Big Data Computing in Clouds– Data Aware Scheduling and Extended MapReduce for Scientific Analytics

A thesis submitted during 2016 to the University of Hyderabad in partial fulfillment of the award of a **Ph.D. degree** in School of Computer and Information Sciences

by

Raghavendra Kune

School of Computer and Information Sciences

University of Hyderabad

(P.O.) Central University, Gachibowli

Hyderabad - 500 046

Telangana

India

April, 2016.

CERTIFICATE

This is to certify that the thesis entitled **"Big Data Computing in Clouds - Data Aware Scheduling and Extended MapReduce for Scientific Analytics"** submitted by **Raghavendra Kune** bearing **Regd. No. 06MCPC11** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under my supervision and guidance, which is a plagiarism free thesis.

The thesis has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Prof Arun Agarwal
Supervisor
School of Computer &
Information Sciences
University of Hyderabad.

Dr. K. Pramod Kumar
Supervisor
ADRIN, 203,
Akbar Road, Tarbund,
Secunderabad.

Dean

School of Computer and
Information Sciences
University of Hyderabad

DECLARATION

I, **Raghavendra Kune,** hereby declare that this thesis entitled **" Big Data Computing in Clouds – Data Aware Scheduling and Extended MapReduce for Scientific Analytics"** submitted by me under the guidance and supervision of **Prof. Arun Agarwal** and **Dr. K. Pramod Kumar;** Co-supervisors **Prof. C. R. Rao** and **Prof. Rajkumar Buyya** is a bonafide research work which is also free from any plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Library is enclosed.

Date:

Name: **Raghavendra Kune**

Signature of the Student

Reg. No: 06MCPC11

Signature of the Supervisors:

DEDICATION

I dedicate this thesis to my parents Sri. Nagappa, Smt. Bhulakshmi, my wife Smt. Sumalatha, and my beloved son Jishnu for their love, support and encouragement.

ACKNOWLEDGEMENTS

I am thankful to God for giving a beautiful life time opportunity – Ph.D journey which is intellectually stimulating and at the same time emotionally exhausting. This journey has taught me a lot and enriched me with life time experiences. Someone has truly said.

Success is a journey, not a destination. The dedication, and consistent performance is often more important for achieving better results.

University of Hyderabad (UOH) has instilled this feeling in me. My supervisors Prof. Arun Agarwal, Dr. Pramod Kumar (ADRIN), Co-supervisors Prof. C. R. Rao, and Prof. Raj Kumar Buyya (University of Melbourne) are always encouraged and inspired me to attain high quality research work and guided me throughout the Ph.d journey. Doctoral Committee members Prof. Hrushikesh Mohanty and Prof. Rajeev Wankar are always with me and revised the work throughout the Ph.D program. My guide and supervisor Prof. Arun Agarwal taught me to reinforce faith in myself while going through difficult times. Prof. Agarwals' domain expertise in Grid, Cloud computing and Image Processing, helped me in expanding my knowledge in related areas of my research work. I would like to express my sincere thanks to Dr. K. Pramod Kumar, who is always an inspiration for me, and his in depth knowledge in cluster scheduling concepts, Remote Sensing Data Processing, and Programming models helped me to carry this journey with a great success.

I would like to express my sincere thanks to Dr. R. Krishnan, former Director of Advanced Data Processing Research Institute (ADRIN), who had readily accepted my request, and agreed to be as guide for my research. However, due to its assignment to Indian Institute of Space Science and Technology (IIST, Trivendrum, India), it had made me to switch over to Dr. K. Pramod Kumar as a

guide for Ph.D journey. I would like to thank Dr. R. Krishnan, for inculcating the mathematical problem solving techniques, and formulating the scheduling problem as Linear Programming models. Later, these foundations made to think over using Genetic approach for solving the scheduling problems for data intensive computing.

I would like to thank my co-supervisors Prof. C. R. Rao, who introduced me to the world of genetic algorithms and its beauty in solving the problems using soft computing and optimization techniques. My co-supervisor Prof. Raj Kumar Buyya, who is always with me, and his guidance in Cloud and Big Data computing, and in depth knowledge of the subject, helped me in all aspects of my Ph.D journey.

I would like to express my sincere thanks to Sri. R. Ramachandran, former Associate Director, ADRIN, who has helped me in getting necessary approvals from the department and encouraged me to pursue the research. I am thankful to Dr. S. Venkat Raman (Group Director, ADRIN), for his constant support during this journey. I would like to thank my Dr. Sudheer Reddy, Scientist/Engineer 'SE', ADRIN, for the conversation we both had about genetic algorithm concepts. I would like to express my gratitude to Smt. Geeta Varadhan (Ex – Director ADRIN), and present director of ADRIN Sri. Santanu Chowdhury, for their kind support, and the encouragement received during my research journey.

I am deeply indebted to my wife Smt. Sumalatha, who made my Ph.D journey smooth to happen. I am indebted to my parents, who had implanted the seed for knowledge quest as a child. My son Jishnu, used to wait for me during nights till I go to bed, and used to wish me during morning, while I go to university, saying "papa bye, saayankala baa" (pappa go to university for thesis writing, comeback in the evening).

I dedicate thesis to my family.

Table of Contents

Abstract

Big Data computing in Clouds is a new paradigm for next generation analytics development, enabling large scale data organization, sharing, and exploration of large volumes rapidly growing variety forms of data using Cloud computing technologies as a back end large scale service-oriented computational infrastructure facility. Advances in information technology and its widespread growth in several areas of business, engineering, medical and scientific studies are resulting in information and data explosion. Knowledge discovery, and decision making from such rapidly growing voluminous data is a challenging in terms of both data organization, access and timely processing, which is an emerging trend known as Big Data computing, a new paradigm which combines large scale compute, new data intensive techniques and mathematical models to build data analytics for intrinsic information extraction. Cloud computing is emerged as service oriented computing model, to deliver infrastructure, platform and applications as services from the providers to the consumers meeting the Quality of Service (QoS) parameters, by enabling the archival and processing of large volumes of rapidly growing data at faster scale based on economy models. Big Data demands huge computing and data resources, and Clouds offer large scale infrastructure, hence both these technologies could be integrated, so as to process large scale Big Data on Clouds infrastructure as back end computing resources. This thesis proposes challenges in integration of both these technologies, and Big Data computing in Clouds as an effective metaphor for the management of large scale data organization and processing in elastically scalable computing and store infrastructures for scientific computing applications. The thesis discusses an architectural framework for Big Data computing in Clouds that supports large scale distributed data intensive applications, Data Aware scheduling model for effectively scheduling the jobs by fetching the data from remote distributed storage repositories using evolutionary genetic approach, followed by an extensions of Hadoop Distributed File System (HDFS) and MapReduce, for distributed data organization and processing for applications of scientific image processing domain. It demonstrates their effectiveness by performing scheduling experiments both in the simulated and real environments using CloudSim and Hadoop clusters respectively.

List of Figures

List of Tables

Abbreviations

NIST	National Institute of Standards and Technology
SMAC	Social, Mobile, Analytics and Cloud
XHAMI	Extended Hadoop and MapReduce Interface for Image Processing Scientific domains
DAGS	Data Aware Genetic Scheduling
HDFS	Hadoop Distributed File System
GA	Genetic Algorithm
RS Big Data	Remote Sensing Big Data
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
MR	MapReduce
CAP	Consistency, Availability, and Partition tolerance
ACID	Atomicity, Consistency, Integrity, and Durability
BASE	Basically, Available, Soft state, and Eventual consistency
EMR	Elastic MapReduce
CCMI	Cloud Computing Management Interface
CS/DMI	Cloud Storage/Data Management Interface
CASI	Cloud Application Services Interface
BDIS	Big Data Infrastructure Services
BDPS	Big Data Platform Services
BDAS	Big Data Analytics Services
IDC	International Data Corporation
IT	Information Technology
TB	Tera Byte
PB	Peta Byte
API	Application Programming Interface
I/O	Input Output
ETL	Extract, Transform and Load

Chapter 1

Introduction

This chapter introduces the context of the research to be presented in this thesis. It starts with an in introduction to the general area of Big Data computing, and discusses the motivation and challenges for integrated Cloud and Big Data computing known as Big Data Computing in clouds. Then, it presents a short view of system architecture, layered framework for Big Data computing in Clouds and elements in the framework, scheduling model for the decoupled computing and storage resources, problem statement, motivation for the scheduling model, extended MapReduce and Data organization model for scientific large scale data problems, and presents the primary contributions of this research. The chapter ends with a discussion on the organization of the rest of this thesis.

1.1 Inspiration for Big Data and Clouds

Big Data computing is an emerging data science paradigm of multi dimensional information mining for scientific discovery and business analytics over large scale infrastructure. The data collected or produced from several scientific explorations and business transactions often require tools for effective data management, analysis, validation, visualization and dissemination, preserving the intrinsic value of the data [1] -[3]. SMAC (Social, Mobile, Analytics, and Cloud) driven growth is enabling the large scale multi dimensional digital data growth worldwide, and IDC [4] report predicted that there could be 40 folds data growth from 2012 to 2020 and expected to double every two years as per the digital universe data growth cycle depicted in Figure 1.1. Since, the advancements in processor, storage, and networking are enabling the resources at considerable low prices and cloud computing technologies are enabling for on demand utility computing for a large scale data preservation and analysis over distributed computing infrastructures based on Service Oriented Architectures (SOA).

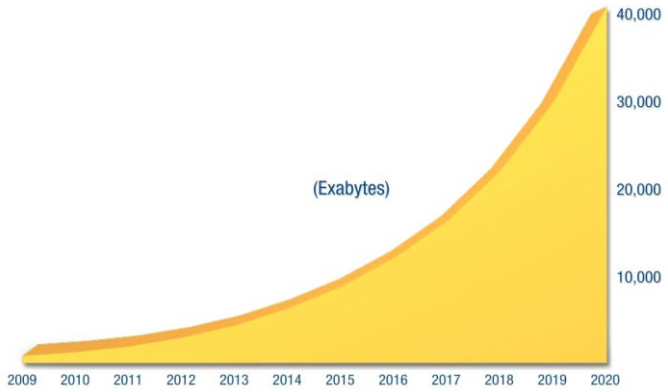


Figure 1.1 Data Growth Cycle (Source: IDC [4])

The wide spread growth of digital content in several fields of social networking, business processing, and scientific analysis are producing the voluminous data in several formats like video, audio, images, relational, text, xml etc. Such data is rapidly growing, often leading to difficulties in storing, and processing in the stipulated time frame using conventional data bases or ware housing systems. To address these problems, Big Data is an emerging as a new technology, to mine untapped information of large scale variety forms of rapidly growing data, using data intensive analytics platforms and computing paradigm over a large scale distributed compute and storage resources.

The data collected or produced from several applications of scientific and business areas like genome study, theoretical physical sciences, weather forecasting, remote sensing, web log mining, business process analytics etc., often require tools for effective data management, analysis, validation, visualization and dissemination, preserving the intrinsic value of the data. Gartner hype cycle for emerging technologies, 2014 as on August is depicted in Figure 1.2, depicts Big Data computing as one among the peak technologies for future research and innovations for the strategists and planners to leverage Big Data Analytics and new technologies to work on large data of various formats for information retrieval and decision making.

Conventional data warehousing systems in general have the pre determined analytics over the abstracted data which is cleansed and transformed into another database known as data marts- which are periodically updated with the similar type of rolled-up data. However, Big Data systems are for non predetermined analytics, hence no data cleansing and transforming

techniques are used. In practice, data warehousing systems organize the data from enterprises financial systems, customer marketing systems, billing systems, point of sale system etc., however, would not capture the operational databases like click streams logs, sensor data, location data from mobile devices, customer support emails and chat transcripts, and surveillance videos etc.

