Studies in Big Data 11

Hrushikesha Mohanty
Prachet Bhuyan
Deepak Chenthati *Editors*

Big Data

A Primer

Springer

Studies in Big Data

Volume 11

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series "Studies in Big Data" (SBD) publishes new developments and advances in the various areas of Big Data- quickly and with a high quality. The intent is to cover the theory, research, development, and applications of Big Data, as embedded in the fields of engineering, computer science, physics, economics and life sciences. The books of the series refer to the analysis and understanding of large, complex, and/or distributed data sets generated from recent digital sources coming from sensors or other physical instruments as well as simulations, crowd sourcing, social networks or other internet transactions, such as emails or video click streams and other. The series contains monographs, lecture notes and edited volumes in Big Data spanning the areas of computational intelligence incl. neural networks, evolutionary computation, soft computing, fuzzy systems, as well as artificial intelligence, data mining, modern statistics and Operations research, as well as self-organizing systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/11970

Hrushikesha Mohanty · Prachet Bhuyan
Deepak Chenthati
Editors

Big Data

A Primer

Springer

Editors
Hrushikesha Mohanty
School of Computer and Information Sciences
University of Hyderabad
Hyderabad
India

Prachet Bhuyan
School of Computer Engineering
KIIT University
Bhubaneshwar, Odisha
India

Deepak Chenthati
Teradata India Private Limited
Hyderabad
India

ISSN 2197-6503 ISSN 2197-6511 (electronic)
Studies in Big Data
ISBN 978-81-322-2493-8 ISBN 978-81-322-2494-5 (eBook)
DOI 10.1007/978-81-322-2494-5

Library of Congress Control Number: 2015941117

Springer New Delhi Heidelberg New York Dordrecht London
© Springer India 2015
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer (India) Pvt. Ltd. is part of Springer Science+Business Media
(www.springer.com)

Preface

Rapid developments in communication and computing technologies have been the driving factors in the spread of the internet technology. This technology is able to scale up and reach out to more and more people. People at opposite sides of the globe are able to remain connected to each other because of the connectivity that the internet is able to provide now. Getting people together through the internet has become more realistic than getting them together physically at one place. This has led to the emergence of cyber society, a form of human society that we are heading for with great speed. As is expected, this has also affected different activities from education to entertainment, culture to commerce, goodness (ethics, spiritual) to governance. The internet has become a platform of all types of human interactions. Services of different domains, designed for different walks of people, are being provided via the internet. Success of these services decisively depends on understanding people and their behaviour over the internet. For example, people may like a particular kind of service due to many desired features the service has. Features could be quality of service like response time, average availability, trust and similar factors. So service providers would like to know of consumer preferences and requirements for designing a service, so as to get maximum returns on investment. On the other side, customers would require enough information to select the best service provider for their needs. Thus, decision-making is key to cyber society. And, informed decisions can only be made on the basis of good information, i.e. information that is both qualitatively and quantitatively sufficient for decision-making.

Fortunately for cyber society, through our presence on the internet, we generate enough data to garner a lot of meaningful information and patterns. This information is in the form of metaphorical due to footsteps or breadcrumbs that we leave on the internet through our various activities. For example, social networking services, e-businesses and search engines generate huge data sets every second of the day. And these data sets are not only voluminous but also in various forms such as picture, text and audio. This great quantum of data sets is collectively christened *big data* and is identified by its three special features *velocity*, *variety* and *volume*.

Collection and processing of big data are topics that have drawn considerable attention of concerned variety of people ranging from researchers to business makers. Developments in infrastructure such as grid and cloud technology have given a great impetus to big data services. Research in this area is focusing on big data as a service and infrastructure as a service. The former looks at developing algorithms for fast data access, processing as well as inferring pieces of information that remain hidden. To make all this happen, internet-based infrastructure must provide the backbone structures. It also needs an adaptable architecture that can be dynamically configured so that fast processing is possible by making use of optimal computing as well as storage resources. Thus, investigations on big data encompass many areas of research, including parallel and distributed computing, database management, software engineering, optimization and artificial intelligence. The rapid spread of the internet, several governments' decisions in making of smart cities and entrepreneurs' eagerness have invigorated the investigation on big data with intensity and speed. The efforts made in this book are directed towards the same purpose.

Goals of the Book

The goal of this book is to highlight the issues related to research and development in big data. For this purpose, the chapter authors are drawn from academia as well as industry. Some of the authors are actively engaged in the development of products and customized big data applications. A comprehensive view on six key issues is presented in this book. These issues are big data management, algorithms for distributed processing and mining patterns, management of security and privacy of big data, SLA for big data service and, finally, big data analytics encompassing several useful domains of applications. However, the issues included here are not completely exhaustive, but the coverage is enough to unfold the research as well as development promises the area holds for the future. Again for the purpose, the Introduction provides a survey with several important references. Interested readers are encouraged to take the lead following these references.

Intended Audience

This book promises to provide insights to readers having varied interest in big data. It covers an appreciable spread of the issues related to big data and every chapter intends to motivate readers to find the specialities and the challenges lie within. Of course, this is not a claim that each chapter deals an issue exhaustively. But, we sincerely hope that both conversant and novice readers will find this book equally interesting.

In addition to introducing the concepts involved, the authors have made attempts to provide a lead to realization of these concepts. With this aim, they have presented algorithms, frameworks and illustrations that provide enough hints towards system realization. For emphasizing growing trends on big data application, the book includes a chapter which discusses such systems available on the public domain. Thus, we hope this book is useful for undergraduate students and professionals looking for an introduction to big data. For graduate students intending to take up research in this upcoming area, the chapters with advanced information will also be useful.

Organization of the Book

This book has seven chapters. Chapter "Big Data: An Introduction" provides a broad review of the issues related to big data. Readers new to this area are encouraged to read this chapter first before reading other chapters. However, each chapter is independent and self-complete with respect to the theme it addresses.

Chapter "Big Data Architecture" lays out a universal data architecture for reasoning with all forms of data. Fundamental to big data analysis is big data management. The ability to collect, store and make available for analysis the data in their native forms is a key enabler for the science of analysing data. This chapter discusses an iterative strategy for data acquisition, analysis and visualization.

Big data processing is a major challenge to deal with voluminous data and demanding processing time. It also requires dealing with distributed storage as data could be spread across different locations. Chapter "Big Data Processing Algorithms" takes up these challenges. After surveying solutions to these problems, the chapter introduces some algorithms comprising random walks, distributed hash tables, streaming, bulk synchronous processing and MapReduce paradigms. These algorithms emphasize the usages of techniques, such as bringing application to data location, peer-to-peer communications and synchronization, for increased performance of big data applications. Particularly, the chapter illustrates the power of the Map Reduce paradigm for big data computation.

Chapter "Big Data Search and Mining" talks of mining the information that big data implicitly carries within. Often, big data appear with patterns exhibiting the intrinsic relations they hold. Unearthed patterns could be of use for improving enterprise performances and strategic customer relationships and marketing. Towards this end, the chapter introduces techniques for big data search and mining. It also presents algorithms for social network clustering using the topology discovery technique. Further, some problems such as sentiment detection on processing text streams (like tweets) are also discussed.

Security is always of prime concern. Security lapses in big data could be higher due to its high availability. As these data are collected from different sources, the vulnerability for security attacks increases. Chapter "Security and Privacy of Big Data" discusses the challenges, possible technologies, initiatives by stakeholders and emerging trends with respect to security and privacy of big data.

The world today, being instrumented by several appliances and aided by several internet-based services, generates very high volume of data. These data are useful for decision-making and furthering quality of services for customers. For this, data service is provided by big data infrastructure to receive requests from users and to accordingly provide data services. These services are guided by Service Level Agreement (SLA). Chapter "Big Data Service Agreement" addresses issues on SLA specification and processing. It also introduces needs for negotiation to avail data services. This chapter proposes a framework for SLA processing.

Chapter "Applications of Big Data" introduces applications of big data in different domains including banking and financial services. It sketches scenarios for the digital marketing space.

Acknowledgments

The genesis of this book goes to 11th International Conference on Distributed Computing and internet Technology (ICDCIT) held in February 2015. Big data was a theme for industry symposium held as a prelude to the main conference. The authors of three chapters in this book presented their ideas at the symposium. Editors took the feedback from participants and conveyed the same to the chapter authors for refining their contents.

In preparation of this book, we received help from different quarters. Hrushikesha Mohanty expresses his sincere thanks to the School of Computer and Information Sciences, University of Hyderabad, for providing excellent environment for carrying out this work. I also extend my sincere thanks to Dr. Achyuta Samanta, Founder KIIT University, for his inspiration and graceful support for hosting the ICDCIT series of conferences. Shri. D.N. Dwivedy of KIIT University deserves special thanks for making it happen. The help from ICDCIT organizing committee members of KIIT University is thankfully acknowledged. Deepak Chenthati and Prachet Bhuyan extend their thanks to their respective organizations Teradata India Pvt. Ltd. and KIIT University. Thanks to Shri Abhayakumar, graduate student of SCIS, University of Hyderabad, for his help in carrying out some pressing editing work.

Our special thanks to chapter authors who, despite their busy schedules, contributed chapters for this book. We are also thankful to Springer for publishing this book. In Particular, for their support and consideration for the issues we have been facing while preparing the manuscript.

Hyderabad
March 2015

Hrushikesha Mohanty
Prachet Bhuyan
Deepak Chenthati

Contents

Editors and Contributors

About the Editors

Hrushikesha Mohanty is currently a professor at School of Computer and Information Sciences, University of Hyderabad. He received his Ph.D. from IIT Kharagpur. His research interests include distributed computing, software engineering and computational social science. Before joining University of Hyderabad, he worked at Electronics Corporation of India Limited for developing strategic real-time systems. Other than computer science research publications, he has penned three anthologies of Odia poems and several Odia short stories.

Prachet Bhuyan is presently an associate professor at KIIT University. He completed his bachelor and master degrees in computer science and engineering from Utkal University and VTU, Belgaum, respectively. His research interests include service-oriented architecture, software testing, soft computing and grid computing. Before coming to KIIT, he has served in various capacities at Vemana Institute of Technology, Bangalore, and abroad in Muscat, Oman. He has several publications in indexed journals as well as conferences. He has been generously awarded by several organisations including IBM for his professional competence.

Deepak Chenthati is currently a senior software engineer at Teradata India Private Limited. His Industry experience includes working on Teradata massively parallel processing systems, Teradata server management, Teradata JDBC drivers and administration of Teradata internal tools and confluence tool stack. His research interests include Web services, Teradata and database management. He is currently pursuing his doctorate from JNTU Hyderabad. He received his master and bachelor degrees in computer science from University of Hyderabad and Sri Venkateswaraya University, respectively.

Contributors

Hareesh Boinepelli Teradata India Pvt. Ltd., Hyderabad, India

Raoul P. Jetley ABB Corporate Research, Bangalore, India

VenkataSwamy Martha @WalmartLabs, Sunnyvale, CA, USA

Hrushikesha Mohanty School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India

P. Radha Krishna Infosys Labs, Infosys Limited, Hyderabad, India

Srini Ramaswamy US ABB, Cleveland, USA

Bhashyam Ramesh Teradata Corporation, Dayton, USA

Sithu D. Sudarsan ABB Corporate Research, Bangalore, India

Supriya Vaddi School of Computer and Information Sciences, University of Hyderabad, Hyderabad, India

Acronyms

AAA	Authentication, authorization and access control
ACID	Atomicity, consistency, isolation and durability
BI	Business intelligence
BSP	Bulk synchronous parallel
CIA	Confidentiality, integrity and availability
CII	Critical information infrastructure
COBIT	Control objectives for information and related technology
CPS	Cyber-physical system
DHT	Distributed hash tables
DLP	Data loss prevention
DN	Data node
EDVAC	Electronic discrete variable automatic computer
EDW	Enterprise data warehouse
ER	Entity relation
ETL	Extract-transform-load
HDFS	Hadoop distributed file system
IaaS	Infrastructure as a service
iMapReduce	Iterative MapReduce
IoT	Internet of things
kNN	k nearest neighbour
MOA	Massive online analysis
MPI	Message passing interface
MR	MapReduce
NN	Name node
NSA	National security agency
PaaS	Platform as a service
PAIN	Privacy, authentication, integrity and non-repudiation
PII	Personally identifiable information
POS	Parts of speech
RWR	Random walk with restart
SaaS	Software as a service
SLA	Service-level agreement

SOA	Service-oriented architecture
SRG	Service relation graph
WEKA	Waikato environment for knowledge analysis
YARN	Yet another resource negotiator

Big Data: An Introduction

Hrushikesha Mohanty

Abstract The term big data is now well understood for its well-defined characteristics. More the usage of big data is now looking promising. This chapter being an introduction draws a comprehensive picture on the progress of big data. First, it defines the big data characteristics and then presents on usage of big data in different domains. The challenges as well as guidelines in processing big data are outlined. A discussion on the state of art of hardware and software technologies required for big data processing is presented. The chapter has a brief discussion on the tools currently available for big data processing. Finally, research issues in big data are identified. The references surveyed for this chapter introducing different facets of this emergent area in data science provide a lead to intending readers for pursuing their interests in this subject.

Keywords Big data applications · Analytics · Big data processing architecture · Big data technology and tools · Big data research trends

1 Big Data

"Big data" the term remains ill-defined if we talk of data volume only. It gives an impression before data size was always small. Then, we run into problem of defining something small and big. How much data can be called big—the question remains unanswered or even not understood properly. With relational database technology, one can really handle huge volume of data. This makes the term "big data" a misnomer.

Days of yesteryears were not as machine-driven as we see it today. Changes were also not as frequent as we find now. Once, data repository defined, repository was

H. Mohanty (✉)
School of Computer and Information Sciences, University of Hyderabad,
Gachhibowli 500046, Hyderabad, India
e-mail: hmcs_hcu@yahoo.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_1

used for years by users. Relational database technology thus was at the top for organisational and corporate usages. But, now emergent data no longer follow a defined structure. Variety of data comes in variety of structures. All accommodating in a defined structure is neither possible nor prudent to do so for different usages.

Our world is now literally swamped with several digital gadgets ranging from wide variety of sensors to cell phones, as simple as a cab has several sensors to throw data on its performance. As soon as a radio cab is hired, it starts sending messages on travel. GPS fitted with cars and other vehicles produce a large amount of data at every tick of time. Scenario on roads, i.e. traffic details, is generated in regular intervals to keep an eye on traffic management. Such scenarios constitute data of traffic commands, vehicles, people movement, road condition and much more related information. All these information could be in various forms ranging from visual, audio to textual. Leave aside very big cities, in medium-sized city with few crores of population, the emerging data could be unexpectedly large to handle for making a decision and portraying regular traffic conditions to regular commuters.

Internet of things (IoT) is the new emerging world today. Smart home is where gadgets exchange information among themselves for getting house in order like sensors in a refrigerator on scanning available amount of different commodities may make and forward a purchase list to a near by super market of choice. Smart cities can be made intelligent by processing the data of interest collected at different city points. For example, regulating city traffic in pick time such that pollution levels at city squares do not cross a marked threshold. Such applications need processing of a huge data that emerge at instant of time.

Conducting business today unlike before needs intelligent decision makings. More to it, decision-making now demands instant actions as business scenario unfolds itself at quick succession. This is so for digital connectivity that makes business houses, enterprises, and their stakeholders across the globe so closely connected that a change at far end instantly gets transmitted to another end. So, the business scenario changes in no time. For example, a glut in crude oil supply at a distributor invites changes in status of oil transport, availability at countries sourcing the crude oil; further, this impacts economy of these countries as the productions of its industries are badly affected. It shows an event in a business domain can quickly generate a cascade of events in other business domains. A smart decision-making for a situation like this needs quick collection as well as processing of business data that evolve around.

Internet connectivity has led to a virtual society where a person at far end of the globe can be a person like your next-door neighbour. And number of people in one's friend list can out number to the real number of neighbours one actually has. Social media such as Twitter, Facebook, Instagram and many such platforms provide connectivity for each of its members for interaction and social exchanges. They exchange messages, pictures, audio files, etc. They talk on various issues ranging from politics, education, research to entertainment. Of course, unfortunately such media are being used for subversive activities. Every moment millions of people on social media exchanges enormous amount of information. At times for different usages ranging from business promotions to security enhancement,

monitoring and understanding data exchanged on social media become essential. The scale and the speed at which such data are being generated are mind bugging.

Advancement in health science and technology has been so encouraging in today's world that healthcare can be customised to personal needs. This requires monitoring of personal health parameters and based on such data prescription is made. Wearable biosensors constantly feed real-time data to healthcare system and the system prompts to concerned physicians and healthcare professionals to make a decision. These data can be in many formats such as X-ray images, heartbeat sounds and temperature readings. This gives an idea for a population of a district or a city, the size of data, a system needs to process, and physicians are required to handle.

Research in biosciences has taken up a big problem for understanding biological phenomena and finding solution to disorders that at times set in. The research in system biology is poised to process huge data being generated from coding information on genes of their structure and behaviour. Researchers across the globe need access to each others data as soon as such data are available. As in other cases these data are available in many forms. And for applications like study on new virus and its spread require fast processing of such data. Further, visualisation of folds that happen to proteins is of importance to biologists as they understand nature has preserved gold mine of information on life at such folds.

Likewise many applications now need to store and process data in time. In year 2000, volume of data stored in the world is of size 800,000 petabytes. It is expected to reach 35 zettabytes by the year 2020. These figures on data are taken from book [1]. However, the forecast will change with growing use of digital devices. We are storing data of several domains ranging from agriculture, environment, house holdings, governance, health, security, finance, meteorological and many more like. Just storing such data is of no use unless data are processed and decisions are made on the basis of such data. But in reality making use of such large data is a challenge for its typical characteristics [2]. More, the issues are with data capture, data storage, data analysis and data visualisation.

Big data looks for techniques not only for storage but also to extract information hidden within. This becomes difficult for the very characteristics of big data. The typical characteristics that hold it different than traditional database systems include *volume, variety, velocity and value*. The term *volume* is misnomer for its vagueness in quantifying the size that is fit to label as big data. Data that is not only huge but expanding and holding patterns to show the order exist in data, is generally qualifying volume of big data. *Variety* of big data is due to its sources of data generation that include sensors, smartphones or social networks. The types of data emanate from these sources include video, image, text, audio, and data logs, in either structured or unstructured format [3]. Historical database dealing with data of past has been studied earlier, but big data now considers data emerging ahead along the timeline and the emergence is rapid so *Velocity* of data generation is of prime concern. For example, in every second large amount of data are being generated by social networks over internet. So in addition to volume, velocity is also a dimension for such data [4]. *Value* of big data refers to the process of extracting hidden

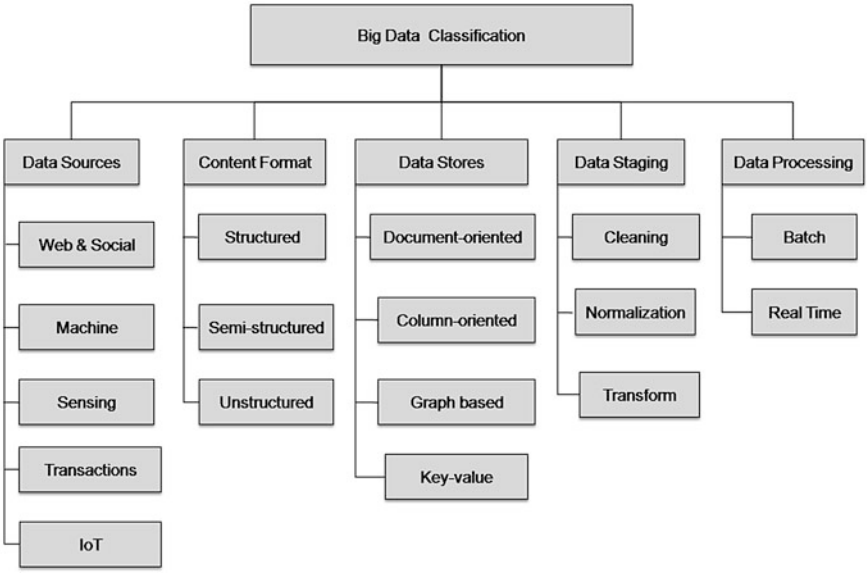

Fig. 1 Big data classification

information from emerging data. A survey on generation of big data from mobile applications is presented in [5].

Classification of big data from different perspectives as presented in [6] is presented in Fig. 1. The perspectives considered are data sources, content format, data stores, data staging, and data processing. The sources generating data could be web and social media on it, different sensors reading values of parameters that changes as time passes on, internet of things, various machinery that throw data on changing subfloor situations and transactions that are carried out in various domains such as enterprises and organisations for governance and commercial purposes. Data staging is about preprocessing of data that is required for processing for information extraction. From data store perspective, here the concern is about the way data stored for fast access. Data processing presents a systemic approach required to process big data. We will again touch upon these two issues later in Sect. 3.

Having an introduction on big data, next we will go for usages of these data in different domains. That gives an idea why the study on big data is important for both business as well as academic communities.

2 Big Data as a Service

In modern days, business has been empowered by data management. In 1970s, RDBMS (Relational Database Management System) has been successful in handling large volume of data for query and repository management. The next level of

data usage has been since 1990s, by making use of statistical as well as data mining techniques. This has given rise to the first generation of Business Intelligence and Analytics (BI&A). Major IT vendors including Microsoft, IBM, Oracle, and SAP have developed BI platforms incorporating most of these data processing and analytical technologies.

On advent of Web technology, organisations are putting businesses online by making use of e-commerce platforms such as Flipkart, Amazon, eBay and are searched for by websearch engines like Google. The technologies have enabled direct interactions among customers and business houses. User(IP)-specific information and interaction details being collected by web technologies (through cookies and service logs) are being used in understanding customer's needs and new business opportunities. Web intelligence and web analytics make Web 2.0-based social and crowd-sourcing systems.

Now social media analytics provide unique opportunity for business development. Interactions among people on social media can be traced and business intelligence model be built for two-way business transactions directly instead of traditional one-way transaction between business-to-customer [7]. We are need of scalable techniques in information mining (e.g. information extraction, topic identification, opinion mining, question-answering, event detection), web mining, social network analysis, and spatial-temporal analysis, and these need to gel well with existing DBMS-based techniques to come up with BI&A 2.0 systems. These systems use a variety of data emanating from different sources in different varieties and at different intervals. Such a collection of data is known as big data. Data in abundance being accompanied with analytics can leverage opportunities and make high impacts in many domain-specific applications [8]. Some such selective domains include e-governance, e-commerce, healthcare, education, security and many such applications that require boons of data science.

Data collected from interactions on social media can be analysed to understand social dynamics that can help in delivering governance services to people at right time and at right way resulting to good governance. Technology-assisted governance aims to use data services by deploying data analytics for social data analysis, visualisation, finding events in communities, extracting as well as forecasting emerging social changes and increase understanding of human and social processes to promote economic growth and improved health and quality of life.

E-commerce has been greatly benefited in making use of data collected from social media analytics for customer opinions, text analysis and sentiment analysis techniques. Personalised recommender systems are now a possibility following long-tail marketing by making use of data on social relations and choices they make [9]. Various data processing analytics based on association rule mining, database segmentation and clustering, anomaly detection, and graph mining techniques are being used and developed to promote data as a service in e-commerce applications.

In healthcare domain, big data is poised to make a big impact resulting to personalisation of healthcare [10]. For this objective, healthcare systems are planning to make use of different data the domain churns out every day in huge quantity. Two main sources that generate a lot of data include genomic-driven study, probe-driven

treatment and health management. Genomic-driven big data includes genotyping, gene expression and sequencing data, whereas probe-driven health care includes health-probing images, health-parameter readings and prescriptions. Health-management data include electronic health records and insurance records. The health big data can be used for hypothesis testing, knowledge discovery as well as innovation. The healthcare management can have a positive impact due to healthcare big data. A recent article [11] discusses on big data impact on host trait prediction using meta-genomic data for gastrointestinal diseases.

Security has been a prime concern and it grows more, the more our society opens up. Security threats emanating across boundary and even from within boundary are required to be analysed and understood [12]. And the volume of such information flowing from different agencies such as intelligence, security and public safety agencies is enormous. A significant challenge in security IT research is the information stovepipe and overload resulting from diverse data sources, multiple data formats and large data volumes. Study on big data is expected to contribute to success in mitigating security threats. Big data technology including such as criminal association rule mining and clustering, criminal network analysis, spatial-temporal analysis and visualisation, multilingual text analytics, sentiment and affect analysis, and cyber attacks analysis and attribution should be considered for security informatics research.

Scientific study has been increasingly collaborative. Particularly, sharing of scientific data for research and engineering data for manufacturing has been a modern trend, thanks to internet providing a pervading infrastructure for doing so [13]. Big data aims to advance the core scientific and technological research by analysing, visualising, and extracting useful information from large, diverse, distributed and heterogeneous data sets. The research community believes this will accelerate the progress of scientific discovery and innovation leading to new fields of enquiry that would not otherwise be possible. Particularly, currently we see this happening in fields of research in biology, physics, earth science, environmental science and many more areas needing collaborative research of interdisciplinary nature.

The power of big data, i.e. its impact in different domains, is drawn from analytics that extracts information from collected data and provide services to intended users. In order to emphasise on vast scope of impending data services, let us discover some analytics of importance. *Data Analytics* are designed to explore and leverage unique data characteristics, from sequential/temporal mining and spatial mining, to data mining for high-speed data streams and sensor data. Analytics are formulated based on strong mathematical techniques including statistical machine learning, Bayesian networks, hidden Markov models, support vector machine, reinforcement learning and ensemble models. Data analytics are also looking into process mining from series of data collected in sequence of time. Privacy security concerned data analytics ensure anonymity as well as confidentiality of a data service. *Text Analytics* aims at event detection, trend following, sentiment analysis, topic modelling, question-answering and opinion mining. Other than traditional soft computing and statistical techniques, text analytics take the help of several well-researched natural language processing techniques in parsing

and understanding texts. Analytics for multilingual text translations follow language mapping techniques. Basically text analytics takes root in information retrieval and computational linguistics. Information retrieval techniques including documentation representation and query processing have become so relevant for big data. Well-developed techniques in that area including vector-space model, boolean retrieval model, and probabilistic retrieval model can help in design of text analytics [14]. Computational linguistics techniques for lexical acquisition, word sense disambiguation, part-of-speech tagging (POST) and probabilistic context-free grammars are also important in design of text analytics [15]. *Web analytics* aim to leverage internet-based services based on server virtualisation, scheduling, QoS monitoring, infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and service-level agreement monitoring, service check pointing and recovery. *Network analytics* on social networks look for link prediction, topic detection, finding influencing node, sentiment analysis, hate monger nodes and monitoring of special activities of business and security concerns. Smart cell phone use has thrown up great expectation in business world for pushing services on cell phones. Light-weight *Mobile analytics* are offered as apps on cell phones. These app applications form an ecosystem for users of several domains. Providing location-based services is the specialisation of mobile analytics. Some of these analytics can predict presence of a person at a place at a given time, possible co-occurrence and prediction of mobility of a person. It can also perform locational service search along with mobility. On cell phone, gaming is also favourite analytics. Analytics of different domains have become so popular that volunteers have started contributing particularly in apps development. The types of analytics, their characteristics and possible applications are given in a tabular form Table 1 [16].

Table 1 Analytics and characteristics

Types of analytics	Characteristics	Examples
Operational analytics	• Complex analytic queries	Real-time fraud detection, ad serving, high-frequency trading
	• Performed on the fly as part of operational business processes	
	• Concurrent, high data volume of operational transactions	
Deep analytics	• Typically multisource	Gaining insight from collected smart utility meter data
	• Non-operational transactions data	
	• Complex data mining and predictive analytics	
	• Real-time or near real-time responses	
	• Uses map reduce-type framework, columnar databases, and in-memory analysis	
Time series analytics	• Analytics with the concept of a transaction: an element that has a time, at least one numerical value, and metadata	Algorithmic trading
Insight intelligence analytics	Analysis over a vast complex and diverse set of structured and unstructured information	

3 Big Data Processing

Big data as told in previous section offers a bountiful of opportunities. But, opportunities always come with challenges. The challenge with big data is its enormous volume. But, taming the challenges and harnessing benefit always have been with scientific tamper. In this section, first we will touch upon challenges big data processing faces and then will talk of broad steps the processing follows, while discussing, we will take help of a conceptual framework for easy understanding. However, some of the prevailing architectures for big data processing will be referred in next section while surveying on big data technology.

Recent conservative studies estimate that enterprise server systems in the world have processed 9.57×10^{21} bytes of in year 2008 [17]. Collaborative scientific experiments generate large data. A bright example of kind is "The Large Hadron Collider" at CERN experiment that will produce roughly 15 petabytes of data annually, enough to fill more than 1.7 million dual-layer DVDs per year [18]. YouTube the popular medium is used heavily for both uploading and viewing. A conservative number as reported at [19] says 100 h of video are being uploaded in every minute while 135,000 h is watched. Multimedia message traffic counts 28.000 MMS every second [20]. Roughly, 46 million mobile apps were downloaded in 2012 and each also collects data. Twitter contributes to big data nearly 9100 tweets every second. From e-commerce domain we can consider eBay that processes more than 100 petabytes of data every day [21]. The volume of big data looks like a data avalanche posing several challenges. Big data service faces challenges for its very characteristics and has generated enormous expectations. First, we will discuss on a broad outline of data service and then refer to some important challenges the service faces.

3.1 Processing Steps

Big data service process has few steps starting from Data acquisition, Data staging, Data analysis and application analytics processing and visualisation. Figure 2 presents a framework for big data processing that models at higher level, the working of such a system. Source of data could be internet-based applications and databases that store organisational data. On acquiring data, preprocessing stage called data staging includes removal of unrequired and incomplete data [22].

Then, it transforms data structure to a form that is required for analysis. In the process, it is most important to do data normalisation so that data redundancy is avoided. Normalised data then are stored for processing. Big users from different domains such as social computing, bioscience, business domains and environment to space science look forward information from gathered data. Analytics corresponding to an application are used for the purpose. These analytics being invoked in turn take the help of data analysis technique to scoop out information hiding in

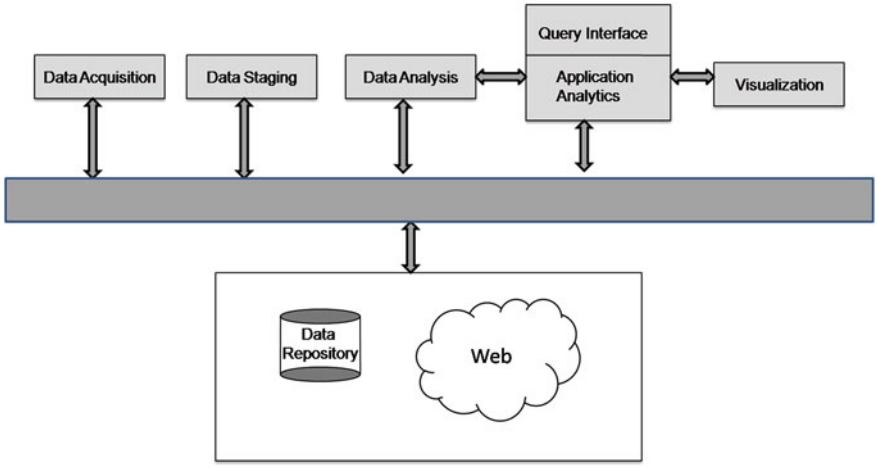

Fig. 2 Big data processing

big data. Data analysis techniques include machine learning, soft computing, statistical methods, data mining and parallel algorithms for fast computation. Visualisation is an important step in big data processing. Incoming data, information while in processing and result outcome are often required to visualise for understanding because structure often holds information in its folds; this is more true in genomics study.

3.2 Challenges

Big data service is hard for both hardware and software limitations. We will list some of these limitations that are intuitively felt important. Storage device has become a major constraint [23] for the presently usual HDD: Hard Disk Drive with random access technology used for data storage is found restrictive particularly for fast input/output transmission [24] that is demanding for big data processing. solid-state device (SSD) [25] and phase change memory (PCM) [26] are the leading technology though promising but far from reality.

Other than storage limitation, there could be algorithmic design limitation in terms of defining proper data structures that are amenable for fast access for data management. There is a need for optimised design and implementations of indexing for fast accessing of data [27]. Novel idea on key-value stores [28] and database file system arrangement are challenge for big data management [29, 30].

Communication is almost essential with big data service for both data acquisition and service delivery for both are usually carried out on internet. Big data service requires large bandwidth for data transmission. Loss of data during transmission is

always of possibility. In case of such loss, maintaining data integrity is a challenge [31]. More to it, there is always data security problem [32]. Cloud environment now has taken up big data storage issues. Many big data solutions are appearing with cloud technology [33, 34].

Demanding computational power has been a part of big data service. Data analysis and visualisation both require high computing power. As the data size is scaling up, the need for computing power is exponentially increasing. Although, the clock cycle frequency of processors is doubling following Moore's Law, the clock speeds still highly lag behind. However, development of multicore processor with parallel computation for the time being is seen as a promising solution [2, 35].

Collection of data and providing data services on real time are of high priority for big data applications such as navigation, social networks, finance, biomedicine, astronomy, intelligent transport systems, and internet of things. Guaranteeing timeliness in big data service is a major challenge. This not only requires high computing power but also requires innovative computing architectures and powerful data analysis algorithms.

The foremost challenge the emerging discipline faces is acute shortage of human resources. Big data application development needs people with high mathematical abilities and related professional expertise to harness big data value. Manyika et al. [36] illustrates difficulties USA faces in human resource for the purpose, but sure it is so for any other country too.

3.3 Guidelines

The challenge big data processing faces, looks for solution not only in technology but also in process of developing a system. We will list out these in brief following the discussion made in [37–39]. Big data processing needs distributed computation. And making for such a system is fairly dependent on type of application we have in hand. The recommended seven principles in making of big data systems are as follows:

Guideline-1: *Choice of good architecture*: big data processing is performed either on batch mode or in stream mode for real-time processing. While for the former MapReduce architecture is found effective but for later we need an architecture that provides fast access with key-value data stores, such as NoSQL, high performance and index-based retrieval are allowed. For real-time big data systems, Lambda Architecture is another example emphasising need for application-based architecture. This architecture proposes three-layer architecture the batch layer, the serving layer, and the speed layer claiming its usefulness in real-time big data processing [38].

Guideline-2: *Availability of analytics*: Making data useful is primarily dependent on veracity of analytics to meet different objectives different application domains look for. Analytics range from statistical analysis, in-memory analysis, machine learning, distributed programming and visualisation to real-time analysis

and human–computer interaction. These analytics must be resident of big data platforms so that applications can invoke on need.

Guideline-3: *Variance in analytics*: There can be a set of analytics for a domain that fits for all types of needs. Analytics can provide good dividends when tailored for an application. It is true so more for exponential increase in big data size. Seems, the trend is towards small data in view of usability of analytics [40].

Guideline-4: *Bring the analysis to data*: Moving voluminous data to analyst is not a practical solution for big data processing mainly for expense in data transmission. Instead of data migration, the issue of migration of analytics needs to be thought of.

Guideline-5: *In-memory computation*: It is now a leading concept for big data processing. In-memory analytic [39] that probes data resident on memory instead of disk is becoming popular for it is time saving. Real-time applications will most benefit of in-memory analytics.

Guideline-6: *Distributed data storage and in-memory analytic*: Distributed data storage is an accepted solution in order to cope up with voluminous data immersing from different sources. Further analytics is to accompany with data that need the analytics. This needs data partitioning and its storage along with the analytics data require. Cloud technology has shown a natural solution for uploading data and associated analytics on cloud so that being invoked big data processing takes place on a virtual super computer that is hidden on cloud.

Guideline-7: *Synchrony among data units*: Big data applications are to be centred around data units where each is associated with requiring analytics. Clusters of data units need to work in synchrony guaranteeing low latency of response for data-driven applications.

3.4 Big Data System

Next, we will have a brief discussion on a typical big data system architecture that can provide big data Service. Such a system is a composition of several subsystems. The framework is presented in Fig. 3. It shows an organic link among components that manage information and provide data service including business intelligence applications. The framework is taken from [41]; the copyright of the framework is with *intelligent business strategies*.

Big data system rather is an environment inhabited by both conventional as well as new database technologies enabling users not only to access information of variety forms but also infer knowledge from it. In literature, big data system is even termed as “Big Data Ecosystem”. It has three-layered ecosystem with bottom one interfacing to all kinds of data sources that feed the system with all types of data, i.e. structured and unstructured. It also includes active data sources, e.g. social media, enterprise systems, transactional systems where data of different formats continue to stream. There could be traditional database systems, files and documents with archived information forming data sources for a big data system.

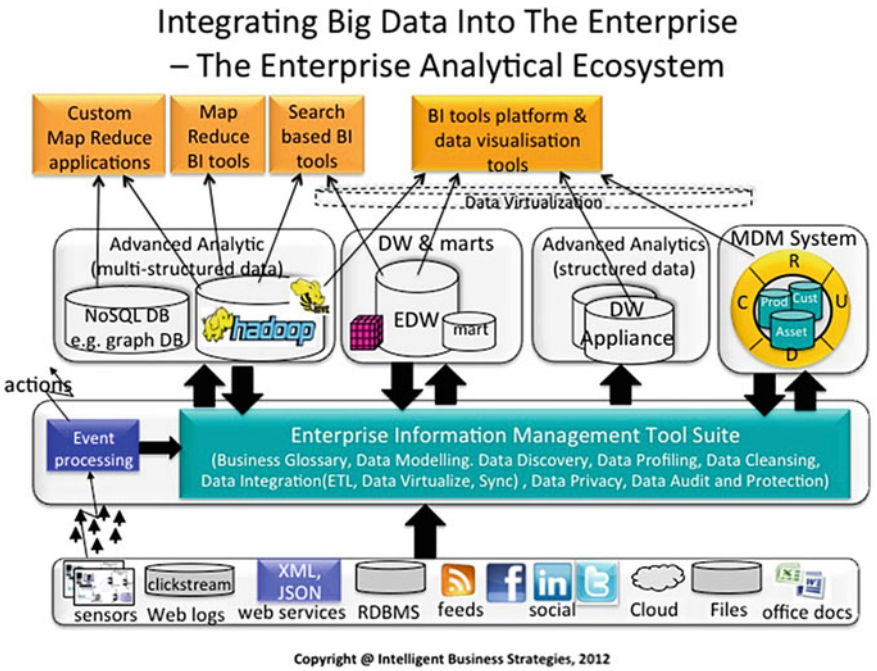

Fig. 3 Big data framework

The middle layer is in responsible of data management that includes data staging, data modelling, data integration, data protection, data privacy and auditing. It can have capability of virtualisation making data availability resilient on cloud environment. The third layer interfacing stake holders provides facilities for running applications. Broadly, the facilities include application parallelisation, information retrieval, intelligent implications and visualisation. The tools and techniques are key to success of a big data system. Usability of such a system increases by making use of appropriate technologies. Big data community is in the process of developing technologies and some of them have caught the imagination of users. Next section presents few popular technologies, though it does not claim a thorough review but make an attempt in citing major technologies made impact in big data processing

4 Technology and Tools

In this section, we point out technology and tools that have big impact on big data service. First we talk of hardware technologies followed by software technologies. Later in this section, we brief on some tools that are made for different purposes in big data service.

4.1 Hardware Technology

Conventional storage technology DRAM to store persistent data faces problem for long-term use because disks have moving parts that are vulnerable to malfunction in long run. DRAM chips need constant power supply irrespective of its usage. So, it is not an energy-efficient technology. Non-volatile memory technology shows a promising solution in future memory designs [42]. There are thinkings on use of NVM even at instruction level so that operating system can work fast. It is a wish to see NVM technology brings revolution to both data store and retrieval. Other than memory, technology looks for improving processing power to address the need for fast data processing. Significant solution towards that includes Data-Centre-on-Chip (DOC) [43]. It proposes four usage models that can be used to consolidate applications that are homogeneous and cooperating and manage synchrony on shared resources and at the same time speed up computation providing cache hierarchies. Tang et al. [44] proposes a hardware configuration that speeds execution of Java virtual machine (JVM) by speeding up algorithms like garbage collection. Same idea can be adopted for big data processing applying hardware technology to speed up data processing at bottlenecks usually found at data being shared by many.

Conventional TCP/IP stack for communication is usually homogeneous and works well for lossless transmissions. Round-trip time (RTT) is usually less than 250 μs in absence of queuing. This technology does not work for big data communication, for its communication, infrastructure requirement is very different. On addressing data-centric communication network problem, all-optical switching fabric could be a promising solution. It proposes computer to directly talk to network by passing the bottleneck of network interface. Processor to memory path can also have optical fibre connection. The first parallel optical transceiver capable of one terabit transmission per second is designed by IBM [45]. Intel is coming out of switches with optical interconnect cable in later versions of Thunderbolt. Hybrid electrical/optical switch Helios architecture [46] is also a promising solution.

Virtualisation technology though came with mainframe technology [47] and gone low for availability of inexpensive desk top computing has come to forefront again for processing big data service on cloud environment. Technologies are coming up for both CPU, memory and I/O virtualisation. For big data analytics, even code virtualisation (like JVM) is being intended for. This technology helps in run-time virtualisation following dynamically typed scripting languages or the use of just-in-time (JIT) techniques [48].

4.2 Software Technology

Here, we next take up developments in software technology taking place for big data services. First, we point out the requirements in software technology in development of big data systems following the discussion made in [49]. The

requirements include storage management and data processing techniques particularly towards business intelligence applications.

Big data storage not only faces challenge in hardware technology but also the challenge with its store management. As the paper [50] through CAP theorem indicates, assurance of high availability as well as consistency of data in distributed systems is always an ideal situation. Finally, one has to relax constraints in maintaining consistency. The nature of applications makes decisive impact assurance of consistency. Some applications need eventual consistency. For example, Amazon uses Dynamo [51] for its shopping cart. Facebook uses Cassandra [52] for storing its varieties of postings. Issues such as file systems [29, 30], data structures [28] and indexing [27] are being actively studied for making big data systems meet growing user needs.

Data processing requirements can be better understood by following the way data are being generated in big data environment. Other than being bulk and heterogeneous, big data characterises with its offline and online processing. Streaming (online) data processing and that at back-end need different approaches as latency for both are diametrically different. Again taking application characteristics into consideration, we find task parallelism is required by scientific applications and data-parallelism by web applications. However, all data-centric applications need to be fault-resilient, scalable as well as elastic in resource utilisation. MapReduce technique is well known for data parallel model used for batch processing. Google's MapReduce [53] is the first successful use of big data processing. It provides scalable and fault-tolerant file system in development of distributed applications. The paradigm has two phases, viz. Map and Reduce. On input, programmer-defined Map function is applied on each pair of (Key, Value): list to produce list of (Key, Value, list) list as intermediate list. And in Reduce phase, another programmer-defined function is applied to each element of the intermediate list. The paradigm supports run-time fault-tolerance by re-executing failed data processing. Opensource version Hadoop [54] supporting MapReduce paradigm is expected to be used for half of the world data in 2015 [55]. There are different versions of MapReduce technique to cater to different types of applications, e.g. for online aggregation and continuous queries, a technique is proposed in [56]. Extension to MapReduce is proposed in [49, 57] to support asynchronous algorithms. Some other extensions of MapReduce to support different types of applications and assuring optimised performance by introducing relaxed synchronisation semantics are proposed in [58, 59]. A survey on ongoing research on MapReduce can be found in [60].

MapReduce technique is found expensive for processing bulk of data required to be analysed iteratively for it requires reloading of data for each iteration. Bulk-synchronous parallel processing (BSP) technique offers a solution [61] to this problem by providing distributed access to data and primitives for intermediate synchronisations. The scheme does not need reloading of data, but augments data store on available of new data. Based on this technology, Google uses a software system *Pregel* for iterative execution of its graph algorithms that holds entire graph in its distributed memory so, avoiding frequent disk access for data processing [62].

Phoebus [63] that works following BSP concept is increasingly being used for analysis of postings made on social networks such as Facebook and LinkedIn. However, getting entire graph in memory though in distributed form will in future not be possible to meet the rise in data generation. There is a need to look for efficient graph partitioning-based data characteristics and use of data.

Event-driven applications are usually time-driven. Basically, it monitors, receives and process events in a stipulated time period. Some applications consider each event as atomic and take action with respect to the event. But, some look for complex events for taking actions for example looking for a pattern in events for making a decision. For the later case, events in a stream pass through a pipeline for being accessed. This remains a bottleneck in processing streaming events following MapReduce scheme, though some systems such as Borealis [64] and IBM's System S [65] manage to work still in future, there is a need for finding scalable techniques for efficient stream-data processing (Stream data: e.g. clicks on Web pages, status changes and postings on social networks). It has been slow in development of cloud-centric stream-data processing [56]. Resource utilisation is key to process scalable data using cloud-centric system. However, such system must manage elasticity and fault-tolerance for making event-driven big data applications viable.

Data-centric applications require parallel programming model and to meet data dependency requirements among tasks, intercommunication or information sharing can be allowed. This in turn gives rise to task parallelism. Managing conflict in concurrent executions in big data processing environment is a formidable task. Nevertheless, the idea of sharing key-value pairs on memory address space is being used to resolve potential conflicts. Piccolo [66] allows to store state changes as key-value pairs on distributed shared memory table for distributed computations. A transactional approach for distributed processing TransMR is reported in [67]. It uses MapReduce scheme for transactional computation over key-value data store stored in BigTable. Work towards data-centric applications is in progress with an aim to provide elastic, scalable and fault-tolerant systems.

Now, as big data applications evolve, there is a need of integration of models for developing applications of certain needs. For example, stream processing followed by batch processing is required for applications that collect clicks and process analytics in batch mode. A published subscribe system can be of a tight coupled system of data-centric and event-based system where subscriptions are partitioned based on topics and stored at different locations and as publications occur then get directed to their respective subscriptions. Though Apache yarn [54] is a system with multiple programming models still much work is to be done for development of systems with multiple programming models.

4.3 Big Data Tools

Engineering of software systems is primarily being supported by tools and progress on big data applications is substantially leveraged by big data tools. There have

been several tools from University labs and Corporate R&Ds. Some of them we mention here for completeness though the list is not claimed to be exhaustive.

Hadoop has emerged as the most suitable platform for data-intensive distributed applications. Hadoop HDFS is a distributed file system that partitions large files across multiple machines for high-throughput data access. It is capable of storing both structured as well as unstructured heterogeneous data. On it data storage data scientists run data analytics making use of its Hadoop MapReduce programming model. The model specially designed to offer a programming framework for distributed batch processing of large data sets distributed across multiple servers. It uses Map function for data distribution and creates key, value pairs for use in Reduce stage. Multiple copies of a program are created and made run at data clusters so that transporting data from its location to node for computation is avoided. Its component Hive is a data warehouse system for Hadoop that facilitates data summarisation, adhoc queries, and the analysis of large datasets stored in Hadoop compatible file systems. Hive uses a SQL-like language called HiveQL. HiveQL programs are converted into Map/Reduce programs; Hadoop has also provision to noSQL data using its component called HBase. It uses column-oriented store as used in Google' Bigtable. Hadoop's component Pig is a high-level data-flow language for expressing Map/Reduce programs for analysing large HDFS distributed data sets. Hadoop also hosts a scalable machine learning and data mining library called Mahout. Hadoop environment has a component called Oozie that schedules jobs submitted to Hadoop. It has another component Zookeeper that provides high-performance coordination service for distributed applications.

Other than Hadoop there are a host of tools available to facilitate big data applications. First we mention some other batch-processing tools, viz. Dryad [68], Jaspersoft BI suite [69], Pentaho business analytics [70], Skytree Server [71], Tableau [72], Karmasphere studio and analyst [73], Talend Open Studio [72] and IBM InfoSphere [41, 74]. Let us have brief introduction of each of these tools as follows.

Dryad [68] provides a distributed computation programming model that is scalable and user-friendly for keeping the job distribution hidden to users. It allocates, monitors and executes a job at multiple locations. A Dryad application execution model runs on a graph configuration where vertices represent processors and edges for channels. On submission of an application, Dryad centralised job manager allocates computation to several processors forming directed acyclic graph. It monitors execution and if possible can update a graph providing resilient computing framework in case of a failure. Thus, Dryad is a self-complete tool to deal with data-centric applications. Jaspersoft BI suite [69] is an open source tool efficient for big data fast access, retrieve and visualisation. Speciality of the tool is its capability to interface with varieties of databases not necessarily of relational databases including MongoDB, Cassandra, Redis, Riak and CouchDB. Its columnar-based in-memory engine makes it able to process and visualise large-scale data. Pentaho [70] is also an open source tool to process and visualise data. It provides Web interfaces to users to collect data, store and make business decision executing business analytics. It also can handle different types of data stores not necessarily of

relational database. Several popular NoSQL databases are supported by this tool. It also can handle Hadoop file systems for data processing. Skytree Server [71] is a tool of next generation providing advance data analytics including machine learning techniques. It has five useful advanced features, namely recommendation systems, anomaly/outlier identification, predictive analytics, clustering and market segmentation, and similarity search. The tool uses optimised machine learning algorithms for it uses with real-time streaming data. It is capable to handle structured and unstructured data stores. Tableau [72] has three main functionalities for data visualisation, interactive visualisation and browser-based business analytics. The main modules for these three functionalities are Tableau Desktop, Tableau Public and Tableau Server for visualisation, creative interactive visualisation and business analytics, respectively. Tableau also can work with Hadoop data store for data access. It processes data queries using in-memory analytics so this caching helps to reduce the latency of a Hadoop cluster. Karmasphere studio and analyst [73] is another Hadoop platform that provides advanced business analytics. It handles both structured and unstructured data. A programmer finds an integrated environment to develop Hadoop programs and to execute. The environment provides facilities for iterative analysis, visualisation and reporting. Karmasphere is well designed on Hadoop platform to provide integrated and user-friendly workspace for processing big data in collaborative way. Talend Open Studio [72] is yet another Hadoop platform for big data processing, but the speciality with it is visual programming facility that a developer can use to drag icons and stitch them to make an application. Though it intends to forego writing of complex Java programs but capability of application performance is limited to usability of icons. However, it provides a good seeding for application development. It has facility to work with HDFS, Pig, HCatalog, HBase, Sqoop or Hive. IBM InfoSphere [41, 74] is a big data analytic platform with Apache Hadoop system that provides warehousing as well as big data analytics services. It has features for data compression, MapReduce-based text and machine learning analytics, storage security and cluster management, connectors to IBM DB2, IBM's PureData, Job scheduling and workflow management and BigIndex—a MapReduce facility that leverages the power of Hadoop to build indexes for search-based analytic applications.

