's scalar types are simple types that appear in most programming languages. With the exception of bytearray, they are all represented in Piginterfaces by `java.lang` classes, making them easy to work with in UDFs:

int

An integer. Ints are represented in interfaces by `java.lang.Integer`. Theystore a four-byte signed integer. Constant integers are expressed as integer numbers, for example, `42`

lon

A long integer. Longs are represented in interfaces by `java.lang.Long`. They store an eight-byte signed integer. Constant longs are expressed asinteger numbers with an `L` appended, for example, `5000000000L`.

昀數oat

A floating-point number. Floats are represented in interfaces by `java.lang.Float` and use four bytes to store their value. You can findthe range of values representable by Java's `Float` type at http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html#4.2.3. Note that because this is a floating-point number, in some calculations it will lose precision. For calculations that require no loss ofprecision, you should use an int or long instead. Constant floats are expressed as a floating-point number with an `f` appended. Floating-pointnumbers can be expressed in simple format, `3.14f`, or in exponent format, `6.022e23f`.

double

A double-precision floating-point number. Doubles are represented ininterfaces by `java.lang.Double` and use eight bytes to store their value.

You can find the range of values representable by Java's `Double` type at http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html#4.2.3. Note that because this is a floating-point number, in some calculations it will lose precision. For calculations that require no loss ofprecision, you should use an int or long instead. Constant doubles are expressed as a floating-point number in either simple format, `2.71828`, or in exponent format, `6.626e-34`.

chararray

A string or character array. Chararrays are represented in interfaces by `java.lang.String`. Constant chararrays are expressed as string literalswith single quotes, for example, `'fred'`. In addition to standard alphanumeric and symbolic characters, you can express certain characters in chararrays by using backslash codes, such as `\t` for Tab and `\n` for Return. Unicode characters can be expressed as `\u` followedby their four-digit hexadecimal Unicode value. For example, the valuefor Ctrl-A is expressed as `\u0001`.

Bytearray

A blob or array of bytes. Bytearrays are represented in interfaces by aJava class `Data Byte Array` that wraps a Java `byte[]`. There is no way to specify a constant byte array.

Complex Types

Pig has three complex data types: maps, tuples, and bags. All of these types can contain data of any type, including other complex types. So it is possible to have a map where the value field is a bag, which contains a tuple where oneof the fields is a map.

Map

A *map* in Pig is a char array to data element mapping, where that element canbe any Pig type, including a complex type. The char array is called a key and is used as an index to find the element, referred to as the value

Because Pig does not know the type of the value, it will assume it is a byte array. However, the actual value might be something different. If you know what the actual type is (or what you want it to be), you can cast it; see Casts. Ifyou do not cast the value, Pig will make a best guess based on how you use the value in your script. If the value is of a type other than bytearray, Pig will figure that out at runtime and handle it. See Schemas for more information on how Pig handles unknown types.

By default there is no requirement that all values in a map must be of the sametype. It is legitimate to have a map with two keys `name` and `age`, where the valuefor `name` is a chararray and the value for `age` is an int. Beginning in Pig 0.9, a map can declare its values to all be of the same type. This is useful if you know all values in the map will be of the same type, as it allows you to avoidthe casting, and Pig can avoid the runtime type-massaging referenced in the previous paragraph

Map constants are formed using brackets to delimit the map, a hash betweenkeys and values, and a comma between key-value pairs. For example, `['name'#'bob', 'age'#55]` will create a map with two keys, “name” and “age”. The first value is a chararray, and the second is aninteger.

Tuple

A *tuple* is a fixed-length, ordered collection of Pig data elements. Tuples aredivided into *fields*, with each field containing one data element. These elements can be of any type—they do not all need to be the same type. A tupleis analogous to a row in SQL, with the fields being SQL columns. Because tuples are ordered, it is possible to refer to the fields by position; see Expressions in foreach for details. A tuple can, but is not required to, havea schema associated with it that describes each field's type and provides a name for each field. This allows Pig to check that the data in the tuple is whatthe user expects, and it allows the user to reference the fields of the tuple by name.

Tuple constants use parentheses to indicate the tuple and commas to delimitfields in the tuple. For example, `('bob', 55)` describes a tuple constant with two fields.

Bag

A *bag* is an unordered collection of tuples. Because it has no order, it is not possible to reference tuples in a bag by position. Like tuples, a bag can, but isnot required to, have a schema associated with it. In the case of a bag, the schema describes all tuples within the bag.