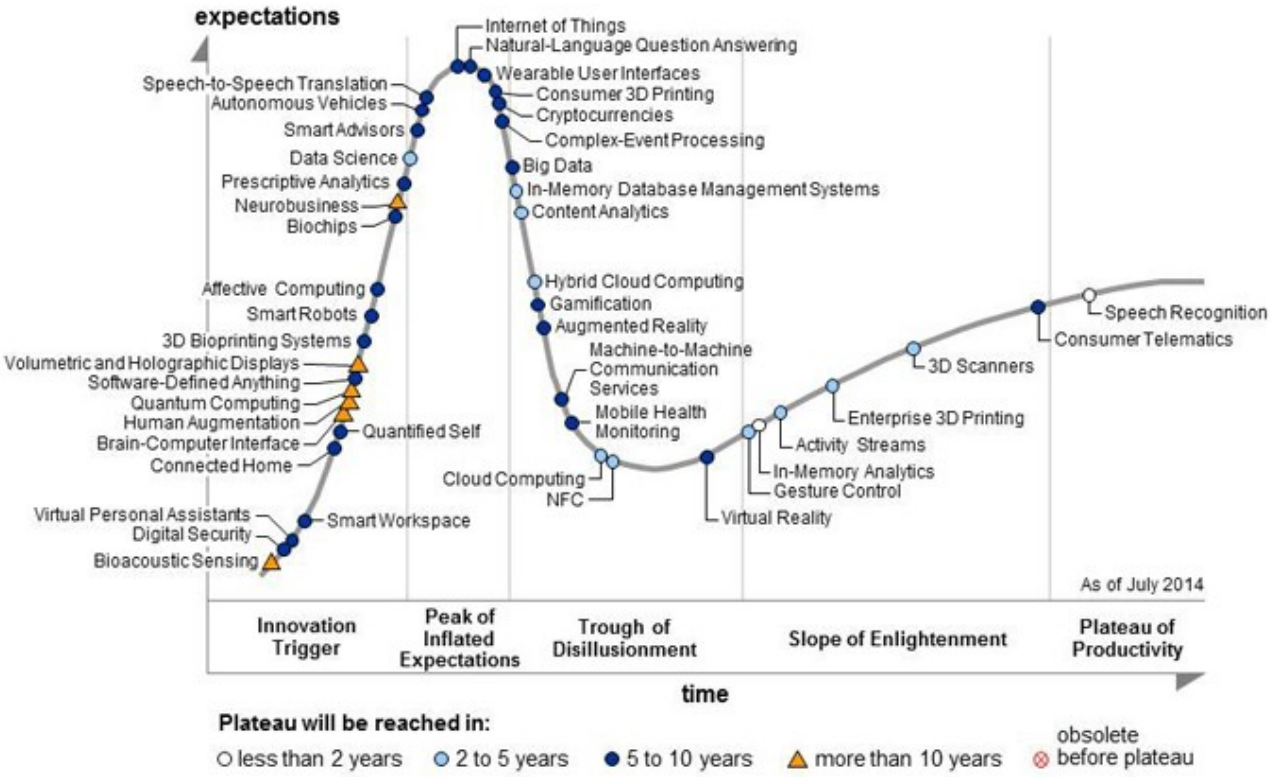


Figure 1.2. Gartner hype cycle for emerging technologies as on 2014 August

Traditional data warehouses work with the abstracted data that has been cleansed and transformed into a separate database (data marts – which are periodically updated with the same type of rolled-up data) for which specific analytics are known in advance. By contrast, Big Data systems maintain raw data whether from operations (log reports), user activity (web site tracking), or other real world usage data. That data is organized in its raw form as its usage is not predetermined, so there is no known target to transform it. Big Data could be organized on distributed storage repositories and large scale computing infrastructure could be utilized for analytics and visualization. However, Big Data and data warehousing systems have the same goals to deliver business value through the analysis of data, but, differ in their scope and the

organization of the data. In practice, data warehouses have traditionally sourced data solely from other databases like enterprise's financial systems, customer marketing systems, billing systems, point-of-sale systems, and so on, however, would not capture the operational databases like click streams logs, sensor data, location data from mobile devices, customer support emails and chat transcripts, and surveillance videos etc. Big Data systems harness these new sources of data, and allow enterprises analyze and extract business value from these immense data sets. Big Data analytics address more complex tasks by mining the information. For instance, Amazon could recommend the ideal book, Google can rank the most relevant website, Facebook knows our likes, and LinkedIn could connect to professionals for whom we know. Similar technologies would be applied to diagnose illnesses, recommend treatments, perhaps even identifying "criminals" before one actually commits a crime. Just as the internet radically changed the world by adding communications to computers, similarly Big Data could change the fundamental aspects of life by giving it a quantitative dimension it never had before.

1.2 Characteristics and key elements

Big Data is characterized into four dimensions called 4V's; Volume, Velocity, Variety, Veracity as depicted in Figure 1.3. Aside, another dimension V (Value/Valor) also used to characterize the quality of the data.

- **Volume:** Volume is concerned about scale of data i.e. the volume of the data at which it is growing. According to IDC [4] report, the volume of data will reach to 40 Zeta bytes by 2020 and increase of 400 times by now. The volume of data is growing rapidly, due to several applications of business, social, web and scientific explorations.

- **Velocity:** The speed at which data is increasing thus demanding analysis of streaming data. The velocity is due to growing speed of Business Intelligence Applications such as Trading, Transaction of Telecom and Banking domain, growing number of internet connections with the increased usage of internet, growing number of sensor networks and wearable sensors.

- **Variety:** It depicts different forms of data to use for analysis such as structured like relational databases, semi structured like XML and unstructured like video, text.

• **Veracity:** Veracity is concerned about uncertainty or inaccuracy of the data. In many cases the data will be in accurate hence filtering the same and selecting the data which is actually needed is really a cumbersome activity. A lot of statistical and analytical process has to go for data cleansing to choose intrinsic data for decision making.

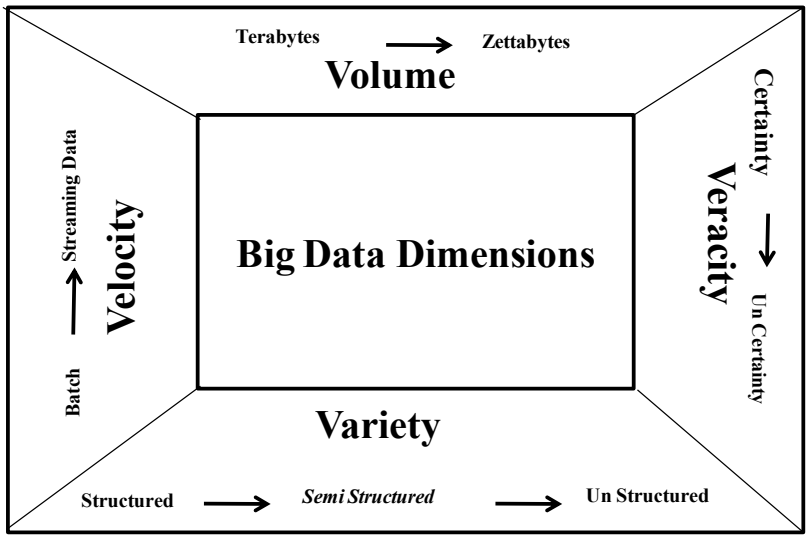


Figure 1.3. Data Dimensions 4V's.

Big Data is an emerging data science paradigm driven from computer science and statistical domains to solve complex data intensive problems in several domains such as science, engineering, healthcare, internet and business intelligence for intrinsic information extraction and decision making. Big Data paradigm enables engineers to analyze far greater quantities and types of data in a shorter time span. It includes structured data sets, such as information stored in databases; semi-structured like XML files and RSS feeds; and unstructured data sets, such as images, videos, text messages, e-mails and documents to uncover insights hidden within these large data sets. The major key elements in the evolution of Big Data are social networking, mobile computing, analytics and Clouds; **SMAC** approach as depicted in Figure 1.4.

Big Data and data warehousing share the same basic goals to deliver business value through the analysis of data, however, differ in the scope and organization of the data. In practice, data warehouses have traditionally sourced data solely from other databases like enterprise's financial systems, customer marketing systems, billing systems, point-of-sale systems, and so on, however, would not capture the operational databases like click streams logs, sensor data,

location data from mobile devices, customer support emails and chat transcripts, and surveillance videos etc.

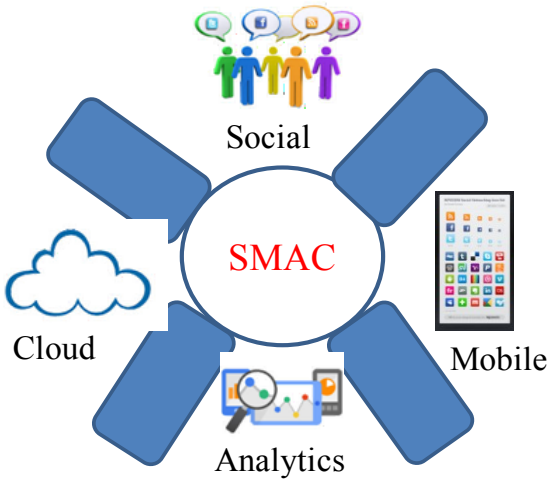


Figure 1.4. SMAC — Big Data Key elements

Big Data is gaining popularity in many fields. Philip et.al [5] presented a survey on Big Data along with opportunities and challenges for Data-Intensive applications stated several areas and the importance of Big Data. Big Data is important both in the business and scientific communities for addressing several problems. As business domains are growing, there is a need to converge a new economic system redefining the relationships among producers, distributors and consumers of goods and services. Obviously, it is not feasible to rely solely on experience or pure intuition to make decisions; hence there is a need to rely on good data services for the decisions. Coming to the scientific community, it is mandatory to extract the information from both the sensor networks and open source information for better decision making using data analytics.

1.2.1 Big Data applications

Big Data is gaining popularity in many fields. Philip et.al [5] presented a survey on Big Data along with opportunities and challenges for Data-Intensive applications stated several areas and the importance of Big Data. Big Data is important both in the business and scientific communities for addressing several problems. As business domains are growing, there is a need to converge a new economic system redefining the relationships among producers, distributors and consumers of goods and services. Obviously, it is not feasible to rely solely on experience or

pure intuition to make decisions; hence there is a need to rely on good data services for the decisions. Coming to the scientific community, it is mandatory to extract the information from both the sensor networks and open source information for better decision making using data analytics. Big Data is applicable to science and engineering, business intelligence, government and industries, scientific explorations etc. the sample application areas are described below.

a) Healthcare industry: Healthcare organizations would like to predict the locations from where the diseases are spreading so as to prevent further spreading [6] . However, it is a challenging job for centers for disease control, to predict exactly the origin of the disease, until there is statistical data from several locations. In 2009, when a new flu virus similar to H1N1 was spreading, Google has predicted this, and published a paper in the scientific journal *Nature*. The authors explained how Google could "predict" the spread of the winter flu in the United States, not just nationally, but down to specific regions and even states. The company could achieve this by looking at what people were searching for on the internet. Since Google receives more than three billion search queries every day and saves them all, it had plenty of data to work with. Google took the 50 million most common search terms that Americans type and compared the list with centers for disease control and prevention (CDC) data on the spread of seasonal flu between 2003 and 2008. The idea was to identify areas infected by the flu virus by what people searched for on the internet.

b) Biotechnology and Drug Discovery: Today, gene sequencing is used extensively by biotechnology companies and pharmaceutical companies to develop more effective drugs or novel therapies for diseases. Gene sequencing involves large collection of data with rapid velocity and biologists have "pipelines" to clean the sequencing data and then "cook" the raw data into usable form. This cooking involves putting all of the "short reads" together into a single human genome and then looking for interesting strings of base pairs in the result. Biologists envision storing sequence of data for millions of humans, so they can perform data mining looking for genome patterns that identify particular diseases. Using current technologies, running a single analysis can take days slowing the scientific process and putting a practical limit on the number of analyses. With new computational technologies for Big Data, we could dramatically reduce current processing times, reducing biologists "time-to-insight" by an order of magnitude.

c) Scientific explorations: The data collected from various sensor sources would be utilized to extract information useful for societal benefits, which would be:

- Physics and astronomical experiments: Large number of scientists collaborating for designing, operating and analyzing the products of sensor networks and detectors such as CERN. A large compute, storage, networking of resources called "Data Grid" are required to analyze petabytes of data [1] [7] [9] [10] [9] . However, extraction and analysis of the data from large distributed compute and storage repositories would demand better analytics.

- Earth Observation Systems (EOS): Information gathering and analytical approaches about earth's physical, chemical and biological systems via remote sensing technologies. It involves collection, analysis and data presentation/visualization. Earth observation is used to monitor and assess the status of, and changes in the natural environment and the built environment. It provides to improve social and economic well-being and its key applications are: weather forecasting, monitoring and responding to natural disasters, and climate change predictions. EOS systems are supported by remote sensing satellites, numerical measurements by various earth sensors with decision support tools.

d) Government: Government provides a large variety of programs and services, which both produce and require massive amounts of data, often unstructured and increasingly in real time. This data comes from numerous sources including historical, video, audio, cell phones, geospatial, imagery, sensors, social media and much more. From crime prevention to transportation, defense, national security, revenue management, environmental stewardship and social services, governments must wrestle every day with managing and using this data. Big Data analytics can improve the efficiency and effectiveness across the broad range of government responsibilities, by improving existing processes and operations and enabling completely new ones. The several areas for harnessing Big Data computing for government sector include:

- Surveillance system analyzing and classifying streaming acoustic signals.
- Cyber security system analyzing network traffic for anomalies.
- Sensors in ocean buoys and river gauges collecting data for faster, more accurate flood predictions.