Now we introduce some tools used for processing of big data streams. Among them IBM InfoSphere Streams, Apache Kafka, SAP HANA, Splunk, Storm, SQLstream s-Server and S4 are referred here. IBM InfoSpere Streams is a real-time data processing stream capable of processing infinite length of stream data. It can process streaming of different structures. It has library of real-time data analytics and can also accept third-party analytics to run. It processes stream of data and looks for emerging patterns. On recognition of a pattern, impact analysis is carried out and necessary measure is taken fitting to made impact. The tool can attend to multiple streams of data. Scalability is provided by deploying InfoSphere Streams applications on multicore, multiprocessor hardware clusters that are optimised for real-time analytics. It has also dashboard for visualisation. Apache Kafka [75] developed at LinkedIn processes streaming data using in-memory analytics for meeting real-time constraints required to process streaming data. It combines offline and online

processing to provide real-time computation and produce ad hoc solution for these two kinds of data. The characteristic the tool has includes persistent messaging with O (1) disk structures, high-throughput, support for distributed processing, and support for parallel data load into Hadoop. It follows distributed implementation of published subscribe system for message passing. Interesting behaviour of the tool is combining both offline and online computation to meet real-time constraints streaming data processing demands. SAP HANA [76] is another tool for real-time processing of streaming data. Splunk [77] as real-time data processing tool is different than others in the sense it processes system-generated data, i.e. data available from system log. It uses cloud technology for optimised resource utilisation. It is used to online monitor and analyse the data systems produce and report on its dashboard. Storm [78] is a distributed real-time system to process streaming data. It has many applications, such as real-time analytics, interactive operation system, online machine learning, continuous computation, distributed RPC (Remote Procedure Call) and ETL (Extract, Transform and Load). Alike Hadoop clusters, it also uses clusters to speed data processing. The difference between two is in making of topology as Storm makes different topology for different applications, whereas Hadoop uses the same topology for iterative data analysis. Moreover, Storm can dynamically change its topology to achieve resilient computation. A topology is made of two types of nodes, namely spouts and bolts. Spout nodes denote input streams, and bolt nodes receive and process a stream of data and further output a stream of data. So, an application can be seen as parallel activities at different nodes of a graph representing a snap shot of distributed execution of the application. A cluster is seen as a collection of master node and several worker nodes. A master node and a worker node implement two daemons Nimbus and Supervisor, respectively. The two daemons have similar functions as JobTracker and TaskTracker in MapReduce framework do. Another kind of daemon called Zookeeper plays an important role to coordinate the system. The trilogy together makes storm system working in distributed framework for real-time processing of streaming data. SQLstream [79] is yet another real-time streaming data processing system to discover patterns. It can work efficiently with both structured as well as unstructured data. It stores streaming in memory and processes with in-memory analytics taking benefit of multicore computing. S4 [80] is a general purpose platform that provides scalable, robust distributed and real-time computation of streaming data. It provides a provision for plug-in-play making a system scalable. S4 also employs Apache ZooKeeper to manage its cluster. Successfully it is being used in production system of Yahoo to process thousands of queries posted to it.

However, other than batch processing and stream processing, there is a need of interactive analysis of data by a user. Interactive processing needs speed for not making users waiting for reply to queries. Apache drill [81] is a distributed system capable of scaling up on 10,000 or more servers processing different types of data. It can work on nested data and with a variety of query languages, data formats and data sources. Dremel [82] is another kind interactive big data analysis tool proposed by Google. Both search data stored either in columnar form or on a distributed file system to respond to users queries.

Different big data applications have different requirements. Hence, there are different types of tools with different features. And for the same type of application, different tools may have different performance. Further, tools acceptance depends on its user friendliness as a development environment. On choosing a tool for an application development, all these issues are to be taken into consideration.

Looking at all these tools, it is realised that good tool must be not only fast in processing and visualisation but also should have ability in finding out knowledge hidden in avalanche of data. Development in both the fronts requires research progress in several areas in computer science. In next section, we address some research issues big data is looking for.

5 Research Issues

Success in big data highly depends on its high-speed computing and analysis methodologies. These two objectives have a natural trade-off between cost in computation and number of patterns in data found. Here, principles of optimisation play a role in finding optimal solution in search of enough patterns in less cost. We have many optimisation techniques, namely stochastic optimisation, including genetic programming, evolutionary programming, and particle swarm optimisation. Big data application needs optimisation algorithms that work not only fast but also with reduced memory [83, 84]. Data reduction [85] and parallelisation [49, 86] are issues also to be considered for optimisation.

Statistics as commonly known for data analysis has also role in processing of big data. But, statistical algorithms are to be extended to meet scale and time requirements [87]. Parallelisation of statistical algorithms is an area of importance [88]. Further, statistical computing [89, 90] and statistical learning [91] are two hot research areas of promising result.

Social networking is currently massive big data generator. Many aspects [92] of social networks need intensive research for obtaining benefits as understanding digital society in future hold key to social innovation and engagement. First, study of social network structure includes many interesting issues such as link formation [93] and network evolution [94]. Visualising digital society with its dynamic changes and issue-based associations are interesting areas of research [95, 96]. Usages of social network are plenty [97] including business recommendation [98] and social behaviour modelling [99].

Machine learning has been in study of artificial intelligence that finds knowledge pattern on analysing data and uses the knowledge in intelligent decision-making that means decision making in a system is governed by the knowledge found by itself [100]. Likewise, big data being a collection of huge data may contain several patterns to discover by analysis. Existing machine learning algorithms both supervised and unsupervised face scale up problem to process big data. Current research aims for improving these algorithms to overcome the limitations they have. For example, ANN being so successful in machine learning performs poor for big

data for memory limitations, intractable computing and training [101]. The solution can be devised by reducing data keeping size of ANN limited. Another method may opt for massive parallelisation like MapReduce strategy. Big data analysis based on deep learning that leads to pattern detection based on regularity in spatio-temporal dependencies [102, 103] is of research interest. A work on learning on high-dimensional data is reported in [104]. Deep learning has been found successful in [105, 106], and it will also be useful in finding patterns in big data. In addition, visualisation of big data is a big research challenge [107], it takes up issues in feature extraction and geometric modelling to significantly reduce the data size before the actual data rendering. Again proper data structure is also an issue for data visualisation as discussed in [108].

Research in above areas aims at developing efficient analytics for big data analysis. Also we look for efficient and scalable computing paradigms, suitable for big data processing. For example, there are intensive works on variants of MapReduce strategy as the work reported in [56] is for making MapReduce work online. From literature, next we refer to emerging computing paradigms, namely granular computing, cloud computing, bio-inspired computing, and quantum computing.

Granular Computing [109] has been there even before arrival of big data. Essentially, the concept finds granularity in data and computation and applies different computing algorithms at different granularity of an application. For example, for analysing country economy one needs an approach that is different than the algorithm required for that of a state. The difference is something like macro- and microeconomics. Big data can be viewed at different granularity and can be used differently for different usages such as pattern finding, learning and forecasting. This suggests the need of granular computing for big data analysis [110, 111]. Changes in computation can be viewed as a development from machine-centric to human-centric then to information- and knowledge-centric. In that case, information granularity suggests the algorithm for computation.

Quantum computing [112] is in its fledging state but theoretically it suggests a framework capable of providing enormous memory space as well as speed to manipulate enormous input simultaneously. Quantum computing is based on theory of quantum physics [113]. A quantum works on concept qubit that codifies states in between zero and one to distinguishable quantum states following the phenomena of superposition and entanglement. The research in this area [114] soon hopefully helps to realise quantum computer that will be useful for big data analysis.

Cloud computing [115] presently has caught imagination of users by providing capability of super computers at ease over internet by virtualisation and sharing of affordable processors. This has assured resource and computing power of required scale [116] for computing. The computing need of big data fits well to cloud computing. As data grow or computation need grows, an application can ask for both computing and storage services from cloud for its elasticity to match with requirement load. And further, billing of cloud computing is made fairly simple for its pay-as-you-use rule. Currently, corporates have shown interest in providing big data application on cloud computing environment. This is understandable for the

number of tools available for the purpose. Further research will lead cloud computing to height as and when big data applications scale up to unprecedented size.

Biology has taken to centre stage in revolutionising today's world in many spheres. Particularly, for computation there has been an effort to get human intelligence to machine. Currently, researchers are interested in looking at phenomena that happen in human brain for storage of huge information for years after years and retrieve an information as need arises. The size and the speed human brain capable of are baffling and can be useful for big data analysis, if the phenomena are well understood and replicated on a machine. This quest has given rise to both bio-inspired hardware [117] as well as computing [118]. Researchers carry forward works in three fronts to design biocomputers, namely biochemical computers, biomechanical computers, and bioelectronic computers [119]. In another work, [120] biocomputing is viewed as a cellular network performing activity such as computation, communications, and signal processing. A recent work shows use of biocomputing in cost minimisation for provisioning data-intensive services [121]. As a whole, biological science and computing science both together project an emerging exciting area of research for taking up problems from both the sides with a great hope of extending boundary of science. The evolving paradigm is also expected to help immensely in big data analysis for possible power of computing mixed with intelligence.

This section points out some research areas that hold promise in big data analysis. As the applications in big data matures, its dimensions will be immersing better and so also new research problems will come up in finding solutions. Next, we conclude this chapter with a concluding remark.

6 Conclusion

Big data is now an emerging area in computer science. It has drawn attention of academics as well as developers and entrepreneurs. Academics see challenge in extracting knowledge by processing huge data of various types. Corporates wish to develop systems for different applications. Further, making big data applications available for users while on move is also on demand. These interests from several quarters are going to define growth in big data research and applications. This chapter intends to draw a picture on a big canvas of such developments.

On defining big data, the chapter goes on describing usages of big data in several domains including health, education, science and governance. This, the idea of big data as a service, makes a ground rationalising intent of study in big data. Next, the chapter makes a point on complexity in big data processing and lists out some guidelines for the same. Processing of big data needs a system that scales up to handle huge data size and is capable of very fast computation. The third section has a discussion on a big data system architecture. Some applications need batch-processing system and some need online computation. Real-time processing of big data is also of need for some usages. Thus, big data architectures vary based

on nature of applications. But, it is seen in general a big data system design mostly keeps two things at priority; firstly managing huge heterogeneous data and secondly managing computation at demanding less and less time. Currently, study on data science has engaged itself for solutions at the earliest.

Success in big data science largely depends on progress in both hardware and software technology. Fourth section of this chapter presents a picture on current developments in both hardware and software technology that are of importance for big data processing. Emergence of non-volatile memory (NVM) is expected to address some solutions that currently memory design for big data processing faces. For fast computations, for the time being data on chip (DOC) could be an effective solution. In addition, cloud computing has shown a way for resource utilisation for big data processing. Not only technology but also computing platform has a hand in success of big data. In that context, there are several tools that provide effective platforms in developing big data systems. This section also presents some references to known tools so that interested readers can pursue more to know details of a tool of its interest.

The sixth section presents research challenges that big data has brought in many areas in computer science. It discusses on several computing paradigms that may hold key to success in big data processing for its ever demanding need for high-speed computation. The chapter also spells needs on developing new variants of algorithms for knowledge extraction from big data. This gives a call to computer scientists to devise new adaptive strategy for fast computation.

Social networking and big data go almost hand to hand for the former being a major source for generation of big data. Thus, study on social networking has a bearing on advancement of big data usage. This also brings many vital social issues onto picture. For example, sharing of data is a very contentious issue. For example, an online analytic making use of data generated due to one's activity in social network raises an ethical question on rights and privacy. Many such ethical and social issues gradually come to fore when usages of big data applications rise. This challenge also invites social scientists to give a helping hand to success of big data. Of course, there could be a question in mind, that is natural to think on sustainability of big data processing. Should there be such huge distributed systems spanning across the globe is the future of computing? Or is there some data that is comprehensive and generic enough to describe the world around that is Small Data? This philosophical question could be contagious to intriguing minds!

Exercise

1. Define big data. Explain with an example.
2. List the possible sources generating big data.
3. Discuss on usage of big data in different domains?
4. Why is it called "big data a Service"? Justify your answer.
5. What makes big data processing difficult?
6. Discuss on the guidelines for big data processing.

7. Draw an ecosystem for a big data system. Explain functionality of each component.
8. Discuss on hardware and software technology required for big data processing.
9. Make a list of big data tools and note their functionality
10. Discuss on trends in big data research.

References

1. Zikopoulos, P.C., Eaton, C., deRoos, D., Deutsch, T., Lapis, G.: Understanding Big Data. McGrawHill, New York, (2012)
2. García, A.O., Bourov, S., Hammad, A., Hartmann, V., Jejkal, T., Otte, J.C., Pfeiffer, S., Schenker, T., Schmidt, C., Neuberger, P., Stotzka, R., van Wezel, J., Neumair, B., Streit, A.: Data-intensive analysis for scientific experiments at the large scale data facility. In: IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 125–126 (2011)
3. O'Leary, D.E.: Artificial intelligence and big data. Intell. Syst. IEEE **28**, 96–99 (2013)
4. Berman, J.J.: Introduction. In: Principles of Big Data, pp. xix-xxvi. Morgan Kaufmann, Boston (2013)
5. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mob. Netw. Appl. **19**, 171–209 (2014)
6. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Ullah, S.: The rise of "Big Data" on cloud computing: review and open research issues. Inf. Syst. **47**, January, 98–115 (2015)
7. Lusch, R.F., Liu, Y., Chen, Y.: The phase transition of markets and organizations: the new intelligence and entrepreneurial frontier. IEEE Intell. Syst. **25**(1), 71–75 (2010)
8. Chen, H., Chiang, R.H.L., Storey, V.C.: Business intelligence and analytics: from big data to big impact. MIS Quarterly **36**(4), 1165–1188 (2012)
9. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734-749 (2005)
10. Chen, H.: Smart health and wellbeing. IEEE Intell. Syst. **26(**5), 78–79 (2011)
11. Parida, L., Haiminen, N., Haws, D., Suchodolski, J.: Host trait prediction of metagenomic data for topology-based visualisation. LNCS **5956**, 134–149 (2015)
12. Chen, H.: Dark Web: Exploring and Mining the Dark Side of the Web. Springer, New york (2012)
13. NSF: Program Solicitation NSF 12-499: Core techniques and technologies for advancing big data science & engineering (BIGDATA). http://www.nsf.gov/pubs/2012/nsf12499/nsf12499.htm (2012). Accessed 12th Feb 2015
14. Salton, G.: Automatic Text Processing, Reading. Addison Wesley, MA (1989)
15. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge (1999)
16. Big Data Spectrum, Infosys. http://www.infosys.com/cloud/resource-center/Documents/big-data-spectrum.pdf
17. Short, E., Bohn, R.E., Baru, C.: How much information? 2010 report on enterprise server information. UCSD Global Information Industry Center (2011)
18. http://public.web.cern.ch/public/en/LHC/Computing-en.html
19. http://www.youtube.com/yt/press/statistics.html
20. http://agbeat.com/tech-news/how-carriers-gather-track-and-sell-your-private-data/
21. http://www.information-management.com/issues/21_5/big-data-is-scaling-bi-and-analytics-10021093-1.html

22. Rahm, E., Do, H.H.: Data cleaning: problems and current approaches. IEEE Data Eng. Bull. **23**, 3–13 (2000)
23. Agrawal, D., Bernstein, P., Bertino, E., Davidson, S., Dayal, U., Franklin, M., Gehrke, J., Haas, L., Han, J., Halevy, A., Jagadish, H.V., Labrinidis, A., Madden, S., Papakon stantinou, Y., Patel, J., Ramakrishnan, R., Ross, K., Cyrus, S., Suciu, D., Vaithyanathan, S., Widom, J.: Challenges and opportunities with big data. CYBER CENTER TECHNICAL REPORTS, Purdue University (2011)
24. Kasavajhala, V.: Solid state drive vs. hard disk drive price and performance study. In: Dell PowerVault Tech. Mark (2012)
25. Hutchinson, L.: Solid-state revolution. In: Depth on how ssds really work. Ars Technica (2012)
26. Pirovano, A., Lacaita, A.L., Benvenuti, A., Pellizzer, F., Hudgens, S., Bez, R.: Scaling analysis of phase-change memory technology. IEEE Int. Electron Dev. Meeting, 29.6.1–29.6.4 (2003)
27. Chen, S., Gibbons, P.B., Nath, S.: Rethinking database algorithms for phase change memory. In: CIDR, pp. 21–31. www.crdrdb.org (2011)
28. Venkataraman, S., Tolia, N., Ranganathan, P., Campbell, R.H.: Consistent and durable data structures for non-volatile byte-addressable memory. In: Ganger, G.R., Wilkes, J. (eds.) FAST, pp. 61–75. USENIX (2011)
29. Athanassoulis, M., Ailamaki, A., Chen, S., Gibbons, P., Stoica, R.: Flash in a DBMS: where and how? IEEE Data Eng. Bull. **33**(4), 28–34 (2010)
30. Condit, J., Nightingale, E.B., Frost, C., Ipek, E., Lee, B.C., Burger, D., Coetzee, D.: Better I/O through byte—addressable, persistent memory. In: Proceedings of the 22nd Symposium on Operating Systems Principles (22nd SOSP'09), Operating Systems Review (OSR), pp. 133–146, ACM SIGOPS, Big Sky, MT (2009)
31. Wang, Q., Ren, K., Lou, W., Zhang, Y.: Dependable and secure sensor data storage with dynamic integrity assurance. In: Proceedings of the IEEE INFOCOM, pp. 954–962 (2009)
32. Oprea, A., Reiter, M.K., Yang, K.: Space efficient block storage integrity. In: Proceeding of the 12th Annual Network and Distributed System Security Symposium (NDSS 05) (2005)
33. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of "big data" on cloud computing: review and open research issues, vol. 47, pp. 98–115 (2015)
34. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**(5), 847–859 (2011)
35. Oehmen, C., Nieplocha, J.: Scalablast: a scalable implementation of blast for high-performance data-intensive bioinformatics analysis. IEEE Trans. Parallel Distrib. Syst. **17**(8), 740–749 (2006)
36. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A.: Big data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey Global Institute (2012)
37. Chen, C.L.P., Zhang, C.-Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. **275**, 314–347 (2014)
38. Marz, N., Warren, J.: Big data: principles and best practices of scalable real-time data systems. Manning (2012)
39. Garber, L.: Using in-memory analytics to quickly crunch big data. IEEE Comput. Soc. **45**(10), 16–18 (2012)
40. Molinari, C.: No one size fits all strategy for big data, Says IBM. http://www.bnamericas.com/news/technology/no-one-size-fits-all-strategy-for-big-data-says-ibm, October 2012
41. Ferguson, M.: Architecting a big data platform for analytics, Intelligent Business Strategies. https://www.ndm.net/datawarehouse/pdf/Netezza (2012). Accessed 19th Feb 2015
42. Ranganathan, P., Chang, J.: (Re)designing data-centric data centers. IEEE Micro **32**(1), 66–70 (2012)

43. Iyer, R., Illikkal, R., Zhao, L., Makineni, S., Newell, D., Moses, J., Apparao, P.: Datacenter-on-chip architectures: tera-scale opportunities and challenges. Intel Tech. J. **11**(3), 227–238 (2007)
44. Tang, J., Liu, S., Z, G., L, X.-F., Gaudiot, J.-L.: Achieving middleware execution efficiency: hardware-assisted garbage collection operations. J. Supercomput. **59**(3), 1101–1119 (2012)
45. Made in IBM labs: holey optochip first to transfer one trillion bits of information per second using the power of light, 2012. http://www-03.ibm.com/press/us/en/pressrelease/37095.wss
46. Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H.H., Subramanya, V., Fainman, Y., Papen, G., Vahdat, A.: Helios: a hybrid electrical/optical switch architecture for modular data centers. In: Kalyanaraman, S., Padmanabhan, V.N., Ramakrishnan, K.K., Shorey, R., Voelker, G.M. (eds.) SIGCOMM, pp. 339–350. ACM (2010)
47. Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Commun. ACM **17**(7), 412–421 (1974)
48. Andersen, R., Vinter, B.: The scientific byte code virtual machine. In: GCA, pp. 175–181 (2008)
49. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. J. Parallel Distrib. Comput. **74**, 2561–2573 (2014)
50. Brewer, E.A.: Towards robust distributed systems. In: Proceeding of 19th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 7–10 (2000)
51. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazon's highly available key-value store. In: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles, SOSP'07, ACM, New York, NY, USA, pp. 205–220 (2007)
52. Lakshman, A., Malik, P.: Cassandra: a structured storage system on a p2p network. In: SPAA (2009)
53. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: OSDI (2004)
54. Apache yarn. http://hadoop.apache.org/common/docs/r0.23.0/hadoop-yarn/hadoop-yarn-site/YARN.html
55. Hortonworks blog. http://hortonworks.com/blog/executive-video-series-the-hortonworks-vision-for-apache-hadoop
56. Condie, T., Conway, N., Alvaro, P., Hellerstein, J.M., Elmeleegy, K., Sears, R.: MapReduce online. In: NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation, p. 21
57. Kambatla, K., Rapolu, N., Jagannathan, S., Grama, A.: Asynchronous algorithms in MapReduce. In: IEEE International Conference on Cluster Computing, CLUSTER (2010)
58. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating mapreduce for multi-core and multiprocessor system. In: Proceedings of the 13th International Symposium on High-Performance Computer Architecture (HPCA), Phoenix, AZ (2007)
59. Improving MapReduce Performance in Heterogeneous Environments. USENIX Association, San Diego, CA (2008), 12/2008
60. Polato, I., Ré, R., Goldman, A., Kon, F.: A comprehensive view of Hadoop research—a systematic literature review. J. Netw. Comput. Appl. **46**, 1–25 (2014)
61. Valiant, L.G.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
62. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: SIGMOD (2010)
63. Phoebus. https://github.com/xslogic/phoebus
64. Ahmad, Y., Berg, B., Cetintemel, U., Humphrey, M., Hwang, J.-H., Jhingran, A., Maskey, A., Papaemmanouil, O., Rasin, A., Tatbul, N., Xing, W., Xing, Y., Zdonik, S.: Distributed operation in the borealis stream processing engine. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD '05, pp. 882–884, ACM, New York, NY, USA (2005)

65. Andrade, H., Gedik, B., Wu, K.L., Yu, P.S.: Processing high data rate streams in system S. J. Parallel Distrib. Comput. **71**(2), 145–156 (2011)
66. Power, R., Li, J.: Piccolo: building fast, distributed programs with partitioned tables. In: OSDI (2010)
67. Rapolu, N., Kambatla, K., Jagannathan, S., Grama, A.: TransMR: data-centric programming beyond data parallelism. In: Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'11, USENIX Association, Berkeley, CA, USA, pp. 19–19 (2011)
68. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: EuroSys '07 Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, vol. 41, no. 3, pp. 59–72 (2007)
69. Wayner, P.: 7 top tools for taming big data. http://www.networkworld.com/reviews/2012/041812-7-top-tools-for-taming-258398.html (2012)
70. Pentaho Business Analytics. 2012. http://www.pentaho.com/explore/pentaho-business-analytics/
71. Diana Samuels, Skytree: machine learning meets big data. http://www.bizjournals.com/sanjose/blog/2012/02/skytree-machine-learning-meets-big-data.html?page=all, February 2012
72. Brooks, J.: Review: Talend open studio makes quick work of large data sets. http://www.eweek.com/c/a/Database/REVIEW-Talend-Open-Studio-Makes-Quick-ETL-Work-of-Large-Data-Sets-281473/ (2009)
73. Karmasphere Studio and Analyst. http://www.karmasphere.com/ (2012)
74. IBM Infosphere. http://www-01.ibm.com/software/in/data/infosphere/
75. Auradkar, A., Botev, C., Das, S., De Maagd, D., Feinberg, A., Ganti, P., Ghosh, B., Gao, L., Gopalakrishna, K., Harris, B., Koshy, J., Krawez, K., Kreps, J., Lu, S., Nagaraj, S., Narkhede, N., Pachev, S., Perisic, I., Qiao, L., Quiggle, T., Rao, J., Schulman, B., Sebastian, A., Seeliger, O., Silberstein, A., Shkolnik, B., Soman, C., Sumbaly, R., Surlaker, K., Topiwala, S., Tran, C., Varadarajan, B., Westerman, J., White, Z., Zhang, D., Zhang, J.: Data infrastructure at linkedin. In: 2012 IEEE 28th International Conference on Data Engineering (ICDE), pp. 1370–1381 (2012)
76. Kraft, S., Casale, G., Jula, A., Kilpatrick, P., Greer, D.: Wiq: work-intensive query scheduling for in-memory database systems. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 33–40 (2012)
77. Samson, T.: Splunk storm brings log management to the cloud. http://www.infoworld.com/t/managed-services/splunk-storm-brings-log-management-the-cloud-201098?source=footer (2012)
78. Storm. http://storm-project.net/ (2012)
79. Sqlstream. http://www.sqlstream.com/products/server/ (2012)
80. Neumeyer, L., Robbins, B., Nair, A., Kesari, A.: S4: distributed stream computing platform. In: 2010 IEEE Data Mining Workshops (ICDMW), pp. 170–177, Sydney, Australia (2010)
81. Kelly, J.: Apache drill brings SQL-like, ad hoc query capabilities to big data. http://wikibon.org/wiki/v/Apache-Drill-Brings-SQL-Like-Ad-Hoc-Query-Capabilities-to-Big-Data, February 2013
82. Melnik, S., Gubarev, A., Long, J.J., Romer, G., Shivakumar, S., Tolton, M., Vassilakis, T.: Dremel: interactive analysis of webscale datasets. In: Proceedings of the 36th International Conference on Very Large Data Bases (2010), vol. 3(1), pp. 330–339 (2010)
83. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. Evol. Comput. **16**(2), 210–224 (2008)
84. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2985–2999 (2008)
85. Yan, J., Liu, N., Yan, S., Yang, Q., Fan, W., Wei, W., Chen, Z.: Trace-oriented feature analysis for large-scale text data dimension reduction. IEEE Trans. Knowl. Data Eng. **23**(7), 1103–1117 (2011)

86. Spiliopoulou, M., Hatzopoulos, M., Cotronis, Y.: Parallel optimization of large join queries with set operators and aggregates in a parallel environment supporting pipeline. IEEE Trans. Knowl. Data Eng. **8**(3), 429–445 (1996)
87. Di Ciaccio, A., Coli, M., Ibanez, A., Miguel, J.: Advanced Statistical Methods for the Analysis of Large Data-Sets. Springer, Berlin (2012)
88. Pébay, P., Thompson, D., Bennett, J., Mascarenhas, A.: Design and performance of a scalable, parallel statistics toolkit. In: 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), pp. 1475–1484 (2011)
89. Klemens, B.: Modeling with Data: Tools and Techniques for Statistical Computing. Princeton University Press, New Jersey (2008)
90. Wilkinson, L.: The future of statistical computing. Technometrics **50**(4), 418–435 (2008)
91. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining Inference and Prediction, 2nd edn. Springer, Berlin (2009). (egy, Russell Sears, MapReduce online. In: NSDI, 2009)
92. Jamali, M., Abolhassani, H.: Different aspects of social network analysis. In: IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006, pp. 66–72 (2006)
93. Zhang, Yu., van der Schaar, M.: Information production and link formation in social computing systems. IEEE J. Sel. Areas Commun. **30**(1), 2136–2145 (2012)
94. Bringmann, B., Berlingerio, M., Bonchi, F., Gionis, A.: Learning and predicting the evolution of social networks. IEEE Intell. Syst. **25**(4), 26–35 (2010)
95. Fekete, J.-D., Henry, N., McGuffin, M.: Nodetrix: a hybrid visualization of social network. IEEE Trans. Visual. Comput. Graph. **13**(6), 1302–1309 (2007)
96. Shen, Z., Ma, K.-L., Eliassi-Rad, T.: Visual analysis of large heterogeneous social networks by semantic and structural abstraction. IEEE Trans. Visual. Comput. Graph. **12**(6), 1427–1439 (2006)
97. Lin, C.-Y., Lynn, W., Wen, Z., Tong, H., Griffiths-Fisher, V., Shi, L., Lubensky, D.: Social network analysis in enterprise. Proc. IEEE **100**(9), 2759–2776 (2012)
98. Ma, H., King, I., Lyu, M.R.-T.: Mining web graphs for recommendations. IEEE Trans. Knowl. Data Eng. **24**(12), 1051–1064 (2012)
99. Lane, N.D., Ye, X., Hong, L., Campbell, A.T., Choudhury, T., Eisenman, S.B.: Exploiting social networks for large-scale human behavior modeling. IEEE Pervasive Comput. **10**(4), 45–53 (2011)
100. Bengio, Y.: Learning deep architectures for ai, Found. Trends Mach. Learn. **2**(1),1–1-1–27 (2009)
101. Seiffert, U.: Training of large-scale feed-forward neural networks. In: International Joint Conference on Neural Networks, IJCNN '06, pp. 5324–5329 (2006)
102. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning—a new frontier in artificial intelligence research. IEEE Comput. Intell. Mag. **5**(4), 13–18 (2010)
103. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
104. Le, Q.V., Ranzato, M.A., Monga, R., Devin, M., Chen, K., Corrado, G.S., Dean, J., Andrew, Y. N.: Building high-level features using large scale unsupervised learning. In: Proceedings of the 29th International Conference on Machine Learning (2012)
105. Dong, Y., Deng, L.: Deep learning and its applications to signal and information processing. IEEE Signal Process. Mag. **28**(1), 145–154 (2011)
106. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
107. Simoff, S., Böhlen, M.H., Mazeika, A.: Visual Data Mining: Theory, Techniques and Tools for Visual Analytics. Springer, Berlin (2008)
108. Thompson, D., Levine, J.A., Bennett, J.C., Bremer, P.T., Gyulassy, A., Pascucci, V., Pébay, P.P.: Analysis of large-scale scalar data using hixels. In: 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV), pp. 23–30 (2011)
109. Andrzej, W.P., Kreinovich, V.: Handbook of Granular Computing. Wiley, New York (2008)
110. Peters, G.: Granular box regression. IEEE Trans. Fuzzy Syst. **19**(6), 1141–1151 (2011)

111. Su, S.-F., Chuang, C.-C., Tao, C.W., Jeng, J.-T., Hsiao, C.-C.: Radial basis function networks with linear interval regression weights for symbolic interval data. IEEE Trans. Syst. Man Cyber.–Part B Cyber. **19**(6), 1141–1151 (2011)
112. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**, 116–123 (1994)
113. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
114. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2009)
115. Furht, B., Escalante, A.: Handbook of Cloud Computing. Springer, Berlin (2011)
116. Schadt, E.E., Linderman, M.D., Sorenson, J., Lee, L., Nolan, G.P.: Computational solutions to large-scale data management and analysis. Nat. Rev. Genet. **11**(9), 647–657 (2010)
117. Sipper, M., Sanchez, E., Mange, D., Tomassini, M., Pérez-Uribe, A., Stauffer, A.: A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. IEEE Trans. Evol. Comput. **1**(1), 83–97 (1997)
118. Bongard, J.: Biologically inspired computing. Computer **42**(4), 95–98 (2009)
119. Ratner, M., Ratner, D.: Nanotechnology: A Gentle Introduction to the Next Big Idea, 1st edn. Prentice Hall Press, Upper Saddle River (2002)
120. Weiss, R., Basu, S., Hooshangi, S., Kalmbach, A., Karig, D., Mehreja, R., Netravali, I.: Genetic circuit building blocks for cellular computation, communications, and signal processing. Nat. Comput. **2**, 47–84 (2003)
121. Wang, L., Shen, J.: Towards bio-inspired cost minimisation for data-intensive service provision. In: 2012 IEEE First International Conference on Services Economics (SE), pp. 16–23 (2012)

Big Data Architecture

Bhashyam Ramesh

Abstract Big data is a broad descriptive term for non-transactional data that are user generated and machine generated. Data generation evolved from transactional data to first interaction data and then sensor data. Web log was the first step in this evolution. These machines generated logs of internet activity caused the first growth of data. Social media pushed data production higher with human interactions. Automated observations and wearable technologies make the next phase of big data. Data volumes have been the primary focus of most big data discussions. Architecture for big data often focuses on storing large volumes of data. Dollars per TB (Terabyte) becomes the metric for architecture discussions. We argue this is not the right focus. Big data is about deriving value. Therefore, analytics should be the goal behind investments in storing large volumes of data. The metric should be dollars per analytic performed. There are three functional aspects to big data—data capture, data R&D, and data product. These three aspects must be placed in a framework for creating the data architecture. We discuss each of these aspects in depth in this chapter. The goal of big data is data-driven decision making. Decisions should not be made with data silos. When context is added to data items they become meaningful. When more contexts are added more insight is possible. Deriving insight from data is about reasoning with all data and not just big data. We show examples of this and argue big data architecture must provide mechanisms to reason with all data. Big data analytic requires all forms of different technologies including graph analytics, statistical analytics, path analysis, machine learning, neural networks, and statistical analysis be integrated in an integrated analytics environment. Big data architecture is an architecture that provides the framework for reasoning with all forms of data. We end this chapter with such architecture. This chapter makes three points as follows: (a) Big data analytics is analytics on all data and not just big data alone; (b) Data complexity, not volume, is the primary concern of big data analytics; (c) Measure of goodness of a big data analytic architecture is dollars per analytics and not dollars per TB.

B. Ramesh (✉)
Teradata Corporation, Dayton, USA
e-mail: vembakkambhashyam@gmail.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_2

Keywords Data lake · Telematics · Data volumes · Business intelligence · HDFS · Sessionizing

1 Introduction

This chapter gives the architecture for big data. The intent is to derive value from big data and enable data-driven decision making. The architecture specified here is for the data-centric portions. It does not cover mechanisms and tools required for accessing the data. The choice of such mechanisms is based at least in part on the interfaces necessary to support a wide variety of access tools and methods. They must support all forms of applications including business intelligence (BI), deep analytics, visualization, and Web access to name a few.

Big data is a broad term. Wikipedia defines big data as "an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process using traditional data processing applications." We agree with this definition of big data.

Benefitting from big data, that is, deriving value from big data requires processing all data big and not so big. In other words, big data processing is not an isolated exercise. Just as data understanding increases with descriptive metadata, value of data in general and value of big data in particular increase with context under which the data is analyzed. This context resides in many places in the organizations' data repository. These must be brought under an overarching architecture in order to fully derive value from big data.

The rest of this chapter is organized as follows:

We discuss data and its key characteristics as foundations for understanding the data architecture. Sections 2–4 cover these discussions. Section 2 defines big data. It describes the three different components of big data. Section 3 defines different characteristics of big data and their evolution. It shows that data growth has been in spurts and has coincided with major innovations in storage and analytics. Big data is the next stage in this data growth story. Section 4 specifies some value metric as a motivation for evaluating the architecture.

We then discuss mechanisms for deriving value from data. Sections 5–7 cover these discussions. Section 5 specifies three functional components of data-driven decision making. It explains the phases that data goes through in order to become a product. Section 6 lays the foundation for the assertion that deriving value from big data is about reasoning with all data and not just big data. Section 7 focuses on data analytics and the evolutions in that space. It argues for an integrated analytics framework.

Section 8 gives the overall architectural framework by building on these earlier sections.

Finally, Sect. 9 puts them all together with a use-case.

2 Defining Big Data

Gartner defines big data in terms of 3Vs—volume, variety, and velocity [1]. These three Vs are briefly explained in the following paragraphs. Some add a 4th V for veracity or data quality [2]. Data quality is a huge problem in practice. Quality of human-entered data is typically low and even automatically collected data from mobiles and sensors can be bad–sensors can go bad, sensors may need calibration, and data may be lost in transmission. In addition, data may be misinterpreted due to incorrect metadata about the sensor. It is important that data should be checked for correctness constantly. We do not address the fourth V in this chapter.

The 3Vs can be viewed as three different dimensions of data.

Volume, the most obvious of the 3V, refers to the amount of data. In the past, increase in data volumes was primarily due to increase in transaction volume and granularity of transaction details. These increases are small in comparison with big data volumes. Big data volume started growing with user interactions. Weblogs was the starting point of big data. Size and volume of Weblogs increased with internet adoption. These machine logs spawned a set of analytics for understanding user behavior. Weblogs seem small compared to social media. Facilities such as Facebook, Twitter, and others combined with the increase in capability and types of internet devices caused a big jump in data production. The constant human desire to connect and share caused an even bigger jump in data production. Social media allows users to express and share video, audio, text, and e-mail in volumes with a large social audience. The corresponding sophistication in mobile technology makes sharing easier and increases the volumes even further. This is the second wave of big data. The next big jump in volume will be from automated observations. All kinds of biometric sensors (heart, motion, temperature, pulse, etc.) and motion sensors (GPS, cellular, etc.) and the ability to move them through cellular, internet, Wi-fi, etc., is the next increase in big data volume. Wearable technologies and the internet of things (IOT) is the next big thing that is waiting to happen. These are referred as observation data. Sensor data and observations will dwarf the volume we have seen so far. The following Fig. 1 shows the relative increases in volume as we move from transactions to interactions to observations.

Data variety is as diverse as the data sources and their formats. Unlike transaction data which is highly structured in relational form, other data forms are either relatively weakly structured such as XML and JSON or differently structured such as audio, video, or without structure such as text, scanned documents. Different data sources such as Weblog, machine log, social media, tweets, e-mails, call center logs, and sensor data all produce data in different formats. These varieties of data make analysis more challenging. Combining variety with volume increases the challenges.

Velocity is the speed at which data flows into the system and therefore must be handled. Data comes from machines and humans. Velocity increases with the number of sources. Velocity increases with the speed at which data is produced such as with mobiles and sensors.

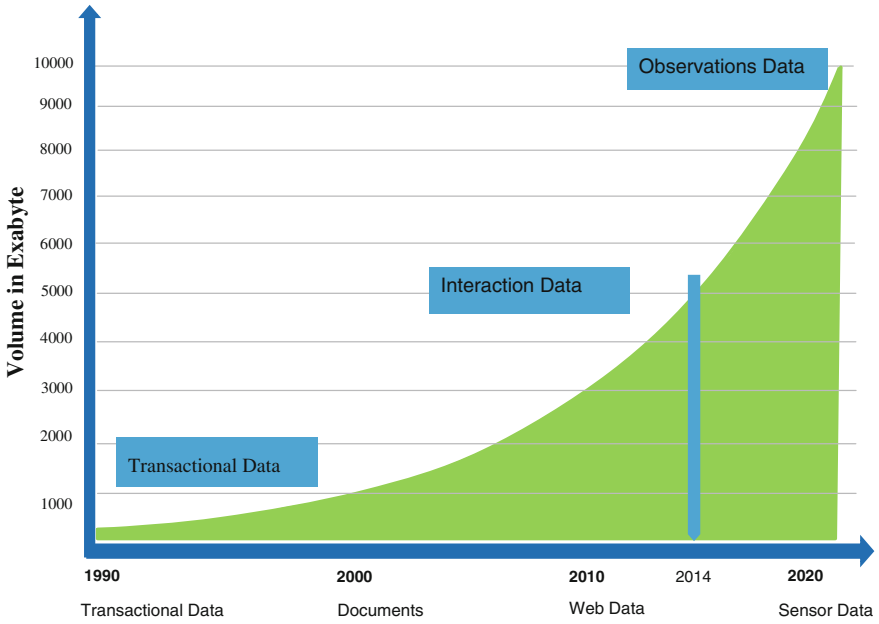

Fig. 1 Volume versus variety

We combine the 3Vs and call it the complexity of data. Complexity refers to the ability to analyze, derive insight, and value from big data. Deriving insight from big data is orders of magnitude more complex than deriving insight from transactional

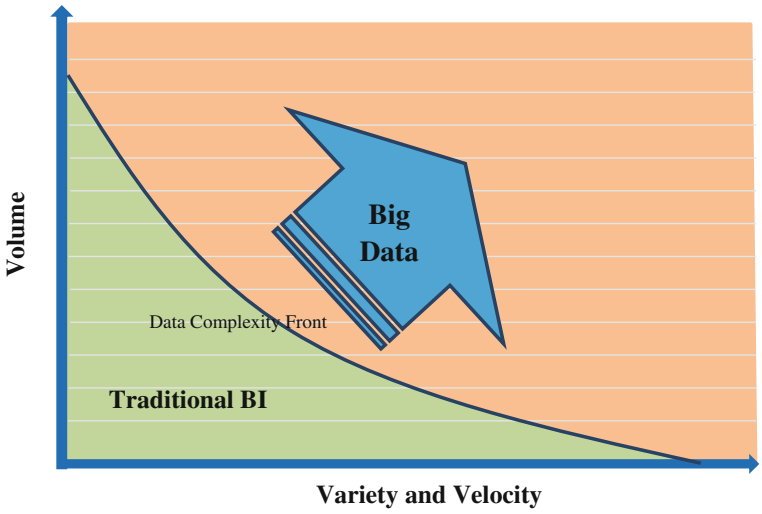

Fig. 2 Moving complexity front related to 3Vs

data. Deriving insight from sensor data is more complex than deriving insight from user-generated data.

Figure 2 shows variety and velocity in one axis versus volume in another axis. Complexity is the moving front. Complexity increases as the front moves to the right and top; complexity decreases as the front moves down and left.

One of the purposes of storing and analyzing big data is to understand and derive value; otherwise, there is no point in storing huge volumes of data. Complexity of gaining insight is directly related to the complexity of data. Analytics on complex data is much harder. It is complex in the type of analytic techniques that are needed and it is complex in the number of techniques that have to be combined for analysis. We cover these aspects later in this chapter. There is also a chapter dedicated to big data algorithms in this book.

3 Space of Big Data

Figure 3 shows the different types of data that make up the big data space. Each type of data exhibits different value and different return on investment (ROI).

Each shaded transition in this picture represents a major plateau in data growth. Each shaded region represents a quantum jump in analytic complexity, data capture, data storage, and management. The quantum jump is followed by a duration of incremental growth in such capabilities until the next quantum jump in such

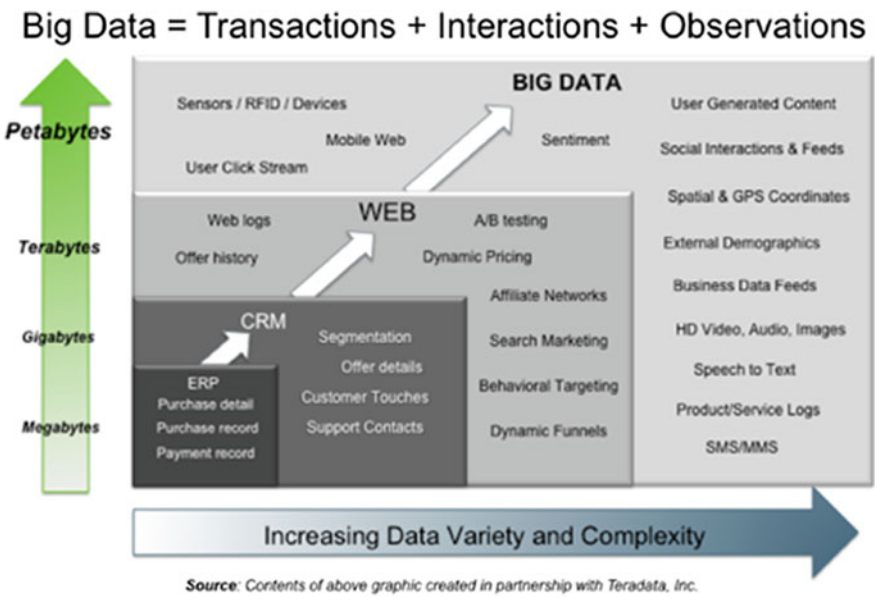

Fig. 3 Big data space

capabilities occur. There have been many plateaus over the years and many minor plateaus within each major plateau. Each has coincided with a jump in technology related to capture, transmission, management, and analytics. Each plateau has required some new forms of analytics. Each jump in technology has enabled a move to the next plateau. Each plateau has meant the ability to process much bigger volumes of data.

There is a ROI threshold for each plateau of data size. This means there is a balance between the cost associated with storing and managing the data and the value that can be derived from the data through application of analytics. Storing them is not useful when cost exceeds the value that can be derived. ROI determines whether the data is useful to be stored and managed. The ROI threshold is not reached until the technology and the cost of doing analysis is cheap enough to get a ROI from storing and analyzing the data. This notion is best described using transactional data as example. In the past retail applications used detail data at the level of store-item-week. Analysis used to be at this level of store-item-week granularity. Users were satisfied doing analysis at this summary level of detail when compared to what they were able to do before. However, users realized more detail than week-level summary will help, but the technology was unaffordable for them to store and manage at more detail levels. When technologies made it possible to go to store-item-day, then to market basket summaries, and then onto market basket detail, these new data plateaus became the new norm. At each point along the way users felt, they had reached the correct balance between cost and capability and any further detail, and data volumes were uneconomical since they did not see any value in them. However, leading-edge users made the conceptual leap to more detail first. They were visionaries in knowing how to get value from more detailed data. Now those plateaus have become common place; in fact, they are table stakes in any analytic environment. It is now relatively inexpensive to store at the smallest SKU level of detail, practical to do the analysis, and suitable to get value from such detailed data. There are such examples in every industry. In the communications industry, the plateau evolved from billing summary data which is few records per customer per month to storing and managing call detail records which are records of every call ever made! The new plateau now in the communications industry is network-level traffic data underlying each call. So there have been such plateaus before in all varieties of data.

Big data is just another plateau. Currently, we are seeing the leading edge of the big data plateau.

The capture and storage technologies appropriate for transactions is inappropriate for Weblogs, social media, and other big data forms. Technologies that can store large volumes of data at low cost are needed. Leading-edge users have found ways to derive value from storing interactions such as clickstreams and user navigation of Web sites. They determine the likes and dislikes of customers from these data and use them to propose new items for review. They analyze social graphs in order to differentiate between influencers and followers with a goal to tailor their marketing strategies. Such analyses are now moving from the leading edge to common place.

Currently, users attempt to reduce the volume of data in an effort to make it cost-effective to manage. They apply filters on acquired raw data in order to reduce its size for analysis. Technology will eventually become cheaper and bigger per unit of floor space and make it cost-effective to store, manage, and analyze all the detailed data instead of summaries of such filtered big data.

3.1 The Next Data Plateau

The next data plateau is on the horizon. The next jump will be from wearable technologies, sensors, and IOT. Many users wonder how to derive value, while leading-edge thinkers are finding innovative ways.

Social networks and the digital stream which are the current focus of big data are medium complexity in comparison with sensor data. Special scientific customers such as the supercollider already store sensor data. IOT will make such amounts common place. Mobile phones with biometric and other sensors collect large amounts of data and transmit them easily. These data are already far beyond any social networks. Wearable computing and IOT will increase this further. IOT is like an electronic coating on everyday things, a sensor embedded on everyday items that makes them seem sentient. IOTs can be linked together to form a network of sensors that can provide a bigger picture of the surroundings being monitored—a Borg-like entity without the malevolent intent.

Leading-edge companies and applications are starting to experiment with the beginning stages of this next plateau. Currently, sensor telemetry is an example of what they are doing. Insurance companies were considered innovators when they used credit rating scores with transaction data to determine premiums. Now they are innovating with big data. They create customized insurance policies based on individual driving behavior such as how far you drive, how well you drive, when you drive, what conditions under which you drive, and where you drive. They monitor every driver action to understand driving behavior. Clearly, a lot of sensors are involved in collecting and transmitting this data. Typically the driver has a small device in the car such as under the steering column. This device monitors the drivers' pattern of driving—speed, rash cornering, sudden stops, sudden accelerations, time and distance-driven, weather conditions under which the car is being driven such as rain and snow, light conditions such as day or night, areas where the car is being driven, the driving speed in different areas—and transmits them in real time to the insurer or aggregator. The insurer uses this data to create a comprehensive picture of the driver, his driving habits, and the conditions under which he drives, perhaps even his state of mind. This allows insurance premiums to reflect actual driving habits. The driver also has access to this data and can use to change his behavior with a potential reduction in premium. There are different names for this form of car insurance such as pay as you drive, pay how you drive, and usage-based premium. Soon sensor telemetry will move from the innovators to become

the norm in the insurance industry. Currently, adoption is restricted by such things as state and country local regulations, privacy considerations, drivers' willingness to be monitored, and other information sharing concerns.

Sensor telemetry is part of a larger movement called telematics. Telematics is the broad term for machine to machine communication and implies a bidirectional exchange of data between endpoints. The data is from sensing, measuring, feedback, and control.

4 Characteristics of Data

This section shows some properties of data that have a bearing on the architecture.

Characteristics of data differ with their type and storage format. Data can be stored in highly modeled form or in a form that lacks a model. There are strong data models for storing transactional data such as relational (ER) models and hierarchical models. There are weaker models for storing Weblogs, social media, and document storage. There are storage forms that lack a model such as text, social media, call center logs, scanned documents, and other free forms of data. The data model has a bearing on the kinds of analytics that can be performed on them.