Bag constants are constructed using braces, with tuples in the bag separated by commas. For example, `{('bob', 55), ('sally', 52), ('john', 25)}` constructs a bag with three tuples, each with two fields.

Pig users often notice that Pig does not provide a list or set type that can store items of any type. It is possible to mimic a set type using the bag, by wrappingthe desired type in a tuple of one field. For instance, if you want to store a set of integers, you can create a bag with a tuple with one field, which is an int.

This is a bit cumbersome, but it works.

Bag is the one type in Pig that is not required to fit into memory. As you willsee later, because bags are used to store collections when grouping, bags canbecome quite large. Pig has the ability to spill bags to disk when necessary, keeping only partial sections of the bag in memory. The size of the bag is limited to the amount of local disk available for spilling the bag.

Pig Latin

The Pig Latin is a data 昀數ow language used by Apache Pig to analyze the data in Hadoop. It is a textual language that abstracts the programming from the Java MapReduce idiom into a notation.

Pig Latin Statements

The Pig Latin statements are used to process the data. It is an operator that accepts a relation as an input and generates another relation as an output.

- It can span multiple lines.
- Each statement must end with a semi-colon.
- It may include expression and schemas.
- By default, these statements are processed using multi-query execution

Pig Latin Conventions

Convention	Description
( )	The parenthesis can enclose one or more items. It can also be usedto indicate the tuple data type. Example - (10, xyz, (3,6,9))
[ ]	The straight brackets can enclose one or more items. It can alsobe used to indicate the map data type. Example - [INNER | OUTER]
{ }	The curly brackets enclose two or more items. It can also be used toindicate the bag data type Example - { block | nested_block }
...	The horizontal ellipsis points indicate that you can repeat a portionof the code. Example - cat path [path ...]

Latin Data Types

Simple Data Types

Type	Description

int	It de的椀nes the signed 32-bit integer. Example - 2
long	It de的椀nes the signed 64-bit integer. Example - 2L or 2l
的斀oat	It de的椀nes 32-bit 的斀oating point number. Example - 2.5F or 2.5f or 2.5e2f or 2.5.E2F
double	It de的椀nes 64-bit 的斀oating point number. Example - 2.5 or 2.5 or 2.5e2f or 2.5.E2F
chararray	It de的椀nes character array in Unicode UTF-8 format.Example - javatpoint
bytearray	It de的椀nes the byte array.
boolean	It de的椀nes the boolean type values. Example - true/false
datetime	It de的椀nes the values in datetime order. Example - 1970-01- 01T00:00:00.000+00:00
biginteger	It de的椀nes Java BigInteger values. Example - 5000000000000
bigdecimal	It de的椀nes Java BigDecimal values. Example - 52.232344535345

Complex Types

Type	Description
tuple	It de的椀nes an ordered set of 的椀elds. Example - (15,12)
bag	It de的椀nes a collection of tuples. Example - {(15,12), (12,15)}
map	It de的椀nes a set of key-value pairs. Example - [open#apache]

Developing and Testing Pig LatinScripts

The last few chapters focused on Pig Latin the language. Now we will turn tothe practical matters of developing and testing your scripts. This chapter covers helpful debugging tools such as `describe` and `explain`. It also covers ways to test your scripts. Information on how to make your scripts perform better will be covered in the next chapter.

Development Tools

Pig provides several tools and diagnostic operators to help you develop your applications. In this section we will explore these and also look at some tools others have written to make it easier to develop Pig with standard editors andintegrated development environments (IDEs).

Syntax Highlighting and Checking

Syntax highlighting often helps users write code correctly, at least syntactically, the first time around. Syntax highlighting packages exist for several popular editors. The packages listed in Table 7-1 were created and added at various times, so how their highlighting conforms with current PigLatin syntax varies.