- Transportation departments using real-time traffic data to predict traffic patterns, update public transportation schedules.
- Police departments analyzing images from aerial cameras, news feeds, and social networks or items of interest.
- Social program agencies gain a clearer understanding of beneficiaries and proper payments.
- Tax agencies identifying fraudsters and support investigation by analyzing complex identity information and tax returns.
- Sensor applications such stream air, water and temperature data to support cleanup, fire prevention and other programs.

e) Financial and business analytics: Big Data could transform business and financial institutions to tailor their products and build strategy road maps aligned with customer expectations. Large scale administrative data sets and proprietary sector data can greatly improve measurement, tracking and describing the economic activity. Text captured from credit applications, account opening interviews, call center notes, mortgage application notes, social media chatter and other customer service interactions could be integrated and analyzed to identify escalation and complaint triggers, understand fraud pattern, manage alerts, reduce credit risk and build social media dash boards. Effective use of Big Data would be a key driver for competition in financial services, and companies that use data more effectively will secure an edge in the market place. Retaining customers and satisfying consumer expectations are among the most serious challenges facing financial institutions. Sentiment analysis and predictive analysis could be used to address the issues and also for other key challenges. The several areas are described below.

- **Travel industry:** Better decision support to enable the customers with targeted options – product/service innovation, extreme search; in which a consumer enters overall budget, the number of passengers, the length of the time for the trip, and the minimum temperature at the destination, the analytics could return the proposals for the trip [11] .

- **Retail industry:** A retail company could recommend specific items to customers based on their individual shopping preferences. Amazon could recommend specific books based

on their interest, from its start, capturing reams of data on all its customers; what they purchased, what books they only looked at but didn't buy, and how long they looked at them. Capturing customer feedback through sentiment analysis which examines customer confidence indices that are driven by specific data elements such as product, functionality, content and price; banks can judge the mood of the market and decide how to best reward their customers. Successful execution drives loyalty and also attracts new customers. Although the technologies behind sentiment analysis are still maturing, many of the tools and techniques are advanced enough for financial services institutions to derive incremental value by understanding customer likes, dislikes and preferences for product and service improvements.

- **Forecast**: Forecast of the airline tickets for predicting whether the price of an airline ticket was likely to go up or down, and by how much, empowered consumers to choose when to go for buying the ticket [6] . The system can work by taking, industry's flight reservation databases, system can make predictions based on every seat on every flight for most routes in aviation sector over the course of a year. The system can save consumers a bundle. However, the similar technique may be applied for hotel rooms, concert tickets, and used cars; anything with little product differentiation, a high degree of variation over a large scale data.

- **Financial industry:** Big Data analytics for fraud detection in financial services would help to detect fraud in real time and proactively identify probable risks without disrupting service to valuable customers. The analytics would process massive amounts of structured and unstructured data from hybrid sources; model and algorithms would be developed to find patterns of fraud and anomalies to predict customer behavior. Several areas of fraud detection credit cards, insurance claims or surrender premium, employee related or third party fraud [12].

Big Data analytics help businesses to make better decisions in several areas which are described below.

- Product strategy: strategies to adopt for promoting the existing or new product by understanding the market by going through the customer data.

- Targeting sales: approaches to improve the sales by targeting the probable customers.
- Just-in-time supply-chain economics: ways of managing manufacturing systems to reduce waste, and lower cost, thus increasing profit.
- Business performance optimization: methods such as productivity, process management, labor optimization for the survival of the firm.
- Predictive analytics and recommendation: predictions about the new products marketing.
- Real time analytics: stringent timing constraints to analyze the data for real time decision making.
- Census data analysis: population analysis for governments for policy makings.
- Health care industry: data maintenance of all the medical related information.
- Telecommunications: transactions and logs of all the customers and log analysis.
- Social network analysis: analysis of the behavior of the people and constructing the strong and weaker social networking groups.
- Stream data mining: Data Analysis of video and streaming data.
- Semantic and web ontology: Ontology building for relationships.
- Customer behavior analysis: analyzing the behavior of the customers for business intelligence.
- IT infrastructure optimization: maximize return on investments (ROI).
- Churn analysis: identify the consumers/customers who are most likely to discontinue their business services and take necessary steps to retain their potential customers.

f) Web analytics: Several web sites are experiencing millions of unique visitors per day, in turn creating a large range of content. This can easily amount of tens of hundreds of gigabytes per day (tens of terabytes per year) of accumulating user and log data. Increasingly, companies want to be able mine this data to understand limitations of their sites, improve response time, offer more targeted ads and so on. Doing this requires tools that can perform complicated analytics on data that far exceeds the memory of a single machine or even in cluster of machines.

1.2.2 Cloud computing – back end infrastructure for Big Data

Clouds are being positioned as the next-generation computing infrastructure platforms for hosting all kinds of IT and scientific applications for deploying, maintaining, managing, and delivery to a wider variety of personal, as well as professional applications and services. The internet, and several scientific experiments in several fields of science and engineering has spawned an explosion in data growth in the form of data sets, called Big Data, that are so large they are difficult to store, manage and analyze using traditional data warehousing and mining techniques. Apart from huge volumes, often the data generated from web logs, and scientific experiments becomes heavily unstructured, and streams are generated so rapidly, which needs to be organized and processed becomes an important factor.

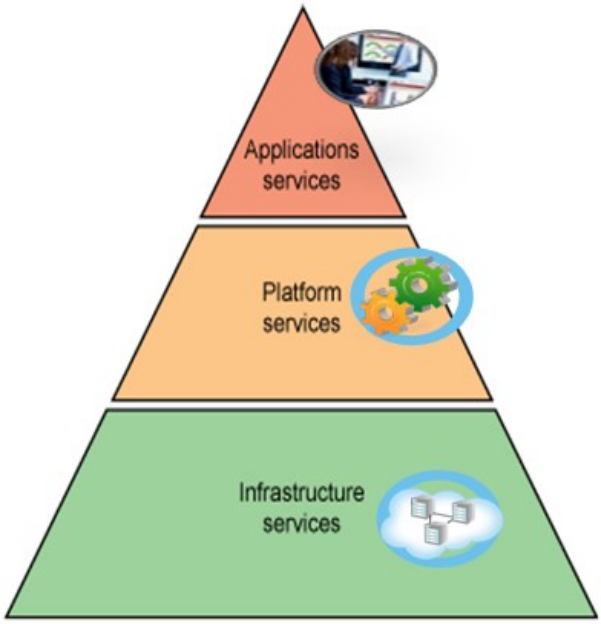


Figure 1.5. Cloud computing architecture

The benefits of Cloud computing over traditional computing models include – multi-tenant, on demand resource provisioning, automated and self orchestrated resource creation and management, dynamically scalable with easy configurable tools and techniques for managing the infrastructures for computing, storage and networking. Clouds offers three fundamental service models as depicted in Figure 1.5 with three layers of service delivery models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and SaaS (Software as a Service). IaaS offers virtualized computing services for storage, compute and networking elements with on demand provisioning, elastically scalable and self management. PaaS is a one

of three fundamental service models of Clouds that provides a platform allowing one to develop, run and manage applications without the complexity of building and maintaining the infrastructures. SaaS is a software distribution model in which applications are made available to the users from remotely without being installed on their local devices. These applications are built using PaaS model and will make use of the IaaS services.

Clouds can be deployed in three modes such as private clouds, public clouds and hybrid clouds to meet the organizational requirements as depicted in Figure 1.6. Private Clouds are set up within the organizational boundaries firewall, to meet the users and application demands. This type of clouds are owned, operated and restricted to particular organizational needs, and not made available to the general public. Public clouds offer the infrastructure, platform and applications services. These are operated by a company and are made available to the general public as on demand pay-as-go computing subscription based services. Hybrid Clouds does the federation of Public and Private Clouds, with an additional infrastructure facility supplemented by Public Clouds as required by the organizational need.

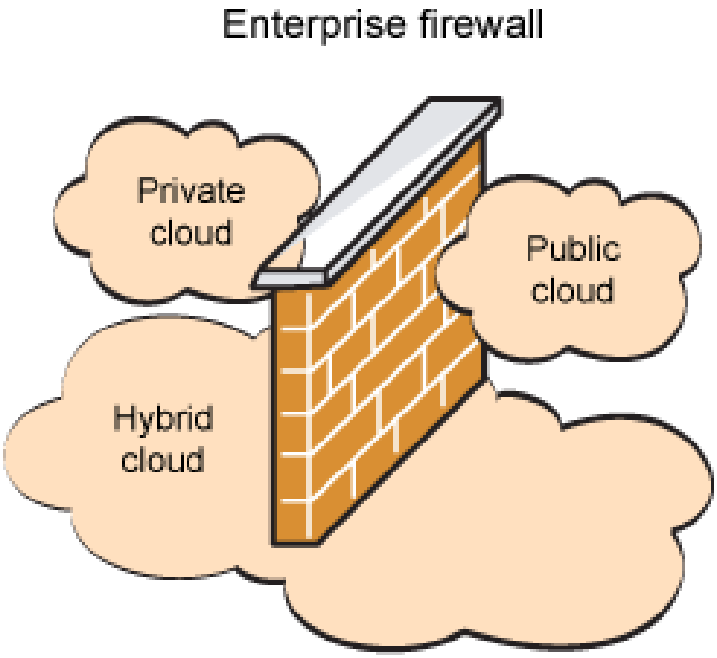


Figure 1.6. Cloud computing deployments

Since Cloud computing technologies are into reality, to deliver compute, storage and software applications as services over private or public networks, and Big Data technologies demand the large scale dynamically scalable computing and storage resources, hence, Clouds can turn as

infrastructure platforms for such large scale big data explorations for intrinsic information extraction.

1.3 Big Data computing in Clouds – Problem statement

Cloud is emerged as service oriented computing model, to deliver infrastructure, platform and applications as services to the end users. As clouds are becoming reality, it is emerging as back end technology by enabling the archival and processing of large volumes of rapidly growing data for further analysis. Big Data computing demands a huge storage and computing resources for data organization, and processing, which could be driven by Cloud computing as under pinning back end technologies for the realization of Big Data Analytics for scientific and business applications.

Here we discuss the challenges in Big Data computing using Clouds as large scale computing infrastructure facilities. We present the elements of Big Data Computing in Clouds, **Taxonomy of Big Data and Clouds, Layered Architecture, followed by a Framework for Big Data Computing in Clouds**. The framework addresses the several elements of the layered architecture.

Here we discuss on one of the areas delineated in the taxonomy – **scheduling the jobs in decoupled data and computing resources for Big Data applications in Clouds based on Data Aware Scheduling using Genetic Approach, family grouping and graph connected components.** To this end, it presents the architecture, problem discussion, model, and problem solving using genetic approach, The thesis discussed the middle ware which is able to discover big data repositories, interface with various repositories, select the data portion for transfer based on the available bandwidth, group them as family, and select suitable resources in order to meet the application requirements, so as to achieve high throughput. . The approach is based on evolutionary methods focused on data dependencies, computational resources and effective utilization of bandwidth thus achieving higher throughputs.

Later, we discuss the handling the image or similar kind of data on the Hadoop Distributed File System and MapReduce model[8] . We focus our discussion on the applications for which the data overlap among the adjacent blocks is mandatory. To meet those requirements, we propose an **extended model of HDFS and MapReduce called XHAMI**, which offers a high

level APIs for data organization and processing of single large volumes of images using distributed file systems and MapReduce programming models.

1.4 List of publications

[1] **R. Kune**, P. K. Konugurthi, A. Agarwal, C. R. Rao, and R. Buyya, Anatomy of Big Data Computing, Journal of Software Practice and Experience (SPE), Volume 46, Number 1, pages: 79-105, ISSN: 0038-0644, Wiley Press, New York, USA, Jan 2016.

[2] **R. Kune**, P. K. Konugurthi, A. Agarwal, C. R. Rao, and R. Buyya, Genetic Algorithm based Data-aware Group Scheduling for Big Data Clouds, in: Proc. International Symposium on Big Data Computing (BDC 2014), London, December 2014, pp. 96-104.

[3] **R. Kune**, P. K. Konugurthi, A. Agarwal, C. R. Rao, and R. Buyya, XHAMI - Extended HDFS and MapReduce Interface for Image Processing Applications, in: Proceedings of the 4th International Conference on Cloud Computing for Emerging Markets (CCEM 2015, IEEE Press, USA), Bangalore, India, November 25-27, 2015.

[4] **R. Kune**, P. K. Konugurthi, A. Agarwal, C. R. Rao, and R. Buyya, XHAMI - Extended HDFS and MapReduce Interface for Big Data Image Processing Applications in Cloud computing Environments, Software Practices and Experiences, Wiley online DOI: 10.1002/spe.2425.

1.5 Thesis Contributions

This thesis makes several contributions towards Big Data computing environments in Clouds with a taxonomy, architectural framework, discusses the gap analysis, and identifies two key elements of Big Data computing in Clouds infrastructure such as Scheduling and large scale data intensive organization and problem solving for scientific analysis. These are as follows.

1. The thesis discusses the key concepts behind **Big Data computing on Clouds**. It compares Big Data Clouds with technologies such as Data Clouds, Storage Clouds and Cluster based Big Data computing setups. Discussed on how Big Data and Cloud technologies, that each has to complement and support for large scale data handling and processing for both scientific computing and business analytics. The objective of this exercise is to understand, how Big Data computing and Cloud computing are converging, to evolve as new generation technology.