1. Value Density
   Value density or information per TB is a measure of the extent to which data has to be processed for getting information. It is different in different data forms. Transaction data has little extraneous data. Even if present, they are removed when they are stored in disk. Social media data on the other hand is sparse. Social media data is typically small amounts of interesting information within a large collection of seemingly repetitive data. Such data may contain the same thought or words multiple times. It may also contain thoughts and words outside the focus of discussion. Similar cases occur in other user produced data streams. Sensors produce data at fixed intervals and therefore may contain redundant data. The challenge is filtering the interesting information from the seemingly extraneous. Filtering useful data reduces the amount of data that needs to be stored. Such filtering occurs upfront in strong data model leading to higher information density. In weaker models, data is stored much closer to how they are produced and filtering occurs prior to consumption.
   The information density or value density increases as data is processed and structured. Value density is a continuum that increases as it moves from unstructured and gains structure.
   Value density of data has a bearing on storage ROI. For instance, low-cost storage may be appropriate to store low value density data.
2. Analytic Agility
   Data organization affects the ability to analyze data. Analytics is possible on all forms of data including transactional and big data. Analytics on structured transactional data with strong data models are more efficient for business

analytics than analytics on unstructured or poorly structured data with weaker data models. However, analytics on weaker data models can lead to insights that were previously unknown, that is, new surprise moments are possible on weaker data models.

There are different types of analytic insights.

One is understand what happened and why. An organization may wish to identify why certain products sell better in one region versus another, or combine sales data with product placement data and marketing campaign strategy in order to determine whether a campaign can succeed, or find out whether a customer is likely to attrition by comparing his interactions with other customers' interactions with similar background and transaction history. Business analysts gain such new insight by finding new ways to traverse the data. In a relational model, this means joining new tables or joining tables in new ways and querying and combining tables in new ways. This is one form of predictive insight.

Another is to find new patterns in data or find the questions that are worth answering. The intent is to find new cause–effect relationships. For instance, knowing that product placement has an impact on marketing campaign, and knowing the factors that cause a customer to attrition away from the vendor, and knowing the factors that influence a customer to buy from a specific vendor or knowing how much influence each factor has on his buying decision are new patterns found in the data.

These two completely different types of insights depend upon the data structure and data organization. When structure is weak or nonexistent, a data scientist applies different learning techniques to reveal different patterns and different structures in the underlying data. Such patterns are further refined using more learning techniques. Such mining can uncover totally new insights. Storing data closer to their original form has the benefit of retaining seemingly noisy data but can become useful with new learning techniques.

3. Frequency and Concurrency of Access

   Frequency and concurrency of data access is another characteristic that affects the data architecture.

   Business users ask repeated questions of the same data. Thousands of such business users operate on the same data at the same time. Transactional data in strong models have high reuse and high concurrency. Transactional data modeled as strong models are more efficient for business uses.

   Data scientists on the other hand ask mining kinds of questions on weakly structured data in order to sift through lots of data. Few data scientists operate on the same system at the same time when compared to business users. Schema-On-Read where the data is structured and a model is built each time the data is used is better for data scientists. Such users try out different models with each interaction in order to discover new structure in the data. Schema-On-Read, where no structure is presupposed, is easier with raw data storage. Data access for such interactions is typically low reuse and low concurrency.

High-reuse and high-concurrency forms of usage mean that it is acceptable to invest in efforts to structure the data. Low-reuse and low-concurrency forms of usage mean that it is acceptable to build structures each time the data is accessed.

One is better performing. The other is more flexible. One is appropriate for BI forms of analytics. The other is appropriate for big data forms of analytics.

Understanding these characteristics is critical to architecting the data solution. All data are important. Organizations cannot afford to throw away data merely because its value density is low. Mining it leads to valuable insight. This is a continuous process and there are quite a few different technologies for deep analysis of data (see Sect. 8.3).

Data has to be productized for broader impact. Data scientists perform investigations on raw data. In order to deliver more statistically accurate results, the data may be transformed to improve its quality, normalize the fields, and give it somewhat more structure for ease of further analysis by more users. Then as business insights are gleaned, the data may be further refined and structured to ease delivery of repeatable processes to deliver those insights. Finally, to integrate the data with the rest of the data in the enterprise, further refinement and structuring including extraction of attributes may be performed to add to traditional data models of the organization in order to enable widespread use of the derived results. At each stage, it is expected that the data size will reduce significantly. The reduction may be several orders of magnitude by the end of the phase, but it will have very large increase in usability by a general business user in the organization.

There are three distinct phases of data in a data-driven architecture—data capture, knowledge discovery, and knowledge usage. This is the focus of the next section.

5 Data-Driven Decision Making

Stephen Yu in his blog [3] divides analytics into four types:

(a) BI. ROI, reporting, and dashboards come under this category.
(b) Descriptive analytics. Dividing customers into segments and profiling profitable customers come under this category.
(c) Predictive analytics. Future profitability of a customer, scoring models, and predicting outcomes such as churn come under this category.
(d) Optimization. Optimization problems and what-if scenarios come under this category.

Data-driven decision making is about performing all these forms of analytics. The architecture must support all of them.

Gaining insight from data and making decisions based on that insight is the goal of data-driven decision making. Analytics creates information, understanding, and

finally insight, prediction, and foresight. Structure is core to analytics. Structure is either predetermined or it is built on the fly. The types of analytics on predetermined structures can be limiting. They are mostly derived from new ways of traversing the structure and combining entities in new and innovative ways to gain new insight. When structure is built on the fly new forms of insights are possible (see Sect. 4, Analytic Agility).

The velocity of big data requires that we capture it without loss. It should be transformed, cured, filtered, analyzed, validated, and operationalized in order to derive its value. Some forms of analytics may be possible to perform where the data lands. Some forms of analytics require structure. If the data fits one of the known structures, those structures are populated for use in every day analytics and decision making. Some forms of analytics require building structure on the fly. Such forms require learn–refine forms of analytic mentioned earlier. Deep analytics are possible using such forms and are performed to gather new structure. Such analytics create understanding of new patterns. All these require the management of three functional phases of data:

1. Data capture and storage. This is referred as data lake.
2. Discovery. This is referred as data R&D.
3. Productize. This is referred as data product.

Data lake is where data lands. It is the part that acquires and stores raw data. Some types of analytics are performed here. Data lake is alternately referred as data platform in this document.

Deep analytics on big data occurs at Data R&D. This is where insights hitherto unknown is gained.

Data product is where analytics such as BI, dashboards, and operational analytics on strong data model occur. This is also where every day analytics occur.

Our descriptions highlight the primary role of each phase. In real implementations, their roles and capabilities blur at the boundaries and are not as sharply differentiated as we have listed. All three aspects perform some form of analytic. The analytics vary in their depth. Data storage and data model have a bearing on the kind of analytics that are possible. The volume and variety of non-relational data have a bearing on where an analytic is performed. More types of predictive and optimization types of analytics occur in data R&D. BI and some forms of predictive analytics occur in data product. Some forms of descriptive analytics and predictive analytics occur in data lake.

Data goes through the following five stages as part of analyzing and gaining insight: acquire, prepare, analyze, validate, and operationalize.

Data lake acquires, stores, extracts, and organizes raw data. Some forms of reporting and some forms of descriptive analytics are performed here. Predictive and operational analytics are performed at both data product and data R&D. They differ on the forms of data on which they operate. Once structure is perceived in data then new structure can be created and loaded onto the data product portion. This in effect operationalizes the analytics. For data without strong structure or in

which structure is not apparent, structure has to be derived in order to perform analytics. The data R&D portion is used to create structure for analytics on big data.

Each of the three functional aspects is elaborated in the following sections.

5.1 Data Lake

Wikipedia defines a data lake as a "storage repository that holds a vast amount of raw data in its native format until it is needed." It further says "when a business question arises, the data lake can be queried for relevant data, and that smaller set of data can then be analyzed to help answer the question."

The above-mentioned definition means that data lake is the primary repository of raw data, and raw data is accessed only when the need arises. The accessed data is analyzed either at the data lake itself or at the other two functional points.

Raw data is produced at ever increasing rates and the volume of overall data an organization receives is increasing. These data must be captured as quickly as possible and stored as cheaply as possible. None of the data must be lost. The value density of such data is typically low and the actual value of the data is unknown when it first arrives. So an infrastructure that captures and stores it at low cost is needed. It need not provide all the classic ACID properties of a database. Until the data is needed or queried, it is stored in raw form. It is not processed until there is a need to integrate it with the rest of the enterprise.

Raw data must be transformed for it to be useful. Raw data comes in different forms. The process of raw data transformation and extraction is called data wrangling. Wikipedia [4] defines the process of data wrangling as "Data munging or data wrangling is loosely the process of manually converting or mapping data from one "raw" form into another format that allows for more convenient consumption of the data with the help of semi-automated tools. This may include further munging, data visualization, data aggregation, training a statistical model, as well as many other potential uses. Data munging as a process typically follows a set of general steps which begin with extracting the data in a raw form from the data source, "munging" the raw data using algorithms (e.g. sorting) or parsing the data into predefined data structures, and finally depositing the resulting content into a data sink for storage and future use." It elaborates on the importance of data wrangling for big data as follows "Given the rapid growth of the internet such techniques (of data wrangling) will become increasingly important in the organization of the growing amounts of data available." Data wrangling is one form of analysis that is performed in the data lake. This process extracts from multiple raw data sources and transforms them into structures that are appropriate for loading onto the data product and data R&D. In general, it cleanses, transforms, summarizes, and organizes the raw data before loading. This process is performed for non-transactional big data besides transactional data. It loads the data R&D for Weblog, social data, machine, sensor, and other forms of big data. The data lake or data platform increases the value density by virtue of the cleansing and transformation process. According to eBay, a

leading-edge big data user, "cutting the data the right way is key to good science and one of the biggest tasks in this effort is data cleaning" [5].

Some of the insights gained here from analysis are stored locally in the data platform and others are loaded onto the data product or data R&D platforms. In order to perform analysis, some form of descriptive metadata about the raw data should also be available in the data platform. Metadata describes the objects in the various systems and applications from where data is collected, much like a card index in a library. Metadata is needed for analyzing the raw data.

Hadoop is the platform often associated with the data lake or data platform. Data is stored in clusters of low-cost storage. Hadoop uses a distributed file system for storage and is called the Hadoop distributed file system (HDFS). The storage format is key-value pairs. It stores data redundantly and is fault-tolerant. Data is processed in clusters of low-cost nodes using MapReduce which is a highly scalable processing paradigm. Hadoop provides horizontal scaling for many kinds of compute-intensive operations. Therefore, many kinds of analytics are possible in data platform.

The performance metric for the data platform is TB per sec of data capture capacity and dollars per TB of data storage capacity.

5.2 Data Product

Insight gained from analytics is productized for strategic and tactical decision making. An enterprise data warehouse (EDW) is the platform often associated with data product. Reports, analytical dashboards, and operational dashboards with key performance indicators are produced at the warehouse. These provide insight into what happened, why it happened, and what is likely to happen.

Data arrives at the warehouse from the data platform categorized and in a form to fit the relational model structure. This structure presupposes how the data is going to be analyzed. This is unlike how data arrives at the data platform.

The data storage structure has a bearing on the insight derivable from data (see section on analytic agility). Data product provides insight on relational forms of storage. Insight comes from traversing the data in complex and innovative ways. The extent to which such traversal is possible is the extent to which new insight can be gained from the data. Such insights include predictive analytics. For instance, it can determine whether a customer has a propensity toward fraud or whether a transaction is in reality a fraudulent transaction. Such analyses require among other things an understanding of the card holders' previous transactions, an understanding of the transactions that typically end up as fraudulent transaction and the conditions under which they become fraudulent. All this goes in deciding whether to approve the current card transaction. Similarly, predicting the success of a marketing campaign is possible depending on the analysis of previous campaigns for related products during related times of year. Similar examples exist in other

domains. Innovativeness of analysis drives the amount of insight that can be derived from data.

Performance, efficient execution of queries, and availability are key requirements of data product platform. ACID property is necessary for such data since this is data repository of record.

Throughput and response time are the metrics and are measured in queries per second, dollars per analytic, and seconds per query.

5.3 Data R&D

Data R&D provides insight on big data. Insight also means knowing what questions are worth answering and what patterns are worth discerning. Such questions and patterns are typically new and hitherto unknown. Knowing such questions and patterns allows for their application at a later time including in the data product portion. Therefore, data R&D is what creates intelligence through discovery for the organization on new subject matters. It also refines and adds to previous insight. For example, it is a new discovery to determine for the first time that card transactions can at times be fraudulent. A refinement of this discovery is to recognize the kinds of behavior that indicate fraud. In other words having the intelligence to ask the question "can transaction be fraudulent?" is a big learning. Determining the patterns that lead to fraud is another form of learning.

Data R&D is the primary site where deep analytic occurs on big data. Model building is a key part of deep analytics. Knowledge or insight is gained by understanding the structure that exists in the data. This requires analysis of different forms of data from different sources using multiple techniques. This portion has the ability to iteratively build models by accessing and combining information from different data sources using multiple techniques (refer Sect. 7). It also constantly refines such models as new data is received.

There are multiple techniques and technologies in the R&D space. The ability to perform analytic on any type of data is a key requirement here.

The metric for this functional aspect of data is dollars per analytic performed and dollars per new pattern detected.

Analysis of big data goes through these stages—acquire, prepare, analyze, train, validate, visualize, and operationalize. The data platform acquires data and prepares it. The analytic part in R&D recognizes patterns in big data. Many big data analytics use correlation-based analytic techniques. A correlation among large volume of data can suggest a cause–effect relationship among the data elements. Analyzing large volumes of data gives confidence that such patterns have a high probability of repeating. This is the validate part which ensures that the patterns recognized are legitimate for all different scenarios. The train part is to automate the pattern recognition process. The operationalize part is to put structure around the data so they can be ready for high-volume analytics in the product platform.

6 Deriving Value from Data

Deriving value from data is not about big data alone. It is about capturing, transforming, and dredging all data for insight and applying the new insight to everyday analytics.

Big data analysis has been about analyzing social data, Web stream data, and text from call centers and other such big data sources. Typical approaches build independent special systems to analyze such data and develop special algorithms that look at each of these data in isolation. But analyzing these data alone is not where the real value lies. Building systems that go after each type of data or analyzing each type of data in isolation is of limited value. An integrated approach is required for realizing the full potential of data.

Analyzing without a global model or context does not lead to learning. There is definitely some learning that exists in such analysis however. Social Web data or textual entries from a call center contain extremely valuable information. It is interesting to analyze a Web clickstream and learn that customers leave the Web site before they complete the transaction. It is interesting to learn that certain pages of a Web site have errors or that certain pages are more frequently visited than others. It is interesting to learn that a series of clicks in a specific order is predictive of customer churn. It is interesting to learn how to improve user experiences with a Web site. It is interesting to learn in depth about the behavior of a Web site. It is interesting to learn from graph analysis of the social Web the influencers and followers. There is value in all this. However, this value is far less than the value that is derived when it is combined with other data and information from other sources from the rest of the organization. This kind of Metcalf's law applies to data —connecting data sources and analyzing them produces far more value than analyzing each of the data sources alone. Combining the behavior of a customer on the Web site with what is already known about the customer from his transactions and from other interactions provides more insight about the customer. For instance, a call center conversation shows a trend in how people talk to each other but combining information about the customer on the call and information about the call center representative on the call (with the customer) adds context about both parties and shows what kind of representative has what kind of effect on what kind of customer. This is more valuable insight than how people talk to each other. Similarly combining product experiences expressed on the Web with product details, suppliers' information, and repair history gives a better understanding of the views expressed in social media. Analyzing social media such as Twitter and Facebook can reveal an overall sentiment trend. For instance, it can tell how people feel about a product or company. It can tell about whether people are complaining about a product. Such information in isolation is interesting but lightweight in value. Linking the active people on the social media with the customer base of the company is more valuable. It combines customer behavior with customer transactions. The sentiments become yet another attribute of a known customer and can

be analyzed holistically with other data. With a holistic view, social Web chatter becomes tailored and more actionable.

Adding global context to any isolated analysis increases the value of such analysis significantly. This is true for transaction analysis. This is true for raw data transformation and cleansing. This is true for big data analysis. For instance, analyzing sensor data with context from patient information and drug intake information makes the sensed data more meaningful.

Therefore, deriving value from big data is not about analyzing only big data sources. It is about reasoning with all data including transactions, interactions, and observations. It is myopic to build an environment for analyzing big data that treats big data sources as individual silos. A total analytic environment that combines all data is the best way to maximize value from big data. This is one of the key requirements for our big data architecture—the ability to reason with all data.

The three functional components we discussed earlier—Data lake, data R&D, and data product—must all be combined for any analysis to be complete. Therefore, reasoning with all data is possible only if the three aspects of data are integrated. We call this integrated analytics.

6.1 The Three Functional Components at Work

The discovery process puts all these three functional components of data-driven decision making together through integrated analytics.

Sales listing applications, barter listing applications, or auction item listing applications are examples of integrated analytics. Such applications determine, through analysis, that a number of different attributes of an item listed for sale impact a buyers' purchasing decision. The goodness metric on each of these attributes affect the order of how the sale items are listed in a search result. For instance if multiple parties list similar products, the search engine would return and list the results based on ranking of these attributes. It is advantageous for better buyer response to have items listed in the beginning of the list rather than lower down in the list. Therefore, sellers prefer their items in the beginning of the list.

There are three functions that are necessary in this application. One, determining the list of attributes that users consider important. This means finding out what attributes impact customer choices. This is a deep analytic. Two, rating the quality of each attribute for each item being listed. Attributes may be objective such as price and age. Attributes may be subjective such as product photograph, product packaging, and product color. Analysis is required to rate the quality of each attribute. Such analysis is harder for subjective attributes. Subjective items such as photograph quality and packaging quality should be analyzed on the basis of background, clarity, sharpness, contrast, color, and perhaps many others. Data platform performs these kinds of analytic using MapReduce. Three, combining all

these to determine the listing order. The data product considers the rating of each of the attribute in order to determine the listing order of each item for sale when a search is performed.

7 Data R&D—The Fertile Ground for Innovation

Data R&D is the challenging part. The other two parts are fairly straightforward. Hadoop clusters meet the requirement of the data platform. Hadoop clusters are scalable and are the preferred solution for the data platform. They can capture all forms of data from an array of sources and store them with redundancy and high availability. They can also do the forms of analytics mentioned earlier using MapReduce. An EDW meets the requirement of data product. Parallel RDBMS can meet the complexity and performance requirements.

The challenge is the middle part or data R&D. Neither MapReduce alone nor relational technology alone is appropriate for this phase. RDBMS requires the data to fit into well-defined data models. Hadoop clusters require super human programmers to program new applications.

Figure 4 shows this challenge.

Data R&D is the focus of a number of innovations and new start-ups. Technologies such as graph engine, text analytics, natural language processing, machine learning, neural networks, and new sensor models are being explored for big data. There are four broad types of efforts currently underway:

1. Bottom-up approaches. These provide relational database-like aspects on top of Hadoop. They add SQL or SQL-like features to Hadoop and HDFS. Some leading Hadoop vendors and Hadoop open source groups fall under this

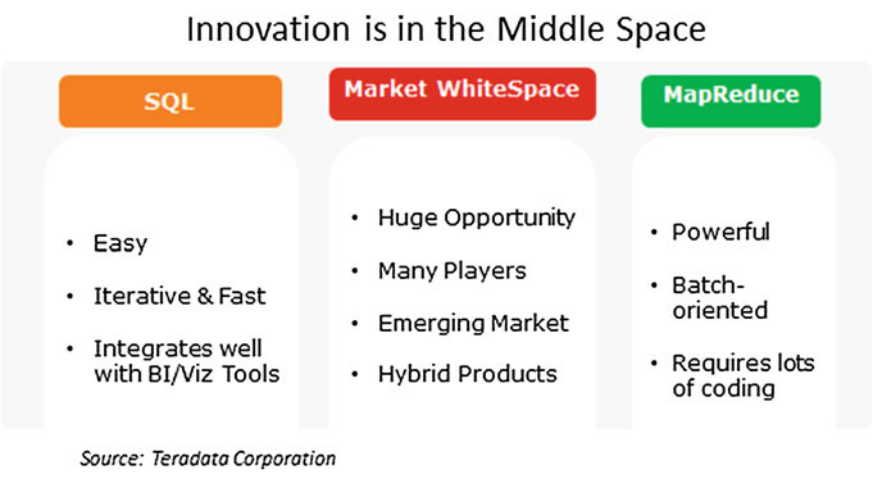

Fig. 4 Innovation rich R&D environment

category. Examples include Apache Hive [6], Cloudera® Impala [7], Hortonworks® Stinger [8], Apache HBase [9] among others. The last one is not relational, but the idea is to provide classical database-like features and consistency property for sparse data.

2. Top-down approaches. This approach starts with a mature SQL engine and integrates other capabilities as callable facilities. SQL is used as a kind of scripting-like facility in order to orchestrate access to other stores. They embed other technologies such as natural language processing functions, pathing functions, and others as SQL callable functions using existing frameworks such as UDFs or through proprietary frameworks. For some technologies, it creates applicable storage natively. Many leading database companies are doing this approach. Besides call-outs, some support other storage and processing technologies natively.
3. Application-specific approach. These are specific solutions targeted toward a specific application. They create targeted technologies and solutions specific to a problem. These may be on top of Hadoop in some cases. Splunk® [10] is one successful example of this approach. It is a specific solution for collecting, analyzing, and visualizing machine-generated data. It provides a unified way to organize and extract real-time insights related to performance, security, and privacy from massive amounts of machine data generated across diverse sources.
4. Technology-specific approach. This is similar to the above-mentioned third category but is technology-specific rather than application-specific. These also target specific problems. Graph engines, path analytics, machine learning, and natural language processing engines are some examples. There are platforms available for this approach from open source and industry.

Specific solutions such as above-mentioned items 3 and 4 are appropriate as stand-alone solutions. They often target a specific problem. They are the best of breed in their area. Splunk® [10] is very successful in analyzing machine logs. SPARQLVerse [11] is good as a traversal engine for graphs representations. Depending on the need, some products and solutions are ideal for a problem but their broad applicability is limited. Therefore, we recommend such solutions only if the problem is confined to what the solution can address.

The choice for data R&D architecture is a choice between the first two approaches. Adding SQL maturity to HDFS seems like a significant effort. This approach assumes all future technology evolutions for the different forms of big data analytics use HDFS.

The SQL foundation approach seems broader in that it attempts to go beyond SQL and integrate MapReduce, HDFS, and other technologies. This is important when big data space and the technologies and algorithms that operate on big data are evolving.

The following picture shows the chaotic evolution for knowledge discovery.

Figure 5 shows stand-alone solutions that target specific problems. This is acceptable if the problem is limited to the application of one such solution.

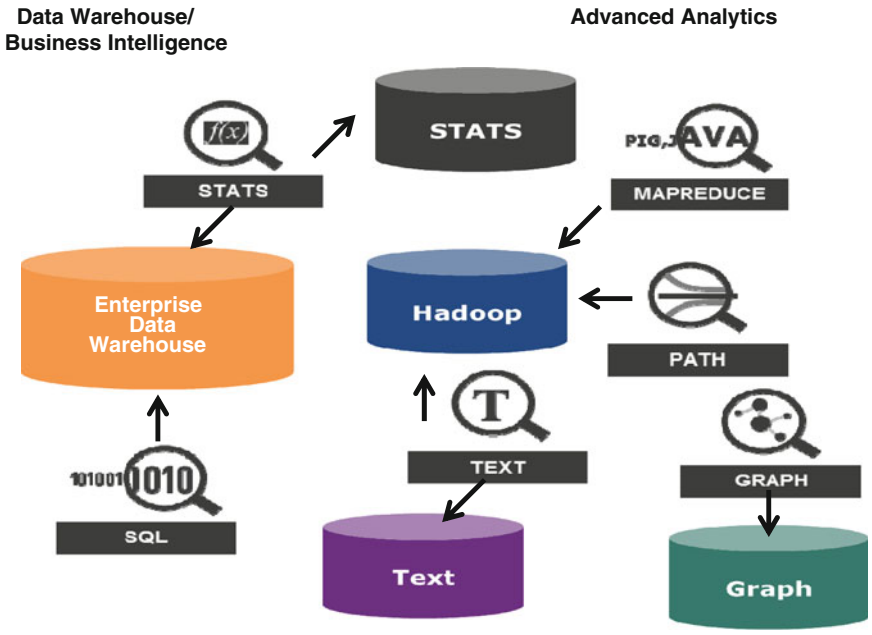

Fig. 5 Bunch of stand-alone solutions is a problem

However, creating insight is an iterative approach where multiple techniques must be applied one after the other. For instance sessionization, followed by text analytics, followed by sentiment analysis, followed by pathing must be applied in order to understand customer behavior across multiple channels. Each technology application refines the input data set and the refined set is fed to another technique. This process continues until insight is gained or structure is found in the data. Such refinement may include reaching out to raw data storage and to SQL stores besides applying multiple big data technologies.

Applying each of the techniques as free-standing techniques repeatedly on an iteratively refined data set requires highly skilled programmers. It is also human intensive when more than a few techniques have to be applied on the same problem. For example, repeated application requires data files to be produced and formatted before each new technique is applied. It is not a scalable solution. By definition, insight creation is a trial-and-error process. The number of application of such trials reduces when each is hard to apply. An integrated mechanism is therefore needed.

Figure 6 shows such an integrated approach. The discovery platform integrates other technologies.

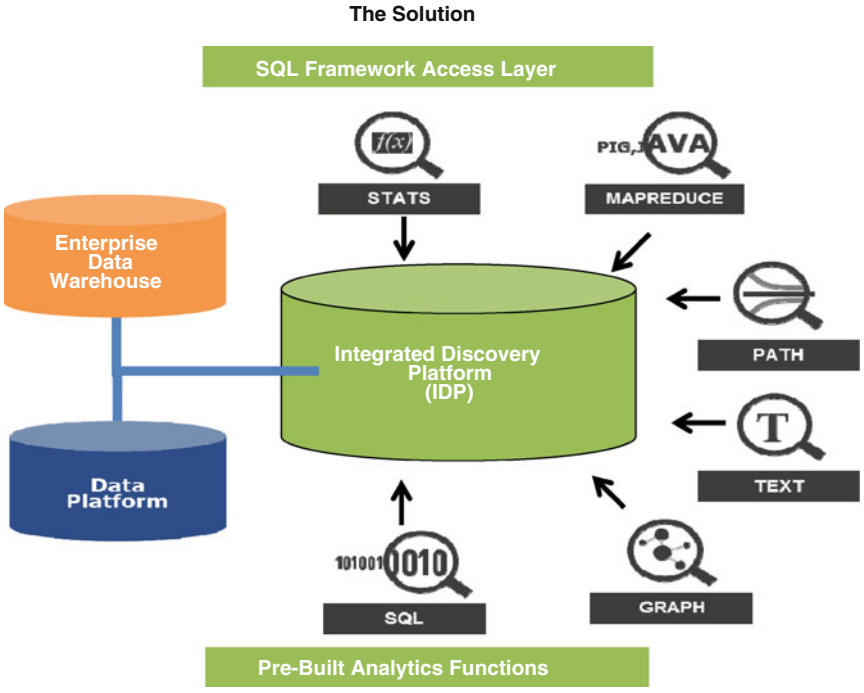

Fig. 6 Integrated discovery platform

The discovery platform starts with a mature SQL engine which acts as a discovery hub operating on mixed data—big data, HDFS data, relational data, and other stores. It has different forms of local store such as a graph store besides SQL store. It uses the SQL language as a kind of scripting facility to access relational store and other stores through UDF and proprietary interfaces. The SQL engine provides the usual SQL platform capabilities of scalability and availability. The platform can also execute MapReduce functions. The discovery platform may natively implement big data analytic techniques such as time series analysis and graph engine. The platform may also natively support other data stores such as HDFS and graph storage. It may also access stand-alone techniques such as machine learning to return results that are refined by that technique.

The industry can encourage this form of discovery through standardization efforts for interaction across platforms for data and function shipping. Currently, these are mostly ad hoc and extensions are specific to the vendor.

8 Building the Data Architecture

Reasoning with all data is possible only if the three functional aspects of data are interconnected.

Earlier, we specified three functional aspects of data. These aspects evolved data from raw to a product by undergoing R&D. These aspects were mapped to a platform:

1. Data lake or data platform. Raw data lands here and is stored. This platform has the capacity and cost profile to capture large volumes of data from varied sources without any loss or time lag. This is typically a Hadoop cluster.
2. Data product. BI, analytical dashboards, and relational forms of predictive analytics happen here. This platform supports high reuse and high concurrency of data access. It supports mixed workloads of varying complexity with large number of users. This is typically a parallel EDW platform.
3. Data R&D. Deep analytics from big data happens here. This platform determines new patterns and new insights from varied sets of data through iterative data mining and application of multiple other big data analytic techniques. The system combines a number of different technologies for discovering insight. This is the integrated discovery platform with tentacles that reach into other stand-alone technology engines besides implementing multiple technologies and multiple data stores natively.

Data lake is satisfactory for stand-alone applications. For example, a ride-sharing company can create a stand-alone booking application in the data lake. The booking application connects passengers to drivers of vehicles for hire. Customers use mobile apps to book a ride, cancel a ride, or check their reservation status and other similar operations. Such applications have a high number of users and a high number of passenger interactions. Such applications can be implemented as a stand-alone entity in the data lake using Hadoop. The application is scalable in terms of storage capacity and processing capacity.

An integrated analytic is, however, required if we want to go beyond stand-alone applications. Such analytic is more valuable. For example, an integrated analytic that reaches elsewhere for data is required if the ride-sharing company wants to consider as part of the booking application other information such as vehicle maintenance records, customer billing records, and driver safety records. These kinds of information usually come from other places in the organizations chiefly from the EDW. Integrated analytics require combining data lake and data product. If in addition the analytic wants to understand driver routes and driving patterns and to perform graph analytics on driver movements and location of booking data, R&D becomes a part of this integration with data lake and data product.

This requires an architecture that integrates all three platforms as shown in Fig. 7.

The focus of this architecture is the data layer and everything else is pushed off to the side. A complete architecture needs all the access tools, data acquisition tools, and all visualization tools. It requires multiple mechanisms to transport and load

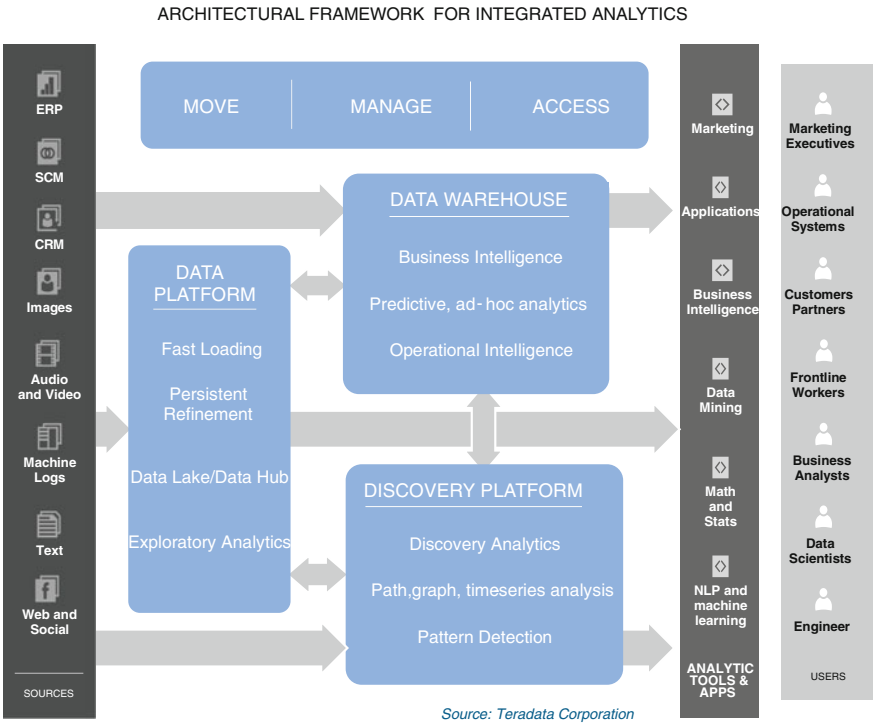

Fig. 7 Integrated analytics architecture

data in parallel. It requires all the client tools that interact with each of these three platforms. Some of these facets of the architecture are merely shown as small boxes but are not elaborated or explained in this picture. Each of these is an entire technology and outside the data architecture portion of this chapter.

Data connections between the three data storage platforms (lines and arrows in this chart) are more important than the data stores. Data stores are an acceptance that there must be different kinds of platforms with different kinds of capabilities and different kinds of attributes to build an overall successful analytic solution. These were elaborated in the previous sections of this chapter. These evolve over time. The arrows show the need for their interaction. As observed earlier, the industry is doing very little by way of standardization in these interaction areas. It is left to individual vendors to evolve their own solution.

Unlike transactional data, the value of big data is often unknown when it initially arrives or is accepted for storage. It is not possible to know all the patterns the data exhibits unless we know in advance all the questions that are worth answering from that data. New questions may come at any time in the future. Therefore, organizing and storing the organized data is not very useful for such analytics. Organizing it upfront loses some of the values that are in the original data even though such value may be unknown at the time of storage.

8.1 Data Platform or Data Lake

The data platform is the data landing zone. All new forms of raw data arrive at this platform for storage and processing. The goal is get the data in, especially new types of data, quickly, and directly as possible; land it in the cheapest storage; and have a platform to efficiently filter, preprocess, and organize the landed data for easier analysis. The primary point of the data platform is to deal with low value density data and increase the value density.

Hadoop is the platform of choice here. It has a high degree of parallelism and low cost of horizontal scaling. MapReduce is agile and flexible. It can be programmed to perform all forms of data transformations. In addition, an increasing number of tools are available to work on data in Hadoop for improving data quality and value density. These range from traditional extract–transform–load (ETL) tools expanding their footprints to new data wrangling tools that are developed specifically for this space.

An example analytic that was covered earlier was to score and rate subjective attributes such as photographs and packaging for their quality for items listed for sale.

Organizing and transforming are forms of analytics that improve value density. Organizing means for instance adding cross-referencing links between store ID and customer ID. Sessionizing is an example of organizing data such as Web records in order to infer that all the Web records are part of a particular conversation with a user. This is a highly resource-intensive and complex operation especially since volumes are high and session logs interleave many different users.

Sessionization is a common form of organization for customer-related interactions. A customer may interact from multiple touch points. Making a chronological order of those interactions organizes the data for analysis. Figure 8 shows customer interactions across multiple channels.

Another example of organizing is text analytics where say four encoded attributes from call center interaction text are extracted and forwarded to the customer record. For example, this customer complained about this problem about this product on this day could be the four attributes. The general goal is to structure the data or add a bit of structure to the data for ease of understanding.

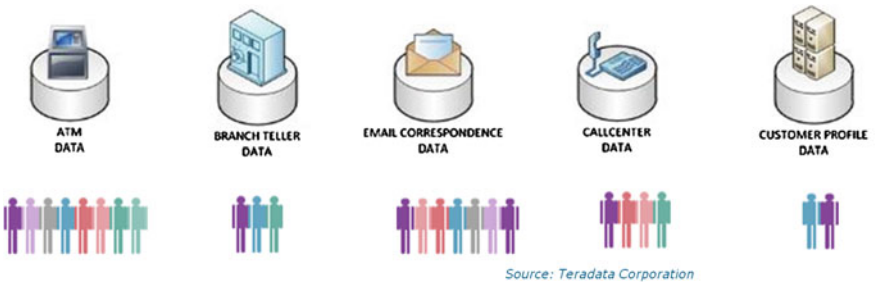

Fig. 8 Sessionization—interaction across different touch points

Figure 8 shows the sessionization process. It shows the importance of this process for all social interaction-related analytics.

Figure 8 shows four example channels of customer interactions—ATM, Teller, e-mail and written correspondences, and call center. It also shows customer profile data.

Many customers arrive at each of these interaction touch points. Many may visit the same touch point more than once at different times. This chaos has to be sorted. The touch point each customer visits is important. The order in which they visit each touch point is important.

Figure 9 shows the result of the sessionization process. Wikipedia [12] defines sessionization and elaborates it this way: "Sessionization is a common analytic operation in big data analysis and is used to measure user behavior. Behavioral analytics focuses on how and why users of eCommerce platforms, online games and web applications behave by grouping certain events into sessions. A sessionization operation identifies users' web browsing sessions by storing recorded events and grouping them from each user, based on the time-intervals between each and every event. Conceptually, if two events from the same user are made too far apart in time, they will be treated as coming from two browsing sessions. A session can be as short as a few seconds or as long as several hours." This Wikipedia definition should help in understanding Fig. 9. Basically, sessionization extracts the series of customer interactions and their order. A customer may come from a single session or from multiple sessions. If the interactions are too far apart in time, they may be considered different sessions. They may come from different IP. Therefore, studying the session ID, the time sequence, and time gap are

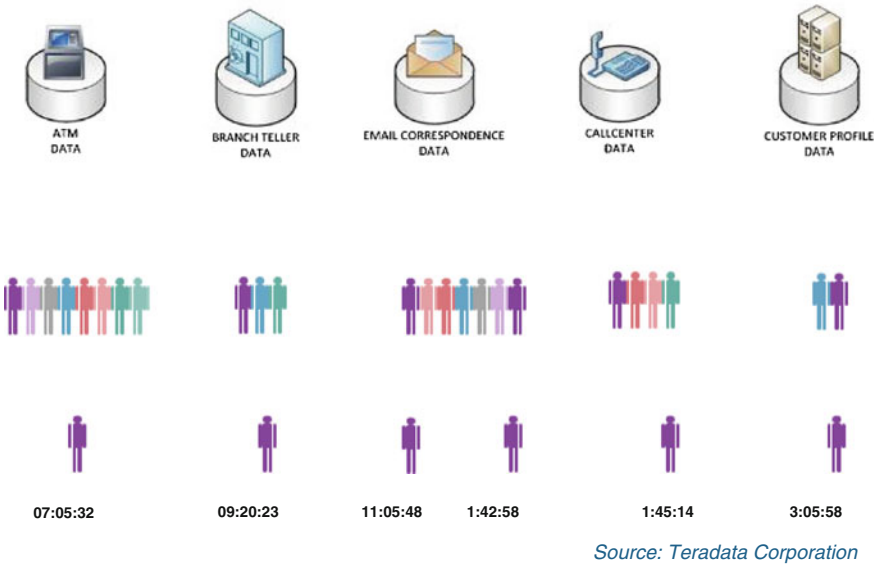

Fig. 9 Sessionized result

all important to determine and stitch together the customer interactions in time order. The customer profile and transaction data are integrated with the sessionized data. The integration of all this data along with path analysis determines the customer interactions that led to the final outcome. Combined with the profitability of the customer, necessary action can be taken. This example shows multiple technologies being combined: sessionization, pathing, and relational databases.

ETL is another form of organization that is also performed on the data platform. ETL applies to both big data and transactional data. ETL on big data includes operations on raw data such as structured and semi-structured transformations, sessionization, removing XML tags, and extracting key words.

Archival is another key operation performed at the data platform. The data landing zone has different retention periods compared to older architectures. In the past, the landing zone was a very transient place where structure was added through ETL processes and then the landed data was destroyed. Only the structured data was retained for regulatory and other archival purposes. The landing zone in this architecture is where raw data comes in and value is extracted as needed. The raw data is archived.

There are privacy, security, and governance issues related to big data in general and the data platform in particular. Access to raw data cannot be universal. Only the right people should have access. Also data life cycle management issues about when to destroy data and the considerations for doing that are important. There are also questions about how to differentiate between insights derived versus third-party data and how to protect each of these. Some of these issues are covered in other places in this book. Some are outside the scope of this chapter. They are mentioned here to indicate these have to be considered in the overall architecture discussions.

8.2 Data Product

Parallel RDBMSs are the platform of choice for the data product platform. BI, dashboards, operational analytics, and predictive analytics are all performed here. Such analytics are based upon relational forms of data storage. Profitability analysis is an example of the kind of predictive analytics this platform performs. The profitability of a customer is based on revenues from the customer versus costs associated with maintaining the relationships with the customer in a specific period. Future profitability is predicted by associating customer characteristics and transaction patterns over a period with other known customers' and their characteristics. Profitability is also predicted during customer acquisition. Questions such as whether the customer will be profitable and is he worth acquiring are answered here. These are not covered further in this chapter.

As mentioned earlier, big data analytics is a combination of analytics using MapReduce on flat files and key-value stores, SQL on relational stores, and other forms of analytics on other forms of storage. Different analytics are combined for discovery in the discovery or R&D platform. Varieties of different technologies are

involved in this area. Innovation is also constantly changing this space. The discovery process is different from the discovery or R&D platform. If the problem is SQL tractable and the data is already in the warehouse, then discovery is done in the EDW. If a particular algorithm is appropriate on Hadoop and HDFS, then discovery is done in the Hadoop platform. The discovery process is done in the discovery platform when big data and SQL have to be combined with SQL, MapReduce, and others technologies such as graph engines, neural nets, and machine learning algorithms.

8.3 Data R&D or Data Discovery Platform

The discovery or R&D platform is one of the key pieces of the architecture for a learning organization. New insights gained here are used by other components to drive day to day decision making. It has the capabilities and technologies to derive new insights from big data by combining data from other repositories. It has the capabilities to apply multiple techniques to refine big data.

Health care is an example of such capabilities. Big data analytics are applied successfully in different care areas. One example application identifies common paths and patterns that lead to expensive surgery. Another application reduces physician offices' manual efforts for avoiding fraud and wasteful activities. These applications combine different kinds of analytics. They pool data from all three platforms. Data sources include physician notes, patient medical records, drug information, and billing records. Analytic techniques applied include fuzzy matching, text analytics, OCR processing, relational processing, and path analytics.

In industries such as insurance, sessionation and pathing techniques are combined to understand paths and patterns that lead to a policy purchase from a Web site or analyze customer driving behaviors for risk analysis and premium pricing.

In aircraft industry, graph analytics, sessionization, and pathing are combined to analyze sensor data along with aircraft maintenance records to predict part failure and improve safety of aircrafts.

As we saw earlier, interactions are much higher in volume than transactions. It is possible to predict the value of a customer from his transactions. It is possible to understand and predict behavior of a customer from his interactions. Predicting behavior is more valuable insight. Data analytic techniques on different forms of user interactions make such predictions possible. There are different techniques to predict behavior.

Clustering is an analytic technique to predict behavior. Collaborative filtering and affinity analysis techniques fall under this category. The idea is that if a person A has tastes in one area similar to a person B, he is likely to have similar tastes in another area. Market basket analysis and recommender systems are examples of this form of analytics. "People who bought this also bought this other item" and "You may be interested in viewing this profile" are some examples of recommendations based on behavior. Combining a customers' historical market basket with the

shopping patterns of "similar" buyers can provide personalized recommendations to the customer. There are also time-ordered collaborative filters which consider the order in which items are likely to be put into the basket. These kinds of analytics combine time series analysis and affinity analysis. There are other forms of cluster analysis such as *k*-means which clusters data such as customers into a specified number of groupings, and canopy which partitions data into overlapping subsets. Each of these techniques requires a chapter of its own and is therefore outside the scope of this chapter.

Statistical analysis is another class of big data analytics. Analytics such as correlation [13] which determines the strength of relationships between different variables, regression which is used to forecast and predict the future based on past observations, classification which identifies the set of population to which a new observation belongs such as patients based on biological readings, and many more are part of this technique.

Text analytics are another class of big data analytics. Sentiment analysis [14], which classifies user statements as positive or negative, and text categorization, which is used to label content such as e-mail as spam, and many more are part of this technique.

Graphs are a technique for modeling relations in any field [15, 16]. In social graphs, they are used to measure a persons' prestige and his ability to influence others. In government, they are used to identify threats through (a) detection of non-obvious patterns of relationships and (b) group communications in e-mail, text, and telephone records. In health care, it is used to detect drug efficacy and outcome analysis. Graph analytics are flexible. It is possible to add to an existing graph incrementally. For instance, it is possible to add new data sources and new relationship to an existing graph in order to support new lines of inquiry. Graph analytics are highly mathematics oriented. Some common graph analysis functions include centrality, pathing, community detection, direction of relationships, and strength of relationships. These are also outside the scope of this chapter. The reader is left to deduce what these analyses imply from the name of the technique. They are mentioned here to convey the kinds of deep analytics on big data that occur at the discovery platform.

9 Putting It All Together

This section shows a use-case that implements our data architecture. It is a big data analytic for preventive aircraft maintenance and for providing part safety warnings to airlines.

Figure 10 shows integrated analytic with all three platforms. The arrows in this picture are data flow and are not indicative of process or control flow.

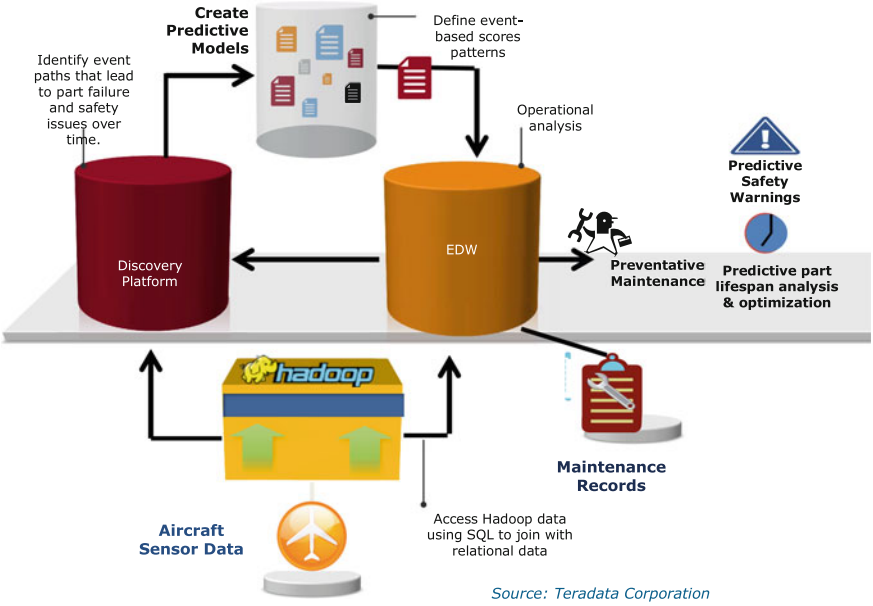

Fig. 10 Data refinery—refining raw data to actionable knowledge

The overall flow is as follows. Sensor data arrives at the data platform. This platform reaches out to the data product and data R&D platforms. Sessionized data is sent to the data R&D platform. The R&D platform models the part and reaches out to the data product and data platforms for information. The product platform performs BI analytics and sends out warnings. It reaches out to the data R&D and data platforms for supporting information.

Each system reaches out to the other for data that is needed in order to make sense of its patterns.

Sensor data are received and analyzed in the data platform (Data lake). However, if such analysis is done in isolation, then it is unwise to execute on it. Sensor data information itself is insufficient. The sensor readings must be organized. It must be sessionized for a time sequence of readings. The discovery platform puts the readings in the context of a part. The part is modeled in order to understand its behavior. This requires knowledge of the part and other sensor readings. These different readings are organized in order to understand part behavior. Some of this is done where the data lands. Most of it is done in the discovery platform (data R&D). Path analysis and statistical analysis are some of the analytical capabilities used to model and identify the pattern of failures of the part.

It is not enough to determine the current state of the part in the aircraft. Information is also needed about when the part was last maintained, the history of the part's behavior on this aircraft, history of the part's behavior in other aircrafts (i.e., the global database), the serial number of the part on this aircraft, and a host of other information. In addition, part advisories from the manufacturer are also needed from documents and other manufacturers release materials. Natural language analytic [17], text analytic [18, 19], and SQL are some of the other techniques used for analysis. The end result of this knowledge refinery is the issuance of a preventive maintenance direction and safety warning for failing parts in order to keep the plane flying safely.

Figure 10 is like a data refinery. Raw data enters the data platform. Interesting stuff is extracted from it and passed for discovery. Data R&D models with enterprise data, transaction data, and dimensional data from the operations side. It is passed to data product which produces the refined output and issues the warnings. Learning from this refinery is used to enhance the product platform for future automated and large-scale application of knowledge.

10 Architecture Futures

We see the future of integrated analytic evolving to satisfy two goals:

(a) The integrated analytic refinery shown in the previous section should be projected as a single system for interaction and control. This means that the data lake, data product, and data R&D are treated as a single integrated entity for control, load, and querying.
(b) The maturity, robustness, and predictability capabilities of EDW should be made available on the integrated entity.

These two goals translate into the following enhancements:

1. A global front end for analytic that optimizes user queries and interactions across all three systems.
2. A data loader that determines where objects should reside. Such decisions are based among other things on different properties of data such as data temperature, frequency of data usage, and frequency of analytic types.
3. A work load manager that guarantees metric of analytic refinery as a whole. Metric such as dollars per analytic performed, throughput, and response time is managed for delivery by the work load manager.

Acknowledgments The author wishes to acknowledge Todd Walter, Teradata Distinguished Fellow, for his many discussions and ideas on this topic. Todd has been a reservoir of valuable information and has been willing to freely share it.

The views expressed in this chapter are the authors' own and do not reflect Teradata corporations' views or ideas.

Questions

1. What is the primary intent of big data?
2. What forms of data constitutes the big data?
3. What are the 4Vs of big data?
4. Can a data warehouse be considered a solution for big data system?
5. Can Hadoop be considered a solution for big data system?
6. Define big data.
7. What caused the large data volume increases? In what order did the volume increases occur?
8. What forms of data are called interaction data?
9. What forms of data are called observation data?
10. How do the 3Vs relate to each other?
11. What are the data characteristics that must be considered for a successful big data architecture? Explain each of them.
12. List the different forms of analytic and write briefly about each of them.
13. What are the three functional phases of data for a big data architecture? Give a brief description of each of them.
14. What are the performance metric of each functional phase of data in a big data architecture? Give a brief analysis of each of this metric and why they make sense.
15. What are the aspects of the big data architecture. Discuss each of them.
16. Discuss some forms of data R&D analytic techniques.
17. Discuss an application of big data architecture.
18. Describe the kind of functions that are performed in each of the three data stores of the architecture with examples.