Table 7-1. Pig Latin syntax highlighting packages

Tool	URL
Eclipse	http://code.google.com/p/pig-eclipse
Emacs	http://github.com/cloudera/piglatin-mode, http://sf.net/projects/pig-mode
TextMate	http://www.github.com/kevinweil/pig.tmbundle
Vim	http://www.vim.org/scripts/script.php?script_id=2186

In addition to these syntax highlighting packages, Pig will also let you check the syntax of your script without running it. If you add `-c` or `-check` to thecommand line, Pig will just parse and run semantic checks on your script.
The `-dryrun` command-line option will also check your syntax, expand anymacros and `import`s, and perform parameter substitution.

describe

`describe` shows you the schema of a relation in your script. This can be veryhelpful as you are developing your scripts. It is especially useful as you are learning Pig Latin and understanding how various operators change the data. `describe` can be applied to any relation in your script, and you can have multiple `describe`s in a script:

```
--describe.pig

divs    = load 'NYSE_dividends' as (exchange:chararray,
symbol:chararray,

            date:chararray, dividends:float);

trimmed = foreach divs generate symbol, dividends;

grpd    = group trimmed by symbol;

avgdiv  = foreach grpd generate group, AVG(trimmed.dividends);



describe trimmed;

describe grpd;

describe avgdiv;
```

```
trimmed: {symbol: chararray,dividends: float}
grpd: {group: chararray,trimmed: {(symbol: chararray,dividends:
float)}}
avgdiv: {group: chararray,double}
```

`describe` uses Pig's standard schema syntax. For information on this syntax, see Schemas. So, in this example, the relation `trimmed` has two 昀椀elds: `symbol`, which is a chararray, and `dividends`, which is a 昀氀oat. `grpd` also has two 昀椀elds, `group` (the name Pig always assigns to the group by key) and a bag `trimmed`, which matches the name of the relationthat Pig grouped to produce the bag. Tuples in `trimmed` have two 昀椀elds: `symbol` and `dividends`. Finally, in `avgdiv` there are two 昀椀elds, `group` and a double, which is the result of the `AVG` function and isunnamed.

Data Types

Hive data types are categorized in numeric types, string types, misc types, and complex types. A list of Hive data types is given below.

Integer Types

Type	Size	Range
TINYINT	1-byte signed integer	-128 to 127
SMALLINT	2-byte signed integer	32,768 to 32,767
INT	4-byte signed integer	2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Decimal Type

Type	Size	Range
FLOAT	4-byte	Single precision 昀敳oating point number
DOUBLE	8-byte	Double precision 昀敳oating point number

Date/Time Types

TIMESTAMP

- It supports traditional UNIX timestamp with optional nanosecond precision.
- As Integer numeric type, it is interpreted as UNIX timestamp in seconds.
- As Floating point numeric type, it is interpreted as UNIX timestamp inseconds with decimal precision.
- As string, it follows java.sql.Timestamp format "YYYY-MM-DDHH:MM:SS.ffffffff" (9 decimal place precision)

DATES

The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD. However, it didn't provide the time of the day. The range of Date type lies between 0000--01--01 to 9999--12--31.

String Types

STRING

The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

Varchar

The varchar is a variable length type whose range lies between 1 and 65535, which speci昀椀es that the maximum number of characters allowed in the character string.

CHAR

The char is axed-length type whose maximum length is 昀椀xed at 255.

Hive data types and file format

Apache Hive supports several familiar file formats used in Apache Hadoop. Hive can load and query different data file created by other Hadoop components such as *Pig* or *MapReduce*. In this article, we will check Apache **Hive different file formats** such as ***TextFile***, ***SequenceFile***, ***RCFile***, ***AVRO***, ***ORC*** *and* ***Parquet*** formats. Cloudera **Impala** also supports these file formats.

Hive Different File Formats

Different file formats and compression codecs work better for different data sets in Apache Hive.

Following are the Apache Hive different file formats:

- Text File
- Sequence File
- RC File
- AVRO File
- ORC File
- Parquet File

Hive Text File Format

Hive Text 昀椀le format is a default storage format. You can use the text format to interchange the data with other client application. The text file format is very common most of the applications. Data is stored in lines, with each line being a record. Each lines are terminated by a newline character (\n).

The text format is simple plane file format. You can use the compression (*BZIP2*) on the text file to reduce the storage spaces.

Create a TEXT file by add storage option as **'STORED AS TEXTFILE'** at the end of a Hive CREATE TABLE command.

Hive Text File Format Examples

Below is the Hive CREATE TABLE command with storage format specification:

```
Create table textfile_table

(column_specs)

Store as textfile;
```

Hive Sequence File Format

Sequence 附椭es are Hadoop flat files which stores values in binary key-value pairs. The sequence files are in binary format and these files are able to split. The main advantages of using sequence file is to merge two or more files into one file.