2. The thesis provides a **comprehensive taxonomy that covers aspects of Big Data Clouds**, underlying elements, Integrated Big Data Access Network, gap analysis and future research directions. Here, identifies the key components of Big Data and Clouds technologies such as Scheduling, and data organization and processing.

3. The thesis presents the **layered architecture and framework for Big Data Clouds**, which is generic enough to accommodate different components and maps well onto the architecture of Clouds. Illustrates the underlying components. identified the two key components in the framework such as scheduling of data intensive applications using both data push and pull models, followed by data organization and processing of the data intensive applications using data intensive MapReduce models.

4. Discussed the significance of **Big Data computing in Earth Observation System (EOS) and Remote Sensing Big Data Clouds** as one of the case study for supporting the proposed architecture and our methodologies.

5. The thesis presents a scheduling model for effectively scheduling the jobs onto computing resources by effectively utilizing the underlying bandwidth. The model takes into account application execution times, replicated data sites, and computing capability to achieve higher throughputs. The proposed model is compared with match making and other heuristic techniques. Developed architecture, and the model for **Data Aware Scheduler using evolutionary genetic algorithm** and graph connected components for scheduling data intensive applications in the decoupled storage and computing infrastructure setups. Discussed how this scheduling model differs from the conventional compute push models like MapReduce, and illustrated the significant role of Data Aware Scheduler in scheduling the scientific applications which majorly demand data push to the computing node rather computing function push to the data node. Developed a mathematical model using data commonality for the jobs (called as family job) and available network bandwidth for parallel chunks of data transfers for achieving high throughputs while scheduling the data intensive jobs. The model is simulated using CloudSim Toolkit using virtual data hosts and virtual clusters, and to meet the requirements of

the scheduling aspects, substantial enhancements were incorporated for CloudSim toolkit to support the features such as data replication, bandwidth awareness, job grouping etc.

6. Demonstrate a model for Big Data Image processing scientific computing applications for which data overlapping is the primary and essential requirement. Developed a methodology and high level programming APIs over Hadoop Distributed File System (HDFS) and extended the conventional MapReduce models called **"XHAMI – Extended Hadoop and MapReduce Interface for Big Data Image Processing Applications in Cloud Computing Environments"**.. XHAMI discusses several key elements of data organization in the Hadoop distributed file system, for image processing applications where overlapped data is the primary requirements and discusses the development for several image processing spatial filters for scientific applications. XHAMI presents a high level data organization model, programming APIs and extended MapReduce functions for image processing applications. The thesis also discusses how XHAMI could be extended for similar kind of image processing domain applications where overlapped data is among the functions is essential requirement.

1.2 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2 presents an over view of data intensive technologies in traditional data warehousing, mining and Big Data technologies. It discusses about the CAP theorem with ACID and BASE properties to differentiate Big Data technology with the traditional data models. It also discusses relevant computing paradigms such as Cloud computing and its deployment models followed by taxonomy of Big Data computing technologies. – derived from publications 1.

Chapter 3 presents an architecture and framework for Big Data computing in Clouds infrastructure. It discusses the several elements of the framework, an integrated Cloud and Big Data computing platform, and illustrates gap analysis in Big Data technologies. The chapter identifies the key elements of the framework such as scheduling the Big Data applications in Clouds infrastructure and also addressing the extended data intensive computing models for Distributed File System. – derived from publication 1.

Chapter 4 discusses Data Aware scheduling model for solving data intensive applications in Clouds infrastructure. It presents the problem statement, motivating applications such as Remote Sensing Big Data in Clouds for Earth Observation Systems (EOS), and addresses Data Aware Scheduler, its architecture, workflow and mathematical model based on data and bandwidth parameters. – derived from publication 2.

Chapter 5 presents, evolutionary based genetic approach for solving data intensive applications using Data Aware Scheduler based on family grouping using graph connected components. It presents the problem formulation for chromosome representation, fitness functions, and scheduling algorithm. It also discusses the simulation and results using CloudSim toolkit, and illustrate the extensions made for CloudSim toolkit for addressing the Data Aware Scheduling. It also illustrates the comparison results of the performance of family Vs non family scheduling and other heuristic techniques – derived from publication 2.

Chapter 6 discusses XHAMI- an extension of MapReduce models for data intensive scientific computing along with the data organization models for Image processing domain applications. It presents the data model and extended function with APIs for MapReduce computing on Hadoop Distributed File System (HDFS). It discusses the results with the performance benefits of XHAMI over traditional. Distributed file system and MapReduce computing models. – derived from publications 3 and 4.

Chapter 7 presents conclusions and future research directions.

Chapter 2

Literature Review – an overview and related technologies

This chapter provides a general overview of Big Data Clouds that covers topics such as key concepts, relevant paradigms, taxonomy of Big Data computing, Layered architecture, Convergence of Cloud computing and Big Data computing. This chapter presents an analysis of the differences between Big Data Clouds, and other technologies such as Data Clouds and Storage Clouds. It ends with a discussion on the convergence between Cloud computing as back end technologies and Big Data is being the computing over back clouds, emerging as Big Data Clouds – a new generation service oriented Big Data Analytics platform.

2.1 Big Data Vs Traditional Data Models

Big Data computing environment consists of applications that demand and produce large amount of data from collection and analysis respectively, sizes increasing from several Tera bytes (TB) to Peta bytes (PB) and beyond. The data is organized as Raw, Object file systems, and are typically stored at geographical distant apart storage repositories, and are offered as services by Cloud providers using web services. The data could be replicated over geographical locations for high availability and effective access so as to minimize the I/O latencies. The data is accessed via APIs offered by providers, or low level distributed programming APIs offered by the underlying technology, in the case of cluster based distributed file system.

In the context of Big Data Clouds represent the providers who offer the data as service through a several set of APIs and Meta data search mechanisms, to retrieve data of interest. Apart from this, they also offer such data replicated sites/clouds for effective access. Computing Clouds refer the providers, those who offer computing infrastructure on demand. These computing resources either could be physical or virtual infrastructure. The computing infrastructure either

could have been established prior for the applications, or could be dynamically provisioned based on the application requirements.

Big Data refers large scale data architectures, and facilitate tools addressing new requirements in handling data volume, velocity, and variability. Traditional databases (data warehousing) assume data is organized in rows and columns and employs data cleansing methods on the data while the data volumes grow over a time period, and often lack on handling such large scale data processing. Traditional Data base / warehousing systems were designed to address smaller volumes of structure data, with the predictable updates and consistent data structure, that mostly operate on single server and lead to operational expenses with the increased data volume. However, Big Data comes in a variety of diverse formats with both batch and stream processing in several areas such as geospatial data, 3D data, audio and video, structured data, unstructured text including log files , sensor data and social media. Below, we discuss the properties of traditional database (Data Warehousing) and Big Data.

Bill Inmo [13] described data warehousing as subject oriented, integrated, time-variant, and nonvolatile collection of data, and helping analysts in decision making process. Data warehouse is segregated from the organization's operational database. The operational database undergoes the per day transactions (On Line Transaction Processing – OLTP) which causes the frequent changes to the data on daily basis. Traditional databases typically addresses the applications for business intelligence, however, lack in providing the solutions for unstructured large volumes rapidly changing analytics in business and scientific computing. The several processing techniques under data warehouse are described below.

- Analytical processing involves analyzing the data by means of basic OLAP (Online Analytical Processing) operations, including slice-and-dice, drill down, drill up, and pivoting.
- Knowledge discovery through mining techniques by finding the pattern and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using visualization tools.

Big Data addresses the data management and analysis issues in several areas of business intelligence, engineering and scientific explorations. Traditional databases segregate the

operational and historical data for operational and analysis reasoning, which are mostly structured. However, Big Data bases address the data analytics over an integrated scale out compute and data platform for unstructured data in near real time. Figure 2.1 depicts several issues in Traditional data (Data warehousing OLTP/OLAP) and Big Data technologies which are classified into major areas like infrastructure, data handling and decision support software as described below.

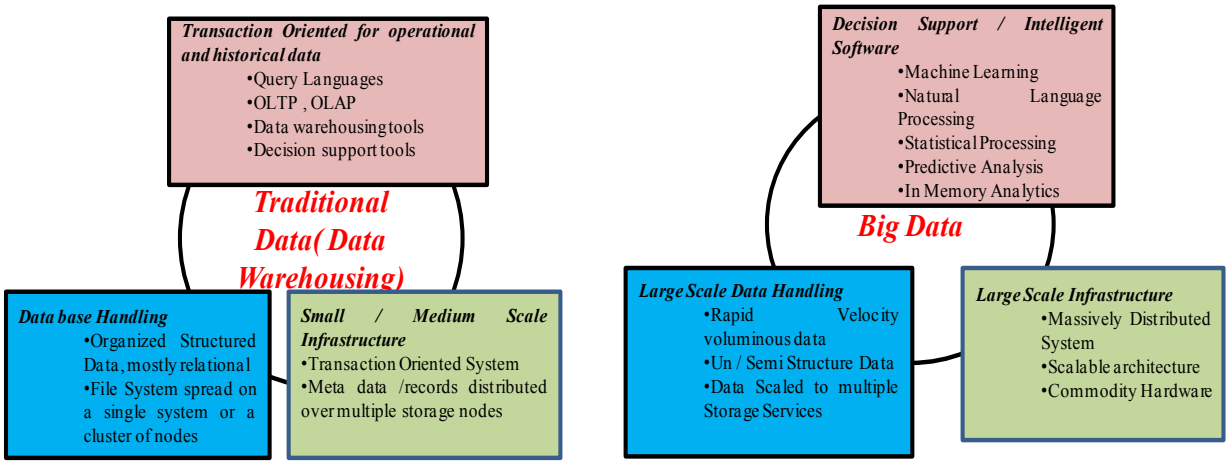


Figure 2.1. Big Data Vs Traditional Data (Data Warehousing) models

- ***Decision support / intelligent software tools***: Big Data technologies address various decision supporting tools for searching the large data volumes and constructs the relations and extract the information based on several analytical methods. These tools would address several machine learning techniques, decision support systems and statistical modeling tools.
- ***Large scale data handling:*** rapidly growing data distributed over several storages and compute nodes with multi-dimensional data formats.
- ***Large scale infrastructure:*** scale out infrastructure for efficient storage and processing**.**
- ***Batch and stream support:*** capability to handle both batch and stream computation.

Table 2-1 illustrates properties of Big Data Vs Traditional Data Warehousing computing.

Table 2-1. Traditional Data warehousing Vs Big Data issues

S. no	*Property*	*Traditional Data Warehousing*	*Big Data specific issues*
1	data volume	Data is segregated into operational and historical data. Applies ETL (Extract, Transformation and Load) mechanisms for processing. As the data volumes are increased, the historical data is filtered from ware house system for faster database queries.	High volume of data from several sources like web, sensor networks, social networks, scientific experiments. Capable of handling operational and historical data together, which could be replicated on multiple storage devices for high availability and throughput.
2	speed	Transaction oriented and the data in turn generated from the transactions is low.	High data growth due to several sources like web and scientific sensors streaming experiments.
3	data formats	Semi/structured data like XML and Relational.	Multi structured data handling such as relational, and un/semi structured such as text, XML, video streaming etc.
4	applicable platforms	OLTP (Online Transactional Processing), Relational RDBMS.	Big data analytics, text mining, video analytics, web log mining, scientific data exploration, and intrinsic information extractions, graph analytics, social networking, In memory analytics, statistical and predictive analytics.
5	programming methodologies/ languages	Query Language like SQL.	Data intensive computing languages for batch processing and stream computing like map/reduce, NoSQL programming.
6	data backup / archival	Files / relational data need to have data backup procedures or	Due to large and high speeds of the data growth rates, the conventional

		mechanisms. Traditional data works on regular, incremental and full backup mechanisms that are already established.	methods are not adequate; hence techniques such as differential backup mechanisms need to be explored.
7	disaster recovery (DR)	Data is replicated at several places to address the disaster.	DR techniques could be separated from mission critical and non critical data.
8	relationship with Clouds	Relational Data bases/ Data ware housing tools as services over cloud infrastructures.	On demand Big Data infrastructure setup, analytic services by several cloud and Big Data providers.
9	data de-duplication	Applicable to transactional record deduplication while merging database records.	File and block level deduplication mechanisms need to be explored for continuous growing and stream oriented data.
10	System users	Administrators, developers and end users.	Data scientists, analytics end users.
11	theorem applicable	Follows CAP theorem [14] with ACID [15] properties.	Follows CAP theorem with BASE properties [16].

2.2 CAP Theorem – ACID and BASE

Traditional databases follow ACID [15] properties, which are the primary standards for relational databases. However, distributed computing systems follow BASE [16] properties to address loss of consistency and reliability as discussed below.