References

1. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
2. http://anlenterprises.com/2012/10/30/ibms-4th-v-for-big-data-veracity
3. http://www.infogroup.com/resources/blog/4-major-types-of-analytics
4. http://en.wikipedia.org/wiki/Data_wrangling
5. http://petersposting.blogspot.in/2013/06/how-ebay-uses-data-and-analytics-to-get.html
6. https://hive.apache.org/
7. http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html
8. http://hortonworks.com/labs/stinger/
9. http://hbase.apache.org/
10. http://www.splunk.com/en_us/products/splunk-enterprise.html
11. http://sparqlcity.com/documentation/
12. http://en.wikipedia.org/wiki/Sessionization
13. Liu, B., Wu, L., Dong, Q., Zhou, Y.: Large-scale heterogeneous program retrieval through frequent pattern discovery and feature correlation analysis. In: IEEE International Congress on Big Data (BigData Congress), 2014, pp. 780–781, 27 June–2 July 2014

14. Park, S., Lee, W., Moon, I.C.: Efficient extraction of domain specific sentiment lexicon with active learning. Pattern Recogn. Lett. **56**, 38–44 (2015). ISSN:0167-8655, http://dx.doi.org/10.1016/j.patrec.2015.01.004
15. Chui, C.K., Filbir, F., Mhaskar, H.N.: Representation of functions on big data: graphs and trees. Appl. Comput. Harmonic Anal. Available online 1 July 2014, ISSN:1063-5203, http://dx.doi.org/10.1016/j.acha.2014.06.006
16. Nisar, M.U., Fard, A., Miller, J.A.: Techniques for graph analytics on big data. In: BigData Congress, pp. 255–262 (2013)
17. Ediger, D., Appling, S., Briscoe, E., McColl, R., Poovey, J.: Real-time streaming intelligence: integrating graph and NLP analytics. In: High Performance Extreme Computing Conference (HPEC), IEEE, pp. 1, 6, 9–11, Sept. 2014
18. Atasu, K.: Resource-efficient regular expression matching architecture for text analytics. In: IEEE 25th International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 1, 8, 18–20, June 2014
19. Denecke, K., Kowalkiewicz, M.: A service-oriented architecture for text analytics enabled business applications. In: European Conference on Web Services (ECOWS, 2010), pp. 205–212. doi:10.1109/ECOWS.2010.27

Big Data Processing Algorithms

VenkataSwamy Martha

Abstract Information has been growing large enough to realize the need to extend traditional algorithms to scale. Since the data cannot fit in memory and is distributed across machines, the algorithms should also comply with the distributed storage. This chapter introduces some of the algorithms to work on such distributed storage and to scale with massive data. The algorithms, called Big Data Processing Algorithms, comprise random walks, distributed hash tables, streaming, bulk synchronous processing (BSP), and MapReduce paradigms. Each of these algorithms is unique in its approach and fits certain problems. The goal of the algorithms is to reduce network communications in the distributed network, minimize the data movements, bring down synchronous delays, and optimize computational resources. Data to be processed where it resides, peer-to-peer-based network communications, computational and aggregation components for synchronization are some of the techniques being used in these algorithms to achieve the goals. MapReduce has been adopted in Big Data problems widely. This chapter demonstrates how MapReduce enables analytics to process massive data with ease. This chapter also provides example applications and codebase for readers to start hands-on with the algorithms.

Keywords Distributed algorithms · Big data algorithms · MapReduce paradigm · Mapper · Reducer · Apache Hadoop · Job tracker · Task tracker · Name node · DataNode · YARN · InputReader · OutputWriter · Multi-outputs · Hadoop example

1 Introduction

Information has been growing at a faster rate, and computing industry has been finding ways to store such massive data. Managing and processing the data from such distributed storage is complex than just archiving for future use. Because the

V. Martha (✉)
@WalmartLabs, Sunnyvale, CA, USA
e-mail: vmartha@walmartlabs.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_3

data carries very valuable knowledge embedded in it and needs to process the data to mine the insights.

1.1 An Example

Here is one example to demonstrate the importance of efficient algorithms to process Big Data. Consider a firm with, say 1000, geo-separated office locations with a million employees altogether and each location collects the number of hours each employee worked everyday at the location. Accounting department of the firm calculates salary of each employee at the end of each pay cycle. One can use a spreadsheet software suite such as Microsoft Excel to do the job here. The software has its limitations on number of rows it can have in a file. It is also possible that a relational database system serves the platform for these computations. It will be a group by operation on 1,000,000 records (1 million employees) and say each pay cycle is 30 days; therefore, 30 days of records adds up to 30 million records. Group by operation on such a large number of rows takes significant amount of time given a single machine database system. Then, the distributed system's platform comes out to rescue from the computation problem. Distributed algorithms are very essential to benefit salient features of distributed systems, in particular when data is big as in the example we discussed here. The benefits from distributed systems for Big Data accompany few challenges that need attention.

1.2 Challenges with Big Data

As mentioned earlier, information has been growing at rapid pace. Industry has been facing several challenges in benefiting from the vast information [1]. Some of the challenges are listed here.

- Since the information is of giant size, it is very important to identify useful information to infer knowledge from it. Finding right data in the collection of data is possible with domain expertise and business-related requirements.
- Big Data is typically archived as a backup, and industry has been struggling to manage the data properly. The data has to be stored in such a way that it can be processed with minimal effort.
- The data storage systems for Big Data did not attempt to connect the data points. Connecting the data points from several sources can help identify new avenues in the information.
- Big Data technology has not been catching with the evolving data. With the fast grown internet connectivity around the world, almost infinite number of data sources are available to generate information at large scale. There is an immediate need for scalable systems that can store and process the growing data.

Researchers have been attempting to address the above-mentioned challenges by advancing Big Data technology. There were times a single computing machine was used to store and process the data. EDVAC is one of the early computer models proposed by Von Neumann [2]. With the advancement in chip fusion technology, a single machine with multiple processors and multiple cores was introduced. Multi-core system is a machine with a set of processors or cores and with facilities to opt computations in parallel. On the other hand, researchers are also focused on connecting several independent machines to run computations in parallel called a distributed system. A distributed system is typically made from an interconnection network of more than one computer or node.

A single machine is proved to be very effective by the generalized bridging model proposed by Von Neumann. Similarly, parallel programming can be effective when a parallel machine is designed as abstract and handy as the Von Neumann sequential machine model. There are two types of parallel systems: One is multi-core system and other is distributed system. With the advancement in distributed systems and multi-core systems, debate started on whether to adopt multi-core system or distributed system.

1.3 Multi-core Versus Distributed Systems

Both multi-core and distributed systems are designed to run computations in parallel. Though their objective is same, there is a clear distinction between multi-core and distributed computing systems that makes them distinguished in their space. In brief, multi-core computing systems are tightly coupled to facilitate shared space to enable communications, whereas distributed systems are loosely coupled that interact over various channels such as MPI and sockets. A distributed system is assembled out of autonomous computers that communicate over network. On the other hand, a multi-core system is made of a set of processors that have direct access to some shared memory. Both multi-core and distributed systems have advantages and disadvantages which are discussed extensively in [3], and the summary of the discussion is tabulated in Table 1.

Given scalable solutions mandated by Big Data problems, industry is inching toward distributed systems for Big Data processing. Moreover, Big Data cannot fit in a memory to be shared among processes, thus to stamp out multi-core system for Big Data processing.

1.4 Distributed Algorithms

Distributed algorithms are developed to perform computation in distributed systems. The algorithms take benefits from multiprocessors in distributed systems and manage computation, communication, and synchronization. There have been

Table 1 Multi-core versus distributed systems [3]

Criteria	Multi-core system	Distributed system
Resources	Dedicated	Utilize idle resources
Resource management	By application	By the system
User per node ratio	Low	High
Processes per node	Few	Many
Node homogeneity	Homogeneous	Heterogeneous
Configuration	Static	Dynamic
Communication paradigm	Message passing	RPC
IPC performance	High	Low
Scalability	Less likely	Yes
Direct access to shared memory	Yes	Now

several attempts to optimize distributed algorithms in generic model, and the following discussion deals with the algorithms.

1.4.1 Random Walks

Let $G = (\vartheta, \varepsilon)$ be an undirected graph representing a distributed system, made of set of nodes ϑ that are connected by links ε. "A random walk is a stochastic process of constructing a course of vertices ϑ visited by a token begins from a vertex i and makes a stopover at other vertices based on the following transition rules: A token reaches a vertex "i" at a time "$t + 1$" from one of the neighbors of the vertex "i" being the token at time "t," and the hop is determined by specific constraints of the algorithm" [4]. Original random walk is designed to address wide range of problems in mobile and sensor networks. The paradigm later adopted for distributed algorithms. A token visits the computing boxes in distributed systems to trigger computations. The token visit is determined by random walk algorithm. A walk of certain length is performed by electing an arbitrary neighbor in every step. The optimization solutions of random walk algorithm attempt to complete the walk in significantly less number of steps. The cover time of the network measures the number of steps needed to all the nodes from a node in the network, in other words steps needed to form a spanning tree of the network. The random walk path is also used to sync the data between nodes on the results from each step.

1.4.2 Distributed Hash Tables

Distributed hash tables (DHTs), in general, are used to perform lookup service in a distributed system [5]. Given that data is distributed among a set of computing/storage machines and each machine is responsible for a slice of information associated with a key. The key to data association is determined by a common hash table. The hash table is pre-distributed for all the machines in the cluster so that

every machine knows where a certain information resides. Looking up a key gives you a node identification metadata that holds the data chunk. Adopting the DHT concept onto computing model, a distributed hash table can be leveraged to find a set of computation machines to perform a task in a distributed system. Computing optimization can be achieved by optimizing the hash table/function for uniform distribution of computing nodes among all possible keys.

1.4.3 Bulk Synchronous Parallel (BSP)

Bulk Synchronous Parallel (BSP) model runs an abstract computer, called BSP computer. The BSP computer is compiled of a set of processors connected by a communication network. Each processor is facilitated with a local memory needed for the processor. In BSP, a processor, means a computing device, could be several cores of CPUs, that is capable of executing a task [6]. BSP algorithm is a sequence of super steps, and each super step consists of an input phase, computing phase, and output phase. The computing phase processes the data sent from previous super step and generates appropriate data as output to be received by the processors in the next step. The processors in a super step run on data received from previous super step, and the processors are asynchronous within a super step. The processors are synchronized after every super step. The communication channels are used to synchronize the process. Direct communication is possible between each processor and every other processor, given absolute authority on distributing the data among processors in the super step. BSP does not support shared memory or broadcasting. BSP is denoted by

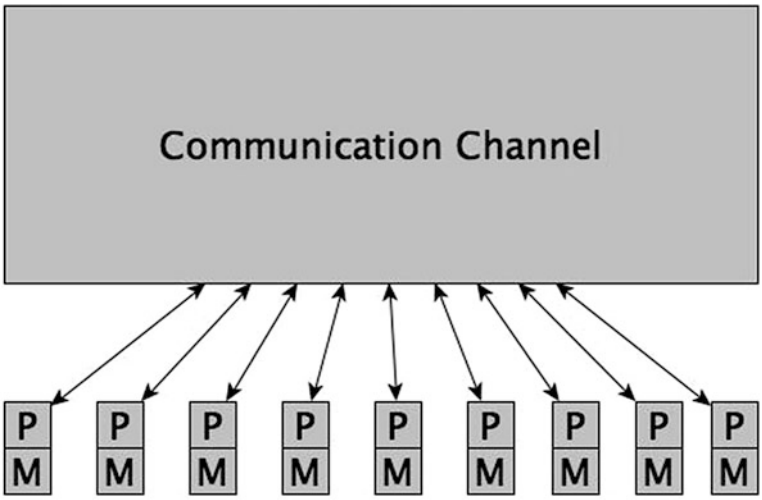

Fig. 1 BSP architecture

(p,g,l) where 'p' is the number of processors, 'g' is communication throughput ratio, and 'l' is the communication latency. The computation cost of BSP algorithm for S steps is $W + H.g + S.l.$ where W is local computation volume $= \sum_{S}^{s=1} w_s$, H is communication cost $= \sum_{S}^{s=1} h_s$, and S is synchronization cost. The BSP algorithm should be designed to optimize computation cost, communication cost, and synchronization cost [7]. The architecture diagram is presented in Fig. 1.

2 MapReduce

2.1 MapReduce Paradigm

Big Data problems typically bid definitive approaches and could sometimes follow non-conventional computing archetypes. All of the approaches have been discussed in the computer science literature for Big Data for decades which follow some kind of out of the box techniques, and MapReduce is certainly not the foremost to drive in this direction. The MapReduce is successful in fusing the several non-conventional computing models together to perform computations on a grand, unimaginable scale. The capability of its design made the MapReduce a synonym for Big Data.

MapReduce paradigm calcifies a distributed system to play authority in processing Big Data with ease. MapReduce, unlike traditional distribution systems, regulates computations on Big Data with least effort. MapReduce exposes specific functions for a developer to implement distributed application and hides internal hardware details. By this, developers can raise their productivity by focusing resources on application without worrying about organizing the tasks and synchronization of tasks.

MapReduce claimed humongous attention from research community as it was with Von Neumann's computing model. Von Neumann proposed 'a model as a bridging model, a conceptual bridge between the physical implementation of a machine and the software that is executed on that machine' [2] for single process machine, long ago. Von Neumann's model served one of the root pillars for computer science for over half a century. Likewise, MapReduce is a bridge to connect distributed system's platform and distributed applications through a design functional patterns in computation. The applications do not need to know the implementation of distributed system such as hardware, operating system, and resource controls.

The engineers at Google first coined the term Map and Reduce as an exclusive functions that are to be called in a specific order. The main idea of MapReduce comes from functional programming languages. There are two primary functions, Map and Reduce. A map function upon receiving a pair of data elements applies its designated instructions. The function Reduce, given a set of data elements, performs its programmed operations, typically aggregations. These two functions, performed

once on a data chunk of input data in a synchronous order coordinated by a synchronous phase called shuffling and sorting, form the basis of MapReduce [8].

There is no formal definition for the MapReduce model. Based on the Hadoop implementation, we can define it as a 'distributed merge-sort engine.' In MapReduce, there are two user-defined functions to process given data. The two functions are Map and Reduce. Given that data is turned into key/value pairs and each map gets a key/value pair. Data is processed in MapReduce as follows:

- Map: The map function takes given appointed key/value pair say (K1, V1) to process accordingly and returns a sequence of key/value pairs say (K2, V2).
- Partition: Among the output key/value pairs from all map functions, all the associated values corresponding to the same key, e.g., K2, are grouped to constitute a list of values. The key and the list of values are then passed to a reduce function.
- Reduce: The reduce function operates on all the values for a given key, e.g. (K2, {V2,…}), to return final result as a sequence of key/value pairs, e.g. (K3, V3).

The reduce functions do not start running until all the map functions have completed the processing of given data. A map or reduce function is independent of others and has no shared memory policy. The map and reduce functions are performed as tasks by Task trackers, also called slaves, and the tasks are controlled by job tracker, or master. The architectural view of MapReduce-based systems is shown in Fig. 2.

Designing an algorithm for MapReduce requires morphing a problem into a distributed sorting problem and fitting an algorithm into the user-defined functions described above. The algorithm should be divided into asynchronous independent smaller tasks.

The pseudo-snippet of MapReduce paradigm can be written as following:

```
Take Input File
Split the input file into List<InputSplit>
For each split S in <InputSplit>
      Convert the content of S to List<key,Value>
      For each Key, Value in List<key, value>
            Intermediate<Key',Value'> <- Map(Key, Value)
      Intermediate<Key', List<Value'>> <-
          Shuffle(Intermediate<Key',Value'>)
      For each Key', List<Value'> in Intermediate<Key',
          List<Value'>>
            Output<Key'',Value''> <- Reduce(Key', List<Value'>)
      OutputWriter(Output<Key'', Value''>)
```

Various implementations of MapReduce are available and Hadoop is one among them. Hadoop not only provides to write MapReduce algorithm but also provides supplementary features that help MapReduce algorithm benefit. The following discussion illustrates the Hadoop tools.

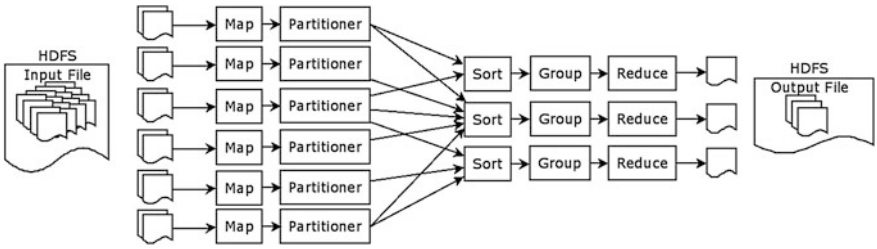

Fig. 2 MapReduce-based systems architecture

2.2 Introduction to Hadoop Tools

Though MapReduce was designed at Google, it has been evolved over time with open-source community. Apache developed several advanced features around the MapReduce technology and released to community to name 'Hadoop.' Sometimes, the words 'Hadoop' and 'MapReduce' are used interchangeably. To draw clear distinction between the two, Hadoop is one of the platforms to implement algorithms of MapReduce paradigm. Hadoop is a collection of several tools closely knit together to make it nearly complete venue to pursue MapReduce. We discuss some of the tools here that are required to develop MapReduce algorithms.

2.2.1 HDFS

MapReduce framework highly depends on the file system upon which the input and output data sits on. To enable various beneficial features, Hadoop constitutes a file system called Hadoop Distributed File System (HDFS) [9]. Similar to many distributed file systems out there, HDFS is also shaped by adopting most of the fundamental concepts from master/slave architecture. HDFS is hoisted from scratch to support very large data. The HDFS system, in succinct terms, constitutes a central server and a set of machines where the data is stored. The central server called Name node (NN) stores metadata while a set of machines are labeled the data nodes (DN) that administer the data comes in. Clients communicate with HDFS Name node to store files. HDFS Name node splits each file into one or more blocks, and then, the blocks are stored in data nodes. HDFS assumes that each file is a sequence of blocks where a block is a sequence of data elements; all blocks except the last one in a file are kept at a fixed size. The size of the blocks in a file is configurable and typically set at 64 MB. If content is less than the specified block size, the rest of the block is left vacant which happens frequently for last block of a file. If the content of the file is less than the block size, the space in lonely block for the file is left empty after filling the content driving the wastage of disk. For the reason, HDFS is unfit for small files, and the more smaller files, the faster they enervates the HDFS. The blocks of a file, are then, are replicated over one or more

Name nodes in the HDFS. The replication facilitates the HDFS to furnish fault tolerance amenity. Analogous to traditional file system, HDFS follows traditional file system in naming files in the system by adopting the tried-and-true directory hierarchy approach. File system operations such as create, rename, and remove files are possible in HDFS as in traditional file systems with an exception for edit operation. One of the most discussed limitations of HDFS from other file systems is that HDFS does not comply with file editing. HDFS is architected to enable the following features to name a few.

1. Big Data: HDFS is designed to support large files and to deal with applications that handle very large datasets. The large files are supposed to split into blocks. By this, HDFS is not recommended for files with small size in particular when one or more files are smaller than a block.
2. High availability (Replication): Applications that read data from HDFS need streaming access to their datasets. High throughput and availability are core features of HDFS. The features are achieved by data replication. The replication factor is configurable per file. An application can specify the number of replicas of a file. Each block of a file is replicated on the number of data nodes as the replication factor. The replication process led HDFS for slower writes but benefits with faster reads. When a file is opened for reading, for each block of the file, one of the data nodes that contain a replica of the block is constituted to serve the data read request. Since there is more than one data node available to serve the data read request, the reads are faster to achieve high availability. To minimize read latency, HDFS tries to satisfy a read request from a replica that is closest to the reader. Once the blocks are written and replicated across data nodes, the blocks cannot be modified leading the HDFS to be termed as write-once-read-many file system.
3. Robustness: HDFS can recover if there is data loss because of data node failure, or disk failure. When data on a data node is corrupted or not available, HDFS administrates the data nodes that replicate the data on the failed node to other nodes to satisfy the replication factor configuration. The re-replication is possible, and each data block is stored on more than one data node. HDFS listens to heartbeats and audits the data blocks for their availability periodically. HDFS constantly regulates all data blocks in the cluster to fulfill the system's configuration and triggers replication process whenever necessary.

HDFS is built on top of two primary components that are responsible in harvesting the above-discussed features. The following discussion presents specifics of the components.

- *Name node*. A machine that manages the HDFS is called 'Name node.' Name node does not store the data itself but administers where to store the data. The Name node executes file system namespace operations such as opening, closing, and renaming files and directories. It also determines the mapping of blocks to DataNodes. The responsibilities of Name node are given as follows:

  - It maintains the directory tree of the files in the file system.
  - It audits the physical location of the data.

- *Data Nodes* There are a set of DataNodes, one per node in the cluster, to store files. The Name node performs a split on a file into several blocks and directs the file system client to send the file contents in the given split sizes to data nodes. The data nodes store the file splits in blocks as per instructions from Name node. The DataNodes also perform block creation, deletion, and replication upon instruction from the Name node. When there is a request for read or write from the file system clients, the data nodes serve the request.

The data flow in a write operation is presented in Fig. 3. HDFS client who wants to write some data file onto HDFS contacts Name node for a list of data nodes that the client can connect to and write the contents of the file. The Name node updates its metadata of the request and responds with a block id and a data node details. The clients upload the content of the file to the data node, while data node copies the received content into the block specified by the Name node. Name node then finds another data node to comply with the replication factor. The Name node instructs the data node to copy the bock to other data node. The replication continues among the data nodes until the system satisfies the replication factor.

The similar and reverse approach is involved in reading the contents of a file from HDFS. The data flow is presented in Fig. 4. Client node, who wants to read a file, contacts Name node for a list of data nodes where the contents of the file reside. Name node responds with a list of data nodes, three data nodes when replication factor is three. Client node chooses one from the list received and contacts the data nodes to get served by the requested data.

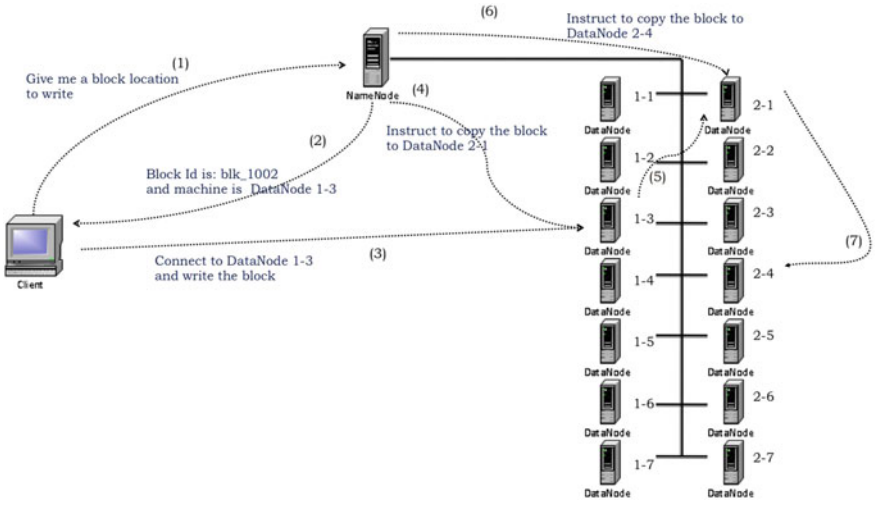

Fig. 3 Data flow diagram in a write operation in HDFS

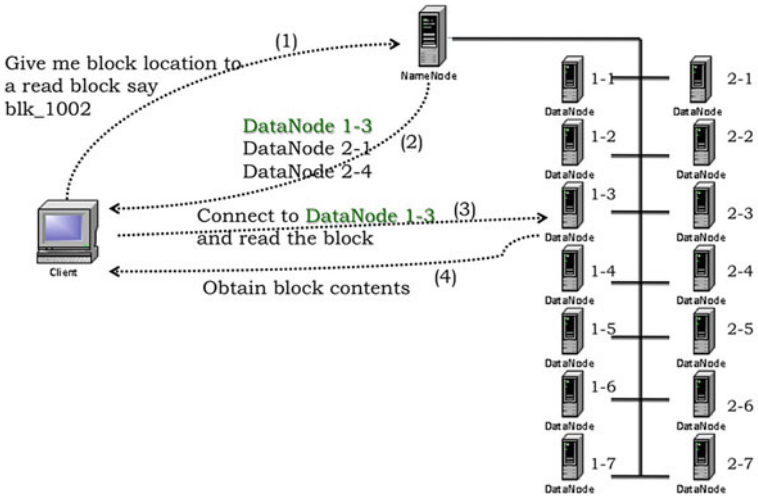

Fig. 4 Data flow diagram in a read operation in HDFS

HDFS is highly available and fault tolerant as client has an option to choose other data node when a data node is down while reading a file. When the client is one of the data nodes, it reads from local file system if the block resides in itself. Such scenarios commonly encounter when an application run in the data nodes. By this, the data does move to application, but application moves to data. Taking this advantage, MapReduce runs its tasks on data nodes. Hadoop implemented a set of tools to achieve MapReduce computing with the advantages taken from HDFS.

2.3 MapReduce Computing Platform

The Hadoop's MapReduce computing platform is depicted in Fig. 5. Hadoop's MapReduce computing platform constitutes two components: One is JobTracker

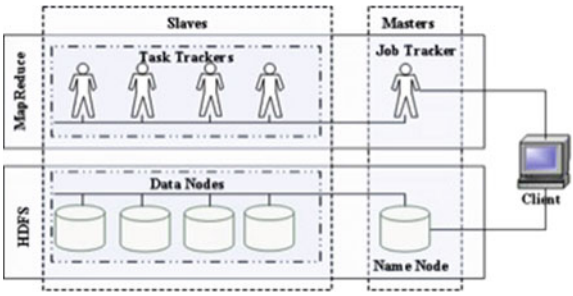

Fig. 5 Hadoop's MapReduce computing platform [15]

acts as a master in the Hadoop cluster while the other is called Task trackers. The Task trackers can be treated as workers.

A typical (simplified) activity flow in Hadoop is as follows [10].

1. A server, called Job Tracker, accepts jobs (MapReduce programs) from clients. A job is associated with a mapper method, a reducer method, and a set of input files and output files.
2. The Job Tracker contacts Name node to get the location of the input files. The Job Tracker applies appropriate scheduling algorithm to assign tasks to a set of computing nodes, called Task trackers. The scheduling algorithm takes data locality into account to optimize data movements.
3. The Job Tracker distributes the mapper and reducer methods among the scheduled Task trackers. In addition to the mapper and reducer methods, the job tracker also distributes the job configuration so that the mapper and reducer methods run based on the provided configuration.
4. The Task tracker performs the assigned mapper task. The Task tracker reads input from data nodes and applies given method on the input.
5. The map task creates and writes to an intermediate key-value pairs to a file on the local file system of the Task tracker.
6. Partitioner reads the results from the map task and finds appropriate Task tracker to run reduce task. The intermediate results emitted by map tasks are then propagated to reduce tasks.
7. The Task trackers that are scheduled to run reduce tasks apply operations programmed in reduce function on the data elements streamed from map tasks and emit a key-value pairs as output. The output data elements are then written to HDFS.

The control flow discussed here is evident enough to assert that the tasks come down to data location to perform designated operations. If there are no computing resources available at the machine where the data located, the computation is carried out in a nearest machine to the data. Such behavior is called data locality, in MapReduce context. Data locality improves the distributed computations by reducing the data movements in the network within the cluster.

2.3.1 Job Tracker

Job Tracker is a server that has the implementation for necessary user interfaces needed to submit and run a map reduce job. Once a map reduce job is submitted to Hadoop cluster, the JobTracker of the Hadoop cluster engineers a scheme to run the submitted job. The scheme involves identifying Task trackers in the cluster to perform map operations, triggering the mappers on Task trackers, monitoring the task while running, etc. Job Tracker runs in listening mode to take requests from clients. When a client submits a job, the job tracker communicates with Name node to obtain a list of machines that carry the input data for the job. The list is used in an

optimization algorithm, and the job tracker comes up with an optimized scheme to run the job on Task trackers. As mentioned earlier, the scheme attempts to reduce network bandwidth utilization within the cluster by adopting data locality feature. By data locality, the preference to run a task to run on a data chunk is (1) the machine where the data is located, (2) the rack where the data is located, and (3) a computing machine in the cluster. It could not always possible to find a node for a map task to run on local data; then, it is highly possible to find a node in the same rack. If there is no node available to take the task, then whatsoever machine in the cluster can perform the task, though not optimal but can let the job advance. One of the reasons a Task tracker not available to take a task is that the Task tracker could have been running the tasks up to its maximum capacity. Whenever the job tracker identifies a Task tracker to run a task, it monitors the task until it terminates. The job tracker finds another optimal Task tracker in case if a task failed on a Task tracker. By restarting a task on another Task tracker, the job tracker ensures that the job does not terminate if there is a task failed once, but attempts to run several times. At the same time, a job cannot be turned into successful states unless all the tasks of the job complete without errors.

The responsibilities of Job Tracker can be summarized as follows:

- Manage Task trackers and its resources as jobs being submitted.
- Schedule tasks among available resources.
- Monitor the tasks of jobs, and restart the failed tasks for configured number of attempts.

Job tracker is the heart of MapReduce computing platform as it is the entry and exit points for clients to submit a job. On the other hand, there is only one Job Tracker process runs on a Hadoop cluster. The JobTracker poses single point of failure for the Hadoop MapReduce service because there is no alternate when the only job tracker shuts down. When the job tracker falls down, all the jobs running on the cluster are deemed to be halted for client.

2.3.2 Task Trackers

Task tracker is a daemon runs on computing nodes of Hadoop cluster. The Task tracker receives instructions to perform map and/or reduce tasks from the Job Tracker. The Task tracker posts the available resources in the node and a heartbeat message at every specific interval of time to the job tracker. The job tracker performs bookkeeping of the Task tracker and corresponding resource and exploits the information in job scheduling. The Task tracker performs the assigned tasks (map, reduce) on given data. The job tracker tracks the Task trackers for the tasks that are running and instructs the Task trackers to where to send the output data.

A Task tracker performs a mapper method or a reduce method. The methods are illustrated in detail as follows.

- *Mapper* Mapper is a function that reads input files of given job in terms of <*Key, Value*> pairs and emits some other <*Key, Value*> pairs. The emitted key-value pairs are to be consumed by reducers and called intermediate key-value pairs. The skeleton of the mapper implementation is presented here.

```
public class MyMapper
   extends Mapper<Object, Object, Object, Object>{

   // Class members definitions goes here

   public void map(Object key, Object value, Context context)
       throws IOException, InterruptedException{
     // process key value pair
     context.write(OutputKey, OutputValue);
    }
}
```

A Mapper, say MyMapper, can be implemented by extending the Mapper interface which is in turn extends MapReduceBase class. A custom mapper is realized by overriding 'map' function in the MyMapper class. The map function takes input key and value as objects and a context that represents the context of the task. The context is then used to read task-specific variable and pass on counters to client.

- *Reducer* The key-value pairs emitted by all mappers are shuffled to accumulate all the intermediate records with a key will see same reducer. The reducer receives set of values for a given key and iterates over all the values for aggregation.

Here is the Reducer implementation, bare-bones code.

```
public class MyReducer
    extends Reducer<Object, Object, Object, Object>{

   // Class members definitions goes here

  public void reduce(Object key, Iterable<Object> values,
                Context context) throws IOException,
                    InterruptedException{
     // process values list for given key

     context.write(OutputKey, OutputValue);
  }
}
```

As for Mapper, a Reducer is developed by extending Reducer interface which in turn extends MapReduceBase class and a custom reduce function can be written by overriding reduce function in the class. Unlike from Mapper, Reducer's second argument is 'Iterable' which splits the values for the given key upon iteration. Context in reducer is leveraged as in mapper.

2.3.3 YARN

The recent version of Hadoop scrapped Job Tracker and Task trackers for a new job processing framework called YARN [11]. YARN stands for Yet Another Resource Negotiator. YARN constitutes Resource manager and Node Manager.

1. *Resource manager* Resource manager is the daemon which governs all jobs in the system. Resource manager regulates the resources in Task trackers. Resource manager schedules tasks based on the resources available on Task trackers. The Resource manager constitutes two primary tools: Scheduler and Applications Manager.

   - The Scheduler is solely designed to apportion the available resources on computing nodes of the Hadoop cluster. The allocation attempts to satisfy the capacity, queue, SLA, etc. There is no guarantee that the scheduler restarts the failed tasks but attempts to reschedule the task on the same node or other node for several attempts according to the scheduling configuration. A scheduler can employ a sophisticated algorithm to allocate resources among jobs submitted; by default, a resource container administers resources such as CPU, memory, disk, and network, which is the basis for scheduling algorithm. Recent version of YARN release supports memory as resource in scheduling. More such resource information is used more in optimal scheduling. YARN allows to use custom-designed schedulers in addition to the FIFO scheduler. Capacity Scheduler and Fair Scheduler are two such schedulers.
     The Capacity Scheduler is developed targeting a goal to share a large cluster among several organizations with minimum capacity guarantee. It is designed for supporting seamless computing flow in a shared cluster being utilized by more than one customer and to optimize resource utilization. Capacity Scheduler allows to maintain more than one queue, and each queue follows its configured scheduling scheme. Each queue complies with the configuration provided for this queue. Clients need to submit a job to the respective queue. The queue is processed in FIFO strategy. The jobs in a queue can be run on the machines that fall under the queue.
     Other scheduler is Fair Scheduler that the jobs equally receive their quota of resources to make progress in the job. There are pools, and each pool is allocated a portion of resources. When a job is submitted to a pool, all the

jobs in the pool share the resources allocated to the pool. In each pool, there is a fair distribution of resources among the jobs running on the cluster. A job never starves to terminate because it is FIFO and time-out waiting in getting resources. The scheduler optionally supports preemption of jobs in other pools in case to satisfy fair share policy.

- The Applications Manager stays open to receive requests to run jobs. When needed, Applications Manager negotiates with clients when the requested resources are not viable or resource allocations are changed to run the job. SLAs exchanged between Applications Manager and client, and when consensus is reached, the job is put on to execution pool. Scheduler takes the job to schedule it to run on the cluster.

2. *Node Manager* Node Manager does the similar job as Task trackers. Node Manager constitutes resource containers and performs operations such as monitoring resource usage and posting the resource status to the Resource manager/Scheduler in the cluster. Node Manager resides on a computing machine, coordinates the tasks within the machine, and informs Resource manager and Applications Manager on the status of the resources, tasks running on the machine.

Irrespective of YARN or Hadoop's early release, Hadoop's underlying design pattern, and client application development, APIs stay nearly unchanged.

There might be several mappers running as the map tasks get completed. As map tasks complete, the intermediate key-value pairs start reaching corresponding reducers to perform reduce operation. The nodes that run map tasks in a Hadoop cluster begin forwarding the intermediate key-value pairs to reducers based on intermediate key.

2.3.4 Partitioners and Combiners

Partitioner: Partitioning, as mentioned earlier, is the process of determining which reducer to work on which intermediate keys and values. There is a partitioner for each node. The partitioner of each node determines for all of mappers output (key, value) pairs running on the node which reducer will receive them. The intermediate key-value pairs of certain key are destined to one reducer despite the key-value pairs generated from different mappers. The partitioning algorithm should be in such a way that there should never be a need to move data from one node to another node. Default partitioning algorithm is hash algorithm. The key-value pairs of keys having same hash go to one reducer. There is also an option to define customized partitioner for a given job. The partitioner should decide which reducer a key-value pair should send to. The number of partitions the algorithm distributes the intermediate key-values pairs is equal to the number of reducers. The job configuration should specify the number of reducers, in other words the number of partitions.

```
public static class CustomPartitioner extends
    Partitioner<Text, Text> {

  @Override
  public int getPartition(Text key, Text value, int
      numReduceTasks) {
      // Find the right partition for the key
      return partition_number;

  }
}
```

Combiner: The objective of the combiner is to optimize the data movements among nodes. The combiner runs on each mapper and performs same task as reducer so that the data is reduced at the mapper node itself. The combiner step is optional in a job. Combiner receives key-value pairs from the mappers in a node and aggregates using the combiner method passed by the job configuration. With the interception of combiner in the pipeline, the output of the combiner is called intermediate key-value pairs and passed to partitioner for determining the appropriate reducer. To summarize, the combiner is a mini-reduce process which operates only on data generated by one machine.

```
public static class MyCombiner extends
      Reducer<Text, Text, Text, Text> {

  public void reduce(Text key, Iterable<Text> values,
      Context context)

        throws IOException, InterruptedException {
        // Aggregate the values for new key value pairs
        context.write(key, new Text(concatenatedData));
  }
}
```

2.3.5 Input Reader and Output Writers

We have been discussing how a job processes given data. Now let us discuss how a job reads the data and writes the output. Each job is associated with a set of input files and a set of output files. A method to define how to read the input file, to generate key-value pairs, is called input reader. Similarly, there is a method to define what format the output should be written, from key-value pairs emitted from reducers, which is called Output Writer.

Input Reader: Input of a MapReduce job is a complete file or directory. When a map is running the task, it receives only a chunk of data from the complete file or directory. A chunk called input split is passed to a mapper. The size of the input split, a mapper should work on, supposed to be defined by client with domain expertise. Default input split size is 64 MB or the same size of a block in HDFS. An input split could be the combination of one or more or partial files. Job scheduler also follows the same input split concept to achieve data locality feature.

Given that the input file of a job is split into several chunks called InputSplits. Each input split constitutes a sequence of bytes, and the bytes need interpretation. A toolkit to interpret the input split is to be developed for each map reduce. RecordReader serves the purpose that translates a given input split into a series of (key, value) pairs. The key-value pairs are in a consumable format when mappers read them. LineRecordReader is one of the record readers out there, and LineRecordReader assumes each line as a record and turns each line into a key-value pair by splitting the line by a delimiter. TextInputFormat is available by default in Hadoop package and can be used to read text files out of the box in a map reducer job. There is few other complex record reader that is capable of decompressing the input splits, deserializing objects, etc. Clients can develop a custom record reader to parse input file data following a specific format.

Output Writer: As the reduce task runs, they generate final output key-value pairs. Output Writer defines how the final key-value pairs should be written to a file. There is an Output Writer for a reducer task. The Output Writer takes the key-value pairs from the reducer and writes to a file on HDFS. The way the key-value pairs are written is governed by the OutputFormat. Therefore, there is a separate file and a common output directory for each reducer; a Hadoop job generates the number of output files as the number of reducers. The output files generated by reducers usually follow the patterns such as part-nnnnn, where nnnnn is the partition identifier associated with a certain reduce task. The common output directory is defined by the user in job configuration. The default output format is the TextOutputFormat which writes a line with key-value pairs in text format separated by tab for each key-value pair emitted by a reducer. In addition, there are two other output formats that are packaged in Hadoop implementation. The following table (Table 2) summarizes the

Table 2 Standard output format implementation

Output format	Description
TextOutputFormat	Given key-value pairs are written to output file in text format with a tab space as delimiter
SequenceFileOutputFormat	Given key-value pairs are written to output file in binary format. This format is useful for jobs reading in another map reduce job
NullOutputFormat	No output is written for this format

output formats. An user-defined Output Writer also allowed to specify job configuration to generate user-specific output format from final output key pairs.

2.4 Putting All Together

Summing up the artifacts learned so far in this chapter is sufficient enough to develop a Hadoop job and make it ready for submission on a Hadoop cluster. Let us go through the process of implementing a Hadoop job in this section to exercise the practices we learned.

Though there are many problems that can be solved by MapReduce paradigm, a Big Data problem yet simple is taken into consideration here to showcase the benefits of the MapReduce phenomenon. Assume that you are given a task to build a search engine on World Wide Web pages similar to Google or Bing. Disregard worrisome features of a search engine and the search engine returns you a list of pages and likely the places where the given keyword is occurred in the World Wide Web. A list of possible keywords in the World Wide Web alongside with corresponding pages and places the keyword occurred can enable the search possible without walking through all the World Wide Web every time there is a search request. Such list is called an index, widely accepted approach. Preparing an index is an easy task when the documents are handful and as well keywords. The real challenge arises when the same task to be done over a manifold of pages which is the case with World Wide Web. The number of documents to be indexed is at high scale and cannot be done on a single machine. MapReduce paradigm lands into overcome the challenges.

It is a typical routine to solve a bigger problem a piece by piece to come up with a viable solution. Endorsing the tradition, let us find a solution for indexing smaller dataset first followed by extending the solution to make it feasible on Big Data. Since MapReduce is partly a composition of map and reduce functions, it is easy to port the solution on smaller dataset onto MapReduce paradigm. The workout on the smaller dataset runs as in the following discussion.

Each page in World Wide Web is a sequence of text character disregarding the non-printing material of the pages. A page, interchangeably called document here, **D** is nothing but a set of text lines. Each text line is a set of keywords. Therefore, **D** = Set{Lines} where Line = Set{Keywords}. Here, each keyword is associated with a line. Now trace back the keyword position. A keyword position is a pair of document id and line number. For each line in the document, generate a pair of the keywords in the line and corresponding positions. The pairs can be aggregated in a method after reading all the contents in the document. The stitching of the above discussion can be written as an algorithm for a document D and is given as follows.

```
public void indexDocument(Document d){
     Map<Position, String> inputPairs = splitDocument(d);

     for( Entry<Position, String> onePair : inputPairs ){
        intermediatePairs.append(indexLine(onePair.Key(),
            onePair.Value()))
     }

     for( String oneKeyword : intermediatePairs.keySet() ) {
          aggregateIndexes( oneKeyword,
              intermediatePairs.get(oneKeyword) );
     }
}
public Map<Position, Text> splitDocument(Document d){
     String[] lines = split(d, newLineCharacter);
     Map<Position, String> inputPairs = new Map();
     for( int i=0; i<size(lines); i++ ){
          inputPairs.append(newPosition(d,i), lines[i])
      }
      return inputPairs
}
public MultipleMapping<String, Position> indexLine(Position
    pos, Line value) {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         emittingMap.append(word, pos);
      }
}
public List<Pair<String, String>> aggregateIndexes(word,
     Collection<Position> positionList) {
      emittingList.append(new Pair(word,
          StringUtils.join(positionList,',')));
}
```

As pointed out earlier, this algorithm works effectively on one document without encountering performance issues. When we carry forward the same application onto millions of documents, it fails to claim its outcomes as cogently as with one document. Harvesting the knowledge gained in this chapter, let us revamp the above algorithm into a MapReduce program. One can simply map the functions in the above algorithm into functions in MapReduce by knowing the spirit of MapReduce paradigm. Table 3 show cases the linear mapping to demonstrate how easy it is to develop an algorithm in MapReduce paradigm.

Now migrating the methods into MapReduce tools such as Map and Reduce methods, though obvious, is presented here. The source code for the MapReduce algorithm is illustrated in the following order: (1) Necessary classes for the MapReduce tools to function, (2) Mapper method, (3) Reducer method, and (4) InputReader implementations to understand the document content.

Table 3 Typical algorithm to MapReduce

Typical method	Method in MapReduce
splitDocument	InputReader
indexLine	Mapper
aggregateIndexes	Reducer

The mapper and reducer classes need to be supported by several supplementary classes, in particular Input Reader and Position. The Position class is defined to standardize on the offset/location of a text in a file. The position is a combination of 'File Name' and 'Offset in the File.' The position class has to be writable to be passed across mappers and reducers. The position class can be implemented as follows.

```
class Position implements Writable {
   String filename;
   Long offset;

   public Position() {
   }

   Position(String filename, long pos) {
      this.filename = filename;
      this.offset = pos;
   }

   @Override
   public void write(DataOutput d) throws IOException {
      d.writeChars(filename + "\\n");
      d.writeLong(offset);
   }

   @Override
   public void readFields(DataInput di) throws IOException {
      filename = di.readLine();
      offset = di.readLong();
   }

   @Override

   public String toString() {
      return filename + ":" + offset;
   }
}
```

A mapper takes a line in the given input with its corresponding offset to generate key-value pairs of tokens and the offset. The mapper class implementation is as follows.

```
public class DI_Mapper extends Mapper<Position, Text, Text,
    Position> {
   Text word = new Text();
   @Override
   public void map(Position key, Text value, Context context)
       throws IOException, InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         context.write(word, key);
      }
   }
}
```

A reducer method takes all offsets emitted by more than one mapper for a given token, and to split aggregate of offsets in a string format. All the offsets of a token are distributed by partitioner. The implementation of reducer class is as follows.

```
public class DI_Reducer extends Reducer<Text, Position, Text,
    Text> {

   @Override
   public void reduce(Text key, Iterable<Position> values,
       Context context) throws IOException, InterruptedException
       {
      String positionsString = new String();
      for (Position onePosition : values) {
         positionsString += onePosition.toString() + ",";
      }
      context.write(key, new Text(positionsString));
   }
}
```

The input to the MapReduce job is a set of text files, and the files should be read to transform the content to appropriate key-value pairs for mapper class. The input reader class does the service for the purpose. The record reader also called input reader takes the input files and generates key-value pairs that would be taken by mappers. The record reader implementation for the mapper here is presented in the following code.

```
public class DI_InputReader extends RecordReader<Position, Text>
    {

   private long start;
   private long pos;
   private long end;
   private LineReader in;
   private int maxLineLength;
   private String filename;
   private Position key;
   private Text value = new Text();

   @Override
   public void initialize(InputSplit genericSplit,
       TaskAttemptContext context) throws IOException,
       InterruptedException {
      FileSplit split = (FileSplit) genericSplit;
      filename = split.getPath().toString();

      Configuration job = context.getConfiguration();
      this.maxLineLength =
          job.getInt("mapred.linerecordreader.maxlength",
            Integer.MAX_VALUE);

      start = split.getStart();
      end = start + split.getLength();

      final Path file = split.getPath();
      FileSystem fs = file.getFileSystem(job);
      FSDataInputStream fileIn = fs.open(split.getPath());

      boolean skipFirstLine = false;
      if (start != 0) {
         skipFirstLine = true;
         --start;
         fileIn.seek(start);
      }

      in = new LineReader(fileIn, job);

      if (skipFirstLine) {
         Text dummy = new Text();
         start += in.readLine(dummy, 0,
               (int) Math.min(
                     (long) Integer.MAX_VALUE,
                     end - start));
      }
      this.pos = start;
   }

   @Override
   public boolean nextKeyValue() throws IOException,
       InterruptedException {
```

```
      // Filename is the key
      key = new Position(filename, pos);

      int newSize = 0;
      while (pos < end) {
         newSize = in.readLine(value, maxLineLength,
               Math.max((int) Math.min(
                           Integer.MAX_VALUE, end - pos),
                     maxLineLength));
         if (newSize == 0)
            break;
         pos += newSize;
         if (newSize < maxLineLength)
            break;
      }
      if (newSize == 0) {
         key = null;
         value = null;
         return false;
      } else {
         return true;
      }
   }

   @Override
   public Position getCurrentKey() throws IOException,
       InterruptedException {
      return key;
   }

   @Override
   public Text getCurrentValue() throws IOException,
       InterruptedException {
      return value;
   }

   @Override
   public float getProgress() throws IOException,
       InterruptedException {
      return start==end?0.0f:Math.min(1.0f, (pos - start) /
          (float) (end - start));
   }

   @Override
   public void close() throws IOException {
      if (in != null) {
         in.close();
      }
   }
}
```

Now, we have every tool needed to run a map reduce job. The map reduce job can be triggered from a method by importing the above-stated implementation. One such method to submit a map reduce job is called Driver. The driver method is named to submit map reduce with given map and reduce implementation to run on specified input files to generate designated output files. Hadoop provides the 'Tool' class to develop such driver classes. The following class implements the Tool class to submit a map reduce job with the given configuration. The main method here calls a method named 'run' with the command line arguments. The command line arguments are input directory that constitutes input files and output directory where the output files to be stored. The code for the driver class is as follows.

```
public class DocumentIndexing extends Configured implements Tool
    {

   public static void main(String[] args) throws Exception {
      int res = ToolRunner.run(new Configuration(), new
          DocumentIndexing(), args);
      System.exit(res);
   }

   @Override
   public int run(String[] args) throws Exception {

      // When implementing tool
      Configuration conf = this.getConf();

      Job job = new Job(conf);
      job.setJarByClass(DocumentIndexing.class);
      job.setJobName("DocumnetIndexing");

      // Assuming key and value of both mapper and reducer are
          text format
      job.setMapOutputKeyClass(Text.class);
      job.setMapOutputValueClass(Position.class);
      job.setOutputKeyClass(Text.class);
      job.setOutputValueClass(Text.class);

      job.setMapperClass(DI_Mapper.class);
      job.setReducerClass(DI_Reducer.class);

      // Input
      FileInputFormat.addInputPath(job, new Path(args[0]));
      job.setInputFormatClass(DI_InputFormat.class);
      // Output
      FileOutputFormat.setOutputPath(job, new Path(args[1]));
      job.setOutputFormatClass(TextOutputFormat.class);

      // Execute job and return status
      return job.waitForCompletion(true) ? 0 : 1;
   }
}
```

To run the above MapReduce application, one needs to compile the Java code and to compress into a jar package. The built jar can be used to submit the job with input and output paths as arguments. The command to submit a job is as follows.

```
Shell> hadoop jar DocumentIndexing-exe-jar-with-dependencies.jar
    documentindexing.DocumentIndexing -D mapred.reduce.tasks=5
    inputPath outputPath
```

The skills learned from this exercise are enough to model most of the other algorithms in MapReduce paradigm. In addition to the basic artifacts in MapReduce paradigm discussed so far, there are several other advanced concepts that make the MapReduce suitable for many domains. Some of the techniques are discussed in the following section.