Create a sequence file by add storage option as **'STORED AS SEQUENCEFILE'** at the end of a Hive CREATE TABLE command.

Hive Sequence File Format Example

Below is the Hive CREATE TABLE command with storage format specification:

create table sequencefile_table

(column_specs)

Stored as sequencefile_table

Hive AVRO File Format

AVRO is open source project that provides data serialization and data exchange services for Hadoop. You can exchange data between Hadoop ecosystem and program written in any programming languages. Avro is one of the popular file format in Big Data Hadoop based applications.

Create AVRO file by specifying **'STORED AS AVRO'** option at the end of a CREATE TABLE Command.

Hive AVRO File Format Example

Below is the Hive CREATE TABLE command with storage format specification:

create table avro_table

(column_specs)

Stored as aveo:

HiveQL: Data Definition

HiveQL is the Hive query language. Like all SQL dialects in widespreaduse, it doesn't fully conform to any particular revision of the ANSI SQL standard. It is perhaps closest to MySQL's dialect, but with significant differences. Hive offers no support for row-level inserts, updates, and deletes. Hive doesn't support transactions. Hive adds extensions to provide better performance in the context of Hadoop and to integrate with custom extensions and even external programs.

Still, much of HiveQL will be familiar. This chapter and the ones that follow discuss the features of HiveQL using representative examples.In some cases, we will briefly mention details for completeness, thenexplore them more fully in later chapters.

This chapter starts with the so-called *data definition language* parts of HiveQL, which are used for creating, altering, and dropping databases,tables, views, functions, and indexes. We'll discuss databases and tables in this chapter, deferring the discussion of views until Chapter 7,indexes until Chapter 8, and functions until Chapter 13.
We'll also discuss the `SHOW` and `DESCRIBE` commands for listing anddescribing items as we go.
Subsequent chapters explore the *data manipulation language* parts of HiveQL that are used to put data into Hive tables and to extract data tothe filesystem, and how to explore and manipulate data with queries, grouping, filtering, joining, etc.

Databases in Hive

The Hive concept of a database is essentially just
a *catalog* or *namespace* of tables. However, they are very useful for larger clusters with multiple teams and users, as a way of avoiding table name collisions. It's also common to use databases to organizeproduction tables into logical groups.

If you don't specify a database, the `default` database is used.

The simplest syntax for creating a database is shown in the followingexample:

```
hive> CREATE DATABASE financials;
```

Hive will throw an error if `financials` already exists. You can suppressthese warnings with this variation:

```
hive> CREATE DATABASE IF NOT EXISTS financials;
```

While normally you might like to be warned if a database of the samename already exists, the `IF NOT EXISTS` clause is useful for scripts that should create a database on-the-fly, if necessary, before proceeding.

You can also use the keyword `SCHEMA` instead of `DATABASE` in all thedatabase-related commands.

At any time, you can see the databases that already exist as follows:

```
hive> SHOW DATABASES;
default
financials

hive> CREATE DATABASE human_resources;

hive> SHOW DATABASES;
default
financials
human_resources
```

If you have a lot of databases, you can restrict the ones listed using a *regular expression*, a concept we'll explain in LIKE and RLIKE, if it isnew to you. The following example lists only those databases that startwith the letter `h` and end with any other characters (the `.*` part):

```
hive> SHOW DATABASES LIKE 'h.*';
human_resources
hive> ...
```

Hive will create a directory for each database. Tables in that database will be stored in subdirectories of the database directory. The exceptionis tables in the `default` database, which doesn't have its own directory.

The database directory is created under a top-level directory specified by the property `hive.metastore.warehouse.dir`, which we discussed in LocalMode Configuration and Distributed and Pseudodistributed Mode Configuration. Assuming you are using the default value for this property, */user/hive/warehouse*, when the `financials` database is created, Hive will create the directory */user/hive/warehouse/financials.db*. Note the *.db* extension.

You can override this default location for the new directory as shown inthis example:

```
hive> CREATE DATABASE financials
    > LOCATION '/my/preferred/directory';
```

You can add a descriptive comment to the database, which will beshown by the `DESCRIBE DATABASE <database>` command.

```
hive> CREATE DATABASE financials
    > COMMENT 'Holds all financial tables';

hive> DESCRIBE DATABASE financials;
financials    Holds all financial tables
  hdfs://master-server/user/hive/warehouse/financials.db
```