- **Basically available**: This property states that, the system guarantees the data availability, however, during the transition/changing state the response would be either delayed or may fail in obtaining the requested data. This scenario is similar as depositing a check in your bank account, and waiting till the check goes through the clearing house, for having the funds made available.

- **Soft state**: The state of the system would change over time, so even during times without input, there may be changes going on due to eventual consistency, thus state of the system is always soft.

- **Eventual consistency**: The system would propagate the data as it is receiving, however, will not ensure the consistency of the data for every transaction. The data would be eventually consistent, whenever it stops receiving the input.

In 2000, Eric Brewer presented CAP theorem, also known as Brewer's theorem [14] for the successful design, implementation and deployment of applications in distributed computing systems. CAP theorem states that any networked shared-data system can provide only two out of the following three properties mentioned below.

- **Consistency**: similar to the consistency property of ACID, the data is synchronized across all cluster nodes, and all the nodes would see the similar data at the same time.

- **Availability**: guaranteed that every request receives a response however, the request is successful/failed in receiving the data which has been requested would not be known.

- **Partition tolerance**: single node failure should not cause the entire system to fail and the system should continue to function even under circumstances of arbitrary message loss or partial failure of the system.

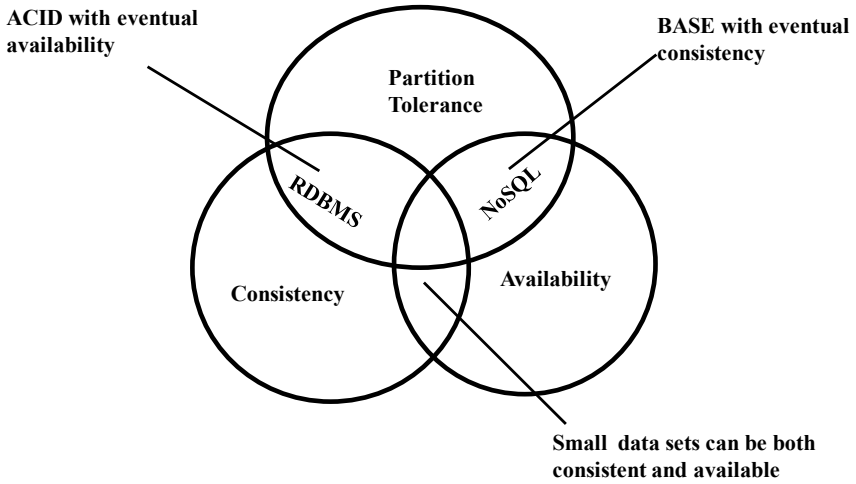


Figure 2.2. CAP theorem with ACID and BASE (Source: NIST [17])

Big Data system adopts Brewer's CAP theorem on the similar lines of BASE. CAP theorem with ACID and BASE is depicted in Figure 2.2.

2.3 Relevant Paradigms in Big Data computing

Cloud computing is emerged as new generation service oriented computing model to deliver the computing, storage and networking resources on demand, The services are offered three types of deployment models such as public, private and hybrid. In this section we will briefly discuss Cloud computing paradigm and relevant technologies that can be leveraged to create Big Data computing environments followed by Cloud computing deployment models.

2.3.1 Cloud computing Taxonomy

As more and more data is generated at a faster-than-ever rate, processing large volumes of data and development of data analysis software is becoming a challenge. Frederic et.al [18] discussed various technologies that demonstrate how cloud computing can meet business requirements and serve as the infrastructure of multi-dimensional data analysis applications.

The digital data collected from several sources like internet web, sensor networks, financial firms, scientific experiments of earth observation are increasing rapidly and clouds are playing a major role for data organization over vast hardware datacenters linked to billions of distributed devices [1] . As, the data volumes are growing, the skills, experience and resources to manage all these bits of data would require a new flexible and scalable IT infrastructure that extends beyond the enterprise cloud computing. However it seems likely that private clouds and public clouds will be common place, exchanging data seamlessly, hence there will not be one cloud bounded by geography, technology, different standards and perhaps even vendor. Hence, storing, analyzing, and delivery of zeta bytes of content need efficient management of infrastructure and efficient analysis tools.

Cloud computing is emerged as the next generation of service oriented computing to deliver resources on demand. Clouds are classified into public, private, and Hybrid clouds [19] based on the services rendered to the users. Cloud computing consists of various tools and technologies to offer the resources and service the requests accordingly. The elements of clouds are virtualization, programming, schedulers, and offerings such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The benefits of the cloud

computing include; access to infinite scalable resources, effective utilization, minimal upfront cost and high return on investments. The taxonomy of Cloud computing is shown in Figure 2.3. The elements of the Cloud computing are described below.

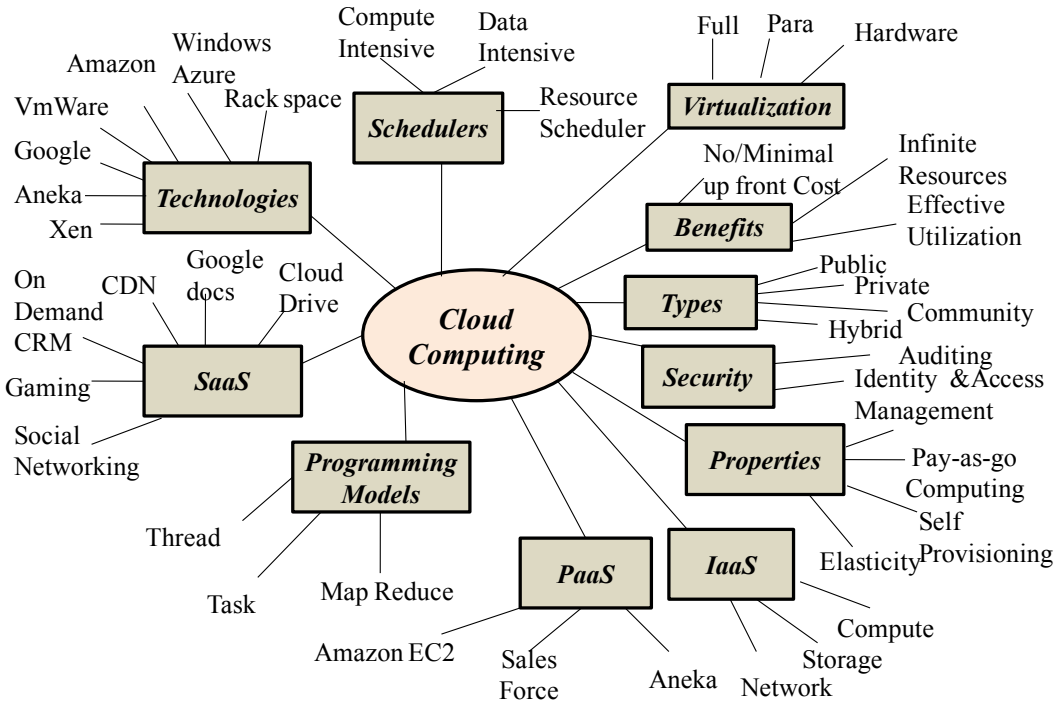


Figure 2.3 Cloud Computing Taxonomy

Table 2-2. Virtualization and Infrastructure Management Technologies

Name	Provider/Technology
Xen	Xen Source
Citrix Xen	Citrix
VMware	VMware
Open Stack	Open Stack
Eucalyptus	Eucalyptus
Hyper-V	Microsoft
OVM	Oracle

i) Virtualization: Virtualization is the basis of the cloud computing technologies. Virtualization tools also known as hypervisors enable the creation of large pools of virtual compute and storage thus provide applications as services on the virtualized network. Two types of

hypervisors exist; bare metal and hosted. Bare metal are further classified into full, para and hardware assisted [20] . Virtualization plays a major role in the creation of Compute Clouds which are intended to delivery compute power in the form of virtual machines. The major players in virtualization Technologies are listed in Table 2-2.

2.3.2 Cloud computing deployment models

Clouds fall into three types such as public, private and hybrid. Public clouds; operate over public networks, rendering the services to the consumers based on pay-as-go computing and subscription models. Private clouds; operate in the closed user community over a secure network and Hybrid clouds federate workloads of the private to public clouds during peak hours. Clouds are classified as Public, Private and Hybrid based on the services each of them renders. Public Clouds; provisions the resources on rent; however billing is based on the usage and subscriptions. Private Clouds are a closed user group virtual data centers, unlike traditional data centers who deliver physical resource, these clouds offer virtual machine instances, storage servers and software over the private network. Unlike Public clouds, Private Clouds won't follow subscription models, however QoS requirements are met. On the other hand, Hybrid Clouds federate. Private and Public Clouds by staging the workloads to public clouds from private virtual data centers as and when needed, especially during the peak hours. Public clouds work outside the firewall boundaries; hence data security may be a major concern. Private clouds work within firewall boundaries and in a secured network; hence data security may not be a major concern. However, Hybrid Clouds work across the firewall boundaries of the data center with semi secured network. Public Clouds offer resources rapidly; hence this model is beneficial to the organizations who would like to immediately concentrate on their product design without worried on infrastructure management. However, Private clouds are good if investment is not a major concern, however, effectively utilize and manage the resources nevertheless security

ii) **Security:** The various security mechanisms needed are auditing, symmetric and asymmetric encryptions apart from Identity and Access Management Mechanisms.

iii) **Benefits:** Clouds offer large pool of resources with effective utilization and less cost on infrastructure and management.

iv) **Schedulers:** Cloud computing addresses both compute and data intensive problems apart from the resource schedulers which deliver the dynamic resources to the user from the virtualized infrastructure.

v) **Properties:** The resources are offered based on pay as per usage basis. Public clouds collect the money for the computational resources based on hourly consumption and storage based on per 1GB and data transfer. The multi-tenancy feature of cloud enables the effective usage of the shared infrastructure.

vi) **Programming Models:** Cloud computing offers programming models such as Thread, Task and Map Reduce models to solve compute and data intensive problems.

vii) **Technologies:** The technology providers of cloud computing are classified into two types, first, who offers tools and SDK for infrastructure setup, others offer resources over the network. The various cloud providers are; Amazon, Windows Azure, RackSpace and Google. The technology developers are Citrix Xen, VMware, Windows Hyper-V, Oracle VM etc.

viii) **Layers:** Cloud computing follows a three layered architecture. The bottom layer is the IaaS (Infrastructure as a Service), the middle is Platform as a Service (PaaS) and the top is the application development layer.

Table 2-3. IaaS technologies/providers

Name	Provider/Technology
EC2	Amazon
VM Role	Windows Azure
Nova	Open Stack
Xen Cloud Platform	Xen Virtualization
Eucalyptus	Open Source

ix) **Infrastructure as a Service (IaaS):** This is the bottom layer of the cloud, which is responsible for Compute, Storage and Network virtualization developments.

The computation is delivered from the virtualized infrastructure. Virtual Machines are provisioned dynamically, which are mapped to the physical infrastructure. The machines can be created on demand, can be migrated easily across the servers. The advantages are effective

utilization of the resources, minimizes the upfront costs and management of the infrastructure is easier. The various Compute Cloud offerings are shown in Table 2-3.

x) **PaaS (Platform as a Service):** The middle layer of cloud computing technologies which offers application development tools, schedulers, APIs for running applications which can run on the infrastructure. Example PaaS systems include: SalesForce.com for on demand CRM, GoogleApp Engine, and Aneka platform for cloud applications development.

xi) **SaaS (Software as a Service):** It is concerned with the delivery of applications services. Example SaaS applications include: Cloud Drive, Google Docs, On Demand CRM, gaming, Face Book and Twitter for social networking.

Clouds are broadly classified into three types of deployment models, based on the services they render to the users, and the network in which they are connecting. The three types of Cloud computing deployments are Public, Private and Hybrid.

a) Public Clouds: The resources are offered as services based on the pay-as-go subscription models. Users can make provision the services and pay according to the usage or consumption of the resources.

b) Private Clouds: Clouds setup within the enterprise within the firewall boundaries for their own usage. These clouds are not accessible from the outside network and hence they are more secured.

c) Hybrid Clouds: These clouds are federation of public and private clouds.

2.4 Data Intensive Scientific Computing

CloudStor [21] is a data intensive computing in the Cloud for serving large scientific data sets to explore new strategies and technologies by investigating applications involving remote-sensed LiDAR data, in conjunction with the Open Topography [22] Project. Data Grids were developed for scientific collaborative study for LHC CERN [23] particle accelerator experiments. BioGrid [24] project works on the single data source and generates huge amount of data that gets distributed across the geographical regions. Virtual Observatory [25] project integrates existing independent sources of data for data exploration. Srikumar et.al presented elements of Data

Grids with focus on data transportation, data replication and resource allocation and scheduling [26]. Ranganathan et.al presented decoupled computation and Data for scheduling Distributed Data Intensive applications in the grid environment [27] . Changjun Hu et.al [28] discussed Data Cloud Model for Management and Sharing of Material Science Data.