3 Advanced MapReduce Techniques

3.1 Hadoop Streaming

Hadoop streaming allows one to create and run Map/Reduce jobs with any executable or script as a mapper and/or a reducer. Standard input and standard output (stdin, stdout) serve the channels to pass data among mappers and reducers. Let us discuss how it works with mapper, and then reducer will be in the similar approach. Mapper application receives input key-value pairs as input from standard input (stdin), and the mapper application has to parse the line came from stdin to extract key-value pairs. Upon completion of map instructions, the output key-value pairs can be emitted by writing to standard output (stdout).

A streaming application splits each line of given text file at the first tab character to recover key and value from the line. A mapper or reducer in streaming applications writes their output to stdout in the same format: key value.

As a mapper task in a streaming application runs, Hadoop converts the content of the given input files into lines and feeds the lines to the stdin of the given mapper process. The mapper program should split the input on the first tab character on each given feed of line to recover key-value pair. Mapper runs the programmed instructions on the given key-value pair and writes the output key-value pair to stdout separated by tab character.

The output of the mapper task is partitioned by a partitioner to feed to reducers. The inputs of a reducer are sorted so that while each line contains only a single (key, value) pair, all the values for the same key are adjacent to one another. Reducer program reads the given lines and recovers list of values for each key. Reducer generates output key-value pairs that are written to output file in HDFS.

Provided a Mapper and Reducer can handle given input in the text format, the mapper or reducer program can be written in any language as long as the nodes in the Hadoop cluster know how to interpret the language.

The following command is a typical way to run a Hadoop streaming job. The command triggers a map reduce job to use 'IdentityMapper' for mapper and shell script-based word count application as reducer.

```
hadoop jar $HADOOP_HOME/hadoop-streaming.jar
   -input myInputDirs
   -output myOutputDir
   -mapper org.apache.hadoop.mapred.lib.IdentityMapper
   -reducer /bin/wc
```

3.2 Distributed Cache

There are plenty of cases where every mapper needs some data in addition to the data to be processed say a configuration file or dictionary file. In the example MapReduce program we discussed, we extended the application where the tokenizer needs to read stop words list to ignore stop words from input text. The stop words file is to be available to every mapper that uses a tokenizer before the mapper start processing given input key-value pairs. Hadoop's distributed cache addresses the problem by providing tools to copy the given data to every node that runs mapper or reducer. The data is only copied once per job and the ability to cache archives which are unarchived on the slaves. The data in the form of files which could be text, archives, jars, etc. Hadoop assumes that the files are in HDFS. The data copying occurs at the time of job creation, and the framework makes the cached files available to the cluster nodes at their computational time.

3.3 Multiple Outputs

As discussed earlier, there are number of output files in a directory for a MapReduce job as the number of reducers in the job. Each reducer writes to a file. Hadoop allows a reducer in a job to generate more than one file and more than one output format. The job that needs to generate multiple outputs takes the output names as configuration. Reducer writes the output key-value pairs to appropriate output named file. Each output, or named output, may be configured with its own OutputFormat, with its own key class and with its own value class.

3.4 Iterative MapReduce

It is very likely that an application cannot be implemented as one single map reduce job. It is also distinctly possible that same map reduce job needs to be repeated

several times to generate expected output. Given the need for more such repetitive scenarios, there have been several implementations to support iterative calls to a map reduce job. iMapReduce is one of them and is discussed here.

Iterative iMapReduce runs mapper and reducer methods iteratively. The iterative calls to mapper and reducer methods improve performance by reducing the overhead of creating jobs repeatedly, eliminating the data movements, and allowing asynchronous execution of mappers [12]. Even though it is not true that every iterative algorithm can benefit from iMapReduce, many machine learning algorithms are quite suitable in iMapReduce.

Following section introduces some of the machine learning algorithms in MapReduce paradigm.

4 Machine Learning with MapReduce

The MapReduce architecture in Hadoop does not support iterative mapper and reducers directly, but there are several implementations that extended Hadoop to support iterative map reduce jobs. Machine learning algorithms take advantage of the iterative map reduce tool to train features on Big Data [13]. Clustering is one of classical machine learning algorithms and is discussed here to illustrate the sense of machine learning in Hadoop.

4.1 Clustering

A cluster is said to be a group of observations with similar interests. Clustering algorithm attempts to identify the groups in the given observations data. Two data points are grouped together if they have similar interests. The interests could be distance, concept, pattern, homogeneous property, etc. The clustering algorithms optimize clustering quality. There are no generic criteria to measure quality of clustering, but different problems follow different methods. K-means clustering is one of the clustering algorithms which optimize the clustering quality by minimizing the distance among observations within clusters.

4.1.1 K-Means Clustering

K-means clustering algorithm takes a set of vectors which we call training data. There are k-vectors to start with as k-cluster representatives. The algorithm distributes the vectors to k-clusters where the clusters are vectors too. The distribution tries to allocate each vector to its nearest cluster. The distribution process repeats iteratively; after every iteration, the new cluster representative is computed. The iteration stops when the cluster representative converges. There is no guarantee that it converges. The iterations stop if it does not converge for configured limit.

Since the same procedure is iteratively performed in the algorithm, now we can do extend the algorithm onto MapReduce. The MapReduce version of the algorithm also iterative so uses iMapReduce. In each iteration, a map reduce job allocates given observations to clusters from previous iteration and computes new cluster representatives. The mapper of the map reduce job distributes the observations into clusters, while the reducer computes new cluster representatives and emits the new cluster. The output clusters from reducer will be consumed by the mappers in the next iteration [14].

The pseudocode of the application is presented here.

```
public void map(Text Key, Text Value){
    previousClusterCenters =
        readPreviousIterationClusterCenters(); // From
        DistributedCache
    int minDistance = 10000; // some maximum value
    nearestCenter = previousClusterCenters[0];

    for (oneCenter : previousClusterCenters){
       distanceFromThisCenter = computeDistance( Value,
           oneCenter );
       if( minDistance > distanceFromThisCenter ){
            minDistance = distanceFromThisCenter;
            nearestCenter =oneCenter
       }
    }
    emit( nearestCenter, Value )
}

public void reduce( Text Key, Iterator<Text> Values ) {
       newCenter = computeCenter( Values );
       if( newCenter != Key )
              IncrementCounterBy1( NumberOfClustersChanged )
       emit ( newCenter, NullWritable );
}

public static void main(){
    iteration = 0
    uploadInitialCentersToHDFS();
    while true {
      ConfigureCentersFileToDistributedCache();
      submitJob()
      ObtainOutputFileAsNewCentersFile();
      currentChanges = getCounters(NumberOfClustersChanged);

      if( currentChanges ==0 || iteration > threshold)
          break;
      iteration++;
    }
}
```

5 Conclusion

MapReduce is one of the well-known distributed paradigms. MapReduce can handle massive amount of data by distributing the tasks of a given job among several machines. By adopting functional programming concept, MapReduce can run two functions, called map and reduce, on Big Data to emit the outcomes. The mappers process given input data independent of each other and sync the emitted results in reducer phase where the reducers process the data emitted by mappers. MapReduce reaps benefits from a suitable distributed file system, and HDFS is one of such file systems. HDFS serves a feature to move a map/reduce task to data location when performing a map reduce job in Hadoop. MapReduce blended with HDFS can tackle very large data with least effort in an efficient way. A problem can be ported to MapReduce paradigm by transforming the problem into asynchronous smaller tasks, map and reduce tasks. If synchronous is needed, it can be done once in a job through partitioning by a partitioner.

This chapter illustrated the primary components of Hadoop and its usage. An example scenario is also discussed to demonstrate how to write a 'HelloWorld' program in a MapReduce paradigm. An advanced MapReduce algorithm, k-means clustering, is presented to show the capability of MapReduce in various domains.

6 Review Questions and Exercises

1. Develop a MapReduce application to prepare a dictionary of keywords from a given text document collection.
2. Discuss the data flow in a MapReduce job with a neat diagram.
3. Develop a MapReduce application to find popular keywords in a given text document collection.
4. How HDFS recovers data when a node in the cluster goes down. Assume that the replication factor in the cluster is Review Question '3.'
5. Design a MapReduce job for matrix multiplication. The input is two file and each contains data for a matrix.

References

1. Tole, A.A.: Big data challenges. Database Syst. J. **4**(3), 31–40 (2013)
2. Von Neumann, J.: First draft of a report on the EDVAC. IEEE Ann. Hist. Comput. **15**(4), 27–75 (1993)
3. Riesen, R., Brightwell, R., Maccabe, A.B.: Differences between distributed and parallel systems. In: SAND98-2221, Unlimited Release, Printed October 1998. Available via http://www.cs.sandia.gov/rbbrigh/papers/distpar.pdf. (1998)
4. Israeli, A., Jalfon, M.: Token management schemes and random walks yield self-stabilizing mutual exclusion. In: Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing (PODC '90), pp. 119–131. ACM, New York (1990)

5. Gribble, S.D., et al.: Scalable, distributed data structures for internet service construction. In: Proceedings of the 4th Conference on Symposium on Operating System Design and Implementation, vol. 4. USENIX Association (2000)
6. Gerbessiotis, Alexandros V., Valiant, Leslie G.: Direct bulk-synchronous parallel algorithms. J. Parallel Distrib. Comput. **22**(2), 251–267 (1994)
7. Leslie, G.V.: A bridging model for parallel computation. Commun. ACM **33**(8), 103–111 (1990)
8. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
9. Borthakur, D.: The hadoop distributed file system: architecture and design. Hadoop Project Website (2007). Available via https://svn.eu.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf
10. Hadoop' MapReduce Tutorial, Last updated on 08/04/2013. Available at http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
11. Vavilapalli, V.K., et al.: Apache hadoop yarn: yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing. ACM (2013)
12. Zhang, Y., et al.: iMAPreduce: a distributed computing framework for iterative computation. J. Grid Comput. **10**(1), 47–68 (2012)
13. Chu, C., et al.: Map-reduce for machine learning on multicore. Adv. Neural Inf. Process. Syst. **19**, 281 (2007)
14. Zhao, W., Huifang, M., Qing, H.: Parallel k-means clustering based on mapreduce. Cloud Computing, pp. 674–679. Springer, Berlin (2009)
15. Martha, V.S.: GraphStore: a distributed graph storage system for big data networks. Dissertaion, University of Arkansas at Little Rock (2013). Available at http://gradworks.umi.com/35/87/3587625.html

Big Data Search and Mining

P. Radha Krishna

Abstract Most enterprises are generating data at an unprecedented way. On the other hand, traditional consumers are transforming into digital consumers due to high adoption of social media and networks by individuals. Since transactions on these sites are huge and increasing rapidly, social networks have become the new target for several business applications. Big Data mining deals with tapping large amount of data that is complex with a wide variety of data types and provides actionable insights at the right time. The search and mining applications over Big Data resulted in the development of a new kind of technologies, platforms, and frameworks. This chapter introduces the notion of search and data mining in the Big Data context and technologies supporting Big Data. We also present some data mining techniques that deal with scalability and heterogeneity of large data. We further discuss clustering social networks using topology discovery and also address the problem of evaluating and managing text-based sentiments from social network media. Further, this chapter accentuates some of the open source tools for Big Data mining.

Keywords Large data analysis · Scalable algorithms · MapReduce · Social graph analysis · Topology discovery · Sentiment mining · Shape-based local neighborhood generalization

1 Introduction

Data science is an emerging discipline in computer science that originated with Big Data revolution. Organizations are overloaded with unimagined scales of digital data, and analysis of such data may potentially yield extremely useful information. Recent trends reveal that organizations show much interest to leverage the extended enterprise data to uplift their business opportunities. Storing, searching, indexing, and mining data at a large (big!) scale with current technologies, databases, and

P. Radha Krishna (✉)
Infosys Labs, Infosys Limited, Hyderabad, India
e-mail: radhakrishna_p@infosys.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_4

architectures is a difficult task. This is mainly due to (i) lack of scalability of the underlying algorithms and (ii) complexity and variety of the data that needs to be processed. Thus, conventional data management techniques cannot be adopted for large-sized data generated by present-day technologies such as social media, mobile data, and sensor data. For instance, due to large size of the data (i.e., volume), it may not fit in the main memory. The complexity further increases for supporting Big Data, where responses are needed in real time. Due to increased adoption of digitization by individuals as wells as organizations, the data is being generated in the order of zettabytes, which is expected to grow around 40 % every year.

Search is often necessary in Big Data analysis to find a portion of data that meets specific criteria, which necessitates a new kind of index structures and associated search technologies. Big Data mining can be defined as *extraction of useful information/insights from very large datasets or streams of data* [8]. It deals with tapping large amount of data that is complex with a wide variety of data types and provides actionable insights at the right time. Big Data, due to its characteristics (such as Volume, variety, velocity, variability, and veracity), necessitates fine-tuning of existing or newly developed data mining technologies.

MapReduce [5], designed by Google, is a parallel programming paradigm that is useful to process Big Data by distributing tasks across various machines/nodes. It provides horizontal scaling to deal with large-scale workloads. The intuition behind MapReduce programming paradigm is that many large tasks can be represented in terms of Map and Reduce operations, which can facilitate parallelism over clusters of machines. The basic structure in MapReduce is key–value pairs (usually represented as <key, value>). Users need to define a single MapReduce job in terms of two tasks, namely *Map* and *Reduce*. The Map task takes key–value pair as an input and produces a set of intermediate key–value pairs. The outputs from the Map task (i.e., intermediate key–value pairs) are grouped by the key and then passed to the Reduce task. Reducer typically merges the values that have the same intermediate key and generates output key–value pairs.

The search and mining applications over Big Data resulted in the development of a new kind of technologies, platforms, and frameworks. This chapter is centered on the notion of search and data mining in the Big Data context and technologies supporting Big Data.

2 Big Data Search and Retrieval

Big Data search and analysis requires management of enormous amounts of data as quickly as possible. Some examples of Big Data search are as follows: (i) page ranking for searching the Web in order to retrieve more relevant Web pages for a given set of keywords, (ii) searching appropriate communities in a social network, which can be treated as a large-scale graph search problem, and (iii) searching the data for queries related to individuals/groups interest and their spending pattern from shopping malls transaction data pertaining to sales of large number of products. The data is usually

stored in distributed file systems, and parallel computing paradigms such as MapReduce and Giraph help in dealing with such large-scale data search and analysis.

Google search engine determines the order of Web pages to be displayed using page rank, which assigns a score to each Web page. Link structure of Web graph can be represented as a matrix and the page rank computation needs matrix-vector multiplication [10]. Fast retrieval of Web pages necessitates parallel implementation of matrix-vector multiplication and page rank algorithm. Below, we present MapReduce implementation of matrix-vector multiplication.

Suppose we want to multiply matrix A of size $n \times n$ with a vector V of size n. An element, say x_i, of this product can be computed as follows:

$$x_i = \sum_{j=1}^{n} a_{ij} * v_j \tag{1}$$

where a_{ij} is the element of ith row and jth column of matrix M and v_j is the jth element of vector V. Below, we describe the MapReduce implementation of matrix-vector multiplication. We assume that the complete vector V is read into the main memory and available to each Map task.

Note that, in Eq. 1, each output element x_i composes multiplication followed by summation. So, in the MapReduce implementation, we can define Mapper to perform multiplication and Reducer to perform summation to produce x_i.

```
//vector V is global
    class Mapper:
        method Map(<i, j>, value):
            emit(i, value * v[j]);

    class Reducer:
        method Reduce(i, values):
            emit(i, sum(values));
```

The input key–value pair for Mapper can be formed by considering the row and column indexes of the matrix as key and the matrix element form as a value. For each row, the Mapper multiplies each matrix element a_{ij} with vector element v_j and emits (or produces) key–value pairs as $<i, a_{ij} * v_j>$. Here, each row of the matrix can serve as a key for Mapper output as well as Reducer input. So, the input key–value pair for Reducer is the *row index* and the *list of all values assigned to corresponding row*. Usually, a Mapper task generates lots of intermediate data in terms of key–value pairs. The emit function saves that data on to the machine's local disk. Later, the data on the local machine will be made available to the appropriate Reducer.

The Reducer emits the key–value pair as $<i, x_i>$, where x_i represents sum of the values in the list that assigned to row i. That is, the Reducer gets all Mapper output values which correspond to one matrix row. A Combiner, similar to Reducer, can be developed to compute the subtotals of one Map task output. The Combiner helps in reducing the size of the intermediate results.

For large vectors which cannot fit into the main memory, one can vertically partition the matrix into equal-sized stripes, say P, and partition the vector into same number of stripes (i.e., P stripes) of equal size. The matrix-vector multiplication now becomes subproblems of multiplying ith portion of the matrix with the ith stripe of the vector. Note that if the entire vector itself can be placed in the main memory, then portioning of matrix and vector is not needed.

Hash and bitmap indexes are commonly used in most of the databases. However, MapReduce framework does not provide any special indexes. The user has to implement suitable indexing mechanisms in order to fetch the data quickly.

The indexing mechanism should provide a very small footprint of the indexes to make the search efficient for Big Data. The flexibility in partitioning these index structures ensures a smooth growth of the indexes and also their availability at the location of processing, thus allowing parallel processing of the operations on such large datasets. For instance, Web search engines such as Google and Yahoo adapted distributed indexing techniques using MapReduce programming paradigm [15]. Similarly, Dean and Ghemawat developed an indexing strategy based on MapReduce framework for document Indexing [5]. The indexing is very useful, especially when data is pushed to multiple Map tasks that are currently running. One of the biggest concerns in indexing problems is the need for an indexing mechanism that generates index at a very high rate both at the initial stage and also during additions, deletions, and updates.

One of the issues for efficient Big Data search is identifying and removing redundant data so that the search algorithms can be applied on useful data only. Most of the analytical and data mining techniques require to search the required data first and then apply appropriate data mining technique(s) to discover patterns.

3 K-Means Clustering

In this section, first we discuss the standard K-means clustering algorithm and then describe the MapReduce implementation of K-means clustering algorithm for Big Data clustering.

Clustering process groups a given set of n objects into different clusters. A good clustering technique satisfies the following two criteria:

- Intracluster distance is minimum (i.e., placing similar objects in one cluster)
- Intercluster distance is maximum (i.e., placing dissimilar objects in different clusters)

Similarity of the objects is computed based on the chosen distance/similarity metric.

K-means Clustering

The input to the K-means clustering algorithm is the number of desired clusters (say, K). The steps involved in the K-means algorithm are as follows:

1. Randomly select K objects from the given N objects as initial K cluster centers.
2. Compute the distance from each data object to each of the K centers and assign that object to the nearest center, thus forming new clusters.
3. Compute K centers for the new K clusters formed in step 2.
4. Repeat steps 2 and 3 till there is no change in the K cluster centers.

In K-means technique, computing distances between objects takes considerable amount of time. The number of distance computations required in each iteration is $N \times K$, where N is the total number of objects and K is the number of desired clusters. Note that the distance computation between an object and K centers does not interfere with the distance computation between another object and K centers. So, one can compute distances from K centers to different objects in parallel. However, the repeating procedure should be done serially as new K cluster centers are resulted in each iteration.

The sequential part of *K-means MapReduce* algorithm is as follows:

```
1. KCentres = Random(dataset, K) //Randomly select K objects from a given
                                    dataset as initial K centres
2. Repeat
3.       newKCentres = MapReduce(dataset, KCentres)
4.       If (KCentres == newKCentres)
5.                Break;
6.       KCentres = newKCentres
7. Return.
```

When compared to the size of the dataset, the number of cluster centroids (i.e., K) is small. So, for designing mapper function, we assume that the K cluster centers are placed in a single file and accessible to each Mapper task. The Mapper function identifies closest cluster center for an object and emits key–value pair `<cluster number, object>`. The Mapper function is given below.

```
Class Mapper:
    Method Map (key, object):
        mindist  = MaximumValue     // Initialize maximum possible value to mindist
        for (i  = 1; i <= K; i++ ) do
            dist = computedistance(KCentres[i], object)
            if (dist < mindist)
                    mindist = dist
                    ClosestClusterID = i
        emit (ClosestclusterID, object);
```

In the above algorithm, the *computedistance* function computes the distance between an object and a cluster center. At the end of for-loop statement, the cluster ID that is closest to the object is determined. Finally, the emit function produces *ClosestclusterID* (i.e., closest cluster ID) and the *object* as key–value pair and stores in the local memory.

The Reducer function updates centroid of each cluster to the new center. The input to the Reducer is the Cluster ID and the objects corresponding to that Cluster ID. The new centroid of the each newly formed cluster is computed using the objects assigned to that cluster. The output of the Reducer is `<Cluster ID, Cluster Centre>`. The new cluster centers are used for next iterations as shown in the sequential part of MapReduce algorithm. To compute the mean of all objects in a cluster, the number of objects in that cluster needs to be tracked. A Combiner can be constructed for each Map task to compute the partial sum of the values of objects assigned to a single cluster. The input key–value pair for the Combiner is `<Cluster ID, object>` and the output is `<cluster ID, (partial sum, number of objects)>`.

4 Social Networks—Community Detection

Social networks are gaining importance due to the involvement of different kinds of interactions and content sharing among individuals, organizations, and communities [20]. Virtual communities can be formed by a group of individuals who have common characteristics, behaviors, interests, etc.

Finding useful communities (also known as *community detection*) from a social network data according to a specific business purpose is a challenging task. The objective here is to find set of clusters from a social network. As stated in Sect. 3, clustering is the process of grouping a set of entities in such a way that entities in the same group (cluster) are more similar to one another than to the ones in other clusters. In the context of a social network, clustering indicates identifying groups, which share a lot of commonalities (relative to other members in the network) and are closely related. That is, social network clustering forms groups of nodes with dense intragroup connections and sparse intergroup ties. The quality of the clusters therefore highly depends on the methodology used to measure the similarity between nodes. Figure 1 shows a sample graph with three different clusters.

Community detection approaches are broadly categorized into four: *node centric*, *group centric*, *network centric*, and *hierarchy centric* [20]. In node-centric community detection technique, each node in a group satisfies certain properties. These properties include complete mutuality, reachability of members, nodal degrees, and relative frequency of within-outside ties. Finding cliques from graph falls under this technique. Clique percolation method (CPM) [9] allows overlap between the communities. Here, two k-cliques are adjacent if they share $k - 1$ nodes, and a community is equivalent to a percolation cluster of k-cliques, in which any k-clique can be reached from any other k-clique via sequences of k-clique adjacency.

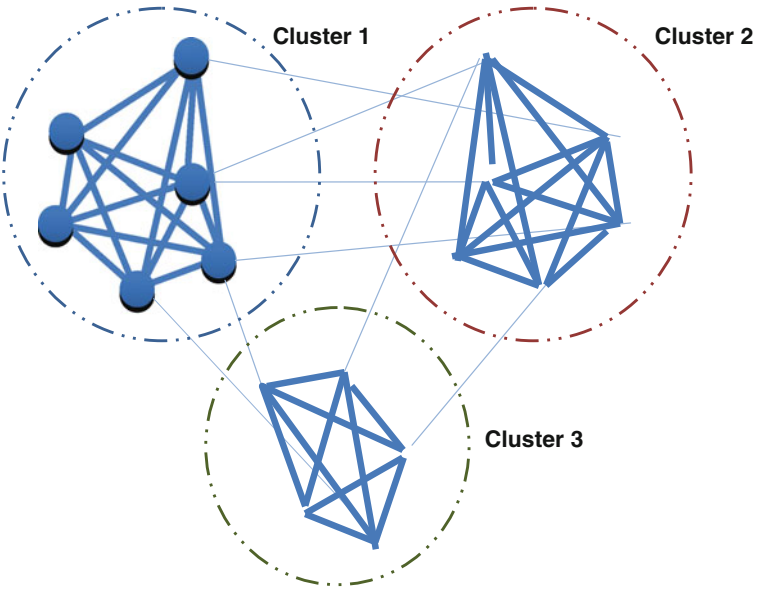

Fig. 1 Node clustering

Group-centric community detection techniques require whole group to satisfy certain properties. Network-centric community detection techniques partition the whole network into different disjoint sets. Clustering based on vertex similarity, latent space models, block model approximation, spectral clustering, and modularity maximization are the approaches of this technique [20].

Hierarchy-centric community detection technique constructs a hierarchical structure of communities. Girvan–Newman algorithm is one of the widely used approaches to find hierarchical communities from a given graph data [16]. Initially, the algorithm considers that all the nodes in a graph are single stand-alone communities. Then, it calculates the modularity between every node pairs and repeatedly merges pair of two communities with largest modularity which results in single community. A scalable community detection approach based on the Girvan–Newman algorithm using MapReduce framework is described in [6].

In the next section, we discuss clustering social networks using topology discovery (e.g., star, ring, and mesh topologies) [19]. The computed topology scores are used to determine the ***extent*** to which the *clusters grow* [4].

5 Social Network Clustering—Topology Discovery

Network structure as well as node attributes plays a major role in evaluating the communities. Discovering network topologies in social networks would provide tremendous benefits for business applications such as finding *key influencers* for

product campaigning and identifying *virtual communities* to recommend music downloads [4]. For example, *star* topology is useful to find key influencers who can motivate a large number of people; *ring* topology would help in determining critical nodes for data transfer; *mesh* topology would help in categorizing active communities/fully connected components which are candidates for promoting new offers [19].

5.1 Measures in Social Network Analysis

Social network measures such as *betweenness*, *closeness*, and *degree centrality*, *clustering coefficient* and *eccentricity* address various types of roles/connections that exist among vertices [19]. These measures in social network analysis give a rough indication of the ***social power*** of a *vertex* based on the pattern in which the vertices of a network are connected and extensively used directly/indirectly in developing most of the social network applications. Table 1 gives the mathematical formulations for these measures.

Betweenness

Betweenness centrality (b) indicates the number of vertices connecting indirectly to a vertex in the network. That is, it gives the importance of a vertex in retaining connectivity among distant vertices of the network. A vertex that occurs on many shortest paths when compared with rest of the vertices would get higher value for betweenness. Faster algorithms exist which find the betweenness centrality in feasible time [2].

Table 1 Formulas for measures in social network analysis [4]

Measure	Formula
Betweenness	$b(v) = \sum_{s \in V, t \in V, s \neq t} \frac{\sigma_{st(v)}}{\sigma_{st}}$
Closeness	$cl(s) = \frac{\sum_{t \in V \setminus s} d_{st}}{\lvert V \rvert - 1}$
Degree	$d(s) = \lvert N_s \rvert$
Clustering coefficient	$cc(s) = \frac{2.E_s}{\lvert N_s \rvert (\lvert N_s \rvert - 1)}$
Eccentricity	$ecc(s) = \max_{t \in V} d_{st}$

Notations $G = (V, E)$ is an undirected graph, where V denotes a set of vertices and E a set of edges. $e_{st} \in E$ if there exists an edge between vertices s and t, where $s, t \in V$

The neighborhood of a vertex s is denoted by N_s, where $N_s = \{t : e_{st} \in E\}$. E_s represents the number of connected pairs among all neighbors of vertex s

σ_{st} represents the number of shortest paths from vertex s to vertex t, where $\sigma_{st}(v)$ represents the number of shortest paths from s to t which pass through vertex v

d_{st} represents the shortest path distance from s to t

Closeness

Closeness centrality indicates the efficiency of each vertex (individual) in spreading information to all the other vertices which are reachable in the network. The closeness centrality (*cl*) of vertex *v* is the average (shortest path) distance from *v* to any other vertex in the graph. The smaller the *cl*, the shorter the average distance from *v* to other vertices. That is, the smaller the closeness value, the more closer the vertex is with all remaining vertices in the network and vice versa. Several algorithms exist which can compute quickly an approximation for the closeness centrality [3, 7].

Degree

The degree centrality (*d*) represents the number of edges a vertex has with its neighbors. It can be defined as the number of vertices $|N_i|$ in its neighborhood N_i.

Clustering coefficient

Clustering coefficient indicates how close the neighbors of a vertex are in forming a *clique* (complete graph). Higher clustering coefficient of a vertex indicates that all the neighbors are nearly connected by every possible edge between them. The clustering coefficient (*cc*) for a vertex *v* is given by the ratio of the total links between the vertices within its neighborhood to the maximum number of links that could possibly exist between them.

Eccentricity

Eccentricity (*e*) of a vertex *v* is the maximum shortest path length that is possible from *v* to all its reachable vertices in the network. Therefore, the lesser the eccentricity, the more the influencing power of a vertex.

The above five measures are considered in [19] to discover star, ring, and mesh topologies. Note that there also exist several other network centrality measures (e.g., radius centrality, eigenvalue centrality, and alpha centrality).

5.2 Finding Top Centrality Vertices of a Topology

Clustering algorithm uses top centrality vertices of a topology (i.e., the vertices with higher topology scores) in order to cluster the network [4]. As the current social networks have vertices in the order of hundreds of millions and also sparse in nature, identifying core members of the topology is the key challenge.

The topology scores indicate the membership of a particular vertex in that topology. These scores are used to identify the potential core members of the desired topology. Below, we define the formulas for scoring different topologies.

Star topology (S)

The star topology has the most influencing vertex at its center of the network (Fig. 2a). Typically, the center vertex possesses large degree. As vertices which connect diverse members, who themselves are not directly connected, are

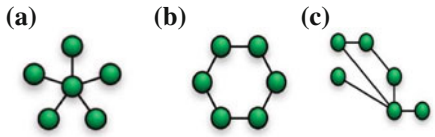

Fig. 2 Topologies. **a** Star-, **b** ring-, and **c** mesh-like structures

interesting, we assign less weight to vertices which connect members who are directly connected among themselves. The topology score of a vertex v can be defined as follows:

$$S_{score}(v) = b(v) \times d(v) \times cl(v) \times \left[1 - cc(v)^2\right]$$

Ring topology (R)

The ring network (see Fig. 2b) is useful when the structure of the network is in the cyclic form. In such a network, each vertex is connected to exactly two other vertices forming a single continuous pathway, i.e., ring. So, a failure in one of the core ring network vertices may result in the disruption of the information flow. Thus, it is critical in some of the applications to identify the core members of the ring topology to propagate information in a quick and fault-tolerant way. The core members of the ring topology are the vertices which have the maximum number of vertices from the same ring as its neighbors. The ring topology score for a vertex v can be defined as follows:

$$R_{score}(v) = ecc(v) \times cl(v) \times d(v)^2$$

Mesh topology (M)

The mesh topology is a lattice in the network graph (see Fig. 2c), where each vertex is connected to all other vertices in the network, thus forming a mesh structure. As the mesh topology represents a completely connected graph, the mesh topology score is directly proportional to clustering coefficient, degree, and closeness and inversely proportional to eccentricity. The mesh topology score of a vertex v can be defined as follows:

$$M_{score}(v) = \frac{cc(v) \times d(v)^2 \times cl(v)}{ecc(v)^2}$$

5.3 Clustering Algorithm to Find Network Topologies [4]

The clustering algorithm first finds the vertices that have the highest topology scores, and then, treating these points as cluster centers, form the clusters by

propagating outward and detecting vertices to be included in the associated cluster. The algorithm is composed of two main steps: edge weight determination and clustering.

Edge Weight Determination

In the edge weight determination step, the social network in consideration is converted into an undirected weighted graph, where weights of the links between vertices are calculated with respect to the degree of the interaction between those vertices.

An example of edge weight calculation for DBLP dataset (http://kdl.cs.umass.edu/data/dblp/dblp-info.html) is shown in Table 2. In this calculation, we have assumed that authors can be linked to each other either by coauthoring a paper or by giving reference to a paper of the other one.

Clustering

In the clustering part, first the topology scores for each node in the undirected weighted graph are calculated. These scores are used to determine the vertices at the center of the clusters as well as the boundaries of the clusters. Figure 3 shows the clustering algorithm. Steps 5 and 6 of the algorithm involve operations on the neighborhoods of the vertices. The neighborhood of a vertex v (i.e., $ne(v)$) is defined as set of vertices, whose distance to v is less than the threshold distance D_v.

Table 2 Formulas for edge weight calculation

Formulas
$w_{ij}^C = \sum_k \frac{\delta_i^k \cdot \delta_j^k}{n_k - 1}$
$w_{ij}^R = \begin{cases} 0 & \text{if } \sum_k r_{ij}^k = 0 \wedge \sum_l r_{ji}^l = 0 \\ \frac{[(\sum_k r_{ij}^k)+1][(\sum_l r_{ji}^l)+1]}{(\sum_k R_i^k)(\sum_l R_j^l)} & \text{otherwise} \end{cases}$
$w_{ij} = w_{ij}^C + w_{ij}^R$
Distance between nodes i and j is $d_{ij} = \frac{1}{w_{ij}}$

Notations Let a_i and a_j be two authors

$W_{ij} =$ *the weight of the link between* a_i *and* a_j

$w_{ij}^C =$ *the weight of the link between* a_i *and* a_j *due to coauthorship*

$w_{ij}^R =$ *the weight of the link between* a_i *and* a_j *due to references*

$\delta_i^k = \begin{cases} 1 & \text{if } a_i \text{ is an author of paper } k \\ 0 & \text{otherwise} \end{cases}$

$n_k =$ *number of authors of paper* k

$r_{ij}^k = \begin{cases} 1 & \text{if } a_i \text{ gives reference to } a_j \text{ in paper } k \\ 0 & \text{otherwise} \end{cases}$

$R_i^k =$ *number of references* a_i *uses in paper* k

Algorithm: Clustering
Input:
G: social network graph
T: number of top score nodes to get
D_c : Threshold distance of the furthermost neighbor to node *c*
d_{cv}: *distance between node c and node v*
ne(c): neighborhood of *c*
R_θ : minimum required number of overlapping elements to be in the same cluster
$R_\theta = \theta \bullet min(\,|ne(c)|,\ |ne(v)|\,)$ where $\theta \in [0, 1]$

1: Calculate topology scores for each node
2: Set N_T as the set of T nodes with highest topology score
3: **for** each $c \in N_T$ **do**
4: $new\ Cluster\ Cl = \emptyset$
5: **for** each $v \in ne(c)$ where $d_{cv} \leq D_c$ **do**
6: if $|ne(c) \cap ne(v)| \geq R_\theta$ then add v to Cl
7: **end for**
8: **end for**

Fig. 3 Clustering algorithm

Parallel implementations of determining centrality measures and clustering algorithm using Giraph to support big social graphs can be found in [11].

Experimental Evaluation

The above clustering approach using topology discovery is evaluated on a synthetic network. Star and mesh topologies are only considered for this study. The parameters used in this experimentation were selected as follows:

- D_c: selected as the average distance of node c to all other nodes
- $\theta \in \left[\theta^0 - 0.3, \theta^0 + 0.3\right]$ where θ^0 is the average clustering coefficient
- T: selected as the topology score at the 10th percentile

A sample sparse graph with 94 nodes and 167 edges is extracted from the synthetic network. The graph consists of 5 connected components. The results for star topology are summarized below, and the clusters obtained are shown in Fig. 4. The six clusters are marked with red color nodes and edges enclosed in dashed circles.

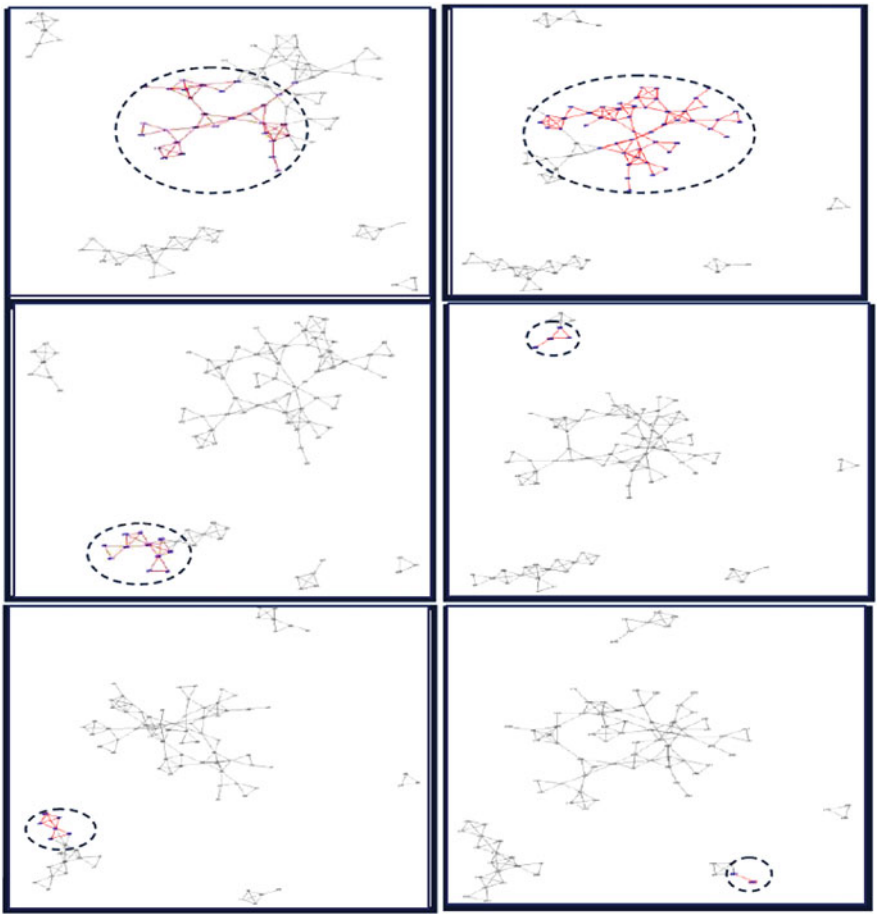

Fig. 4 Six clusters formed using star topology

Theta	Number of clusters	Number of nodes left	Number of overlap nodes	Number of total overlaps	Cluster sizes
0.803	6	10	19	19	(6, 30, 4, 11, 2, 50)

Similarly, the results for mesh topology are derived from the same sample network. The results are summarized as given below:

Theta	Number of clusters	Number of nodes left	Number of overlap nodes	Number of total overlaps	Cluster sizes
0.653	9	7	16	16	(17, 16, 10, 19, 4, 19, 4, 11, 4)

6 Social Network Condensation

Social networks are very large, and condensing them is useful to carry out analysis easier. Densely connected groups, such as meshes or cliques, might constitute good candidates for simplification of networks. The reason is that tightly connected groups typically have very homogeneous opinions and share similar traits, thoughts, preferences, and behaviors. Additionally, such groups possess excellent information diffusion properties, where new ideas are effectively shared between group members. This section provides an approach to merge a subset of mutually connected nodes in such a way that relationships along with their intensity are preserved.

Figure 5 shows a graph where the nodes encircled have to be merged and simplified to one virtual node representing the original cluster of vertices. The cluster merging algorithm is designed in such a way that it gradually selects two nodes and combines them to a new virtual node. Merging the nodes has to be done in such a way that associated connections are reflected accordingly, that is, the relational strength to their neighbors is captured correctly. Figure 6 illustrates an example of how the node merging process could look like. Nodes of the cluster are combined and connections updated until there is only one single representative node left.

In the following, we explain how the connections of a neighboring node have to be combined and updated, once two adjacent nodes are being merged. The two nodes X_1 and X_2 of Fig. 7 are related to one another with strength w_{12} and have to be merged to one single virtual node X_{12}. Node Y is adjacent to node X_1 and/or X_2, with w_1 and w_2 indicating the relational strength to the two nodes.

The weight of the resulting edge after the merging process can be symbolized by a function that combines the weights w_1, w_1, and w_{12} of involved edges. The approach demonstrated here considers the edge weights w_i, as probabilities, i.e., the likelihood that information is exchanged over the respective edge. By considering all paths from node Y to either node X_1 or X_2, the joint probability to reach from

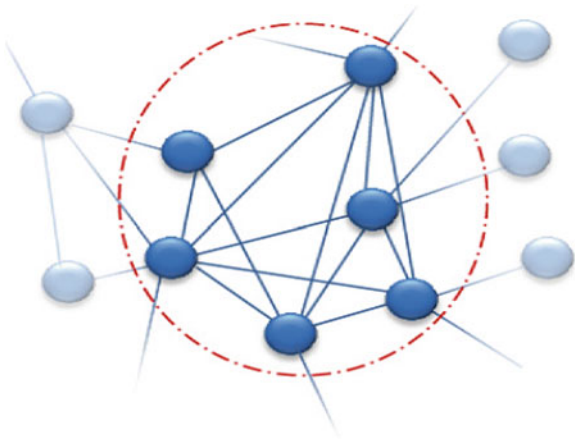

Fig. 5 Cluster merging

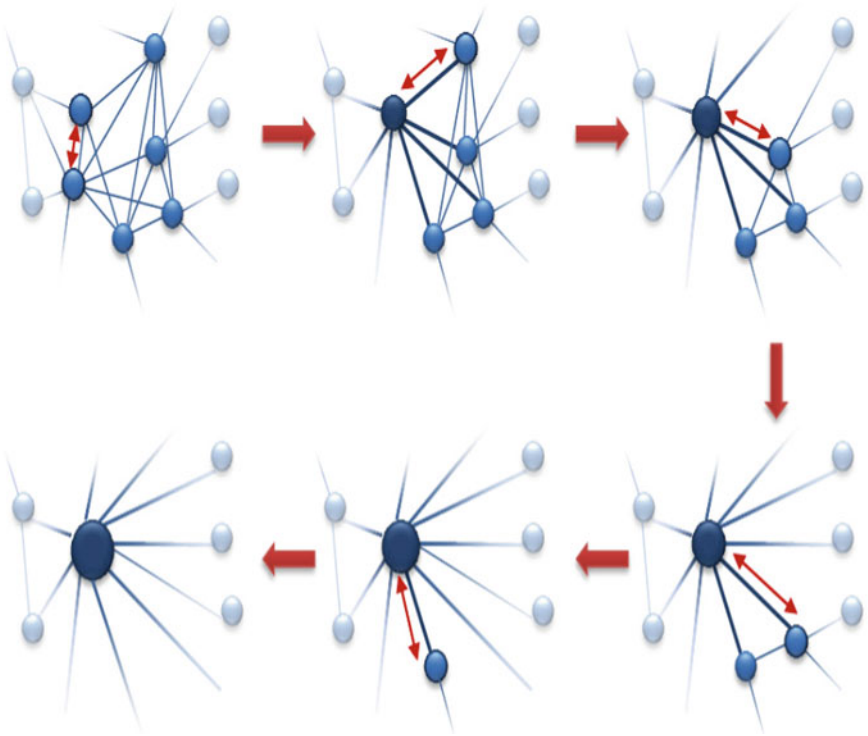

Fig. 6 Cluster merging process

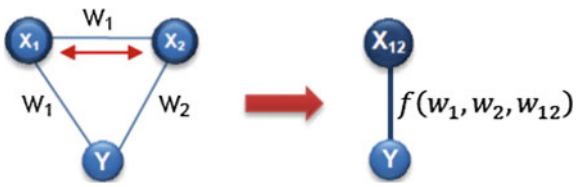

Fig. 7 Node merging process

node Y to the new virtual node X_{12} is calculated. There are four paths from node Y to either node X_1 or X_2:

$$\begin{aligned} P_{Y,X_1} &\Rightarrow Y \rightarrow X_1 \\ P_{Y,X_1,X_2} &\Rightarrow Y \rightarrow X_1 \rightarrow X_2 \\ P_{Y,X_2} &\Rightarrow Y \rightarrow X_2 \\ P_{Y,X_2,X_1} &\Rightarrow Y \rightarrow X_2 \rightarrow X_1 \end{aligned}$$

The likelihood that information is passed through a path is given as follows:

$$\begin{aligned} L_{Y,X_1} &= w_1 \\ L_{Y,X_1,X_2} &= w_1 w_{12} \\ L_{Y,X_2} &= w_2 \\ L_{Y,X_2,X_1} &= w_2 w_{12} \end{aligned}$$

The likelihood that information is passed from node Y to either node X_1 or X_2 via any paths is calculated as follows:

$$f(w_1, w_2, w_3) = 1 - [(1-w_1)(1 - w_1 w_{12})(1-w_2)(1 - w_2 w_{12})]$$

Hence, for each node that is directly related to node X_1 and/or X_2, there is a straightforward way of aggregating respective edge weights.

Note that in case neighboring node Y only has one connection to either node X_1 or X_2, the weight w_i of the missing edge can simply be set to 0.

7 Text Sentiment Mining

In this section, we discuss an approach for analyzing text sentiment using *local neighborhood generalization* for user feedback. We address the problem of evaluating and managing sentiments that are coming from social network media continuously.

Text sentiment is highly subjective to the end-user. A post on a social network media channel can convey different information to the end-user, depending on the relevant product, its brand, and potential target customers. This requires the sentiment evaluation system to be adaptive to the end-user. Building such a system is a challenging task where large corpus of posts are present (sometimes in millions and billions) in the database. The sequence of posts can be treated as *streams*. These streams need to be analyzed in a more real-time manner to understand the sentiments up to date. Therefore, a mechanism is required to update the sentiment evaluation system incrementally over time. We present an approach for incorporating user feedback in sentiment evaluation for a large number of social media posts. In this direction, a *shape-based local neighborhood generalization scheme* is presented, which is simple to implement and effective in reducing time to incorporate system update based on user feedback.

7.1 Sentiment Evaluation

Social media is a useful medium to capture customer sentiment on products from user reviews in various forums. Therefore, sentiment derived from social media

posts such as Twitter, Facebook, and blogs can be utilized for marketing new products. Sentiment evaluation system is an important component in several solutions such a brand management.

Sentiment evaluation from text in social network media has gained significant interest recently. There are several methods for evaluating sentiment from text data, which can be broadly classified into (a) rule-based methods and (b) supervised learning methods. Rule-based methods apply a set of rules on regular expressions and parts of speech (POS) derived from text to evaluate sentiment. The supervised learning methods use sample posts annotated with associated sentiment for training a single or ensemble of classifiers. A new post is then classified based on its characteristic sentiment. We have considered posts with two kinds of sentiments, *positive* and *negative* (indicated by **+** and **?**, respectively). A post can also be *devoid* of any sentiment value (indicated by ***NO***).

Sentiment evaluation is highly subjective to the end-user of the system and therefore can impact the marketing strategy implemented by an enterprise. A mechanism is required to update the system over a period of time subjective to the end-user. User feedback on the sentiment evaluated for a post can be used for retraining the system. Retraining the entire sentiment evaluation system can be challenging when there are a large number of posts available in the database. This may be impractical in terms of system responsiveness, as a result adversely affecting its usefulness. Complexity of retraining also increases when the number of user feedbacks received is not sufficient for retraining the entire system.

Below, we present a simple to implement, yet effective approach, for incrementally updating sentiment evaluation based on user feedback (Fig. 8). This scheme triggers a system update in real time on every occasion a user feedback on a post is received. An interface is available to the user to accept or reject the sentiment against each post. On rejection, a post with negative sentiment is considered to exhibit positive sentiment and vice versa. A ***shape-based local neighborhood generalization scheme*** is used to handle the growing number of posts over time. The end-user is provided better control on the way sentiment is evaluated by sentiment evaluation system in real time. We also illustrate a workflow to extract

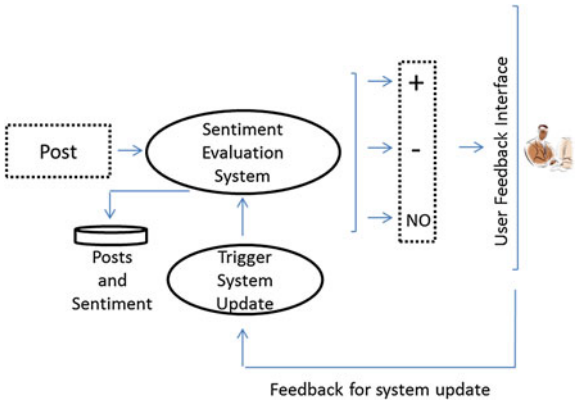

Fig. 8 User feedback-driven sentiment evaluation system

sentiment data from Web posts and use it for deriving necessary trends and inferences relating to the commercial offering such as brand management.

Incorporating user feedback in sentiment classification and retrieval problems has been addressed by methods focusing on *concept drift* and *active learning* schemes. These methods address the issue of subjectivity of sentiment evaluation depending on the end-user and change of sentiment associated with a post with respect to time, also known as concept drift.

Active Learning

Active learning is useful to build a corpus from a large set of posts. It provides a framework for strategically selecting posts in the feature space such that few labeled posts are required for training a classifier.

Active learning can be used to incorporate user feedback on selected posts for retaining the sentiment evaluation system. A post for which the sentiment classifier is most uncertain is first sent to the user for their feedback. Once required feedback is recorded, it can be used by the next classification model to reclassify all the available posts. A problem with such an approach is that it requires reclassification of all the posts before any meaningful analysis can be performed on the sentiment values. Reclassification of a large set of posts cannot be performed online as the classifier will not scale up. Also, selection of data points for active learning by the user requires a minimum set of samples depending on the dataset before any change in the classification model can be performed. Rather than recording change in the system based on each user feedback, the task of providing feedback is forced upon the end-user till necessary data points are accumulated.

Concept Drift

Sentiment associated with a post may change over a period of time. It can be due to a change in the sentiment of key terms in the topic of the post. Concept drift is particularly relevant in the classification of online news content where the perception for news can change over time. For example, in brand management, concept drift can occur with respect to changing perception of the brand manager for a given topic depending on the market. This can affect the relevance of metrics derived from posts if the change in user perception is not incorporated immediately in the system. Usually, methods for handling concept drift require relabeling of some or all of the previously classified posts or expect new data samples for retraining the sentiment evaluation system. Few systems even depend on alternate and less accessible information such as user bias to detect sentiment. Such methods typically expect the system update to wait for sufficient new annotated samples before a change can be affected in sentiment evaluation. Rather, a scheme is required that can handle online update in sentiment evaluation as and when user feedback is provided, so that useful metrics can be derived by brand manager in real time. Such an approach should be scalable and performed in real time and should incrementally update the sentiment evaluation system.

Below, we present an approach for incorporating user feedback in sentiment evaluation for a large number of social media posts. In addition to updating the

(*positive*, *negative*, *neutral*) sentiment for the query document, a local neighborhood generalization is also applied to neighboring posts.

7.2 Consumer Sentiment from Web

There is dramatic change in the way people express their opinions on the Web. Accessibility, ease of use, and to a certain extent anonymity have bolstered the growth of Web content related to people's views and opinions on popular products and services available to them. Reviews on Web pages, blogs, and tweets are few of the informal ways of sharing opinions where sentiment of the user regarding products and services is available. Identifying opinions and related information from natural language text is known as sentiment analysis. There are several challenges in natural language processing (NLP)-based sentiment classification including POS identification, handling negative statements and slang [14]. Given the informal nature of Web-based posts, the presence of highly informal words and abbreviations which are not available in the standard language is very common. Other conventional Web content such as emails, instant messaging, weblogs, and chat rooms also provide rich resource for monitoring the sentiment of the users online for devising quick marketing strategies.

There is abundance of techniques for sentiment classification from text using NLP-based techniques (e.g., [14]). A Web document may contain a mixture of *positive* and *negative* opinions about the subject. Based on this, there are several levels of sentiment analysis, which may be performed on the document. Analysis at *word level* determines the sentiment of a word or a phrase, whereas *sentence* or *document level* methods identify dominant sentiment of a sentence or documents. We focus on implementing these sentiment analysis methods on Web data, which come in many forms and are available mostly in unstructured formats.

Figure 9 shows the workflow for capturing, cleaning, and extracting sentiment information from Web documents. Data capture is performed using Web crawlers and specific application interfaces on channels such as blogs, market research reports, emails, Twitter, and social media. All posts are not equally relevant, and at

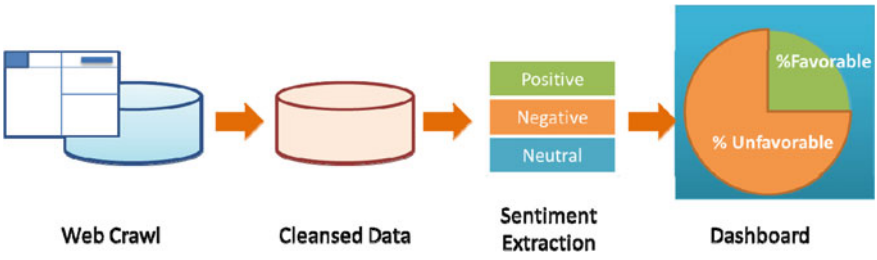

Fig. 9 Workflow for sentiment extraction and analysis from web data (such as review comments, blogs, and tweets)

the same time, some posts may not relate to specific product or service in question. Therefore, crawled data from the Web is cleaned based on the relevance to the subject in question and converted to structured/semistructured formats using various NLP techniques.

Sentiment analysis on the cleansed data is performed to categorize each instance of Web data as containing positive, negative, or neutral sentiment regarding the product or service. This sentiment-related information is used by the enterprise systems to derive useful inference and is employed by end-users for devising marketing strategies. Each instance of Web data is generally referred as a *post* in this work and is represented as a point in the feature space as illustrated in Fig. 10. The end-user (e.g., brand manager) of the system analyzes findings discovered from Web sentiment data in the form of dashboard illustrated as graphs, pie charts, and bar charts. Effective visualizations on dashboards allow drilling down the data and quickly generate data maps for analysis at various levels. Such analysis provides opportunities to correlate information from enterprise-structured data with respect to informal data sources such as Web posts and identify trends in sentiment. Such analysis presumes the effectiveness of sentiment extraction from Web posts. Given the challenges in performing sentiment extraction from complex and unstructured Web data, it is practical to assume that result of automatic sentiment classification will require user feedback from time to time to achieve realistic inferences. Local shape generalization for capturing feedback provides an effective means to record the user feedback quickly and at the same time ensures that its impact is not restricted locally (see Sect. 7.3).

We now look into local shape generalization approach for capturing user feedback on sentiment data in detail.

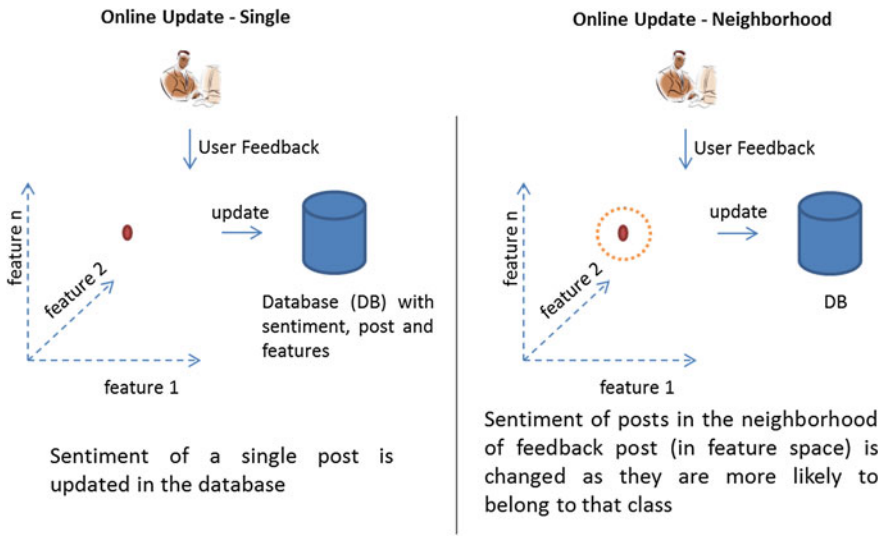

Fig. 10 Local update: updating post sentiment in feature space

7.3 Local Shape Generalization

Retraining the sentiment evaluation system can be an off-line or online action. Since update based on user feedback is an expensive task, off-line update is performed when the system is idle. In the online update, sentiment evaluation system is updated immediately after the user feedback is performed. This can be an expensive task as the number of posts in the system could be fairly large. Therefore, the logic for such an update should not affect the responsiveness of the system. We perform online update using a local neighborhood analysis scheme.

7.3.1 Local Update

The impact of feedback on the system on performing local update should be configurable by the end-user. Every post in the corpus of posts can be observed as a point in a multidimensional orthogonal numerical feature space (see Fig. 10).

Each dimension corresponds to a feature used for describing the post and is comprised of *word frequency*, *POS frequency, co-occurrence frequency*, *Web site impact metric*, *user impact metric*, *word length of post*, and *word distance*. A useful subset of these features can be used to describe each post in the feature space based on the business domain/end-user. Position of a post with negative/positive user feedback on sentiment is identified in this feature space (denoted as a feature vector).

One possibility is to update sentiment corresponding to the post with feedback in the database. A rule can be created based on this update for the range of feature values corresponding to the post (see Fig. 10, left). A new post with the similar numerical features will be classified with the correct sentiment based on user feedback. Instead of a single update, user can also select a neighborhood around the post with feedback in order to influence the sentiment of nearby posts. Posts in the neighborhood of a given post are highly likely to exhibit similar sentiment. Therefore, in the second method, a neighborhood operation is performed in the feature space around the post where feedback is received. Sentiments of the neighboring posts are updated based on this feedback (see Fig. 10, right). A rule is created to update post sentiment between the ranges of feature values based on the neighborhood selected. Any new post falling in this range of feature values will be assigned respective sentiment. Various *kernels* in feature space can be used to represent the neighborhood.

7.3.2 Kernels in Feature Space

The local update using neighborhood analysis is performed using one of the following three structures: *n-dimensional ellipsoid*, *n-dimensional spheroid*, and *n-dimensional rectangle*. Representative two-dimensional shapes are shown in Fig. 11. End-user can select any of these shapes to represent the neighborhood

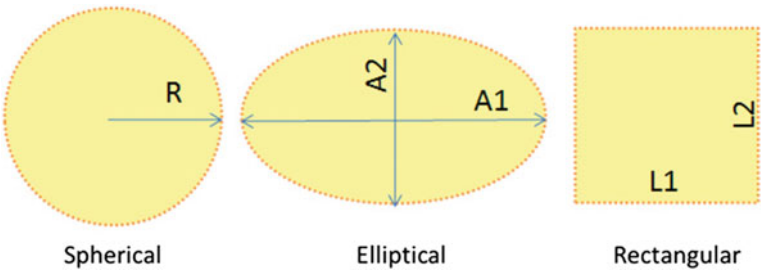

Fig. 11 Representative two-dimensional kernels

shape and size. Size of the neighborhood can be defined by the following equations. Here, f_i corresponds to ith feature values for the post with user feedback, n is the number of dimensions, and K_i is a user-defined threshold on ith dimension, $0 < K_i \leq 1$, and $1 \leq i \leq n$.

Radius of n-dimensional spheroid (R) is

$$R = \frac{1}{2n} \sum_{i=1}^{n} K_i(|\max(f_n) - \min(f_n)|)$$

Axis lengths for n-dimensional ellipsoid are

$$\begin{aligned} A_1 &= K_1|\max(f_1) - \min(f_1)| \\ A_2 &= K_2|\max(f_2) - \min(f_2)| \\ A_n &= K_n|\max(f_n) - \min(f_n)| \end{aligned}$$

Side lengths for n-dimensional rectangle are

$$\begin{aligned} L_1 &= K_1|\max(f_1) - \min(f_1)| \\ L_2 &= K_2|\max(f_2) - \min(f_2)| \\ &\ldots \\ L_n &= K_n|\max(f_n) - \min(f_n)| \end{aligned}$$

Here, size of the kernel is controlled based on the distribution of existing posts in the feature space. K_i is large when feedback has a high weight and vice versa. User can also set a limit on the maximum number of records that can be modified while specifying K_i.

Different dimensions in the feature space may be treated very differently for reacting to a particular user feedback. This can be achieved by optimizing the thresholds based on the importance of the features using the generalized shape kernels. Each threshold can be assigned the same value if each of the features is expected to contribute equally toward the evaluation of sentiment.

7.3.3 User Feedback on Neighborhood Updates

The end-user can also accept or reject the local neighborhood update performed after the initial feedback. A graphical user interface is provided for this feature where each updated post in the neighborhood and its new sentiment is displayed. Depending on the feedback on local update, the neighborhood size (kernel size) used for initial local update is modified. A representative rectangular kernel in feature space with two feedback points (posts) is shown in Fig. 12. Note that the local neighborhood is modified to accommodate the rejections by the end-user. The rule set on feature values is updated accordingly.

7.4 Experimental Results and Discussion

The effectiveness of local neighborhood updates in place of performing a global update is described in [4]. A representative n-dimensional dataset containing 1000 feature vectors is used with $n = 5$. Each feature vector in the dataset represents a post. Features generated randomly using a normal distribution where feature values range between [0–1]. Comparison of global and local updates is performed using a k-nearest neighbor (k-NN) sentiment classifier. k-NN classifier has been traditionally used in information retrieval and also used for several text sentiment classification works [17]. Training set for sentiment classification is derived from the dataset containing 20 % of records. All feature vectors in the training set are randomly assigned positive and negative sentiments as class labels. The remaining records in the dataset are treated as the test set and are updated using k-NN classifier with $k = 3$ for simplicity. Test set is presented to the user for recording the feedback.

Local updates are performed for each feedback received by the user on the test set. An n-dimensional spheroid is used in the experiments. The configurable neighborhood size, $R = 0.05$, is used such that only a small neighborhood is updated around each post where the user feedback is received.

The first experiment is performed to study the effect of user feedback on local and global updates. User feedback is recorded on a predetermined percentage of the test dataset. Posts with feedback are used for retraining the sentiment classifier (global update). Both local and global updates are performed based on recorded

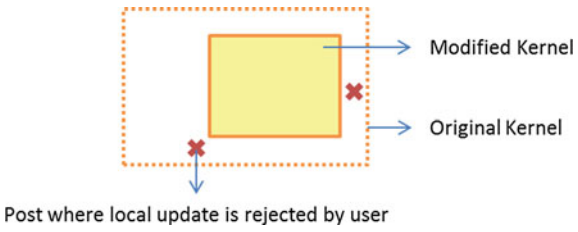

Fig. 12 Updating kernel parameters based on user rejections in local update

user feedback. Accuracy of local update is determined as the percentage of updated posts with matching sentiment value as global update.

Figure 13 shows the trend of accuracy with increasing percentage of annotated posts in 4 trials. Each trial is performed to represent a different user updating the sentiment for a percentage of posts. In trial-1 sentiments for 1–5 % of posts are changed. Successively, in 2nd, 3rd and 4th trials, sentiments for 1-10%, 1-15% and 1-20% of posts are changed. We can observe that as the percentage of feedback is increased, the results of local updates and global updates have better match for sentiment. This increasing trend of matching post sentiment is apparent till around 12 % of posts are annotated through user feedback.

Low matching accuracy is observed initially as only a small percentage of posts have update sentiment value as a result of local update. A maximum accuracy of 64 % is observed. A decreasing/stable trend in matching post sentiments is noted with increasing number of feedbacks. Therefore, it may be desirable to rely on global update for sentiment classification after at least 12 % posts are annotated by the user for the dataset under consideration.

The second experiment is conducted to observe the effect of increasing dimensionality of features on the performance of local update. Table 3 lists the maximum accuracy of sentiment match with a different feature length for posts. We can observe that feature length is not a significant factor in deciding the post sentiment when using the local update. Both the experiments used an n-dimensional spheroid for local update where equal weight is assigned to each feature dimension during update. It may be desirable to use other kernels if the user considers some features more useful than the others in deciding sentiment of posts.

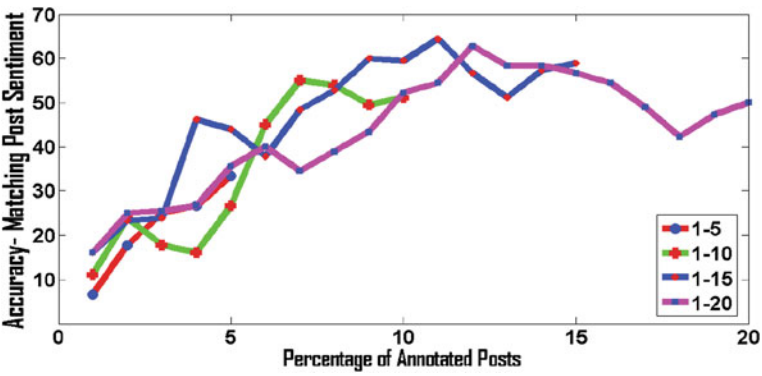

Fig. 13 Percentage of matching posts with increasing annotation (feedback)

Table 3 % of matching posts with increasing feature length

Feature length	10	20	30
Annotated (%)	12	12	12
Max. accuracy (%)	63	64	66

This presented feedback-based system for sentiment analysis can be used to analyze the user sentiments. Our approach enables the sentiment evaluation system to evolve over a period of time based on user feedback. It also enables sentiment evaluation to be customized for the end-user in real time without impacting the responsiveness of the system.

Unlike single or global system updates, local neighborhood generalization provides a scalable framework to incorporate feedback in real time with manageable effect on classification accuracy of sentiment. Given the size of Web data available in the form of posts, such a framework is of high significance. Local kernel shapes allow users to dynamically update significance of various features while providing feedback. The user, for various features, can follow both equal and biased approach of weightage with different kernels. While multidimensional kernels of spherical, rectangular, and elliptical shapes result in symmetrical update of the neighborhood, asymmetrical kernels can be explored for local neighborhood generalization.

8 Big Data Mining and Analysis Tools

Big Data mining is the process extracting useful information from complex and extremely large datasets of Big Data scale [8]. Traditional data mining tools are not scalable to support Big Data due to its characteristics such as *volume*, *variety*, *variability*, and *velocity*. Recently, there are several emerging data mining tools that support Big Data analytics. Below are few **open source** tools:

Mahout (http://mahout.apache.org)

Apache Mahout is a Hadoop-based scalable machine learning and data mining tool. It supports various data mining techniques including association rule mining, classification, and clustering and also supports recommendation techniques. Most of the Mahout implementations are enabled by MapReduce computing paradigm.

R (http://www.R-project.org)

R is a statistical computing and visualization tool mainly designed for large datasets for statistical analysis. ***R*** has number of libraries which can enable building a powerful Big Data mining solutions. ***R*** was initially developed at the University of Auckland, New Zealand, by Ross Ihaka and Robert Gentleman in 1993, and later a core group is formed for development and releasing new versions of R [18].

Massive Online Analysis (http://moa.cms.waikato.ac.nz/)

Massive Online Analysis (MOA) is an open source analytics framework for data stream mining. It is mainly designed based on concept drift and real-time analysis of Big Data streams. MOA supports scalable and distributed machine learning algorithms for a wide variety of techniques such as classification, outlier detection, clustering, frequent pattern mining, regression, change detection, and recommendation [1].

MOA is written Java and can be interconnected with WEKA (the Waikato Environment for Knowledge Analysis), which is popular open source data mining tool.

GIRAPH (https://giraph.apache.org/)

In MapReduce programming model for analyzing graphs, for each job, output is written to disk, and for new job, graph is again loaded into the memory. This loading of graph every time a job is started is an overhead for large graphs. To avoid such overheads, a new parallel computing platform namely GIRAPH. GIRAPH is an alternative to MapReduce for easy graph analysis.

GIRAPH is introduced by Apache, which suits very well for large graph applications. It provides more intuitive application programming interface to deal with graph problems.

GraphLab (https://dato.com/products/create/open_source.html)

GraphLab, another alternative to MapReduce implementations, is a high-performance distributed computing platform for processing large graph data. It provides parallel implementations of various machine learning algorithms and suitable for Map operations that characterize overlapping, iterative, and dependent computations. That is, GraphLab bridges between high-level MapReduce abstractions and their implementations into low-level parallel constructs.

GraphLab project developed in C++. It was initiated at Carnegie Mellon University by Carlos Guestrin in 2009 [13].

Pegasus (http://www.cs.cmu.edu/~pegasus/)

Pegasus is a peta-scale distributed data processing platform that supports big graph mining algorithms such as PageRank, Random Walk with Restart (RWR), diameter estimation, connected components, and eigensolver. Pegasus project is built at *Carnegie Mellon University*, and the software is developed in Java using MapReduce on top of Hadoop platform [12].

9 Summary

This chapter introduces the basic intuition of Big Data search and mining. "MapReduce" is a programming model for processing parallelizable problems across huge datasets using a large number of computers, collectively referred to as a cluster. We presented MapReduce implementations for matrix-vector multiplication and k-means clustering algorithm. Then, we studied community detection approaches and social network clustering based on various topologies such as *star*, *ring*, and *mesh* and also described a method for condensing social networks to support Big Data mining more effectively. We also discussed about handling

sentiment streams that arrive continuously by developing a shape-based local neighborhood generalization scheme. Finally, we provided few open source tools that support Big Data mining.

Exercises

1. Define (a) Big Data search and (b) Big Data mining
2. What type of intermediate data does MapReduce store? Where does MapReduce store them?
3. Write Mapper and Reducer functions for k-means clustering algorithm.
4. Give the algorithm to find the page ranking.
5. List ideas to improve/tune (existing) text processing and mining approaches to support big data scale.
6. (a) Explain the significance of social networks and their role in the context of Big Data.
   (b) List challenges of Big Data in supporting social network analytics and discuss approaches to handle them with justification.
7. How the centrality measures and structure of the social networks are useful in analyzing social networks.
8. What is community detection? Discuss various community detection approaches.
9. Explain social network clustering algorithm (using topology discovery) that allows overlap clusters.
10. Explain the concepts of active learning and concept drift. How these concepts are useful for big data search and mining.
11. What is sentiment mining? Describe an approach for extracting sentiments from a given text with examples.
12. Develop a suitable architecture for supporting real-time sentiment mining and discuss their components.
13. List open source tools and their characteristics to perform Big Data analytics.
14. Discuss various alternative mechanisms to MapReduce along with their merits

References

1. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. (JMLR) **11**, 1601–1604 (2010)
2. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Sociol. **25**(2), 163–177 (2001)
3. Chan, S.Y., Leung, I.X., Li.: Fast centrality approximation in modular networks. In: 1st ACM International Workshop on Complex Networks meet Information and Knowledge Management (CNIKM '09), ACM, pp. 31–38 (2009)

4. Celen, M., Satyabrata, P., Radha Krishna, P.: Clustering social networks to discover topologies. In: 17th International Conference on Management of Data (COMAD 2011), Bangalore, India (2011)
5. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: Sixth Symposium on Operating System Design and Implementation (OSDI'04), San Francisco, CA, pp. 137–150 (2004)
6. Dhaval, C.L., Somayajulu, D.V.L.N., Radha Krishna, P.: SE-CDA: a scalable and efficient community detection algorithm. In: 2014 IEEE International Conference on BigData (IEEE BigData14), Washington DC, 2014, pp. 877–882 (2014)
7. Eppstein, D., Wang, J.: Fast approximation of centrality. J. Graph Algorithms Appl. **8**(1), 39–45 (2004)
8. Fan, W., Bifet, A.: Mining Big data: current status, and forecast to the future. SIGKDD Explor. **14**(2), 1–5 (2012)
9. Imre, D., Palla, G., Vicsek, T.: Clique percolation in random networks. Phys. Rev. Lett. **94** (16), 160–202 (2005)
10. Ipsen, I.C.F., Rebecca, S. Wills: Mathematical Properties and Analysis of Google's PageRank. http://www4.ncsu.edu/∼ipsen/ps/cedya.pdf
11. Jyoti Rani, Y., Somayajulu, D.V.L.N., Radha Krishna, P.: A scalable algorithm for discovering topologies in social networks. In: IEEE ICDM workshop on business applications and social network analysis (BASNA 2014) (2014)
12. Kang, U., Tsourakakis, C.E.,Christos Faloutsos. PEGASUS: a peta-scale graph mining system —implementation and observations. In: IEEE International Conference on Data Mining (ICDM), Miami, Florida, USA (2009)
13. Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J.M.: Graphlab: a new parallel framework for machine learning. In: Conference on Uncertainty in Artificial Intelligence (UAI), Catalina Island, California, USA (2010)
14. Manuel, K., Kishore Varma Indukuri, Radha Krishna, P.: Analyzing internet slang for sentiment mining. In: Second Vaagdevi International Conference on Information Technology for Real World Problems (VCON), pp. 9–11 (2010)
15. McCreadie, R.M.C., Macdonald, C., Ounis, L.: Comparing Distributed Indexing: To MapReduce or Not?. LSDS-IR Workshop, Boston, USA (2009)
16. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**, 066133 (2004)
17. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
18. R Development Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2013) (ISBN 3-900051-07-0)
19. Radha Krishna, P., Indukuri, K.V., Syed, S.: A generic topology discovery approach for huge social networks. In: ACM COMPUTE 2012, 23–24 Jan 2012
20. Tang, L., Haun, L.: Chapter 16: Graph mining application to social network analysis. Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data, Springer, pp. 487–513

Security and Privacy of Big Data

Sithu D. Sudarsan, Raoul P. Jetley and Srini Ramaswamy

Abstract Big data, with its diversity (voice, video, structured and unstructured), has brought in unique challenges to security and privacy due to its shear scale. They are expected to be distributed, cloud-based and hosted by service providers. Security challenges in terms of cryptography, log/event analysis, intrusion detection/prevention, and access control have taken a new dimension. Privacy of online and cloud data is being addressed by governments, researchers, policy makers, as well as professional/standards bodies. The book chapter would cover challenges, possible technologies, initiatives by stakeholders and emerging trends with respect to Security and Privacy.

Keywords Security · Privacy · Big data · Cloud · Internet of things (IOT) · Anonymity · Critical information infrastructure (CII) · Confidentiality · Integrity · Availablity · Online privacy · Offline privacy

1 Introduction

Data security and privacy issues are no strangers to us. What is it that makes "big data" security and privacy different? And, why do we need to have a dedicated chapter on this? Just as a kingdom and an empire have different characteristics and need to be governed with different strategies, even though it is all about administering the land with its resources and inhabitants, big data differs in its characteristics from traditional data which warrants a detailed discussion.

S.D. Sudarsan (✉) · R.P. Jetley
ABB Corporate Research, Bangalore, India
e-mail: sudarsan.sd@in.abb.com

R.P. Jetley
e-mail: Raoul.Jetley@in.abb.com

S. Ramaswamy
US ABB, Cleveland, USA
e-mail: srini@ieee.org

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_5

Security mechanisms to safeguard data of a given type, say voice or email, have reached a level of maturity, and one can talk about accepted common practice or best practices. Here, in big data, we are dealing with possibly a diverse set of data ranging from voice, video, images, and structured as well as unstructured text. Even in the absence of scale, accepted security mechanisms for such diverse data do not exist today. Similar to the kingdoms in an empire, where each kingdom will have its own law, sometimes in conflict with another kingdom within the same empire, security, and privacy requirements of specific data within a big data need not be same but could be even at loggerheads.

Data storage mechanisms, till recently, were based on normalization, codification, extraction, and so on. The underlying criteria were based on high cost of storage and transport which were true to the technology of twentieth century. With time, practices based on these criteria have become the norm. In today's world, the cost of storage is close to zero and availability of connectivity is a foregone conclusion. We are now looking at data being stored as they are created, with sufficient redundancy to ensure availability in a distributed way. Obviously, the security mechanisms that were relevant to normalized single point storage do not suffice for distributed and redundant storage. A summary of key considerations of the past as against the emerging trends is provided in Table 1.

Cloud provides resources on demand and at least, in theory, expected to be fully elastic. Elasticity implies elimination of resource constraints in terms of platform, software, and infrastructure. In the absence of resource constraints, availability is improved through redundancy. While redundant resources increase the availability, they throw additional challenges to confidentiality and integrity. Even if confidentiality is managed, handling integrity with multiple copies is a challenge as all the copies need to be in sync. Cloud is also expected to involve third-party providers, perhaps

Table 1 Design considerations of past and future

Criterion	Waning (past)	Emerging (futuristic)
Storage cost	High. Minimize data by extraction, encoding, and normalizing	Close to zero. Retain as much source data as possible
Storage security	Uncommon. Mostly physical access control. Once accessed, entire file system is accessible. Granular secure storage only on need basis	Secure and safe storage is expected by default. Strong logical access control mechanisms
Communication cost	High	Connectivity is a given and cost is close to zero
Communication security	Optional. Security using crypto algorithms only when needed	Default, e.g., IPv6
Computation	Limitations due to single core and serial algorithms	Multi-core is default and advantages of parallel algorithms
Accessibility	From a single access point with optional backup access/redundancy	From multiple access points and as much redundancy as possible
Data synchronization/ integrity	As provided by the storage provider. Explicit mechanisms needed only if backup/redundancy storage exists	Challenge to be addressed due to redundant multiple copies

except for few fully owned private clouds. Security while dealing with third-party at a global level is another challenge. To achieve scale and redundancy, use of cloud as well as service providers have become imperative. Security and privacy issues applicable to cloud are thus very relevant and part and parcel of handling big data.

Internet of things (IoT) enables any device to be able to connect any other device using the internet. To highlight any device aspect, "internet of everything" is also used. At the same time, to specifically support industrial devices, "industrial IoT" is used. Irrespective of the universalization or restriction, IoT is a key ingredient of the upcoming industrial revolution "industry 4.0." This revolution promises seamless communication across any and every connected device. Each physical device has corresponding cyber device ensuring cyber-physical system in the true sense. IoT is a delight to research and business community at large with the need for new protocols, security solutions, applications, products, and systems. Many of the IoT proponents expect every device to be uniquely identifiable, which is fast tracking IPv6 adoption. The current transition is reflected in the increasing support for dual IP stack, which supports both IPv4 and IPv6. The presence of dual stack is a security challenge starting from the limitations of traditional firewalls to intrusion detection systems. If new communication protocols are added to the mix, security challenge compounds multifold.

Cloud and IoT along with other factors are catapulting the amount of data, and we have "big data" to deal with. Early database designers had to deal with the high cost of storage and communication. They went to the extent of not just storing only basic data but also encoded and normalized them to have the least amount of storage. This also ensured that only necessary data were transmitted and information was derived from this data as computation was considered relatively inexpensive. This also augured well for security practitioners since confidentiality was achieved typically with access control and cryptography, while integrity was achieved with error checking mechanisms. The key challenge was to ensure the availability as data were stored only at one location in a secure way. Later, backup and standby systems came up as the cost of storage and communication reduced over time. With ability to provide elastic storage with as much redundancy as needed, cloud has changed the very assumptions of security practitioners. IoT-enabled devices would generate and transmit so much data that security issues as well as managing the life cycle of those data are other dimensions that need to be addressed.

As devices become part of the cyber world with cloud and IoT, the criticality of cyber security takes another dimension. During cyber-attack, cyber-physical systems by their very nature create damages both in cyber as well as physical world. This is very important to understand, since in typical IT systems, cyber-attacks seldom cause physical damage. One can compare the effects of cyber-attack on a laptop to that of a surgical robot or an automobile resulting in their malfunction.

Yet another aspect of big data is about the nature of data itself. Often one hears about the quality of data, and efforts are made to get clean data. In fact, if the data are not clean or as per the quality requirements, they tend to be rejected. Redundant data are ignored. Any copy of the data is only for the purpose of backup and disaster recovery. However, with big data, one expects any data and every data to

be stored. Redundancy is expected to improve the availability. One may appreciate the challenge of keeping the integrity of data and synchronizing multiple copies.

Security has traditionally focused on confidentiality, integrity, and availability (CIA) in descending order of importance. Confidentiality has been handled using access control mechanisms and cryptography. Integrity is achieved using signatures, digests, and error detecting/correcting codes. Availability is managed with redundant communication links and standby storage systems. With big data and cyber-physical systems, the current practice of decreasing order of importance of CIA is unlikely to hold water.

To summarize, the emergence of cloud and IoT and the resultant big data brings in new security dimensions and paradigms. Challenges from cloud architecture to third-party reliance to making every device uniquely identifiable in the cyber-physical domain to big data security and analytics are all there. Exciting times ahead!

2 Security Versus Privacy

Security and privacy seem to be at loggerheads at all times and is a highly debated subject for quite sometime among researchers and lawmakers. One of the important reasons for this inconclusive debate is the subjective nature of the definitions attributed to these terms. Security aims to reduce risk. To reduce risk, specific information about a resource is often called for. As soon as the resource is related to entities such as individuals and corporates, privacy concerns creep in. One way this issue is typically addressed is by anonymizing data.

Yet with big data if at all anonymity can be provided has become a question mark. We do have standing examples of using anonymized data and yet identify

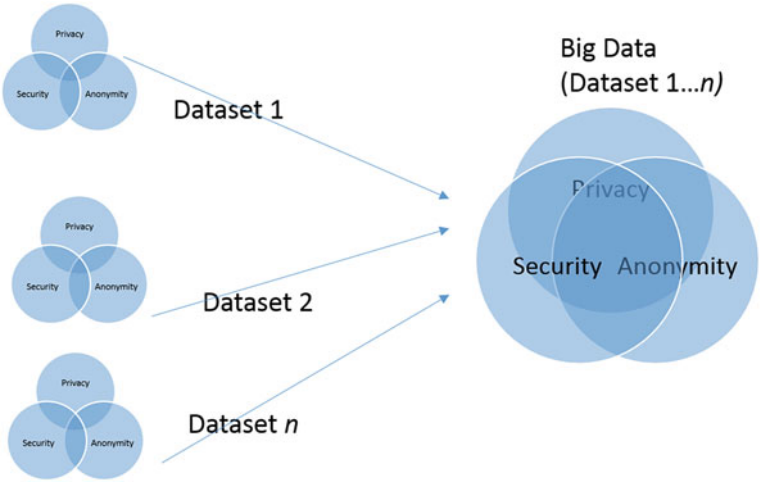

Fig. 1 Effect of big data on privacy and security

individuals with certainty. As early as in 2002, Latanya Sweeney was able to correlate data from the Group Insurance Commission (GIC) and voter registration list for the city of Cambridge to identify Massachusetts Governor Weld unambiguously [1]. In this case, the GIC data had personally identifiable information (PII) removed and yet it was possible to recover by correlating two different data sets made available for good reasons. With big data, we are going to have several such data sets that could be correlated and hence ensuring privacy, while enabling security is going to be one of the grand challenges to mankind. As we access more and more data sets, then privacy keeps losing its place, while anonymity also becomes questionable with security becoming the lone criteria as depicted in Fig. 1. As it stands today, unless new ways of preserving PII are found, we are staring at a world where there is no such thing as privacy.

3 Security

Security incidents happen when risk materializes. Risk is the chance of occurrence of hazard. Corollary is that only when all potential hazards and their chances of occurrence are known will it be possible to provide mitigation to avoid/handle the risk resulting in appropriate security. However, it is often not possible to identify all possible hazards. Secondly, even if certain hazards are known, mitigation may not be possible or feasible. Further, hazards themselves are not constant and set in stone. They keep changing with time and situation. This cannot be more true than in the connected internet world.

A system is considered secure when it satisfies the requirements of CIA. This is not, of course, the only measure. Another popular measure is based on the privacy, authentication, integrity, and non-repudiation also known as PAIN.

Availability and the resultant accessibility are to be taken seriously. Open Security Foundation[1] reports that while 58 % of data loss events involving PII is by outsiders, the rest 42 % is by insiders including 19 % accidentally. As much as 31 % of data loss is attributed to hackers over the last decade; however, in 2013, this went up to 47 %. While these data are based on reported incidents in USA, the rest of the world cannot turn a blind eye.

3.1 Big Data Security in IT

Information system security is commonly referred as IT security the scope of which includes several sectors such as business, finance, and banking. Confidentiality of data is of prime importance here. In case of a pharmaceutical industry, formula of a

[1] http://www.datalossdb.org.

specific drug could be a trade secret. In case of banking sector, customer data is to be protected and several laws enforce such a requirement. Espionage is one of the key threats to security, and elaborate studies have been made on this subject. System-level security policies specifically address this issue. With the advent of big data, as we increase availability, the number of ways in which espionage attempts could be made increases by orders of magnitude. Integrity of information is another aspect to be addressed. Several security mechanisms are in place including hash and signature. Yet another important characteristic in case of transactions is non-repudiation. Non-repudiation is a property that ensures that the owner or sender of the information cannot dispute it later. Even defense information systems have similar requirements in a stricter way. In this case, secrecy is supreme which keeps the adversary guessing and when needed provides the "surprise" element, so often decisive.

So much water has flown in case of IT security that not only we have the best practices, but also we have several standards and certifications such as Federal Information Processing Standards (popularly known as FIPS), ISO/IEC 27001 information security management system, ISO/IEC 15408 (popularly known as "Common Criteria"), Control Objectives for Information and related Technology (COBIT), and Information Technology Information Library (ITIL). However, as big data comes in, these standards need to be upgraded to handle the scale and complexity.

3.2 Big Data Security in Critical Infrastructure

Importance of critical infrastructure has gained traction over the years, and several countries have taken explicit steps to protect them. For our discussion, critical information infrastructure (CII) part of the critical infrastructure is particularly interesting, and big data has important ramifications as one looks to protect CII. Identification and definition of "critical infrastructure" has varied from country to country. Certain sectors such as banking and finance, energy/electricity, transportation, and health services are common across several countries. A handbook by ETH covering CII and its protection was published almost a decade ago [2]. Defense has always been a critical sector, and while some countries protect it discreetly, other countries make it part of critical infrastructure. When it comes to infrastructure, the biggest challenge is not confidentiality but availability. In fact, confidentiality may not be an issue at all. As a case in point, if we take a passenger reservation system, the most important requirement would be availability. Same thing could be said about the availability of a grid supplying electricity.

For long CII was protected by not connecting it to public internet. Where such a connection became essential, air gap and other mechanisms were put in place to address security requirements. Those days are now receding past and more and more CII is accessible from public internet. This has added to the data volume. Yet another important factor is the digitization of data collection. For example, manual metering of electricity usage is giving way to automatic/smart meters. The metering

Table 2 Comparison of IT and CII systems

Criterion	IT system	CII
Priority of security requirement	Confidentiality, integrity, and availability (CIA)	Availability, integrity, and confidentiality (AIC)
Application type	General purpose; can also be used for special purpose	Special purpose
Attack surface	Reasonably well understood; public	Not very well understood; not public
Security awareness	Openly discussed	Opaque

information, instead of being maintained and processed locally, is becoming part of a large data repository and processed in some form of centralized or clustered manner. This enables consolidation, trend analysis, and various types of report generation. At the same time, availability of such information also enables vested interests to play with, for example, pricing based on demand and create artificial shortage or surplus. Hence, securing the big data is a key aspect as we become more "smart" with smart home, smart plant, smart appliance, etc.

In contrast to IT system standards that focused on security, CII standards focused on safety and availability. Commonly referred as industrial control protocols, e.g., IEC 61850[2] and Modbus/TCP,[3] a closer study clearly indicates that safety and timing issues are addressed and traditional security is outside the scope of these protocols. Several industrial control protocols are currently under revision which will address some of the security issues. However, existence of legacy systems and backward compatibility requirements are the challenges to overcome.

Another way of looking at CII is that they are in one way or another cyber-physical systems (CPS). This distinction and understanding is critical from security point of view as a compromise and security incident of an IT system results in information disclosure or financial loss, whereas in case of CII the result could be physical as well as financial. For example, security issues related to implantable medical devices are outlined in [3].

A quick comparison of IT systems and CII systems is depicted in Table 2. From a security point of view, the order of CIA requirements is in the opposite order. As an example, confidentiality is perhaps the most important characteristic in a business tender, while availability takes the top priority in case of a defibrillator during a medical emergency. Integrity remains at the center. Figure 2 provides a pictorial view of confidentiality versus availability across different types of systems. As far as hardware and application types go, IT systems typically prefer general purpose, commercial off the shelf solutions to keep the cost low and avoid vendor lock-in. However, for CII systems, special purpose hardware and industrial-type systems are used that are tailor made for specific applications. For example, the mission computer in a space application and a single board computer in a submarine will have different

[2] http://www.iec.ch/smartgrid/standards/.

[3] http://www.modbus.org/.

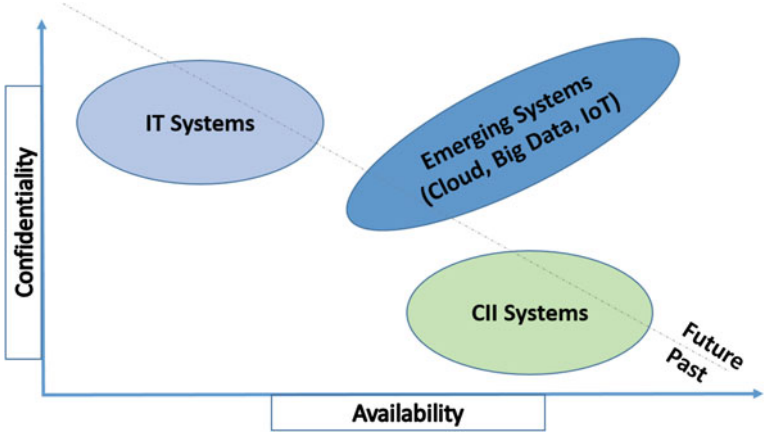

Fig. 2 Generalized view of confidentiality versus availability needs

requirements and are designed specifically based on the use case. General purpose solutions do not fit the bill here. Attack surface and attacker characteristics are reasonably well understood and documented for IT systems. Also, being non-mission critical, public declaration of vulnerability and patching them within a reasonable time frame is an accepted norm. However, for CII systems, due to the strategic nature, attacks are not disclosed. When they do become public, the system will become non-operational till the same gets fixed. For example, if a vulnerability resulting in potential brake failure in an automobile is discovered, then no vehicle owner will drive that type of vehicle till it is fixed to the satisfaction of the user. This also forces the vendor to work behind the doors and come up with a fix to deliver, possibly without even disclosing the real reason for delivering the fix. This restricts pooling in of resources. However, of late just like CERT[4] advisory for general purpose IT systems, ICS-CERT[5] for industrial control systems has become active. Hopefully, the benefit of community effort and pooling together of resources will happen.

3.3 Confidentiality

Confidentiality is the most discussed topic when it comes to data security. It implies placing a set of rules or restrictions with the aim of limiting access to the data in question. Confidentiality is hence achieved through access control mechanisms. While confidentiality relates to any type of data, if the same is associated with PII, then this may be referred as privacy.

[4] http://www.cert.org/.

[5] https://ics-cert.us-cert.gov/.

Traditional security mechanisms have relied based on restricting data access in one or more of the following ways as required:

- Store data in plain text, but use cryptographic techniques during transmission for security. Here, the assumption is that the data in transit is vulnerable.
- Store data in plain text, but some authentication is needed to access. Password protected files fall under this category.
- Store data in an encrypted way and decrypt whenever access is required, often used in portable devices.

The mechanism used to implement confidentiality services is usually referred as authentication, authorization, and access control (AAA). Authentication refers to the establishment of user identity. Authorization refers to determining the resources that the authenticated user can or cannot have access to. Access control is the process of enforcing access permissions to authorized resources while restricting access to other resources. Such an access control could be with additional constraints such as time of the day or IP address from which connection is made.

Use of passwords for authentication has proven to be futile in most cases.[6] Multifactor authentication is recommended but with caveats.[7] For example, compromising point of sale units is highlighted in [4].

With the quantity of data going northwards and redundancy becoming the norm rather than exception, access control is a daunting task. Parts of data are likely to be provided with anonymity, security, privacy as well as public in any repository. The amount of anonymization or sanitation that is needed mandates development of automated and intelligent tools. To address access control during transit, IPv6 has some kind of encryption as part of the protocol as against IPv4 which does not care about it. Of course, IPv4 was designed with the idea enable sharing and IPv6 aims at secure sharing.

Crypto algorithms have become synonymous with confidentiality. The strength of crypto algorithms has relied on key strength and difficulties in brute forcing to break them. However, analysts agree that it is possible to decrypt any encrypted message given sufficient samples. Big data just does that—provide sufficient, perhaps more than sufficient samples.

3.4 Integrity

Integrity implies trust. In terms of security, it means that the data did not get modified and is the same as the time at which it was created/edited by authorized entities. This could be source data or a result of certain calculation or editing in an authorized manner. Integrity also implies consistency of data across multiple

[6] http://www.hongkiat.com/blog/keeping-online-data-safe/.

[7] http://www.asd.gov.au/publications/csocprotect/multi_factor_authentication.htm.

copies. While in paper world, original or authenticated copy is used to confirm integrity of any new copy, in cyber world finding the origin of data itself is a challenge.

Data integrity issues could occur because of one or more of the following:

- Hardware errors,
- Software errors,
- Intrusions, and
- User errors

Preserving data integrity is an issue with storage mechanisms themselves and several tools such as *fsck* utility in Unix operating system. At a file system level, transactional file system could help [5]. They help identify data corruption. If we consider a distributed file system such as the Google File System [6], then it employs a checksum-based technique called chunkserver to detect data corruption. Mirroring, RAID, and checksum are commonly employed techniques to handle data integrity. Several host intrusion detection systems perform integrity verification[8,9] to detect intrusions.

Data loss or theft or breach is another issue that affects integrity. Once copied, it could be tampered and released to create confusion and integrity issues. Data loss prevention (DLP) is a common technique used to avoid data loss [7]. However, it is designed for traditional systems where specific computers implement DLP. As soon as data is accessed from a non-DLP system, the purpose is defeated. In addition to security, data loss often compromises privacy as well.

3.5 Availability

Availability implies the ability to access authorized data as and when required. Availability of critical data is achieved mostly by designing "high availability" (HA) systems. HA systems are designed with backup servers and alternate communication links. Disaster recovery and standby systems address HA needs. With big data and cloud, availability in itself is unlikely to be the issue due to the redundancy of data and multiple access routes. The challenge would be ensuring access control to authorized users and entities. Ensuring controlled access in the world of IoT does look intimidating.

With the emergence of IoT, entities such as sensors and appliances also need access to data. The resource constrained entities such as sensors and point-of-sale terminals being part of the mix, breaking into the network, and achieving privilege escalation to overcome access controls is a distinct possibility. In a way, this complexity is creating challenges for security teams to detect security incidents.

[8] http://www.tripwire.com/.

[9] http://www.la-samhna.de/samhain/.

This can be seen from the fact that in 2012, about 37 % organizations were able to detect intrusions by themselves, while in 2013, it came down to about 33 % [4].

4 Privacy

Privacy is essential in many ways. Personal elements in life such as intimacy, friendship, role play, and creative experimentation need to remain private. However, IT-enabled and IT-networked world of today is posing challenges in maintaining privacy. In data world, privacy typically refers to PII. In several countries, privacy is protected legally, e.g., Health Insurance Portability and Accountability Act (HIPAA 1996 US) and Sarbanes–Oxley Act (SOX 2002 US). Privacy aims to protect information that is considered personal and should not be shared without informed consent. Even when shared, use of PII is often restricted to specific purposes. However, to protect citizens, countries have enacted laws that define constituents of PII. Whenever there is a need to share information which also contains PII, then the PII components are anonymized or sanitized before sharing. For example, it is a very common practice to understand drug effectiveness on patients by studying medical records. While such a study is essential to understand and develop more effective treatment, PII part of the medical record needs to be removed by anonymization to protect individual patients.

Accessing anonymized data is becoming easier by the day as connectivity gets better. Even if accessible, limitations in terms of compute capability and knowledge to process data were challenges, not so long ago. However, it is a thing of past. With free online tools and powerful personal computers, processing capability has reached one and all.

What can big data do to privacy? In another world, not long ago, a person can go for shopping and come back while pretty much remaining anonymous. Today, with online monitoring, one can be traced from the time one leaves his/her home, the route taken and items bought in a store. The security/surveillance system at home records the car leaving the garage. Traffic monitoring cameras record the route. The store card and/or the credit card used to buy the items ensures recording of shopping info. And this is for real! Today we capture so much data that were never captured and stored in the past, which is what big data is doing. What was anonymous is suddenly traceable to individuals and privacy issues crop up.

Big data has impacted privacy so much that the USA is contemplating changes to the "Consumer Privacy Bill of Rights" of 2012 [8] and initiated big data and privacy review in January 2014 [9].

4.1 Online Privacy

Availability of data online takes away the ability of individuals or organizations to decide by themselves sharing of information about them. Personalizing, it could be

re-phrased as my losing control over what, when, how, and how much I would like to share information about me. Any data with PII online is at once a security and privacy issue. While traffic monitoring is a security issue, tracing the route taken by an individual is a privacy issue. Tracing the route and owner of the vehicle may be acceptable from security point of view for crime investigation, the same will be violation of privacy as often is the case, e.g., with paparazzi. As shown in Fig. 1, one can argue that there is no such thing as privacy, particularly once data is online.

What constitutes PII is generally as defined by the law(s), which typically is a list of information types, e.g., first/middle/last name, driving license number, date of birth, and credit/debit card number. The very definition of what constitutes PII permits use of other information which sufficiently enables identifying specific target audience. Value of PII has been understood by businesses, and hence, today data with PII has gained so much traction that a plethora of branded credit/debit cards, loyalty cards, frequent flier programs, and the like have emerged. They do provide some benefit to the customer in terms of discounts, and receiving targeted information. Yet it should be mentioned and noted that several online agencies and service providers sell their products and services by highlighting their ability to target identified users. Anyone browsing internet today can see advertisements popping up that are very specific which is possible only if some uniquely identifying data is available.

Very often, customers are provided with notice as well as terms of use and similar options while installing or using services. The information shared based on the agreement or consent given is for that instance and isolated. However, the moment multiple independently consented data are correlated and analyzed over time, what gets revealed is not what was consented for! Because big data reveals patterns and information that could not be found otherwise. A more interested reader can refer to [10].

4.2 Offline Privacy

Is there an issue with offline data privacy? After all, the data is not online! Let us consider a paper document containing PII. As such, it is perhaps safe to assume that without physical access to the document PII remains protected. Governments own a large number of data containing PII. Laws like the Freedom of Information Act enacted in USA or the Right To Information Act enacted in India enable access to public records. It is therefore reasonable to assume that at least data lying with government agencies offer only limited protection of PII. At the same time, as countries enact one or another form of paper reduction acts as well as physical storage constraints along with environmental issues, paper documents are continuously being replaced by electronic versions and hence making them online. As we move forward, with smart home, smart appliances, surveillance, and monitoring in public and work places, there will be nothing offline. In a nut shell, offline privacy is evaporating fast and sooner or later offline privacy will become a thing of past.

4.3 Privacy Post Archival

Archiving data for historic preservation to legal mandate is not uncommon. The moment archived data is accessed and become "actionable," the question of privacy crops up. Predicting future behavior based on past behavior is a trivial task. Such predictions need observation or access to the past behavior. By analyzing archived data, we are just enabling predicting future behavior, and in a sense identifying "stereotypes." Till few years back, most email providers offered a limited mail storage option to the users. This forced the user to delete emails that are no longer relevant or possible to manage without. However, of late, unlimited storage option is provided by email providers and data will continue to be archived. The traditional life cycle management of data where there is a definite phase to destroy is becoming less relevant. With social networks having become so popular, a childhood prank may make someone a suspect or unsuitable for certain positions at a much later stage in life, due to potential "similar" future behavior due to the past behavior!

The challenges of archival and disposal has been discussed by the National Electronic Commerce Coordinating Council in their report [11].

5 Emerging Trends

5.1 The Story So Far

Emergence of cloud has taken virtualization and elasticity to new heights. Service providers are already offering software as a service (SaaS) to provide on-demand software, platform as a service (PaaS) to provide platforms on demand to build/run applications, and infrastructure as a service (IaaS) to provide on-demand provisioning of resources (e.g., virtual machines). On demand and elasticity are there to use.

The addition of "smart" to everything enabling IoT has catapulted the number of entities that could communicate in the cyberspace while erasing boundary lines across several segments of industry. Smart devices and broadband access have resulted in the amount of data generated to unthinkable limits, and no amount of prediction seems to be close to reality resulting in "big data." Such a rapid change in the infrastructure, communication, and data generation has left system designers grappling for an understanding of potential security and privacy threats. Increase in availability has resulted in potential access to a large number of entities hitherto not part of the design considerations.