David Tracey et.al [29] discussed a holistic architecture for IOT, Sensing Services and Big Data with the using Constrained Application Protocol (CoAP) with Hadoop HBase [30] data store. Amazon Web Services (AWS) offering Hadoop Big Data as a Service over Amazon computing infrastructure [31] . Rackspace is offering Hadoop Big Data platform over their private clouds, with the choice of managing Big Data platforms with the choice of storage devices, platforms, architectures and network designs [32] .

HPCC (High Performance Computing Cluster) [33] also known as DAS (Data Analytics Supercomputer), is an open source, data-intensive computing system platform. HPCC platform incorporates software architecture on commodity computing clusters to provide high performance for big data applications, with parallel batch data processing (Thor), high performance online query applications using indexed data files (Roxie), and data-centric declarative programming language called ECL [34].Previous works on scheduling in Data Grids [27] [37] have been more concerned with the relationship between job assignment and data replication based on computation and data proximity. Mohammed et.al [38] discussed a Close-to-Files algorithm, searching the entire solution space for a combination of computational and storage resources for minimizing the processing time with the restriction of one dataset per job for execution.

Srikumar [39] described scheduling the distributed data intensive applications on global grids based on a set coverage approach for cost and time minimizing problems. This approach is based on the availability of both computation and data resources; however, data transfer from replicated sites and the selection of efficient computing nodes for minimizing the execution times are not addressed.

Big Data computing frameworks such as Apache Hadoop [40] is an open source implementation for MapReduce scheduling methods; the examples are Capacity [41], and Throughput [42]. Fair Scheduler [43] is a pluggable group scheduler where in each group gets equal time slots for computation. Capacity Scheduler is similar to FIFO within each queue, but

limiting the maximum resources per queue. Throughput Scheduler reduces overall job completion time on heterogeneous cluster by actively assigning tasks to computing nodes based on the server capabilities. Shared Scan Schedulers S^3[44] allows sharing the scan of a common file for multiple jobs arriving at different time intervals thus improving the performance of multiple jobs which are operating on a common data file. Here, we discuss a scheduling methodology, where computational resources and data storages are decoupled with the data replicated over storage repositories which are geographical dispersed. Here, the problem is focused on grouping the jobs based on the data requirements, and the objective is to minimize the total makespan considering both computational resources and communication bandwidth effectively.

2.5 Big Data Taxonomy

Several years the organizations have captured transactional structured data using traditional relational data bases and used transactional query processing [4] [45] for the information extraction. This traditional way of computing exploited data patterns rather than the whole data. Later the decisions were made by retrospective justifications based on patterns of business operations or scientific experiments. In recent years, technologies are evolving to perform the investigations on the complete data which is becoming possible due to the lower costs and improvements in data capture, data storage and data analysis enabling the organizations to store the captured data from several sources like blogs, social media feeds, audio, video and scientific experiments. The options to store and process the data have expanded dramatically using technologies such as map reduce and in-memory computing [46] with highly optimized capabilities for different business and scientific purposes.

Due to the advancements in storage capacity, data handling and processing tools, the analysis of data can be carried out in real time or close to real time, acting on full data sets rather than on the summarized elements, leveraging tools and technologies enough to address the issue. In addition, the number of options to interpret and analyze the data has also increased, with the use of various visualization technologies. All these developments represent the context within which **"Big Data"** technology is placed. The benefits of the Big Data systems are described below.

- Provides handle for large volumes of un conventional information, together with traditional relational data.

- Allows to deal frequently, with rapidly changing information structures.
- Gain business value from unconventional information like weblogs, application logs, social media, etc.
- Capability to run analysis on unstructured data.
- Support of cost effective complex data processing and analysis using commodity hardware.

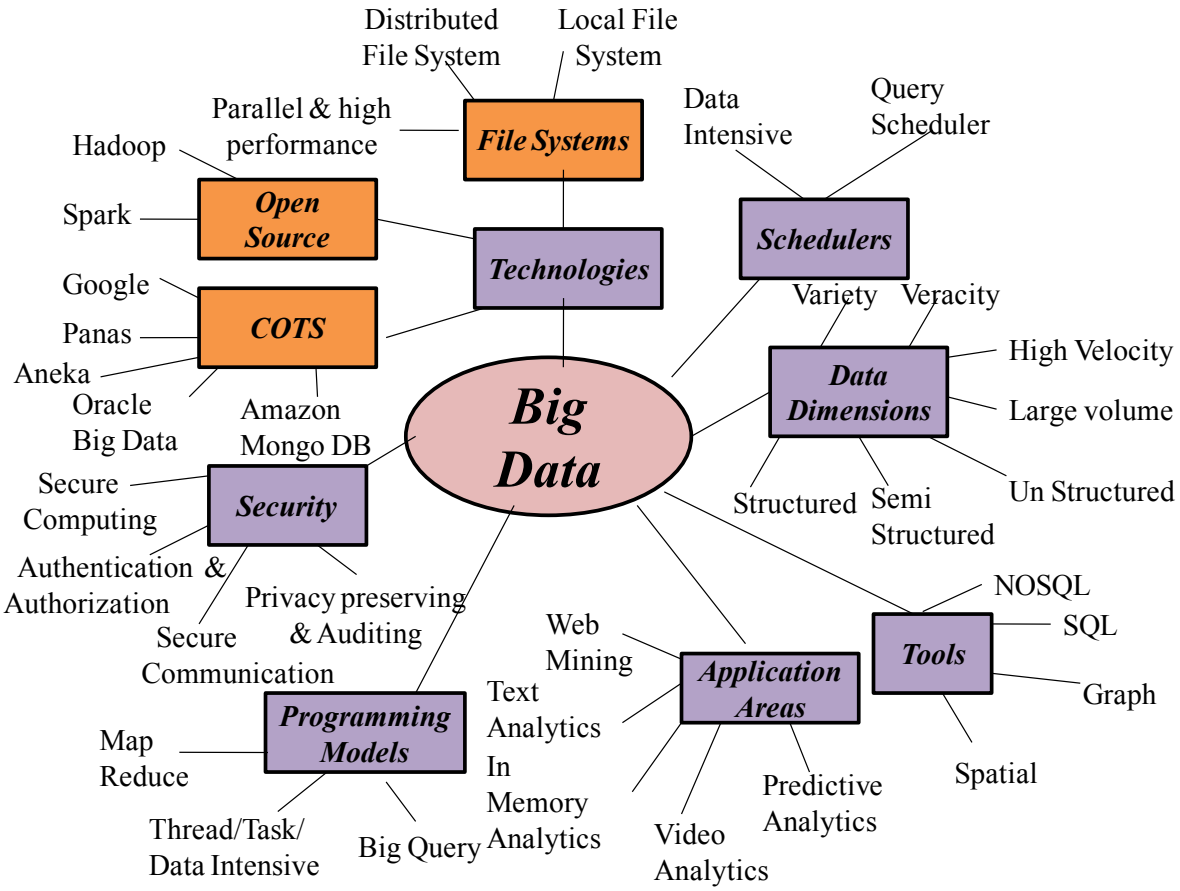


Figure 2.4. Big Data Taxonomy

Figure 2.4 depicts the taxonomy of Big Data and its elements are discussed below.

i) Big Data Dimensions

Big Data is characterized by volume, velocity, variety and veracity.

ii) Analytics - Application Areas

Analytics is the process of analyzing the data using statistical models, data mining techniques and computing technologies. It combines the traditional analysis techniques and mathematical models to derive information. Basically, Analytics and Analysis performs the same function,

however, analytics is the application of science to analysis. Big Data Analytics refers to a set of procedures and statistical models to extract the information from a large variety of data sets. A few major Big Data Analytics application areas are discussed below.

- **Text Analytics**

Text Analytics [47] is the process of deriving information from text sources. These text sources are forms of semi-structured data that include web materials, blogs and social media postings (such as tweets). The technology within text analytics comes from fundamental fields including linguistics, statistics and machine learning. In general, modern text analytics uses statistical models, coupled with linguistics theories, to capture patterns in human languages such that machines can "understand" the meaning of texts and perform various text analytics tasks. These tasks can be as simple as entity extraction or more complex in the form of fact extraction or concept analysis.

- **In Memory Analytics**

In memory analytics [46] is the process which ingests the large amounts of data from a variety of sources directly into the system memory for efficient query and calculation performance. In-memory analytics is an approach to querying data when it resides in a computer's random access memory (RAM), as opposed to querying data stored on physical disks. This results in vastly shortened query response times, allowing business intelligence (BI) applications to support faster business decisions.

- **Predictive Analysis**

Predictive analysis [48] is the process of predicting future or unknown events with the help of statistics, modeling, machine learning and data mining by analyzing current and historical facts.

- **Graph Analytics**

Graph analytics [11] studies the behavior analysis of various connected components, especially useful in social networking web sites to find the weak or strong groups.

- **Web Log Mining**

Web Log mining is the study of the data available in the web. This involves searching for the texts, words and their occurrences. One example for web log mining is searching for the words and their frequencies by Google Big Query Data Analytics [49] uses Google Big Query platform to run on the Google cloud infrastructure.

iii) Big Data Tools

NoSQL database management systems (DBMSs) are designed for use in high data volume applications in clustered environments. They often do not have fixed schema and are non-relational unlike the traditional SQL database management system (also known as RDBMS). Because they do not adhere to a fixed schema, NoSQL DBMS permit more flexible usage, allowing high speed access to semi-structured and unstructured data. Relational databases are based on relational algebra, which is based on set theory. Relationships based on set theory are effective for many datasets, however, where there is parent-child or distance of relationships are required; set theory is not very effective. In other words, relational databases will perform poor on the key-value pairs and where more data context is needed. Hence, several NoSQL programming tools and databases are increasing which are mostly based on graph theory to solve several data formats like key-value pairs, document databases, column family/big table databases, and graph databases. The brief descriptions of the tools are as below.

- **Key-Value Stores:** Key-value pair (KVP) tables are used to provide persistence management for many NoSQL technologies. The concept is; the table has two columns- one is the key; the other is the value. The value could be a single value or a data block containing many values, the format of which is determined by program code. KVP tables may use indexing, has tables or sparse arrays to provide rapid retrieval and insertion capability, depending on the need for fast look up, fast insertion or efficient storage. KVP tables are best applied to simple data structures and on the Hadoop Map Reduce environment. Examples of key-value data stores are Amazon's Dynamo [50] and Oracle's Berkeley DB [51] .

- **Document Oriented Database:** A document oriented database is a database designed for storing, retrieving and managing document oriented or semi-structured data. The central concept of a document oriented database is the notion of a "document" where the contents within the document are encapsulated or encoded in some standard format such as JavaScript Object

Notation (JSON), Binary JavaScript Object Notation (JSON) or XML. Examples of these databases are Apache's CouchDB [52] and 10gen's MongoDB [53] .

- **Column family/big table database**: Instead storing key-values individually, they are grouped to create the composite data; each column contains a row of data. This is useful for streaming data such as Web logs, time series data coming from several devices, sensors etc. examples are HBase[30]

- **Graph database:** A graph database contains nodes, edges and properties to represent and store data. In a graph database, every entity contains direct pointers to its adjacent element and no index look-ups are required. A graph database is useful when large-scale multi-level relationship traversals are common and is best suited for processing complex many-to-many connections such as social networks. A graph may be captured by a table store that supports recursive joins such as Big Table and Cassandra [114] . Examples of graph databases include Infinite Graph [54] from Objectivity and the Neo4j open source graph database [11] .

iv) Technologies

Big Data technologies majorly address data organization and computing. In the data organization, file organization plays a major role. The several technologies for Big Data file systems are described below.

a. File System

File system is responsible for the organization, storage, naming, sharing, and protection of files. It makes use of the indexing mechanisms such as inode, FAT etc., and directory services to arrange files into hierarchical structure. The major services offered by file systems are; read/write calls for the data retrieval and storage, access privileges for security. In general, file systems are classified into two types; local and remote or distributed. Local file systems organizes the files onto the local disk storages and are interfaced to the system directly, however, remote file systems organize the data onto the remotely located storage devices and offers a degree of transparency for access.

Big Data file management is similar to distributed file system, however the read/write performance, simultaneous data access, on demand file system creation, efficient techniques for file synchronizations would be major challenges for design and implementation. The goals in

designing the Big Data file systems should include certain degree of transparency to the user as mentioned below.