5.2 On the Horizon

As we look forward, it is clear that several things are unraveling. Cloud is moving from a monolithic option to offer variety in terms of private, public, community,

and hybrid clouds. Business verticals are expected to use community clouds to leverage on consortia approach. Private cloud would be the choice for strategic areas such as defense. The current offerings will mostly fall into public cloud and will become more of a commodity offering. Hybrid clouds are likely to be adopted by large organizations to keep critical operations on their private cloud and keep the majority on public or community cloud to optimize on the cost.

IoT itself will get better defined with new protocols defined. Variants of IoT such as industrial IoT would emerge. The current internet protocols may get replaced by new ways of networking. Techniques to uniquely identify devices connected to IoT will emerge. The question of whether the streams of data generated by smart devices and sensors need to be stored, and if so, how and where to do are all the questions that will be debated for possible solutions. Replacing compromised and/or defunct sensors will be major challenge that will be addressed by researchers and industry.

Big data requires big time crunching. We see plans announced by the National Security Agency (NSA) that their Utah Data Center has plans to have exaflop (1018 bytes), zettaflop (1021 bytes), and yottaflop (1024 bytes) by 2018, 2021 and 2014, respectively [12]. The compute capabilities will not only enable analytics, but also challenge strength of crypto algorithms. We can expect novel crypto or even new technologies from unchartered territories may emerge.

5.3 Research Challenges

Characteristics of big data and the possibility of global access to such data have not been understood sufficiently. Ability to mine patterns from autonomous sources has resulted in privacy issues. Anonymization of individual data sets is proving to be insufficient. Key research challenges include:

- Understanding characteristics of large volumes of data.
- Understanding characteristics of large and diverse data sets.
- Identifying duplicate data sets.
- Security requirements of big data.
- Access control mechanisms when availability is difficult to control.
- Keeping integrity of multiple copies of data.
- Challenges in maintaining anonymity of data.
- Anonymizing data containing PII.
- Designing of crypto algorithms whose strength does not depend on sample size.
- Identifying best practices for security and safety, including new type of data sources and sets.
- Real-time monitoring with encrypted streaming data.
- Achieving reliable computations in distributed multi-core environments.
- Reducing the information overload for security and information managers.

- Privacy preserving data analytics.
- Audit and compliance verification of information systems.

The list of challenges above is not exhaustive but indicative. We can look forward to researchers trying to achieve technical solutions to these issues. Government and standards bodies will be fully occupied with defining policies, drafting laws and standards, identifying best practices, and promoting their adoption. Academics will define new curricula. Businesses will need to re-orient and redefine strategies as they get access to more data, while their own data gets shared faster and wider.

5.4 Summary

In this chapter, we covered several security-related topics in a concise manner. We started with a discussion on cloud, IoT, and big data as well their effect on security. This was followed by a short discussion on security and privacy as well as the effectiveness or otherwise of anonymization with the emergence of big data. Security was then addressed in some detail. In particular, the contrasting requirements on confidentiality and availability with the convergence of IT and CII was highlighted. We briefly covered CIA aspects as well. Privacy, being an issue that affects everyone, was discussed. Going further emerging trends including a partial list of challenges were highlighted.

Even though we do hear pessimistic view like there is no such thing as privacy in an online world, it may be worth remembering and recalling that mankind has seen much tougher challenges and it has risen to the occasion each time and has come up with a solution to each challenge. This time also we can trust our scientists and engineers to come up with workable solutions to the issues at hand.

Questions

1. List the reasons stating why big data security concern is different than that of conventional data.
2. Explain the association between cloud technology and big data and state the security challenge the association brings in.
3. Discuss on trade-off between security and anonymity of big data.
4. Draw a comparison between IT and CII systems on security issue.
5. List the mechanisms used to assure data confidentiality and illustrate on each.
6. Why big data could be vulnerable for confidentiality?
7. Present a scenario how big data can invade one's privacy.

References

1. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertainty Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
2. Dunn, M., Wigert, I.: Critical Information Infrastructure Protection. https://www.emsec.rub.de/media/crypto/attachments/files/2011/03/ciip_handbook_2004_ethz.pdf (2004)
3. Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., Fu, K.: They can hear your heartbeats: non-invasive security for implanted medical devices. In: Proceedings of ACM SIGCOMM, Aug 2011
4. Threat Report: M-Trends beyond the breach. https://www.mandiant.com/resources/mandiant-reports/ (2014)
5. Gal, E., Toledo, S.: A transactional flash file system for microcontrollers. In: Usenix '05: Proceedings of the Usenix Annual Technical Conference, pp 89–104. Usenix, Anaheim, CA, USA (2005)
6. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03), pp 29–43. Bolton Landing, NY, Oct 2003
7. Kanagasingham, P.: Data Loss Prevention. SANS Institute. Aug 2008
8. The White House Washington: Consumer data privacy in a networked world: a framework for protecting privacy and promoting innovation in the global economy. http://www.whitehouse.gov/sites/default/files/privacy-final.pdf (2012)
9. Podesta, J., Pritzker, P., Moniz, E.J., Holdren, J., Zients, J.: Big data: seizing opportunities, preserving values. http://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf (2014)
10. Sloan, R.H., Warner, R.: Unauthorized Access: The Crisis in Online Privacy and Security. CRC Press, Boca Raton (2013). ISBN:978-1-4398-3013-0
11. NECCC: Challenges in managing records in the 21st century. National Electronic Commerce Coordinating Council 2004. https://library.osu.edu/assets/Uploads/RecordsManagement/Challenges-in-21st-e-recs-neccc.pdf (2004)
12. Bamford, J.: The NSA is building the country's biggest spy center (watch what you say). WIRED. http://www.wired.com/2012/03/ff_nsadatacenter (2012)

Big Data Service Agreement

Hrushikesha Mohanty and Supriya Vaddi

Abstract World today having several services on Web generates high volume of data around, so much so that one cannot just afford to ignore. These data volumes together usually referred as big data, though look heterogeneous and unrelated at a glance; still, big data carry striking relations among them implicitly as well as explicitly, so that many users across the world may get interested of data patterns being generated at a far end of the world. Thus, there is a need to deliver big data available on cyberspace to users as per their needs. This brings in a notion of big data service. For such service, we here propose a cyber infrastructure that takes user's data request and finds an appropriate data service provider to deliver data. We also propose an approach for SLA: service level agreement specification. SLA for a service is derived from SLAs of data consumer and provider on negotiation. Matching of consumer's SLA to that of a provider SLA and a process of negotiation between these two are illustrated.

Keywords Big data SLA · Big data as a service · Service agreement · Agreement specification · Data service · Matching · Negotiation

1 Introduction

Currently, as internet penetration grows exponentially, many services are being offered on internet. People while using internet generate digital footprints that make goldmine of information useful for innovation as well as better service management. Chapter "Big Data: An Introduction" discusses on big data and its usages in different

H. Mohanty (✉) · S. Vaddi
School of Computer and Information Sciences, University of Hyderabad,
Hyderabad 500046, India
e-mail: hmcs_hcu@yahoo.com

S. Vaddi
e-mail: supriya_vaddi@yahoo.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_6

domains. Now, it is well understood that providing big data to intended users is itself a service. An agency can provide a cyber infrastructure that collects data from different service providers and offers data analytics for data consumers for their uses and decision-making. This is termed here as big data service. In order to bring order in data requisition and delivery, a concept of service level agreement (SLA) needs to be introduced as the provision that has been there already for Web services. Here, in this chapter, the issue of SLA for big data service has been dealt with.

SLA needs to be specified in such a way that a provider and a consumer can specify their service as well as requirement particulars, respectively. And the same could be unambiguously understood not only by both but also by a third party who could be a party for processing of SLAs. SLA processing takes place in two steps, i.e. *matching* and *negotiation*. Matching of SLAs of providers and consumers is to find a provider which meets the data requirement of a consumer. It has certain similarity with webservice match making. On finding a provider, negotiation is initiated to avail the data service at the cost and QOS a consumer is interested in. And then, agreement is reached and ready to be enforced. Enforcing an SLA in a service makes it to monitor and adapt in an appropriate way.

Next section presents a motivating example illustrating the need of SLA in making business decisions. Then a framework for data service is proposed. The functionalities of components of the framework are illustrated. An example of SLA specification is provided in the fourth section. Syntax for spelling out data of a service provider and to detail customer requirements is presented. Readers may kindly note the words customer and user are used interchangeably for the same purpose in our illustration here. Next section, i.e. fifth lists out issues in SLA processing. Particularly, two algorithms one for SLA matching and another for SLA negotiation are presented. Agent-based negotiation is the highlight of the proposed approach. Some related works are reviewed in the sixth section. In review, we have also taken up the literature that is closely related. This chapter ends with a concluding remark.

2 Big Data Service and Agreement

2.1 A Scenario

Service-based industry has been the order of the day having opted for a lifestyle that happens to be primarily guided by internet. For the purpose, we will take a retail agency *MartBridge* that supplies home needs to people. The agency also sources its merchandise from farmers. Farmers sale grains and greens to MartBridge and then it does processing and packaging before putting the items on shelves for sale. Customers on internet put their requirements for purchase. The agency on delivery of requested items closes a transaction initiated by a customer. This puts up a simple scenario where MartBridge interfaces many from producers to consumers. A scenario

of this kind has been there in ages since trading has taken a root in our civilisation. Now the present age has offered technology that plays definite role in our life style. We will draw a picture of interactions the stakeholders have in such a scenario.

Consumers are at the front who interacts with the service provider MartBridge that retails goods consumers need. MartBridge at the back-end takes services of different service providers that drive the business. Consumers may not be aware of the roles of these back-end service engines. In this case, let *MartFresh*, *MartPackage*, *MartTransport* and *MartWare* are the service providers entrusted with definite roles to play in provisioning their services to their respective clients. MartBridge serves to its consumers, i.e. people put order of purchases. And the retail agency supplies it on availability of items. A service transaction made between a consumer and retailer has many aspects like time taken in accepting a service order, time taken in delivering a service, availability, quality of service items, cost of items, service cost and like many more features. These are the data generated from a service, termed as *service-generated-data*. And these data can be of immense use for both a service provider and a service consumer. A service provider can improve upon its performance on making a study of its performance at different times, i.e. at peak hour of demand. Similarly, a consumer would like to know performance of service providers for selecting one of its choices. Vice versa, a service provider can develop a consumer relation on collecting data on consumers. It is true before, we have such trading scenario but difference now is with its abundance and speed. For pervading internet, millions of people and similarly a large number of services exist on World Wide Web and at a moment a huge number of transactions is on execution. For these transactions, footprints of services and stakeholders leave a large amount of data for further processing and reasoning out on subjects of interests. In addition to *volume* and *velocity*, the *veracity* of data is also of concern. Primarily, these three make **big data** difficult to handle with traditional database management systems.

Coming back to our discussion on the scenario at hand, let us consider other two co-services, MartFresh and MartTransport, that take part in a supply chain connecting farmers to consumers through the retailer MartBridge. The service MartFresh sources grains and greens from farmers and then MartTransport picks up to a warehouse that is managed by a service provider MartWare. At warehouse, MartPackage carries out processing and packaging of grains and vegetables, and stores those for retailing. On demand again MartTransport takes the merchandise from warehouses to MartBridge. A supply chain that runs through these services is shown in Fig. 1.

MartChain is a collection of services and each has its own objectives. The services are connected in a sequence that exhibits the way they work together for a purpose. Let us call it SRG, i.e. Service Relation Graph. It also explains how a service composition works as a workflow to deliver service required by a consumer. It is not real that on receiving a customer order a workflow connecting all services gets activated. At different levels, different parts of a flow get invoked at different conditions. Such a graph can be further qualitatively enriched by labelling nodes and edges with service specification stating limitations and constraints at which

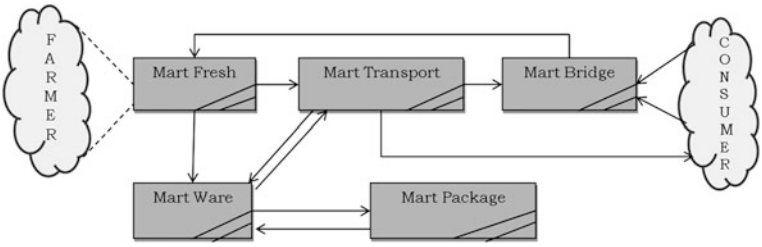

Fig. 1 MartChain: service relation graph

service would promise to work. While services on SRG are in execution state, analytics for each service (at each node of SRG) can be deployed to gather information on service performance. Some of these performances are listed below:

- invocation time;
- processing time;
- rate-of-failure;
- trust;
- security;
- negotiation time;
- maintenance time;
- user recommendation;
- associations;
- obligation;
- cost estimation;
- Amount of data processed;
- Amount of data received;
- Amount of data emitted;
- Data quality of input and output;
- Data persistence time;
- Data movement latency in/out of service; and
- Data losses due to velocity of data that comes in

Let us brief on usages of such analytics with the help of the example presented here. A customer before putting an order may like to get assured how efficient MartBridge is in taking a call. That is invocation time, which could be computed averaging the agency's historical behaviour in servicing calls. Processing time denoting an average time MartBridge takes to deliver goods at consumers' doorstep is computed by an analytic called *processing time*. Similarly, analytics to compute rate of service failure and trust metric are of use for MartBridge service users. MartBridge may compute worthiness of MartFresh in supplying good-quality grains. Similarly, grain growers would like to get trust worthiness of MartFresh to get assured of their payments as well as of good sale prices. Thus, MartFresh needs to have an analytic to compute trust. It may be noted that trust analytics at MartFresh has two facets, i.e. one for MartBridge and another for grain growers.

MartBridge customers may feel secured of their transactions with the retailer for secured money transaction, merchandise delivery and privacy preservation, etc. Negotiation is a primitive activity in business. For example, MartBridge may go with negotiation with MartWare for storing its merchandise. Negotiation must have a protocol to decide on consensus that needs to be followed during service execution. The consensus, i.e. agreed upon terms and conditions, is termed as **SLA**: Service Level Agreement. There could be an analytic to manage a service run in accordance with the service agreement. The maintenance analytic would fix service execution to follow a given SLA. Recommendation is another practice that exists in service paradigm. For example, *MartBridge* may recommend *MartPackage* to MartWare for packaging service. MartBridge can recommend a loan scheme to its consumer for purchasing merchandise of higher volume. A customer may wish to know the services to which the retailer MartBridge is associated with for its confidence building. For this, an analytic querying on associations can be invoked. With reference to Fig. 1 some services have direct interactions and some have indirect. Like, MartBridge service directly interacts with *MartTransport* but indirectly connected to MartPackage. The former relation is called *strong association,* whereas the later is termed as *weak* association. Such characteristics associated with services are required to be understood, quantified and collected for decision-making.

A service consumer can make use of such analytics to make a decision on choice of a service and to define service requirements. For example, a consumer may define the quality of items, delivery period, cost range, etc. while choosing a service. Thus, the defined service analytics can be invoked to measure service performance during service execution. Using the example, we have shown the usages of analytics and the data (on their performance) they generate, in conducting business especially on cyberspace. Availing such data in cyberspace is called here as *big data as a service.* Next, we look at the notion of big data in a concrete form.

2.2 Big Data

The term is a buzzword among computer science professionals as well as researchers. Here, we would like to seek what makes this word so emergent and why it has caught the imaginations of interested community. We have been modelling our information world so successfully with *ER* Entity-relationship modelling framework. A two-dimensional structure *table* has been used to model both data and their relationships. Instantiations related to a table populate the table as tuples. The characteristics of such a data repository are regularity in dimension and static in growth; though now and then there could be modifications to the repository to ensure that the repository reflects true of its world. As we have discussed through *MartBridge* scenario, the world today changes fast and so happens to the generation of data. A true reflection of world through traditional database management systems is almost impossible for rapid velocity in data generation. Data, now being created

has a time of its birth and in some cases has a lifetime too. More, such data are to be stored for posthumous analysis and future prediction. In past, such data repositories are usually of relational databases. Now, fast evolving heterogeneous data types provide opportunity as well as challenge to look for different data representation frameworks to make useful for applications of new genre.

Other than velocity, the next issue is of data dimensionality. We have been restricted to < *data*,*relation* > in case of relational databases. Currently, data are viewed in versatile situations with varying dimensions. Some of these include location, data owner, features, consumer, time of generation and lifetime. Thus, data descriptions vary from a kind of data to another. For example, a share value needs time and stock exchange (location) details for its description but international oil price needs only time reference but not location reference. Thus, *veracity*, i.e. heterogeneity in data dimensions, is an identifying factor for big data.

Changes in world of observation also bring changes to data repository that models the world. In classical databases, provisions for such modifications are taken care. But, repository of present kinds, for example data storage of *facebook* footprints, is quite different than classical data repository. Here, in every second, millions leave their footprints on their walls and as well as on other walls. Features, i.e. velocity of data generation and volume of generated data, are the two that standout big data from classical databases. These two along with the other feature, i.e. data veracity and big data management, faces an uphill task not only to decide on storage of data but also to manage it. The task gets further complex for its spread across the globe. For example, *facebook* or that matter of any such social networking services generate data across the globe. The data are not only structurally heterogeneous but also so semantically. As discussed earlier, information-rich big data is being made available as a data service on internet. In next section, we talk of agreement-based big data services.

2.3 Service Agreement and Management

We, in the previous section, have introduced the concept of big data. And also we have told that such data can be of use for many applications in different domains including governance, business, manufacturing, entertainment, education and health. Through an example we have reasoned, data generated on web can be tactfully utilised to provision services to consumers at agreed upon business conditions. This shows an organic relation between big data and service that can be built on it. Following the scenario described with MartChain let's detail on a concept called *big data as a service*.

Say MartBridge (Fig. 1) wants to source items for sale. There could be many agencies like MartFresh providing items that MartBridge requires. And each of these may have different levels of performance. Further, MartBridge business objective may be required to maintain certain levels of performance. In order to maintain its performance level, it needs MartFresh like associated services to maintain certain

level of performance. This kind of decision-making can be facilitated by collecting business data and applying analytics for decision-making. The next level of business intelligence is achievable by big data service, mainly on internet, so that a consumer requiring a data service can locate its service provider and avail this service by invoking appropriate analytics meeting its requirements. Materialising big data service on internet needs an infrastructure with required access provisions for data service providers as well as consumers. Like any other services, big data service also needs a SLA among the stakeholders those are data providers and consumers. For smooth service provisioning, such an infrastructure needs to generate a SLA agreement and to monitor adherence to the agreement by the stakeholders. Next section presents a framework of such an infrastructure.

3 Big Data Service Framework

As emphasised, big data being generated online is a mine of information that needs to be harnessed for usages in different domains. For example, *MartFresh* would like to know sales on different items at different outlets like *MartBridge*. Say *MartBridge* can supply its data only if *MartFresh* has signed up with the former and both have agreed with data sharing terms and conditions. Before, coming on to details on how terms and conditions are specified formally in a SLA, in this section we discuss on the framework that could be appropriate for the purpose. Figure 2 provides a framework for big data service. While discussing on the proposed framework with our example, we dwell upon agreements that can be operational for both data acquisition as well as data delivery.

The main components of the proposed framework are *ServiceMonitor*, *DataDispenser*, *SLAManager* and *DataMaker*. *SData* is a repository that stores data related to service execution. It also stores the service data in different forms as per users' requirements. The other repository *SLAR* stores SLAs that are used for service data collection (from different services) and delivery to users. Functionality of the proposed framework includes the following:

- Data request processing
- SLA management;
- Data acquisition
- Data delivery

As a whole, one can summarise to say, this framework for big data service collects service-generated data and delivers to users who need it. And the process is guided by an agreement among three players having stakes in a data service.

The proposed framework in Fig. 2 for the data services interfaces both users (data consumers) as well as service providers. Service execution footprints are the data users may query on. *DataDispenser* interfaces to users for collecting users' data requirements and conditionality. It also delivers data to a user based on its conditionality. This conditionality is built to specify desire of availing service in

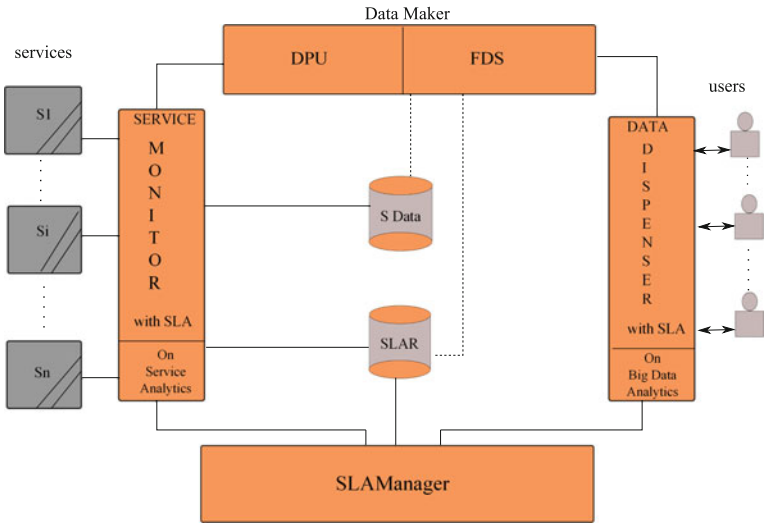

Fig. 2 A framework for big data ServiceProvider

terms of a cost and corresponding QoS. *SLAManager* on receiving users' terms and conditions negotiates with service providers and formulates an agreed document called SLA that is to be followed by both, for provisioning services that users ask for. It stores the SLAs at *SLAR,* a repository of SLAs. *ServiceMonitor* has dual functionality. It collects data from service footprints made during service execution. The collected data are stored in *SData*. It also monitors adherence of a SLA meant for a data service to a user.

In order to make the proposed framework operationally useful, different apps and applets may be implemented for desired services. The component *DataDispenser* contains interfaces to users' applications to serve specific purposes like placing data service request, specifying format for data display and specifying terms and conditions for a SLA. For these applications even apps can be made available on this framework for aiding users on move in availing the service of the framework. DSR (u,R) is a module in DataDispenser component; the module takes request *R* from user *u*. A request R $< \text{p},d_n,\text{sla} >$ specifies data provider p, required data names d_n and 'sla' specifying conditionality of users choices at which the service is intended. Levels in a service are defined on the basis of its output, timeliness and cost. DDU(u,F) is a module to display data for user u, in a format F, a tuple $< p,d_n,v >$ of data provider, data name and visual form intended by the user. Other than these applets and apps, the component can have analytics that provides a service statistics on users, service providers and their interactions. Analytic like $vdt(u,p,T)$ computes volume of data transacted between user u and provider p in time *T*. Alike computing analytics query on user service satisfaction, recommendation and other necessary issues such as security and trust on services can be made available for users.

DataMaker component works as request processing engine that takes user requests *R* provided by *DSR*(*u,R*) of component *DataDispenser*. With respect to a given *R* from *u*, *FDS*(*u,R*) module fetches data from SData and hands over to *DataDispenser* to display (by module *DDU*(*u,F*)). In the process, it consults the SLA if there any in between a provider and a user. DPU(), i.e. Data processing Unit, has manifold functionalities. One is of processing data that is required for a user request. With respect to a user request, it also involves CUD() at *ServiceMonitor* in collecting data for a user.

The component *ServiceMonitor* as told earlier interfaces service providers and collects data, services generate while providing services to their respective clients. Here again, we would like to reiterate the framework is meant for big data as a service. The component makes provision for data collection by both service providers and data users. The former, a service provider *p* requests by calling module *CPD*(*p,D*), collect provider data D, a tuple $< d_n, sla_p, T >$ with data names, *p*'s SLA and time specificity *T*. For example, MartBridge wishes to put its stock details, say once in a day on data service infrastructure. Let us say this data service of MartBridge is made available on some service conditions, e.g. availability only to registered users or guest users on payment at a given rate specified in sla_p. A second approach for data collection would be of data user initiation. DataMaker upon receiving a data request from user *u* may call *CUD*(*u,O*) that collects user data as specified by *O* as tuple $< d_n, sla_u, T >$ with data name, user SLA and time specification, respectively. Data information, i.e. service foot points on being collected, are stored on data repository SData with timestamp. This is done by ServiceMonitor calling module *SSD*(*x,B*) to store service data pertaining to *x*–a user or a provider. The parameter B is a tuple $< p, u, sla_p, sla_u, d_n, t >$ with provider p, user u, SLA for service provider sla_p and that of user sla_u, data names d_n and time stamp *t*. Further, the component may be inhabited by applets and apps as analytics to measure performances of service providers as well as of the monitor itself. For example, analytics to compute average uptime *aut*(), trust and predicting service response time *psrt*() of a service provider are of interest.

SLAManager component has a module *GAD*(*x,C*) to get agreement from user (through DataDispenser) and service provider through *ServiceMonitor* and to store those on *SData* for references. It also can be used to edit or delete the SLAs. In *GAD* (*x,C*), x specifies either a producer or user; C contains either a sla_u or sla_p. Further, the component SLAManager has an important role, i.e. negotiation on SLAs. NPD (N_P, N_D) is a module to negotiate and produce negotiated document N_D; a SLA being agreed upon by user and provider is mentioned in N_P. SLA negotiation process either can be automated or intervened one. On this we deal exclusively in coming section. There can be some analytics to compute performance of SLAManager. For example, some analytics could be on average time taken for an automated or intervened negotiation, percentage on acceptance of a negotiation *pan*(), average lifetime of a SLA *asla*() and average time on SLA modification. For big data, as the data scenario

often changes, so the performance of SLAManager must match with the speed of changes in evolving data.

The modules that make different components of the proposed framework for big data service are discussed above; and a comprehensive view on these modules is presented in Table 1.

Table contains modules and analytics, residing with each component of the proposed framework. Module names are in capital letters while that of analytics are in small letters. Below, we present an overview on behaviour of the framework with the help of an interaction diagram as shown in Fig. 3, on the timelines of each component shown as actors. Vertical dotted lines for each component are their timelines and horizontal solid lines show the interactions they make. On receiving sla_u^s and sla_p^s, SLAs of a user u and provider p on service s, SLAManager starts negotiation to generate agreement document (by invoking its module NPD()). This is an asynchronous invocation of the module on occurrence of an event i.e. receipt of a SLA from any one of the stakeholders. Behavioural aspects of the framework are self-explained in Fig. 3 with the help of the description presented in Table 1.

Table 1 Framework modules

Component	Module/analytic	Functionalities
DataDispenser	DSR(u,R)	Data service request from a user
	DDU(u,F)	Data Display for user
	vdt(u,p,T)	Volume of data transacted
	uss(u,p)	User service satisfaction
	rds(u,p)	Recommended data service
DataMaker	FDS(u,R)	Fetch data from store for user
	DPU()	Data processing unit
	arp()	Average request processing time
	ptl()	Peak time load
	ltl()	Lean time load
ServiceMonitor	DMC(u,R)	Direct monitor to collect for user
	CPD(p,D)	Collect provider data
	CUD(u,D)	Collect user data
	SSD(x,B)	Store service data
	aut()	Average up time of a service
	psrt()	Predict service response time
SLAManager	GAD(x,C)	Get agreement document
	NPD(N_P,N_D)	Negotiate produce agreement document
	PUS(u,sla_u^s)	Put user u SLA for service s
	PPS(p,sla_p^s)	Put provider p SLA for service s
	ant()	Average negotiation time
	pan()	Percentage on acceptance of negotiation
	alsla()	Average life SLA
	SLA_Processing(R,u)	Processing SLA for a request R from user u

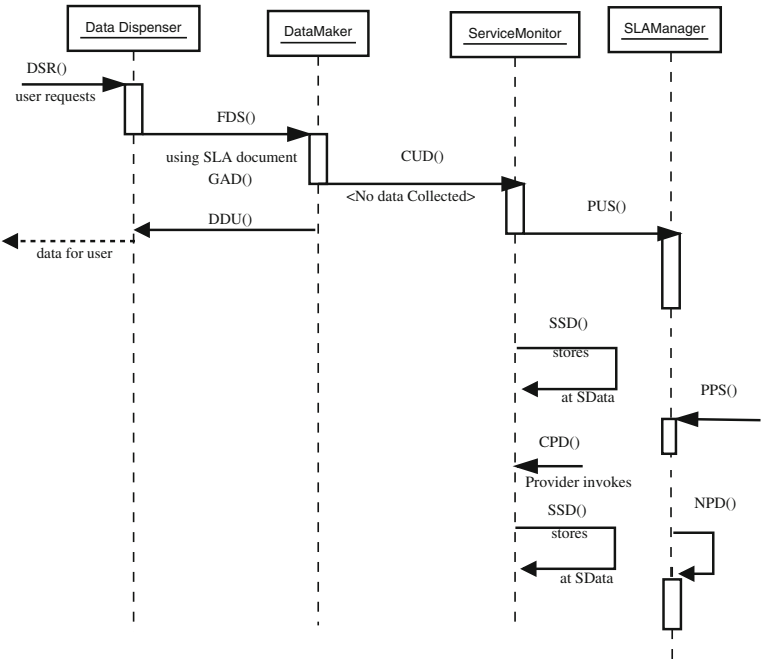

Fig. 3 Framework behaviour overview

The framework proposed explains how data evolved from different services are collected and delivered to intended users based on compatibility of conditions provisioned by both user as well as provider. So, the points of importance here include first, how conditionality can be expressed unambiguously considering the aspects involved in big data. Second, how a consensus among data providers and users can be evolved for data delivery. The first issue we take up in the next section and then in succeeding section details on the second issue.

4 Service Agreement Specification

Here, we refer to big data as a service and have proposed a service level specification details on grades of provisioned service and the conditionality associated with. Primarily, unlike webservices, the proposed specification addresses issues that are specific to big data. Data collection and visualisation are two important operations performed in big data services. Either a service provider offers its footprint data at a conditionality or a consumer wishes to avail such data at its own conditions. The conditions are defined on data quality and its associated cost. It also specifies at what interval data are to be collected and at what volume. Further, conditionality can be defined on forms of data such as audio, visual, or text. Thus,

the proposed specification is defined on *volume, velocity, and veracity* of data being created from footprints of a service execution. SLA specification for both user and provider follows a generic form we present below

Basically, the above SLA format can be used by both user and provider. A user *u* while preparing a SLA for a data service may explicitly specify the service name whose data the user is interested in. In case the user is not aware of a service who generates data of its interest then may choose to leave it to SLAManager (by specifying s*) to find an appropriate service. In case user needs, it can specify the locations at which service should be hailed from; this is defined by (< *lName* >). In data field, user specifies the names of data (*dName*), its formats (*dForm*) such as text, audio, and graphics, grade on quality (*dQoS*), volume of data (*dVolume*) and associated cost *dCost*.

It can also indicate time choices at which data are to be collected. SLA to avail a data service needs to be time sensitive as time has bearing on both data geneartion as well as data transmission. Temporal specification for a SLA is detailed as:

$$\begin{aligned}\text{SLA} :: (&service((< sName > [< lName >])|(< s* > across< lName >))\\ &user(< uName > | < u* > across< lName >)\\ &data(< dName >< dForm >< dQoS >< dVolume >< dCost >< time >))\end{aligned}$$

Like a user as stated above, a service provider can specify its SLA for providing data to a chosen user or to some which can be fixed by SLAManager. This scenario can be seen in the framework of publish–subscribe model that makes a match between published sets of $\{sla_u\}$ and $\{sla_p\}$ SLAs by users and providers, respectively. *DataDispenser* component by its module *DSR* allows users to specify its sla_u for the data request it makes. Similarly, the component *ServiceMonitor* by its module *CPD*() collects sla_p from service providers. Thus, collected SLAs are stored at *SLAR*, the repository of SLAs. In next section, we detail on actions of *SLAManager*. Here, next we put two examples of SLAs of a provider and a user quoting our example MartChain (Fig. 1).

Say, service MartTransport wishes to provide information *ItemL*: items transport list, *TptL*:List of vehicles and *OrderL*:List of transport orders, to any user (u*) located in Odisha or Andhra Pradesh. For each data item, it specifies a form of representation, e.g. text, image, volume of data in KB or MB, cost and time. For example, transport list *TptL* is represented in image form of volume 10 MB and can be delivered at 5 pm everyday, and for cost of rupees one thousand only. In our proposed SLA specification, a SLA for the purpose is presented as:

$$\begin{aligned}sla_{MartTransport} \equiv\ &(service\ MartTransport\ across\ Odisha\\ &user\ u*\ across\ Odisha, AndhraPradesh\\ &data\ (ItemL\ Text\ 5KB\ 100Rs\ before\ 12am,\\ &\quad TptL\ Image\ 10MB\ 1000Rs\ at\ 5pm,\\ &\quad OrderL\ Text\ 5KB\ 100Rs\ after\ 6am))\end{aligned}$$

Similarly, a user say 'Chabisha' wishes to avail data of any transport service, can specify its conditionality on different aspects of data service as given below.

$$sla_{uuu} \equiv (\ service\ s*\ across\ Odisha,\ AndhraPradesh \\ user\ Chabisha \\ data\ (OrderL\ Text\ 5KB\ 50Rs\ before\ 6pm, \\ TptL\ Image\ 1KB\ 20Rs\ at\ 6pm))$$

These two examples of SLAs by a user and a provider exhibit possibility of matches between their requirements. This matching is carried out during negotiation. Analytics that may aid to negotiation are listed in Table 2.

Like these, there could be even some more analytics defined on service provider, user and big data infrastructure. And these analytics provide the information on performance of the varieties of stake holders. For example, both the service provider and the user may like to know the average time *SLAManager* takes for a negotiation.

Until now, we have been talking of SLA specifications for two types of stake holders, viz. service providers and users. In between users and service providers, infrastracture plays a vital role for data acquisition as well as delivery. The framework Fig. 2 we have proposed can be viewed as an*IAS: infrastructure as a service*. Entities using this infrastructure for their usages are required to sign a SLA with the infrastructure. A generic SLA specification to use analytics the infrastructure provides is stated as:

$$sla_{IAS}^{Anal} \equiv (analytic < aname > \\ for \quad * | < bname > \\ with(< cost >< useType >))$$

An analytic supported by IAS needs to specify its *aname,* i.e. name of an analytic. And *bname* is the name of a benefactor for the analytic *aname*. Instead of specifying particular *bname*s, one can skip it telling the unrestricted access to the analytic for usages. It also specifies terms and conditions applied for availing the analytics. Term and conditions are specified with usage type *useType* and cost associated with.

IAS uses service providers and users' behavioural data, to provide these analytic service. The stake holders may sign a non-disclosure agreement for the service provided by analytics.

Table 2 Some analytics on data service

AnalyticName	Defined on	Functionality
TrustOnService(S)	service provider	Computes trust on a service provider
TrustOnUser	service user	Computes trust on a user
ServiceUser(S)	service provider	Lists the users source data of a service
UserService(U)	service user	Finds the services from whom user fetches data
AvgNegoTime()	SLAManager	Computes average time taken for negotiation

In this section, we have introduced schema for SLA specification by data service provider, consumer and infrastructure service provider. Initiation of a service includes both match of data supply and demand (by provider and customer, respectively) and their conditions. The next section deals with service match.

5 Data Service

The big data infrastructure presented in Fig. 2 acts as a facilitator to provide data services to service providers as well as users. Both users and providers wishing data services publish their respective SLAs invoking *PUS*() and *PPS*() available with SLAManager Fig. 3. A data service between a provider and a user is initiated on matching their respective SLAs. Thus, *data as a service* is viewed in publish–subscribe model. Matching of SLAs may result to either a match or a mismatch. In case of a mismatch, negotiation between a provider and a user is expected for possible atonements of their respective SLAs towards matching. Thus, SLA matching precedes to SLA negotiation. We detail these two here in sequence.

5.1 SLA Matching

For any two given SLAs, respectively, of a user '*u*' and a provider '*p*', let us extract their data components and label both, respectively, as SLA^u.dc and SLA^p.dc, each containing a set of conditionality as

$$\{(< dName >< dForm >< dQoS >< dVolume >< Time >< dCost >)\}$$

We are interested in investigating the match between SLA documents of users and providers. The documents SLA^u.dc and SLA^p.dc contain requirements and offers of a user and a provider, respectively. Process of match between two includes data match, QoS match and cost match. 'Data match' is of finding similarity between requirement and supply of data, i.e.

$$\begin{aligned} &matchD((SLA^u.dc.dName,\ SLA^p.dc.dForm)\\ &\qquad (SLA^u.dc.dName,\ SLA^p.dc.dForm))\\ &matchT(SLA^u.dc.Time,\ SLA^p.dc.Time)\\ &matchV(SLA^u.dc.dVolume,\ SLA^p.dc.dVolume)\\ &matchQ(SLA^u.dc.dQoS,\ SLA^p.dc.dQoS)\\ &matchC\ (SLA^u.dc.Cost,\ SLA^p.dc.Cost) \end{aligned}$$

Data match has five components. The first component *matchD* is to ensure the data supplied is as required. The second component *matchT* is to match temporal characteristics between the two, i.e. the provider and the user. For example, a user

wants to receive a data service at a given time interval. Then matchT returns true if there exists a provider who can provide data at the given interval of time. The third component *matchV* checks match between volume of data required and that of supply. Match operation *matchQ* is designed to check match between service QoS desired by a user and supplied by a provider. Similarly, *matchC* compares the cost wanted by a user to the cost quoted by a supplier. *Algorithm*1 for SLA matching performs the following.

Algorithm 1 SLA Matching Algorithm

```
SLA_Match(SLA^u.dc,SLA^p.dc)
Begin
SLA_Nego =0
md = matchD((SLA^u.dc.dName, SLA^p.dc.dForm),
                    (SLA^u.dc.dForm, SLA^p.dc.dForm))
if md =0 then
    abort
else
    mv = matchV(SLA^u.dc.dVolume, SLA^p.dc.dVolume)
    mt = matchT(SLA^u.dc.Time, SLA^p.dc.Time)
    mq = matchQ(SLA^u.dc.dQoS, SLA^p.dc.dQoS)
    mc = matchC(SLA^u.dc.Cost, SLA^p.dc.Cost)
end if
if ((mv ==0)∨ (mt ==0)∨(mq ==0) then
    SLA_Nego=0
else
    SLA_Nego=1
end if
End
```

The function *matchD* of *Algorithm*1 returns '0' if the required data are not available with the provider. In that case, the algorithm is aborted otherwise *md* assumes value 1. Then, the algorithm proceeds to match for other features, viz. *matchV*, *matchT*, *matchQ* and *matchC*. Each of these matchings returns 0 or 1 or 2 for not matched or exactly matched or subsumed match of user requirements to providers SLA specification. For not matching of any of these features, the match fails. For other two cases, i.e. exactly matched and subsumed match, algorithm indicates to proceed for next stage for SLA processing, i.e. SLA negotiation by setting the variable SLA_Nego = 1. It is to be noted that in case of exact match still, we go for negotiation as current business pressure may call for negotiation by provider or user or even by both.

5.2 SLA Negotiation

One can view negotiation as a protocol between a producer and a consumer initiated by one and responded by the other with respect to a merchandise or a service transaction. The protocol continues until either both agree at a point of negotiation

or one of them/both decide(s) to stop the negotiation. Here the proposed negotiation model is based on service cost. At the start both user and provider quote costs of their choices. During negotiation, a consumer, on receiving a cost quote from a provider, responds with an increase to its earlier proposed cost, provided the provider has shown interest in continuation of negotiation. For a provider, its response to a customer proposal is just reverse, i.e. decrease of cost asked for, in case customer is agreeing to continue with negotiation. With this basic concept, we present a negotiation protocol. Before that, below we present a concept called negotiation pressure: *NP*.

NegotiationPressure is a business analytic provided by this framework that suggests actors engaged in a business negotiation either to continue or to get disengaged. In former case, the negotiation protocol continues, whereas for the later it stops. Now we illustrate on computation of negotiation pressure at a user as well as at a provider. For each, three factors contribute to negotiation pressure. Let us consider a case of a consumer. The three factors that build up negotiation pressure for a customer to continue with negotiation with a provider are : (1) *Goodness* of the provider; (2) *Intensity of the service requirement; and* (3) *Competition among consumers for the provider*. We formalise negotiation pressure for a user u towards a provider p for its service s as

$$NP_p^u(s) = Qg_p^u(s) + Qr^u(s) + Qn_p^u(s)$$

$NP_p^u(s)$ Negotiation pressure on user u for service s from provider p

$Qg_p^u(s)$ Goodness of provider p at user u is the ratio between the number of successful transactions and the total number of data service transactions initiated between p and u. In case of no previous history, it assumes a zero value

$Qr^u(s)$ Intensity of requirement of s for u is defined as the ratio between the number of business goals of u requires s and the total number of business goals u has. In case of no goals, the term assumes zero value and value 1 shows the maximum dependence of u on s

$Qn_p^u(s)$ Competitions among customers (users) for p are defined by the ratio between number of users wanting the service from p for the service s and the total number of people looking for the same service at p and also at other providers. In case there are no such users then the term assumes the zero value

Thus, each factor may assume a value between (0,1) and total negotiation pressure for a consumer may vary in the range (0,3). At value negotiation pressure zero, customer will not initiate negotiation. Else for some value say x ϵ (0,3), negotiation starts calling an upward revision of cost. i.e. *quote_cost*^{u}() computed by function *NewQuote*() ref. *Algorithm*2. This computation can be done in many ways taking purchase practice a customer follows. A simplest way one can see is a linear increase at each step of negotiation. The number of steps for negotiation can be

fixed either by count or by amount of increase every call must make. For this, the other input required is *MxQuote*, i.e. maximum cost increase a user may call for.

A similar concept can be laid for a provider to compute its negotiation pressure that evolves from goodness of a concerned customer, ratio of unfulfilled business targets and market competition due to number of providers competing for the service. So, negotiation pressure for provider p for service s is computed as

$$NP_u^p(s) = Qg_u^p(s) + Qr^p(s) + Qn_u^p(s)$$

The semantic of each term is alike to that of a user we have discussed prior to it. Alike to $NP_p^u(s)$, $NP_u^p(s)$ assumes a value in range (0,3). For our model, we assume negotiation pressure at both provider and user ends. Negotiation pressure pushes stake holders to negotiate.

Here in our discussion, we have put negotiation over cost only, this means all other parameters such as quality, volume and time are being matched. However, if a user is ready to explore all possible combinations of parameters while selecting data service options, then a variant of *Algorithm* 2 can be designed to propose different options with different combinations of parametric values that may evolve. However, on going back to our strategy of cost-based negotiation, we here propose a negotiation protocol that solicits cost calls from each at each round of negotiation. Taking a cue from business practices, negotiation process is initiated by a user quoting lowest cost it wishes to give. In reverse, provider proposes the highest cost it wishes to gain in the proposed service transaction. On a negotiation step, user increases its quote, whereas the provider decreases its in case both have pressure to continue with the negotiation. Negotiation is an iterative process that solicits quotes from both at each step. The process continues till the quotes from both gets the same or one of them do not desire to continue with it. Figure 4 schematically presents a process of negotiation. The labels in angular brackets show conditionality. Exchange of data *quote_cost* between two is shown as a label above a horizontal line showing a message passing.

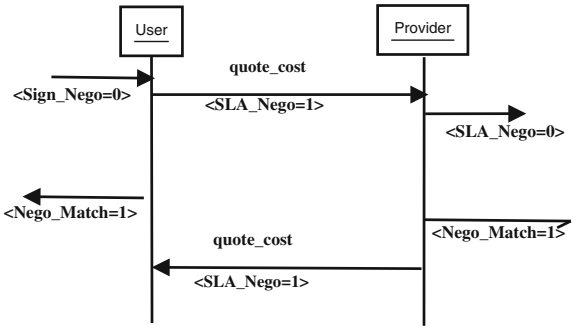

Fig. 4 Negotiation diagram

5.2.1 Negotiation: A Distributed Process

This section rolls out a scheme to carry out negotiation between a data service provider and a user. As discussed before, negotiation decision is privy to an individual engaged in negotiation process. Hence, the process needs to be executed at each, thus negotiation is a distributed process where each sends its negotiation cost to other by message passings. The distributed process eventually comes to an end when a negotiation termination condition is met at any one of the stakeholders. Further to make the process scalable, we have adopted $Agent - based\ negotiation$ where each of users and providers is represented by a software agent. At a time, there could be thousands of providers and users intending to avail data services over a service framework as suggested in Fig. 2. Implementing a negotiation process that is interactive, becomes highly time-consuming when participants have to be online and participating in the process. Instead, we have chosen agent-based negotiation that automates a negotiation process where agents participate on behalf of user and provider engaged in negotiation (note here we are considering 1–1 negotiation between a user negotiation agent *uNagent* and a provider negotiation agent *pNagent*). Further to our proposed 1–1 negotiation, we introduce a third-party agent called observing agent *oNagent* that initiates and subsequently monitors a negotiation process being carried out between a *uNagent* and a *pNagent*. A schematic representation of the proposed negotiation is presented in Fig. 5 ABN: Agent-Based Negotiation.

DataDispenser module of the proposed framework in Fig. 2 on receiving a data request from a user 'u', DSR(u,R) forwards the request along with SLA^u to *SLAManager* for SLA processing. *SLAManager* executes algorithm *SLA_Processing*. All users and providers need to put their SLAs for data services. This is done by *SLAManager*'s GAD() function that is available to both user and provider through DataDispenser and ServiceMonitor components, respectively. For a given request R on finding a provider p, system gets their respective SLA documents $SLA^u.dc$ and $SLA^p.dc$. These two are taken as inputs to the algorithm $SLA\_Match()$. In case the algorithm decides for negotiation, then $SLA * \varepsilon\gamma\omega\tau\iota\alpha\tau\iota\omega\nu$ algorithm (Algorithm 2) is called for the purpose. The variable *nid* assumes a unique number representing a particular process of negotiation between a producer and a user for a data service.

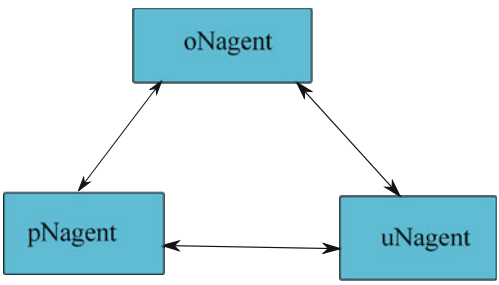

Fig. 5 Agent-based negotiation

Algorithm 2 SLA Negotiation Algorithm

```
var:: o : oNagent, p : pNagant, u : uNagent
// Executed at oNagent
At oNagent: On receiving SLA_Negotiate(cost_quote^{u,nid}, cost_quote^{p,nid})
begin
nego=1; u.nego=1; p.nego=1
send MakeAQuote(nid,nego) to p and u
end
// Executed at pNagent
At pNagent: On receiving MakeAQuote(nid,u.nego) do
begin
if ((NP_u^p(nid.s) > θ_u^p(nid.s)) ∧ u.nego) then
    //Provider under pressure to negotiate for service s with user u
    quote_cost^{(p.nid)} = NewQuote(nid) //Returns zero for no possible quote
    if ((quote_cost^{(p.nid)} < MnQuote(nid.s)) ∧ (quote_cost^{(p.nid)})) then
        p.nego=1;send GetAQuote(quote.cost^{p.nid}, p.nego) to o
    else
        p.nego=0; send GetAQuote(quote.cost^{p.nid}, p.nego) to o
    end if
end if
end

// Executed at uNagent
At uNagent: On receiving MakeAQuote(nid,p.nego) do
begin
if ((NP_u^u(nid.s) > θ_p^u(nid.s)) ∧ p.nego) then
    //User under pressure to negotiate for service s with provider p
    quote_cost^{(u.nid)} = NewQuote(nid) //Returns zero for no possible quote
    if ((quote_cost^{(u.nid)} < MxQuote(nid.s)) ∧ (quote_cost^{(u.nid)})) then
        u.nego=1;send GetAQuote(quote.cost^{u.nid}, u.nego) to o
    else
        u.nego=0; send GetAQuote(quote.cost^{u.nid}, u.nego) to o
    end if
end if
end

// Executed at oNagent
At oNagent: On receiving GetAQuote(quote.cost^{(x,nid)}, x.nego) do
begin
if (quote_cost^{(u,nid)} ∧ quote_cost^{(p,nid)}) then
    if (quote_cost^{(u,nid)} = quote_cost^{(p,nid)}) then
        SignSLA(nid)
    else
        if ((x = p) ∧ (quote_cost^{(u,nid)} ≠ quote_cost^{(p,nid)}) ∧ (u.nego)) then
            send MakeAQuote(nid,p.nego) to u
        else
            if ((x = u) ∧ (quote_cost^{(u,nid)} ≠ quote_cost^{(p,nid)}) ∧ (p.nego)) then
                send MakeAQuote(nid,u.nego) to p
            else
                send NegoFail(nid)to u and p
            end if
        end if
    end if
    quote_cost^{(u,nid)} = quote_cost^{(p,nid)} = 0
end if
end
```

The algorithm *SLA* ∗ εγωτιατιων holds key to the strategy guiding the proposed negotiation protocol. An observing agent *oNagent* initiates a negotiation sending *MakeAQuote* messages to *pNagent* and *uNagent*. Each on receiving this message do (i) Based on negotiation pressure and willingness of other to negotiate, it generates a new quote *quote_cost* by executing function *NewQuote* for a particular negotiation *nid*. (ii) Then, new derived *quote_cost* is tested for acceptability. This means, for the user the new quote must not be more than the maximum quote *MxQuote*. Vice versa for a provider it should not be less than the minimum cost *MnQuote*. (iii) The new quote is sent to their observing agent by a message *GetAQuote*.

The observing agent on receiving a *GetAQuote* message, does one of these two *only if new quotes from both provider as well as user are available*. (i) If both the quotes are the same then the agreement between the two is made by executing *SignSLA*. (ii) In case the two differ in quotes and the counter part is still willing to negotiate then the observing agent *oNagent* asks the other to make a quote. (iii) Else the negotiation fails.

Algorithm 3 SLA Processing Algorithm

```
1: SLA_Processing(R,u)
2: Begin
3: SLA_Nego =0,p=DMC(R,u)
4: if (SLA_Match(SLA^u.dc,SLA^p.dc)) then
5:     break;
6: end if
7: if (SLA_Nego) then
8:     nid = assignNo(u,p)
9:     cost_quote^{u,nid} = SLA^u.dc.cost
10:    cost_quote^{p,nid} = SLA^p.dc.cost
11:    CreateAgent(nid)
12:    send SLA_Negotiate(cost_quote^{u,nid},cost_quote^{p,nid}) at oNagent
13: end if
14: End
```

SLAManager executes *Algorithm* 3 *SLA_Processing* and if required invokes *oNagent* to start execution of *SLA Negotiation*. A copy of *SLA Negotiation* also gets executed at *uNagent* and *pNagent*. Thus, the algorithm *SLA Negotiation* is distributed in nature and gets executed at three agents being invoked among themselves until a negotiation process gets terminated. On successful negotiation, a negotiated SLA document is signed and comes for enforcement. The signed SLA becomes a document of reference during delivery of data service. As big data service environment varies and service providers and users are loosely coupled as well as geographically at distance, there is a need for third-party monitoring of service provisioning to ensure fairness in data service. This needs SLA monitoring mechanism. The module serviceMonitor of our proposed framework can have monitoring analytics for the purpose. This is an issue this chapter does not take up.

6 Related Work

Agreement for availing a service has been in real world long before. Idea of an agreement is sacrosanct for both the parties involved. internet-based services also need to follow the practice of agreement-based business delivery, and now it is essential when more and more services are being placed on internet. Here, we review some of the recent works on SLA specification, negotiation and monitoring. We refer to some work on web services that have enough resemblance to the issues we have taken up here towards data service agreement.

At the beginning, we take up [1] to highlight a framework required to provide big data service on internet. Currently, having service on internet, a lot of data on services are available from the footprints, e.g. service logs these services generate. This chapter proposes a framework capable of delivering these data to interested users. Big data as a service is employed to provide common big data-related services like access to service-generated big data and data analytics. These services are to be provided based on agreements data providers and users have. This agreement known as SLA needs to be specified.

Big data are huge in size as well as heterogeneous. Locations of data generation and of their usage could be at distance needing transportation of data at optimised cost. The work in [2] shapes it as a multi-criteria optimisation problem for data transfer in a cloud from data origin to place where they are needed. It proposes a scheduling policy and based on it two greedy algorithms are proposed for data transfer. Algorithms respect individual constraints and transfer data at minimised transfer times, thus bringing down the transfer cost as well as waiting time for the ones who need data. Performance of these algorithms is studied by simulation.

SLA specification has been studied, mostly at the advent of webservices that has accelerated ecommerce for making business over internet. Fully automated e-commerce needs rigorous SLA specification. Mostly, SLA has been addressing QoS issues and there have been works like [3] proposing a language *SLAng* to specify service QoS. This chapter introduces two interesting concepts Horizontal SLA that defines a contract between different parties providing the same kind of service. Vertical SLA regulates the supporting parties that make the underlying infrastructure for a service. The usages of SLAng are studied for webservices for processing images across multiple domains.

Since 2001, IBM has come out with Web service level agreement (WSLA) language specification [4]. WSLA defines means to specify agreements among the parties involved in a service. It defines the obligation of a service provider to perform a service according to agreed upon QoS parameters. Still, it also provides freedom to implement implementation policy and its supervision [5]. A similar specification by opengrid forum is also proposed [6].

Guaranteeing system performance as per defined SLA is interesting particularly for network-based systems for varying behaviour of network itself. Currently, cloud-based systems face a challenge to perform as per agreed upon SLA. A recent work [7] proposes a framework that addresses the challenge. It proposes a novel

cloud model that enables systematic and transparent integration of service levels in SLA into a cloud. A specification language CSLA is introduced to describe QoS-oriented specification. A control theoretic approach is proposed to enable a cloud to guarantee system performance as agreed upon. The idea has been applied to Map Reduce service, multi-tier e-commerce service and a locking service.

We have discussed the role of SLA in availing webservices and big data services. Similar usage of SLA has also been studied for webservice selection. Researchers in [8] propose a method for customers to select a service of kind it requires. A service provider prepares a SLA document and publishes on digital market place for customer consideration. They have proposed a scheme called WYSIWYG 'What You See Is What You Get' that customers use for selection. A customer takes a snapshot of SLAs (of providers) available on market place, i.e. on service infrastructure required for hosting a service as well as delivery of a service. A filtering framework collects SLAs and stores those on a SLA repository for processing. This chapter also proposes a data model for a SLA repository. The applicability of the proposed framework is studied by integrating it to web application layer of a service-oriented architecture (SOA). A guideline for integrating SLAs to SOA environment is proposed by Software Engineering Institute [8]. This report surveys the state of practice in SLA specification and offers guidelines for ensuring service quality including high availability, security, performance and other QoS parameters, if any. It also talks of mechanisms to monitor adherence to an agreement and to assure quality service as agreed upon, specified in an agreement.

Selection of a provider may lead to find one who can deliver a service that is required. But, there could be some unresolved issues such as cost or some other QoS parameters that needs to be agreed upon by both provider as well as service consumer. This process is called negotiation. This requires an algorithm for the ranking of functionally equivalent services that orders services on the basis of their ability to fulfil the consumer requirements at the cost agreed to both. A work in [9] proposes an adaptive algorithm for automated negotiation. It employs trusted negotiation broker that carries out bilateral bargaining of SLAs between a service provider and consumer. They define mathematical models to map high-level business requirements to low-level parameters used for decision-making and fixing bargains. The algorithm is adaptive for making decisions on receiving counter parts' bargains. Thus, the technique ensures the satisfiability for both engaged in negotiation.

On managing an agreement with a SLA, the next issue is enforcement of a SLA on service framework so that it can be used during service delivery as a document of reference. This requires a right kind of architecture. In [10], researchers proposed a conceptual architecture SLAWs for SLA enforcement. It builds a flexible SLA enforcement layer that could be integrated seamlessly to an existing service providing infrastructure. This architecture is designed on the principles of SOC: Service oriented computing outlining the elements and their relationships in providing services and also considers the utility of WSDL into account. Thus, it enables a webservice platform to perform both functional logic and to control non-functional properties. For enforcing a SLA, the question of monitoring comes

to our attention. In our chapter, we have not dealt with monitoring. Still, for interested readers, we would like to bring out this issue considering how this is done for composite services [11]. This chapter proposes event-based monitoring technique. In addition, prediction of potential violations is carried out using machine learning techniques. Run time prevention of those violations is provisioned by triggering adaptation techniques.

7 Conclusion

This chapter makes a case for data services on an infrastructure that collects footprints of service executions and delivers data to customers those who need it. Big data for tremendous growth on internet usage provide an excellent business opportunity for data services. It is different than other paradigms such as data warehouses for its very nature with high volume, veracity and velocity, thus making big data service unique on its own terms. We have illustrated this with an example.

Three important contributions this chapter makes are SLA specification, SLA processing including matching and negotiation, and a framework for big data service. For each, narration has gone to a level of refinement that is quite near to the implementation. Students and researchers intending to start work in this area can start with implementation of the proposed framework for taking a leap forward to further research. The reference discussed in the previous section provides a lead to further study.

Some related issues the chapter has not addressed include SLA enforcement and monitoring. SLA processing, i.e. matching, negotiation and SLA enforcement, for big data requirements is a research challenge that we wish researchers to take up for big gain.

Practice Questions

Q.1. What is SLA ? Why is it required?
Q.2. Define ‘big data service’ and explain it with your example.
Q.3. Present big data service specification language and specify the service for your example in Q.2.
Q.4. Write an algorithm implementing publish–subscribe model to find a data service for a given requirement.
Q.5. Present a framework that hosts the Q.5 solution for a big data service.
Q.6. Implement the framework proposed as a solution to Q.6. Choose appropriate environment and programming language for implementation.
Q.7. Explain why negotiation over internet needs a protocol.
Q.8. Write a distributed negotiation algorithm for a big data service.

Q.9. Present a framework for executing your algorithm for Q.8.
Q.10. Implement the framework due to Q.9 and simulate the performance of your negotiation algorithm proposed for Q.8.

References

1. Zheng, Z., Zhu, J., Lyu, M.R.: Service-generated big data and big data-as-a-service: an overview. In: 2013 IEEE International Congress on Big Data, pp. 403–410 (2013)
2. Nita, M.C., Chilipirea, C., Dobre, C., Pop, F.: A SLA-based method for big-data transfers with multi-criteria optimization constraints for IaaS. In: Roedunet International Conference (RoEduNet), pp. 1–6 (2013)
3. Davide Lamanna, D., Skene, J., Emmerich, W.: SLAng: a language for defining service level agreements. In: Proceedings. The ninth IEEE workshop on future trends of distributed computing systems, 2003. FTDCS 2003, pp. 100, 106. 28–30 May 2003 (2003)
4. Ludwig, H., Keller, A., Dan, A., King, R.P., Franck, R.: Web Service Level Agreement (WSLA) Language Specifications vol. 1, IBM Corporation, 2001–2003
5. Keller, A., Ludwig, H.: The WSLA framework: specifying and monitoring service level agreements for web services. J. Network Syst. Manage. **11**(1), 57–81 (2003)
6. Andrieux, A. et al.: Web Service Agreement Specification (WS-Agreement) Open Grid Forum, March 14, 2007
7. Serrano, D., Bouchenak, S., Kouki, Y., Ledoux, T., Lejeune, J., Sopena, J., Arantes, L., Sens, P.: Towards QoS-Oriented SLA Guarantees for Online Cloud Services. CCGRID 2013, pp. 50–57
8. Bianco, P., Lewis, G.A., Merson, P.: Service Level Agreements in Service Oriented Architecture Environments. Technical Note CMU/SEI-2008-TN-021, September 2008
9. Zulkernine, F.H., Martin, P.: An adaptive and intelligent SLA negotiation system for web services. IEEE Trans. Serv. Comput. **4**(1), 31–43 (2011)
10. Parejo, J.A., Fernandez, P., Ruiz-Corts, A., Garcia, J.M.: SLAWs: Towards a conceptual architecture for SLA enforcement. In: Proceedings of the 2008 IEEE Congress on Services—Part I (SERVICES '08), pp. 322–328. IEEE Computer Society, Washington, DC (2008)
11. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: Monitoring, prediction and prevention of SLA violations in composite services. In: IEEE International conference on WebServices, pp. 369–376 (2010)

Applications of Big Data

Hareesh Boinepelli

Abstract Over the last few decades, big business houses in various disparate areas have been accumulating data from different departments in various formats and have been struggling to correlate the datasets and make any valuable business decisions. The key stumbling block has been the inability of the available systems to process large data when the data are part structured and part unstructured. As witnessed in the previous chapters, the technology strides made over the last few years have broken the stigma of processing large datasets and have enabled mining and analysis of large data. Corporations in the data warehousing space have seen this trend as the next big opportunity to help their clients mine their historical data and help further their businesses in terms of adding strategic and tactical value based on the insights gained from their accumulated data over decades. In this chapter, we will see typical examples of how different businesses analyze their data and enhance their business objectives. We will present some examples in the fields of financial services, retail, manufacturing, telecommunications, social media, and health care.

Keywords Basket analysis · Fraud detection · Customer churn · Path analysis · Predictive analysis · Sentiment analysis · Social networking analysis · Sessionization · Graph analysis · Data visualization · K-means clustering

1 Introduction

All the major corporations face a highly competitive environment with constant pressure to increase the profitability by identifying avenues for operational efficiencies and at the same time keeping the business risk to a minimum. All of the big businesses have realized the importance of analyzing loads of historical data that

H. Boinepelli (✉)
Teradata India Pvt. Ltd., Hyderabad, India
e-mail: hareeshkb@gmail.com

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5_7

they have collected over the years, and analysis of this data has become an integral part of making strategic business decisions for these corporations. There is a big push to setup an integrated data management systems and utilize the business intelligence and analytic techniques for improving their businesses.

Over the recent past, big data analytics has found its ways into multiple applications in diverse areas with widespread interest from both the academia and industry. Although this area has made significant progress over the last decade or so, many more challenging problems still exist and finding avenues to new and complex problems in this growing market is ongoing. Various techniques in modeling, statistical analysis, data mining, and machine learning are used to forecast the future events and predict customer behaviors and then proactively act on them to safeguard and enhance business objectives.

In the following sections, we will give a high-level overview of the challenges faced by different industries and how big data are being used to solve them in their respective market segments. Even though big data analytics has the potential in multiple industry domains, we will restrict ourselves to only a few, namely banking and finance (Sect. 2), retail (Sect. 3), manufacturing (Sect. 4), telecommunications (Sect. 5), social media (Sect. 6), and health care (Sect. 7).

2 Big Data Reference Architecture

Figure 1 shows the high-level architecture framework [1] of a typical big data analytics system that includes the following components

1. Acquisition of data from various sources,
2. Infrastructure to do data transformations,
3. Store the data into multiple repositories,
4. Running through high-performance analytic engines, and
5. Reporting and visualization toolset.

Sources of data could be from operational systems which have a nice structure to it (schema/tables/columns/etc.) or can be unstructured such as social media data, click stream data, event logs, and multimedia data. Most of the structured data are stored in the traditional data warehousing environments and the semi-structured and non-structured data on Hadoop clusters. Data are distributed to downstream systems, such as data marts and analytic engines of various types, where the end users can query using SQL-based reporting and analysis tools. Depending on the application, various analytic techniques such as correlation analysis, pattern and trend analysis, collaborating filtering, time series analysis, graph analysis, path analysis, and text analysis are performed before presenting the data on the dashboards using various visualization techniques. In-depth coverage of these components is presented in the previous chapters.

Teradata and IBM are two of the many vendor companies that provide solutions based on the above reference architecture. Figure 2 shows the big data analytics

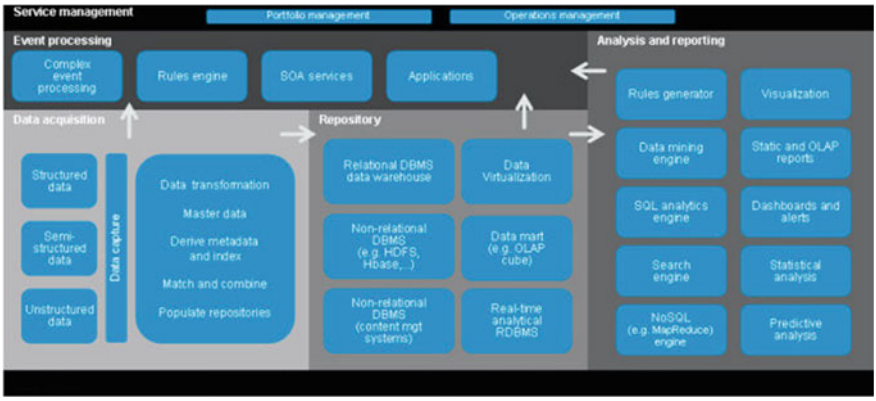

Fig. 1 Big data infrastructure architecture

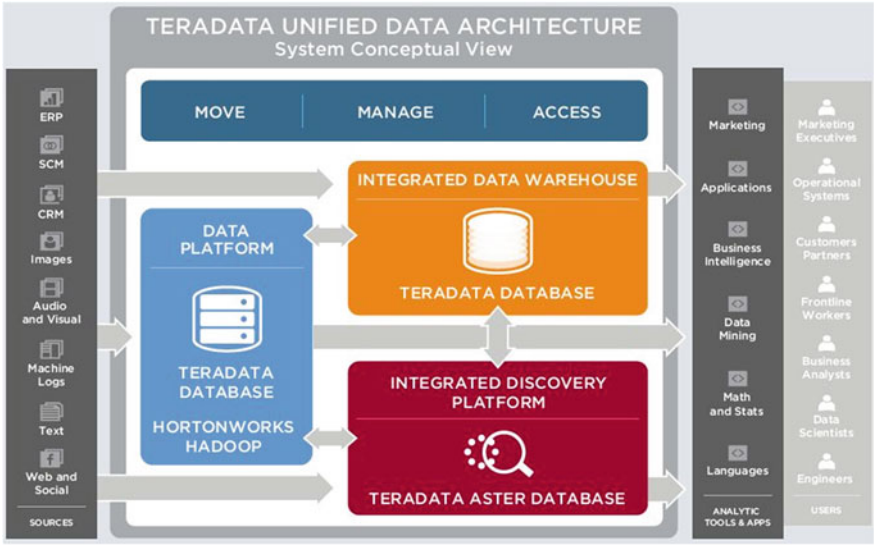

Fig. 2 Teradata unified data architecture

platform from Teradata called the Unified Data Architecture platform [2] with the capabilities to

1. Capture and transform data from variety of sources that are structured, semi-structured, or unstructured data.
2. Ability to process huge data volumes through the use of Hadoop with the data discovery and integrated data warehouse.

3. Support for a number of prepackaged analytic functions in categories such as path analysis, cluster analysis, statistical analysis, predictive analysis, text analysis, relational analysis, and graph analysis.
4. High scalability and performance.

More details on the big data analytics solution can be gathered from the Ref. [2] provided.

Although the reference architecture in Fig. 1 captures the complete set of capabilities required for any big data application, it needs to be noted that not all subsystems represented are required for every application. In the following sections, we will present the frameworks and its components for industry-specific application.

3 Applications in Banking and Financial Industries

Massive amounts of data are being generated by the banking and financial industries through their various service offerings such as checking/savings accounts, mobile banking, credit and debit cards, loans, insurance, and investment services. Most of these data are structured data. Also, most of these organizations have set up their presence online for better serviceability and marketing through which lots of data are collected. As indicated in Fig. 3, some of the channels include

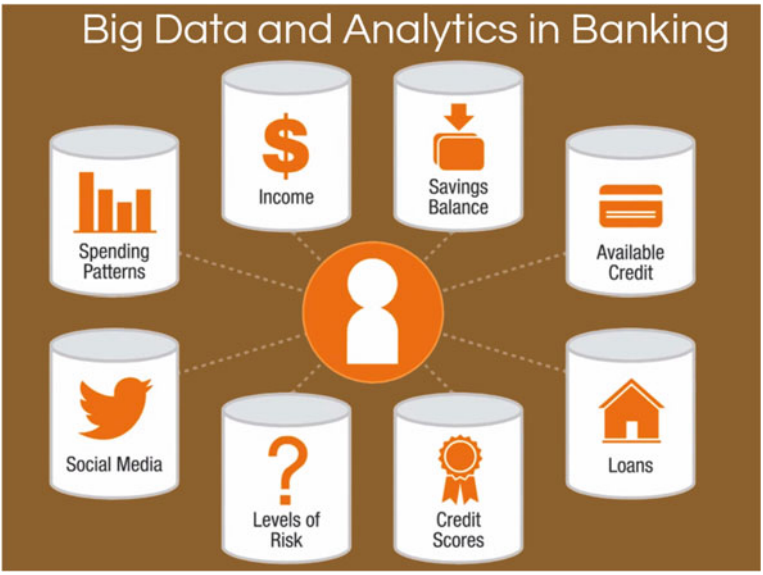

Fig. 3 Big data analytics in banking industry [3]

- Customer interaction through e-mails and chat logs;
- Social networks through tweets and Facebook feeds/posts; and
- Semi-structured data through Web logs and customer reviews.

Most of the data collected are unused, and the industry is looking to various new technologies in data mining and business analytics to help understand and identify customer needs and offer new services which will enhance their business opportunities and increase their margins and profitability. The industry is also looking for solutions in risk management and fraud detection which will help minimize the business exposure. Another area of interest for the industry is on the strategies of retaining customers.

In the following sections, we will cover how big data analytics is applied to the few of the most important areas in more detail.

3.1 Fraud Detection

Various surveys and studies [4] indicate that banking and financial services industry is the victim of the most of the fraud cases among various industries. Following are some of the widely known frauds in the banking industry:

1. Online Banking Fraud: Involves fraudsters taking over the access to the victim's account and performing transactions which siphon the funds out of the accounts.
2. Card Fraud: Involves fraudsters stealing the card information and transact fraudulent transactions.
3. Insider Fraud: Involves fraud by bank's employees.
4. Money Laundering: Crime involving transactions with mainly foreign banks to conceal the origins of illegally obtained wealth.

The traditional approach of sifting through the reports manually and applying various rules is only useful for compliance process and not for detecting fraud and stopping losses. The financial industry requires real-time fraud detection to effectively identify the fraudulent transactions real time and stop them from executing [5].

The key element of fraud detection is the use of analytics to detect patterns of fraudulent behavior. This requires clear understanding of the customer's past behavior in terms of the nature of the transactions so that the distinction of fraudulent versus non-fraudulent transactions can be made effectively by analyzing the transaction against the customer profile which may also include a risk score. This process of scoring the transactions needs to account for the unpredictable nature of transaction activity with varied customer base which includes normal customers and criminals.

Hence, the fraud detection involves a 2-step process which includes

1. Creating the customer profiles based on the historical transactions and identifying the patterns of transactions that lead to fraud
2. Using these customer profiles to catch any outliers or map the transaction sequences/events to the pre-defined fraud patterns and flag any probable fraudulent transactions

Building of customer profiles involves usage of statistical techniques through calculation of statistical averages, min/max values, standard deviations, etc., on the historical transactions to capture the typical transactions mix. Another aspect of the customer profile is capturing the relationships with whom the transactions take place. Graphing techniques [6] are used to capture the network of relationships by mapping transactions between customers along with the modes of payment. Figure 4 captures the flow of creating these customer patterns and profiles.

Figure 5 presents the flow of real-time fraud detection and isolation during the execution of a transaction. If the submitted transaction does not fit the profile of the customer in terms of the transaction amount, transaction connection, etc., it is flagged off for next level of investigation. Statistical outlier detection based on the historical statistics in the customer profile is one technique to detect the suspect transaction.

Pattern analysis [7] on the transaction events and comparing against the pre-recordings of patterns of fraudulent activity is the popular technique used to catch any fraudulent customer activity real-time. Time series analysis techniques [8] are also employed to identify whether the customer activity fits the well-defined business rules of fraudulent activity.

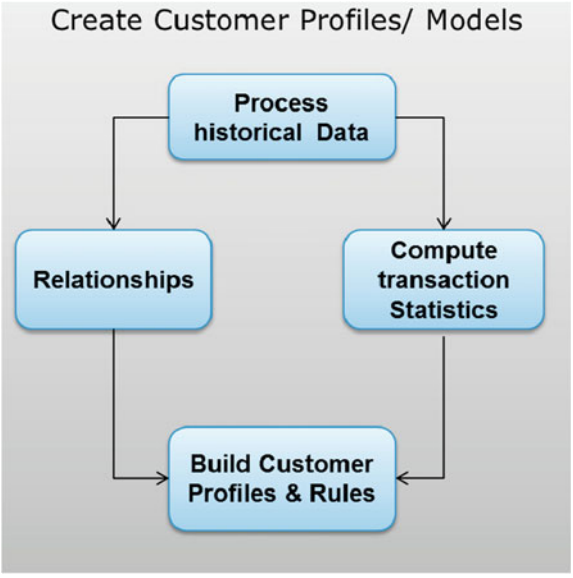

Fig. 4 Building customer profile for fraud detection

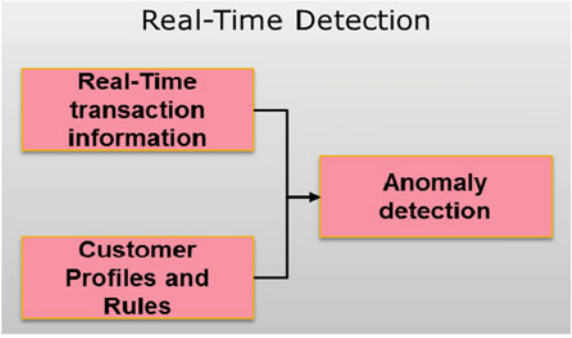

Fig. 5 Real-time fraud detection using customer profile

3.2 Money Laundering

Money laundering is a more sophisticated fraud, and detecting it requires setup of more complex and integrated multi-dimensional database system with data from different sources such as bank transactions, and law enforcement databases.

Complex networks of relationships are identified by linking data gathered over sources such as phone, e-mail, Web browsing, and travel records thereby identifying the connections between known and unknown players. Graphs of interconnected banking institutions, customer accounts, and transactions at certain times using certain devices are used to help identify potential money laundering schemes. Data analysis techniques such as clustering, classification, outlier identification, and data visualization tools [9] can be used to detect patterns in transactions involving large amounts between specific set of accounts. These techniques have the potential to identify key associations and patterns of activities that help identify suspicious cases for further investigation.

3.3 Risk Analysis

In general, banks and financial institutions have mechanisms for quantifying risk and mitigating it. Various market forces play their part in various types of risk, and a clear understanding of the potential losses for all the possible situations is needed.

Out of the various types of risks in the financial industries [10], prediction of default on loan accounts and credit card accounts is one of the important areas due to the enormity of these accounts and mitigating losses from these accounts becomes fundamental to the business. Prediction of various factors that are responsible for defaults is done using data mining techniques related to attribute selection and attribute relevance (Fig. 6). Based on the outcomes of the analysis, banks can identify customers who belong to low-risk category or offer favorable payment plans to the customer.

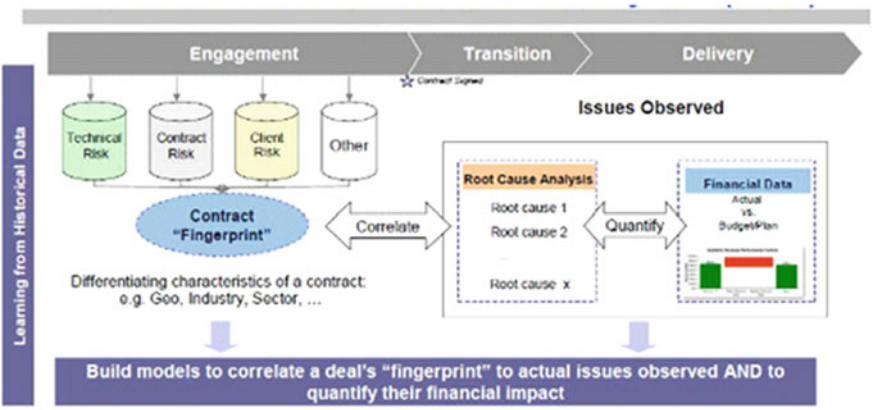

Fig. 6 Financial risk analytics framework

4 Applications in Retail Industry

Most of the big name retailers such as Costco, Wal-Mart, Target, and Amazon use big data for various operations including inventory management, product recommendations, tracking customer demographics, and tracking and managing the side effects of product recalls. Some retailers have used this customer data to improve the quality of service and enhance the customer loyalty.

4.1 Recommendation of Products

One of the well-known strategies that the retail companies employ to increase their revenues is to recommend products to the customers that they might be interested in based on what the customer is currently purchasing. This is typical of an e-retailer whose back-end systems run product recommendation engines by cross-referencing the items among sales records from various customers who may have purchased the same item earlier.

The retailers who have presence online and offline (brick and mortar) can use the data collected across multiple channels and come up with the purchase patterns and recommend products online. Path and pattern analytics are used on the historical customer purchasing behavior across multiple channels to generate high-quality recommendations. Collaborative filtering techniques are used on a customer's historical purchases and searching patterns, and comparing against other customers to predict the next recommendation.

Collaborative filtering technique is used in the recommendation systems by the e-retailers [11] such as amazon for recommending products, and the same techniques are used by the movie recommendation engine that Netflix uses. These same

techniques are employed offline in coming up with the weekend fliers, advertise on sales receipts, or coming up with the promotions by bundling items in order to promote sales.

4.2 Predicting Trends

Retailers collect huge amount of data about customers including location, gender, and age from their various transactions. Mining of retail data can help identify customer buying patterns and trends which will in turn help identify customer needs for effectively plan for product promotions and attract more customers and increase revenues/profits [12]. Multi-dimensional analysis and visualization tools of the dataset can be used for the prediction which could help with the company planning of the logistics/transportation of the needed goods.

5 Applications in Manufacturing

Manufacturing companies have become highly competitive across the world with the margins of doing business going down every day. The manufactures are always on the lookout for optimizing costs in running factories thereby increasing the margins. Big data analytics is helping in a couple of areas as discussed below [13].

5.1 Preventative Maintenance

In the automated world of manufacturing, sensors are used everywhere in monitoring the assembly line so that the failures can be quickly identified and fixed to minimize the downtime. The root cause of plant failure could be due to one or more of the numerous possible parameters spread across different subsystems linking the assembly line. Huge amount of sensor data, all unstructured data, is accumulated over the running of the manufacturing plant. Historical maintenance records for the various subsystems are also gathered in the semi-structured format. And logs related to the productivity relative to the peak capacity are also gathered along with the maintenance records and sensor data.

Time series analysis of the various subsystems based on their respective sensor data and performing pattern matching against the failure case is used for catching the potential failures. Also, path analysis and sessionization techniques are used to capture the critical events based on correlations between the sensor readings, historical maintenance records, and logs to predict the probable failures. This helps take preventative measure to keep the line running for extended period of time without interruptions and also help with improving the safety of running the operations.

5.2 Demand Forecasting

The most important factor in businesses which are tied to manufacturing industry is to optimally use the resources where the day-to-day orders keep changing dynamically. Forecasting sales and the time frame when they happen will help plan for timely acquisition of raw materials, ramping up/down production, manage warehousing, and shipping logistics. In the short term, overestimating demand leaves the manufacturer with unsold inventory which can be a financial drain and underestimating implies missed opportunities. In the long term, demand forecasting is required to plan for strategic investments and business growth. Hence, for effective running of a business with maximum profitability requires a solid forecasting system.

Time series analysis is a popular forecasting technique used to predict future demand and is based on the historical sales data. This simplistic method in generating future forecast is inaccurate when the environment is dynamic with factors such as changing customer requirements and impact of competition.

Predictive modeling [14] is a more advanced and accurate forecasting technique which has the capability to factor in all the variables impacting future demand. The model also facilitates with testing various scenarios and helps understand the relationship between the influencing factors and how they affect the end demand.

6 Applications in Telecommunications

With the expansion of telecommunications services across the globe, the telecommunications industry is trying to penetrate various markets with diverse service offerings in voice, video, and data. With the development of new technologies and services across multiple countries, the market is growing rapidly and has become highly competitive between various service providers.

Figure 7 shows the big data analytics framework for telecom domain that is used as the basis for formulating the strategies for better business. Business insights for different departments are mined based on the data collected across various platforms. Some of these include

1. Customer/subscriber data: Personal information and the historical relationship with the provider.
2. Usage patterns.
3. Customer service records: Service-related complaints or request for additional services and feedback.
4. Comments on social media.

In the following sections, we will review a couple of areas where the industry is trying to identify avenues for revenue preservation and generation using big data.

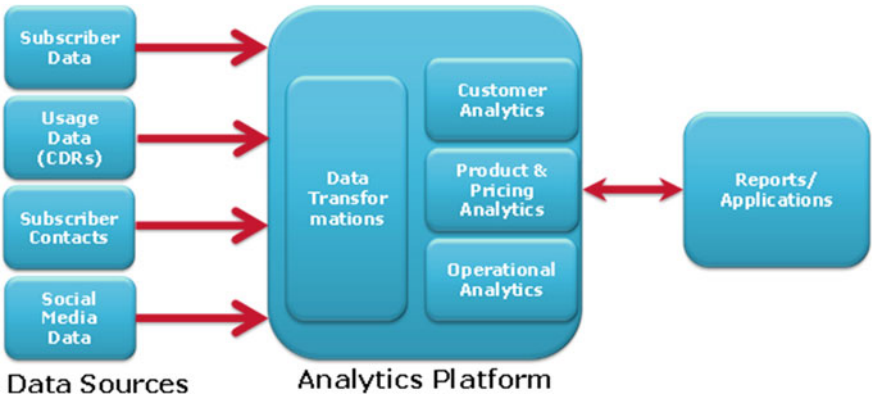

Fig. 7 Big data framework: telecommunications domain

6.1 Customer Churn

It is well known that the customer churn is a big headache for the all the telecom service providers. Customers leaving the existing service provider and signing up with a competitor cause revenue/profit/loss. It is a costly affair to acquire new customers with new promotions and has an effect of increased marketing costs which in turn has the effect on profitability.

Studies have shown that proactively identifying the key drivers for churn and developing strategies in retaining customers help minimize the revenue and profit erosion. The service provider can then focus on upgrading the underlying network infrastructure for better quality of service and better support services to retain and grow the customer base.

Various statistical data analysis techniques [15] are traditionally used to identify the triggers to customer defections and apply these triggers to the existing subscribers and evaluate chances of canceling their service and moving to another provider. Using customer behavior data collected on different channels such as calling profiles, customer complaint calls to the call centers, comments over e-mail, and feedback surveys, better churn prediction can be done to identify high-risk customers. In order to figure out the patterns of events leading to the churn, path analysis techniques are used. Using the Naive Bayes classifier for text analysis, a model is built to identify the high-risk customers.

Another popular technique used is graph engines [16] to represent connections between users based on the call detail records and then identify communities and influencers within the user communities. One of the remedial actions is to engage the high probable churn customers and offer incentives and extend the contracts for additional time period.

6.2 Promotion of Services

Telecom service providers are constantly looking to increase their revenues by recommending auxiliary services to customers that they might be interested in based on the current subscription plan. This is done either through cross-referencing with the customers with similar profiles. Another strategy is to promote the next best plan for a small incremental price. The data analytic techniques used for these recommendation engines [17] are fundamentally same as used in e-tailing business.

7 Applications in Social Media

Online social media is growing leaps and bounds as witnessed by the growth in the active user base and the amount of data that it generates. Sites such as Facebook, Twitter, Google+, LinkedIn, Reddit, and Pinterest are some of the most popular online hangout places these days. Even big corporations have started using social media as a business channel by having their presence through Facebook accounts, Twitter accounts, YouTube channels, and company blogs to name a few. The inherent openness of the social media to everyone to hear and voice their opinions and build new relationships has paved way to the creation of wealth of data. This has caught the attention of data scientists in exploring the use of social media in various areas.

Figure 8 illustrates a typical framework for applications involving social media analysis [18] with the various components highlighted. Social media analysis involves gathering and analyzing huge data that the social media generate to make

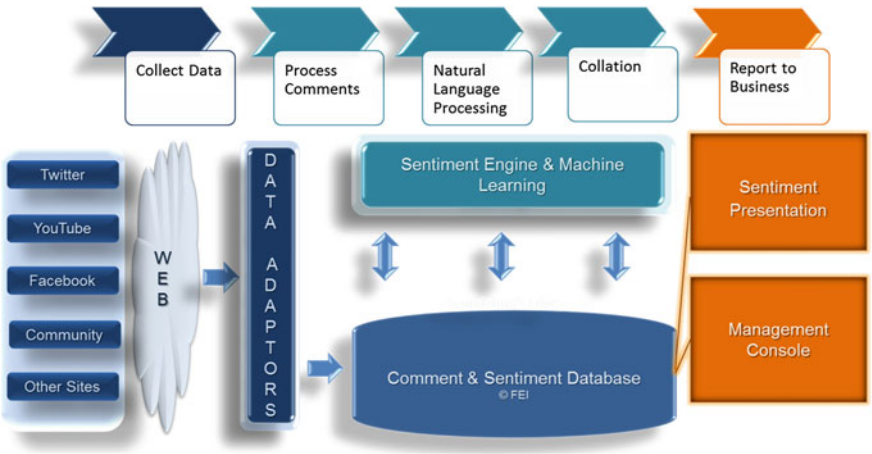

Fig. 8 Typical framework for social media analysis

business decisions. The goals of this analysis include strategies on product marketing, brand promotion, identifying new sales leads, customer care, predicting future events, foster new businesses, etc.

The work flow includes the phases of data collection, data transformation, analysis, and presentation dashboard. The social media data consist of mostly unstructured data ranging from blog posts, and its comments link to Facebook friends, tweets/retweets, etc. Based on the specific objective of the analysis, the data filtering is performed on the raw data which are then analyzed for the understanding and predictions on the structure and dynamics of community interactions. Sentiment analysis [19] and reputation management are few of the applications where natural language processing is applied to mine blogs/comments. Graph analysis techniques are applied to identify the communities and influencers within the communities.

7.1 Social Media Marketing

In social media, it is well known that different people have different levels of influencing others based on various factors, the prime being the number of connections he/she has. Representing the user-to-user connections in a graph as shown in Fig. 9 helps identify the key influencers [20] who then can be targeted with the product advertisements. This has been shown to help in creating brand awareness and facilitate viral marketing of new products.

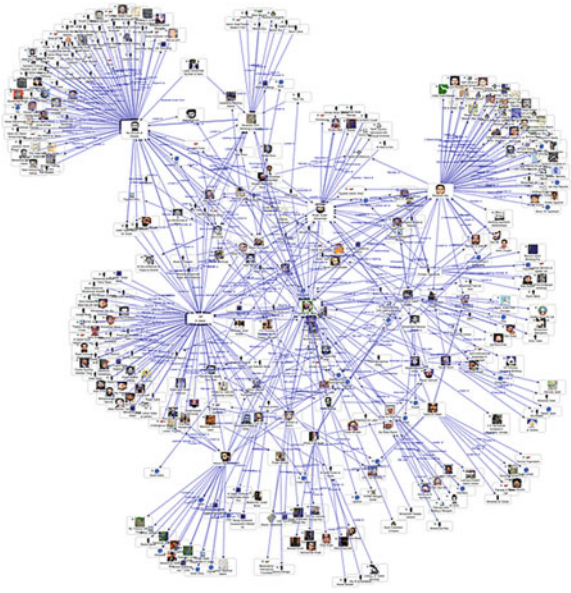

Fig. 9 Identifying influencers using K-means clustering techniques

Also, by offering incentives to the customers with most influence in a community, and leveraging his/her influence, customer churn can be contained.

7.2 Social Recommendations

Graph and link analysis is used extensively in professional social networking sites such as LinkedIn to identify and recommend other professionals that a user may be interested in establishing connection based on the existing connection mix.

Reddit site uses similar analysis of graphs built using the articles/posts and the interests of the users reading them to recommend new articles/posts to users with similar interests. List of articles in the database, user profiles, and profile of user interests are analyzed to come up with the recommendations across multiple users. Analytic technique used to organize data into groups or clusters based on the shared user attributes is K-means clustering algorithm.

8 Applications in Health care

Application of big data analytics is gaining importance in the health care industry due to the characteristics of the business involving huge dataset of customer electronic health records, the goal to deliver service at minimum cost, need for critical decision support, etc.

Figure 10 shows a typical framework for applications in health care industry capturing various components of the typical platform. Huge amounts of health care data are collected that includes clinical data such as laboratory records, doctor's notes, medical correspondence, electronic medical records (EMRs), claims, and finance. Advanced analytics on this data is used to improve customer care and

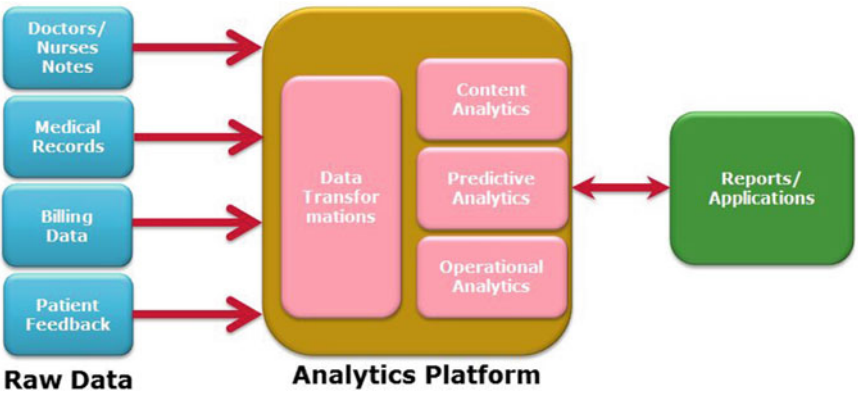

Fig. 10 Analytics framework for health care

results, drive efficiencies, and keep the costs to minimum. Analytics is also used to do a thorough investigation and detect adverse side effects of drugs which then enable quick recall of those drugs.

Following are a few examples of big data analytics in health care industry:

Finding New Treatments
National Institutes of Health [21] in USA maintains the database of all the published medical articles on various health topics and has opened up access to all the interested researchers. This dataset of documents is huge, and mining meaningful information is a challenge.
Researchers have used the semantic searches on this database to uncover new relationships between therapies and outcomes. Graph analysis [6] is used by researchers focusing on cancer who discovered that immunotherapy performs better than chemotherapy in certain cases of cancer. Visualization techniques [22] are used to find the correlations quickly.
Multi-Event Path to Surgery
Applying path and pattern analysis techniques to the data obtained from the patient records with the different procedural codes, it is possible to identify sequence of events leading to expensive surgeries. Using this info, better preventative care can be provided to avoid surgery and help reduce the medical costs.
Reduction in Claim Review
Evaluation of medical claims involves looking at doctor notes, medical records, and billing procedural codes which is time consuming and laborious process especially in cases where the treatments were complex involving multiple procedures. In order to reduce this manual effort, text analytic techniques, namely FuzzyMatch, are employed to determine inaccurate billing practices as well as potential abusive, fraud, or wasteful activity.

9 Developing Big Data Analytics Applications

The framework for a big data analytics application is conceptually similar to that of a traditional business intelligence application with the following differences.

- The main difference lies in how the structured and unstructured data are stored and processed as demonstrated in the big data framework chapter (Chap. 2). Unlike the traditional model where the BI tool is run on the structured data on mostly a stand-alone node, the big data analytics application, in order to process the large scale of data, breaks down the processing and executes across multiple nodes accessing the locally available data.
- Unlike the classical BI tools, the big data analytics tools are complex and programming intensive and need to be able to handle data residing in multiple formats.

- A different application development framework that takes advantage of running lots of parallel tasks across multiple nodes.

Development of big data applications involves awareness to various platform-specific characteristics such as

- Computing Platform—A high-performance platform which includes multiple processing nodes connected via a high-speed network;
- Storage System—A scalable storage system to deal with massive datasets in capturing, transforming, and analyzing;
- Database Management System;
- Analytics Algorithms—Develop from scratch or use the third-party open-source or commercial software suites
- Performance and scalability needs

Other than the knowledge of general platform architecture to which the targeted applications are developed, big data application developers need to be exposed to the popular big data application frameworks supported on the platform. The most popular software suite/framework that enables big data application development is called Apache Hadoop which is a collection of multiple open-source projects. Hadoop framework comprises of various utilities on top of the Hadoop distributed file systems (HDFS) and a programming model called MapReduce as described in Chap. 2 along with various other infrastructure components supporting the framework. These include PIG, HIVE, JAQL, and HBase.

Building sophisticated analytic applications requires the expertise of the data mining techniques and algorithms on top of the platform architecture and framework for which these applications are intended for. The implementations of the popular algorithms are available as open source and some are proprietary implementations. Examples of the open-source implementations include

- R for statistical analysis,
- Lucene for text searches and analysis, and
- Mahout Library—A collection of widely used analysis algorithms implemented using the map/reduce paradigm on Hadoop platforms are used for building applications. These include collaborative filtering, clustering algorithms, categorization, text mining, and market basket analysis techniques.

Analytic function implementations provided by third-party vendors or the open source have a specific programmers interface. One of the main challenges to the application developers is the complexity involved in some of the key elements of incorporating the APIs in the application. These include the following:

- Integration of open-source packages into the overall system and how the libraries are exposed to the developers.
- Support in acquiring the required input for the functions from database tables, raw files, etc.
- Support in saving function results into tables, temporary buffers, files, etc., and

- Ability to cascade multiple analysis methods in a chain to have an output of one function as an input of the next to simplify the implementation of the overall application.

The commercial big data platform solutions offered by corporations such as IBM [23] and Teradata [2] include their own proprietary application frameworks. Integration of various open-source packages and implementation/support of proprietary packages where the open-source library lacks the functionality are the key for the sale ability of the platform. These integrated commercial solutions promote the ease of use compared to the challenges using the open-source solutions as one of the strengths when marketing their platforms.

10 Conclusions

Majority of large companies are dealing with the problem of finding value in the huge amount of data that they have collected over the years. Depending on the market segment the business is addressing, different data analytic techniques are used to identify new markets, optimize operational efficiencies, etc., so as to increase the bottom line.

In this chapter, we tried to present handful of areas in different industries where the applications of big data and analytics have been effectively used. Identifying new areas and exploring new solutions will be the area of focus for the future. Corporations have started seeing value in putting dollars in data-driven strategies, and the realization that the big data strategy is a key component of business, in order to stay competitive, is gaining ground.

Exercises

1. Write an application to recommend new music labels for users based on the historical listening profiles of the user base. The basket analysis can use the dataset that is available at http://ocelma.net/MusicRecommendationDataset/lastfm-360K.html.
2. Yahoo! Messenger is a popular instant messaging application used by many users to communicate to their friends. A sample dataset of so-called friends graph or the social network is available at http://webscope.sandbox.yahoo.com/catalog.php?datatype=g titled "Yahoo! Instant Messenger Friends Connectivity Graph." Write an application to identify the top 5 users who have most influence in the given social network.
3. Visualize the social network of users for the dataset indicated in Exercise 2 above. Use the open-source graph analysis tool called Gephi for this

visualization (available at http://gephi.github.io/users/download/. Use the quick start tutorial at http://gephi.github.io/tutorials/ to render and identify the communities for the above dataset.

4. Microsoft Corp. has published a dataset which captures the areas of www.microsoft.com that users have visited over a one-week time frame. This dataset is freely available to users at http://kdd.ics.uci.edu/databases/msweb/msweb.html.
   Write an application to predict the areas of www.microsoft.com that a user can visit based on data on what other areas he or she visited.
5. Using sentiment analysis concepts/algorithms gained in the earlier chapters, analyze the movie reviews/feedback data available at http://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data to build a model to predict the positive, negative, and neutral sentiment of the reviewers. Use 75 % of the data for the model and the remaining 25 % of the data to validate the model.
6. Using R open-source statistical and data analysis tools, write an application to predict the movement of a stock belonging to DOW Jones. The sample dataset is provided at https://archive.ics.uci.edu/ml/machine-learning-databases/00312/
7. Demonstrate with an example how to build a prediction model based on Naive Bayes for text. And then demonstrate with an example using the built model to do text prediction.

References

1. Big Data Calls for New Architecture, Approaches. http://tdwi.org/articles/2012/10/24/big-data-architecture-approaches.aspx
2. Big Data: Teradata Unified Data Architecture in Action. http://www.teradata.com/white-papers/Teradata-Unified-Data-Architecture-in-Action/
3. How Bigdata can help the banking industry: A video post. http://www.bigdata-startups.com/BigData-startup/big-data-analytics-banking-industry-video/
4. Global Fraud Study: Report to the Nations on Occupational Fraud and Abuse, Association of Certified Fraud Examiners http://www.acfe.com/uploadedFiles/ACFE_Website/Content/documents/rttn-2010.pdf
5. Whitepaper from ACL: Fraud detection using Data Analytics in the Banking Industry. http://www.acl.com/pdfs/DP_Fraud_detection_BANKING.pdf
6. Diane, J.C., Holder, L.B.: Mining Graph Data. Wiley, New York (2007)
7. Jean-Marc, A.: Data Mining for Association Rules and Sequential Patterns. Springer, Berlin (2001)
8. Nettleton, D., Kaufmann, M.: Commercial Data Mining Processing, Analysis and Modeling for Predictive Analytics Projects, Elsevier, North Holland (2014)
9. Analytics in Banking Services. http://www.ibmbigdatahub.com/blog/analytics-banking-services
10. IDC White Paper: Advanced Business Analytics Enable Better Decisions in Banking. http://www.informationweek.com/whitepaper/Customer-Insight-Business-Intelligence/Analytics/idc-white-paper-advanced-business-analytics-enabl-wp1302120869
11. Su, X., Taghi, M.K.: A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009**, 421425 (2009)

12. Das, K., Vidyashankar, G.S.: Competitive advantage in retail through analytics: developing insights, creating value. Inf. Manage. http://www.information-management.com/infodirect/20060707/1057744-1.html (2006)
13. Big data: The next frontier for innovation, competition, and productivity. http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
14. Makridakis, S., Wheelwright, S., Hyndman, R.J.: Forecasting: Methods and Applications. Wiley, New York (1998).
15. Analytics in Preventing Customer Churn. http://www.teradata.com/Resources/Videos/Prevent-Customer-Churn-Demonstration/
16. Richter, Y., Yom-Tov, E., Slonim, N.: Predicting customer churn in mobile networks through analysis of social groups. In: SDM, Columbus (2010)
17. Big Data Analytics Use Cases—Ravi Kalakota. http://practicalanalytics.wordpress.com/2011/12/12/big-data-analytics-use-cases/
18. Foundations Edge: Media Analysis Framework. http://www.foundations-edge.com/media_analytics.html
19. Ogneva, M: How companies can use sentiment analysis to improve their business (2010). http://mashable.com/2010/04/19/sentiment-analysis/
20. Blog postings on Social Media Marketing. http://practicalanalytics.wordpress.com/category/analytics/social-media-analytics/
21. Big Data Disease Breakthroughs—William Jackson: Information Week. http://www.informationweek.com/government/big-data-analytics/big-data-disease-breakthroughs/d/d-id/1316310?
22. Periodic Table of Data Visualization Methods. http://www.visual-literacy.org/periodic_table/periodic_table.html
23. Big Data Technology. http://www.ibm.com/big-data/us/en/technology/

Author Biography

Hareesh Boinepelli is presently an engineering manager at Teradata Corp. Past work experience includes Kicfire Inc., McData Corp., Sanera Systems, and Nortel Networks over the last 15+ years. Experience covers the areas of telecommunications networks, storage networks, data warehousing, data analytics, etc. Academic qualifications include Ph.D. from Telecommunication Research Center, ASU, Arizona, M.E (ECE) from IISc, Bangalore, and B.E. (ECE) from College of Engineering, Osmania University.

Index

© Springer India 2015

H. Mohanty et al. (eds.), *Big Data*, Studies in Big Data 11,
DOI 10.1007/978-81-322-2494-5