- Distributed access transparency: Unified directory services, clients are unaware that files are distributed and can access them in the same way local files are accessed.
- Location transparency: A consistent name space encompassing local as well as remote files without any location information.
- Concurrent access and transparency: consistency and coherence, with all the clients have the same view of the state of the file system. This means that if one process is modifying a file, any other processes on the same or remote systems that are accessing the files will see the modifications in a coherent manner.
- Failure handling: The application programs and the client should operate even with the few components failures in the system. This can be achieved with some level of replication and redundancy.
- Heterogeneity: File service should be provided across different hardware and operating system platforms.
- Scalability : The file system should work well in small environments from one machine to bigger number and also scale gracefully to huge from one hundred nodes to tens of thousands of nodes.
- Replication transparency: To support scalability, we may wish to replicate files across multiple servers. Clients should be unaware of this.
- Migration Transparency: Files should be able to move around without the client's knowledge.
- Support fine-grained distribution of data: To optimize performance, we may wish to locate individual objects near the processes that use them.
- Tolerance for network partitioning: The entire network or certain segments of it may be unavailable to a client during certain periods (e.g. disconnected operation of a laptop). The file system should be tolerant enough to handle the situations and applies the appropriate synchronization mechanisms.

Big Data File system could be broadly classified into two three categories as mentioned below. The features of these three file systems are similar; however they differ in the way they are served to the clients or application programs.

- Object based storage: Offer the services through web service calls such as REST/HTTP calls. Examples are Amazon S3 [67] , Open Stack Swift [68] Google drive, Sky Drive, Drop box etc.
- Block Storage: Network Level exposure through SCSI/ATA/iSCSI mounted to the server via operating system. Users can format and partition to create their own file systems. E.g. include SAN (Storage Area Networks) e.g. Amazon Elastic Block Storage [56] , Luster [107] Windows Distributed File System [108] , Google File System (GFS)[106] , IBM's GPFS [109] , Andrew File System (AFS) [110] etc.
- Network File storage: These file systems mounted to the system that are accessed via network such as Network File System (NFS), Server Message Block (SMB), CXFS etc.

b. Open Source Tools

Several open source tools for Big Data computing are described below.

- **Apache Hadoop [40] :** An open source reliable, scalable and distributed computing platform. It offers a software library and framework that allows distributed processing of large scale distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than relying on hardware to deliver high availability, the system is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures.

- **Spark [121] :** Apache Spark is a fast and general engine for large scale data processing. This covers Shark SQL, Spark Streaming, MLib machine learning and Graphx graph analytics tools. Spark can run on Hadoop YARN [40] cluster manager, and can read any existing Hadoop data.

- **Storm [133] :** Distributed real time stream oriented computing for real time analytics, online machine learning, continuous computation, distributed RPC, ETL etc. Storm topology

consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation.

- **S4 [134] :** Platform for processing continuous data streams. S4 is designed specifically for managing data streams. S4 apps are designed combining steams and processing elements in real time.

c. Commercial tools

Google offers Big Query [49] to operate on Google Big Tables [113] . Amazon supports Big Data through Hadoop cluster and also NoSQL support of columnar database using Amazon DynamoDB [50] . RackSpace [112] offers Hadoop framework on their storage platforms. Aneka [20] offers a data intensive platform to build map Reduce applications on .NET platform.

v) Programming Models and Schedulers

Several programming models such as data intensive, stream computing, batch processing, high performance/throughput, query processing, column oriented data processing described below.

- Map Reduce: data intensive programming model, with high level constructs for Map and Reduce functions. Map function does the assignment of the programming code/logic to the system to perform the computation on the data; whereas, Reduce function aggregates all the results of the map function by shuffling/sorting to generate the final result. Map Reduce programming is a type of recursive programming model to operate the similar logic on multiple distributed data sets, the examples include Hadoop Map Reduce [28] .

- Thread/Task Data Intensive Models: Thread programming model is used for high performance applications, where in the logic demands more number of cores or high end cores for processing within the reasonably less time. Task Programming models are applied for workflow programming models, e.g. Aneka [62]

- Machine learning tools: new generation of machine learning tools for decision making. Few tools available are Hadoop Mahout [111]

- Big Query Languages: new generation of Query Languages, examples are Google Big Query [49]

Big Data computing majorly need to address two types of scheduling mechanisms; Query scheduler and Data Aware Scheduling. Query schedulers addresses several mechanisms for querying the data managed by Big Data systems, Data Intensive schedulers addresses several computing mechanisms, examples include Capacity scheduler [41] , and Fair scheduler [43] .

vi) Big Data Security

Big Data project can uncover tremendous value for an enterprise, by revealing customer buying habits, detecting or preventing fraud, or monitoring real time events. However, a poorly run Big Data project can be a security and compliance nightmare, leading to data breaches. Big Data must be protected, to ensure that only the right people have appropriate access to it. Big Data security addresses several mechanisms for large scale high volume rapidly growing varied forms of data, analytics and large scale compute infrastructure. As, the data volumes and compute infrastructures are very large, traditional methods of computing and data security mechanisms, which are tailored for securing small scale data and infrastructure, are inadequate. Also, the use of large scale cloud infrastructures, with a diversity of software platforms, spread across large networks of computers, also increases the attacks. The onion model of defense for Big Data Security is depicted in Figure 2.5.

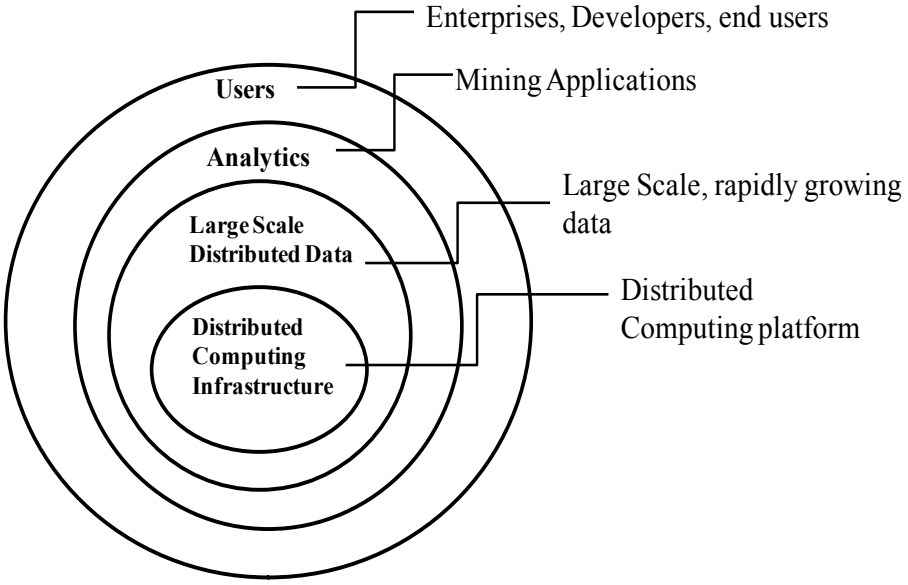


Figure 2.5. Big Data Security Onion Model of Defense

Below, we describe the several elements in the Big Data security Onion model of Defense.

- Distributed Computing Infrastructure: Mechanisms for providing security while data is analyzed over a multiple distributed systems.
- Large Scale Distributed Data: Privacy preserving mechanisms, encryption techniques for the data stored on large scale distributed systems.
- Analytics Security: Offering security to the applications and delivery of the right analytics tools to the end users.
- Users Privacy and Security: Confidentiality, Integrity and Authentication mechanisms to validate the users.

Big Data security refers to both data and infrastructure security, protecting the information throughout the data life span, from the initial creation on through the final disposal. The information must be protected while in motion, rest and processing. During its life time, information would pass through many different processing systems, demanding protection at each stage. Based on the security elements to be provided, Big Data security could be classified into three major areas and the related functions are described below.

a) User Level Security

Authentication, authorization and access control mechanisms for computing, and data access needs to be addressed based on the user privileges and roles, such as, Identity Access Management (IAM) Controlled access for all the users and resources in an automated fashion. This ensures that access privileges are granted according to policies and all individuals and services are properly authenticated, authorized and audited. Poorly controlled identity access management processes may lead to regulatory non-compliance because if the organization is audited, management will not be able to prove that company data is not at risk for being misused. Granular Access Control like Role Based Access Control (RBAC) approach permits access to the system resources based on their role policies to authorized users.

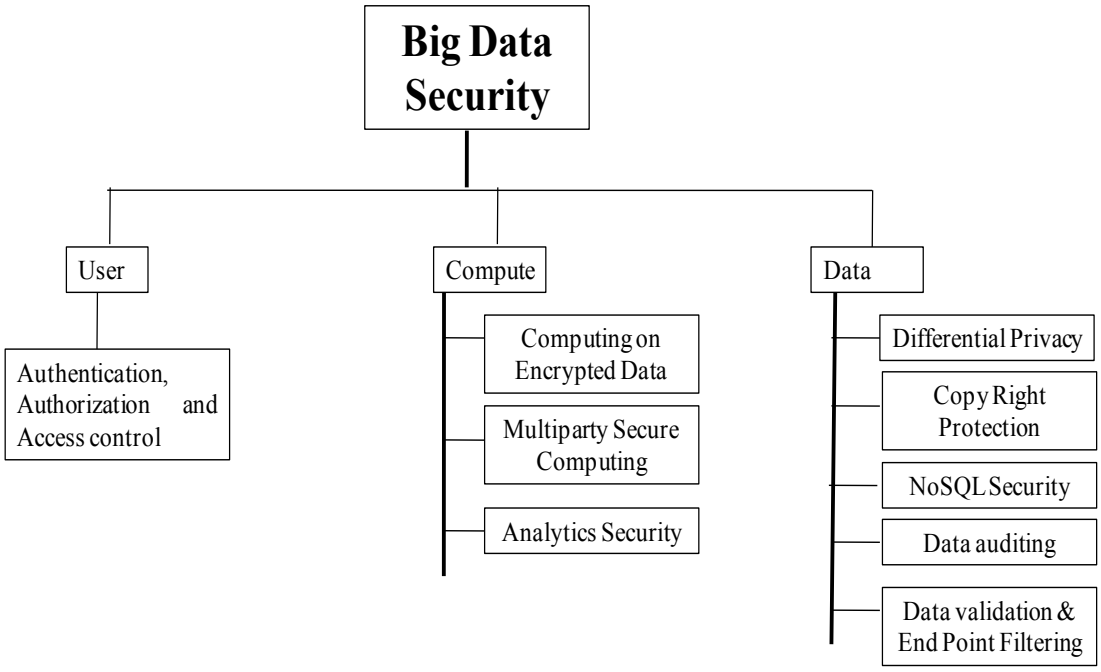


Figure 2.6.Big Data Security Components.

b) Privacy preserving computing

In the Big Data environment, there is a great potential for advancements in several fields like healthcare, education, financial/economic growth, social and scientific studies, however, there are huge risks involved regarding the loss of control over private data. The risks include sacrificing data integrity, losing the anonymity, profiled and giving up competitive edge when everyone has access to the privacy information. Hence, several methods, needs to be developed to obtain the benefits without/minimizing the risks without losing control on the private data. Several such methods are explained below.

a. **Computing on encrypted data:** Computing on the encrypted data without being decrypted. Since, the data is too large; decryption would be a costly affair. The technologies for computing on encrypted data includes-

i. Fully Homomorphic Encryption (FHE) [122] [123] : computing on cipher text and generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

ii. Functional encryption (FE) [124] : A type of public key encryption in which possessing a secret key allows one to learn a function of what the cipher text is

encrypting. This generalizes several existing primitives including Identity-based encryption (IBE) and Attribute-based encryption (ABE).

iii. Program obfuscation [125] : Method of creating obfuscated source or machine code that is difficult for humans to understand.

b. **Multiparty secure computing** [126] : parties jointly compute a function over their inputs, while at the same time keeping these inputs private.

c. **Analytics security:** Security mechanisms for the analytics offered as services over Big Data cloud platforms.

c) Data security at rest and move

Secure communications plays a major role in encrypting the data and user identity over the network, which is broadly addressed by cryptography mechanisms as discussed below. It is said that, there is a need to design a cryptographically secure communication framework. Sensitive data is routinely stored unencrypted in the cloud. The main problem to encrypt data, especially large data sets, is the all-or-nothing retrieval policy of encrypted data, disallowing users to easily perform fine grained actions such as sharing records or searches. Mechanisms such as Attribute Based Encryption procedures alleviates this problem by utilizing a public key cryptosystem where attributes related to the data encrypted serve to unlock the keys. On the other hand, there can be unencrypted less sensitive data as well, such as data useful for analytics. Such data has to be communicated in a secure and agreed-upon way using a cryptographically secure communication framework. Cryptographically enforced secure communication: To ensure that the most sensitive private data is end-to-end secure and only accessible to the authorized entities, data has to be encrypted based on access control policies. Specific research in this area such as attribute-based encryption (ABE) has to be made richer, more efficient and scalable. Access Controlled secure communication: Data communication based on levels of access mechanisms Processing on encrypted data: Full and partial homomorphic encryptions needs to be evolved to process on the encrypted data without fully decrypting.

Many Big Data use cases in enterprise settings require data collections from many end-point devices. For example, a security information and event management system (SIEM) may collect

event logs from millions of hardware devices and software applications in an enterprise network. A key challenge in the data collection process is input validation; validating that a source of input data is not malicious and filtering out malicious input from our collection. Input validation and filtering is a daunting challenge posed by non-trusted input sources, especially with bring your own device (BYOD) model. Both data retrieved from weather sensors and feedback votes sent by an iPhone application share a similar validation problem. A motivated adversary may be able to create "rogue" virtual sensors, or spoof iPhone IDS to rig the results. This is further complicated by the amount of data collected, which may exceed millions of reading/votes. To perform these tasks effectively, algorithms need to be created to validate the input for large data sets. Several elements of Big Data security are described in Table 2-4.

Table 2-4. Big Data security elements

S.no	Security element	Description
1	System Security	Big Data setup would be either confined to an enterprise or could be a large collection of several enterprises, social and scientific collection of disparate sources distributed system.
2	Privacy, Security and Confidentiality	Not revealing private and confidential information to unauthorized users. For example, in a mailing system, secrecy is concerned about preventing the users from finding out the passwords of other users.
3	Integrity	Improper modification of information. It is concerned with trustworthiness. For example, in a pay roll system integrity addresses the issues by preventing the employee not to update their salary details.
4	Availability	Availability of information resources. For example, the data should be made available as and when it is needed. An information system which is not available, when we need it, is at least as bad as none at all
5	Authentication and Authorization	Preventing resource access to the one who are not empowered to use.
6	Data Base and File Security	Role based access mechanisms.
7	Privacy Preserving Data Mining	Data sharing among the users thus protecting privileged information for mining applications.
8	Secure Computing	Multi-party computation preventing information disclosure of the party to another party.
9	Data Auditing	Auditing mechanisms for data integrity which could be offered by neutral third party providers.

10	Access Control	Access restrictions to the data at several levels.
11	Data Security at Rest and Move	Encryption mechanisms for the data at transfer as well for the data at rest.
12	Analytics Security	Preventing unauthorized access to applications thus retaining integrity.
13	Copy Right Protection	Protecting Intellectual Property (IP) rights for the data Protection among several coordinated and content owners.
14	Data Migration across multiple domains / Inter Cloud Migration	Protecting the data and managing applicable policies.
15	NoSQL Security	Security mechanisms need to be evolved for column, document, key-value and graph data models. In order to maintain fast access to data, NoSQL databases come with little built-in security. They have what's called BASE (Basically Available, Soft state, Eventually consistent) properties; rather than requiring consistency after every transaction, the data base just needs to eventually reach a consistent state.
16	Data validation and filtering	Finding the data which is needed further so as to minimize the expenditures of storage and computational cost.

2.6 Selected Big Data technologies

Big Data is gaining popularity and several open source and commercial technologies are under evolving to address Big Data Analytics either cloud or physical infrastructure. These include frameworks, scheduling models, file systems, NoSQL column oriented databases, machine learning tools and libraries. In this section we describe the technologies and present the table to describe the comprehensive list of the technologies. Table 2-5 summarizes the list of Big Data technology providers.

Table 2-5. Big Data Technologies.

Provider	Name	Type	File System	Tools	Description
Apache	Hadoop	Open source	Hadoop Distributed File System(HDFS)	HBase Map Reduce	A framework for Big Data. Cluster setup with replication of the data across the nodes. It is a good tool to address the applications which demand computing for processing large volumes of data.
Amazon	Elastic Map Reduce(EMR)	Commercial	HDFS	Map Reduce	Hadoop cluster offered over Amazon Infrastructure.
Amazon	DynamoDB	Commercial	Amazon S3 & Simple DB	Column oriented key/value pair	NoSQL implementation over Amazon Cloud.
Google	Big Query	Commercial	Google File System(GFS) (latest version is called Colossus)	Map Reduce	Data Analytics for the websites of your interest.

Open Stack	Hadoop Cloud	Open source and enterprise support	HDFS	Nova, Swift	Hadoop on virtual private Cloud infrastructure.
Panasas	ActiveStor	Commercial	PanFS	High performance Parallel File System.	Simulation and collaborative design.
Manjrasoft	Aneka	Commercial	Windows distributed file system.	Map Reduce.	PaaS model for the development of .NET based map reduce applications development over cloud infrastructure [66] , [104]
MongoDB(formerly 10gen)	MongoDB	Open source and enterprise support	Uses BSON (Binary JSON) serialization format to store the documents and make remote procedure calls. Uses GridFS file systems for storing document files larger than 16MB [127]	Document oriented database over a cluster of nodes with indexing and searching capabilities. Applies NoSQL queries.	Full index support, replication and high availability, auto sharding (horizontal scaling), query, Map/Reduce functionality with flexible aggregation and data processing.

Google	Google Spanner	Commercial	Temporal multi version database, schematized semi-relational tables.	NewSQL	Globally distributed database by Google, the successor to Big Table [128] . Data is stored in schematized semi-relational tables; data is versioned, and each version is automatically time stamped with its commit time; old versions of data are subject to configurable garbage-collection policies; and applications can read data at old timestamps. Spanner supports general-purpose transactions, and pro-vides a SQL-based query language.
Apache	CouchDB	Open source	Works on the base operating file system	Document databases with NoSQL based on Erlang implementation.	CouchDB is ideal for web applications that handle large amounts of loosely structured data.
Neo Technology	Neo4j	Open source and enterprise support	Cross platform works on the operating file system	Graph database	Embedded disk based fully transactional java persistence engine that stores data structured in graphs rather than in tables.

EMC[2]	Scale-Out NAS for Big Data Systems	Commercial	OneFS	Support for petabytes of storage and supported by REST, object based, block and file level access with data life cycle tier tools.	Unified storage technology and operating system with integrated system of file system, volume manager and data protection schemes.
Storm	Apache	Opensource	Stream/In memory	Stream computing	Similar to S4 , software for streaming data-intensive distributed applications

Table 2-6. Hadoop ecosystem

Sno	Tool Name	Description
1	HDFS	Distributed redundant file system for Hadoop
2	HBase [30]	A key-value pair/column oriented database system that runs on HDFS scaling to billions of rows
3	Hive [78]	A system of functions that support data summarization and ad hoc query of the Hadoop Map Reduce result set used for data warehousing. A Data warehouse system with SQL like access
4	Pig [103]	High Level language for managing data flow and application execution in the Hadoop environment
5	Mahout[111]	Machine Learning system implemented on Hadoop. A Library of machine learning and data mining algorithms
6	Zookeeper[40]	Centralized service for maintaining configuration information and naming, providing distributed synchronization and group services with coordination
7	Sqoop [40]	A tool designed for transferring bulk data between Hadoop and structured data stores such as relational databases

8	Ambari	Deployment, configuration and monitoring tool
9	Flume	Collection and import of log and event data tool
10	HCatalog	Schema and data type sharing over Pig, Hive and MapReduce
11	MapReduce	A Parallel computing programming model on server clusters using HDFS
12	OoZie	Orchestration and Work flow Management System
13	Whirr	Cloud agnostic deployment of clusters
14	Default scheduler – FIFO	Firs In First out, scheduler with priority support. Processes one job at a time
15	Fair scheduler	Pluggable Scheduler provided from Face book. Group scheduling, however each group gets equal time slots
16	Capacity scheduler	Pluggable Scheduler provided by Yahoo. Similar to FIFO scheduling within each queue, however limits the maximum resources per queue.

2.7 Discussions and Summary

In this chapter we have discussed conventional data models and its short comings in handling large scale, variety forms of data, followed by, emerging technologies such as Big Data computing and their relevant paradigms computing models for solving several applications in scientific, social networking and business domains. We have discussed Cloud computing technologies which are emerged as large scale elastically scalable infrastructure technologies as services based on demand. We have discussed taxonomy of Cloud computing, its deployment models, and several applications of business, scientific and social networking applications that Clouds are addressed as service oriented computing. We have presented taxonomy of Big Data computing, and illustrated several elements of it such as application domains, data dimensions, File Systems, available open source technologies and their tools, security aspects of Big Data, Scheduling approaches for data intensive applications, and programming models such as MapReduce, Task and Thread models. In the next chapter, based on the survey we have presented in this chapter, we will discuss the architecture and framework for Big Data Computing in Clouds. Later, we will identify gaps in both Clouds and Big Data technologies, and identify two key components of the framework such as scheduling data intensive applications in Clouds, and data organization/processing models for scientific computing over distributed file system using MapReduce computing models for image processing applications.

Chapter 3

Big Data Clouds Framework – A Proposal

This chapter introduces Big Data Clouds architecture and Framework for large scale data organization, processing on dynamically scalable Clouds infrastructure. First, we describe the layered architecture followed by framework and present the elements that are necessary for processing big data over Clouds. We illustrate the functions of each of the layers and how they are distinguished from the Cloud computing layered architecture. Based on the described layered architecture, we identify the key elements and describe the framework. In the framework, we identify the several key components, and describe the functions of each of the components those are mentioned.

3.1 Layered Architecture

Big Data Cloud architecture is similar to cloud computing architecture and adopts the four layered model, the layers from bottom to top are cloud infrastructure, Big Data fabric layer, Big Data platform and Big Data analytics layer as shown in Figure 3.1.

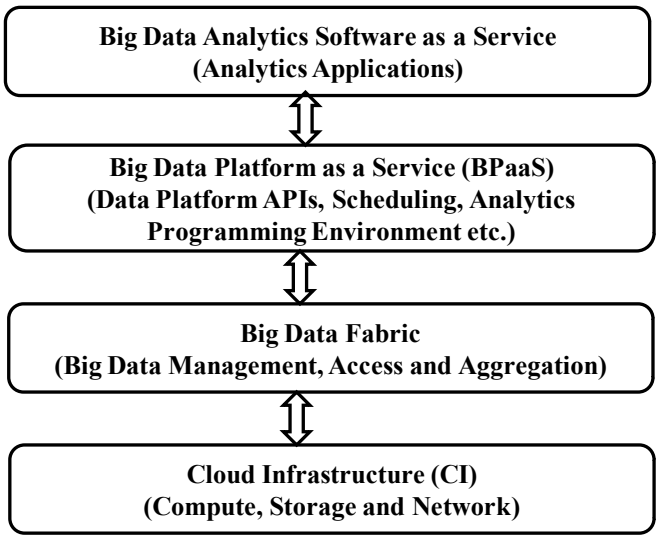


Figure 3.1. Big Data Cloud Reference Architecture.

The cloud infrastructure layer handles the scalable dynamic infrastructure that could be delivered either from clouds or from physical infrastructure. The second layer, Big Data fabric layer; addresses the several tools for data management, access and aggregation. The third layer is the platform layer which addresses the tools and technologies for data access and processing, programming environments for analytics development and scheduling etc. the top layer is the Big Data analytics Software as a Service offers several analytics. The brief functional description of each of the layers is described below.

- **Cloud infrastructure (CI):** Large scale management of dynamic and elastic scalable large infrastructure of compute and storage resources as services. Virtualization technologies are used for on demand provisioning of the resources based on SLAs and QoS parameters. The services rendered by this layer are

    a. Offers the large scale infrastructure to setup Big Data platform on demand.
    b. Dynamic creation of virtual machines for Big Data computing.
    c. Large scale offerings for File/Block/Objected based storages on demand.
    d. Ability to move the data seamless way across the storage repositories.
    e. Able to create the virtual machines and auto mount the file system to the compute node.
    f. Information defined data storage, access and retrieval mechanism.

- **Big Data fabric:** This layer addresses standard tools and APIs to access storage, compute and application services. This layer offer standard interoperable protocol APIs to connect multiple cloud infrastructures as specified in the standards [65].

- **Big Data Platform as a Service (BPaaS):** The third layer from bottom which offers core middle platform services to access storage/data services, compute based on SLAs and QoS. This layer consists of middleware management tools such as schedulers, data management tools such as NoSQL tools for data processing. Big Data Platform layer address development of platform tools and SDKs to address Big Data Analytics.

- **Big Data Analytics:** Big Data analytics offered as software services from the Big Data cloud providers. Users can quickly use this analytics software without investing on infrastructure and pay only for the resources consumed. This layer organizes the

repository of software appliances and quickly deploys on the infrastructure and delivers the end results to the users, the pricing would be computed based on the usage, quality of service provided etc.

3.2 Framework

Big Data clouds architecture is similar to cloud computing model, however, the services offered by Big Data layers are specific to the development of Big Data analytics. Cloud computing providers offer the infrastructure and platform tools for the usage of the infrastructure. Also, they may offer software services readily available for usage. However, Big Data clouds are specific to the development of analytics for information mining, using Cloud computing technologies in due course. Major layers, sub layers and each layer how related to layered reference architecture is shown in Table 3-1.

Table 3-1. Layers mapped to reference architecture

S. No	Layer name	Sub layers	Reference layer of architecture
1	Infrastructure	Resource and Interface layers	Cloud Infrastructure (CI), Big Data fabric
2	Big Data Platform	Foundation, Runtime, Programming modeling layer, SDK	Big Data platform as a Service
3	Applications	Analytics, Big Data services	Big Data Analytics software as a Service

Big Data cloud layered architecture from bottom to top, layered components, interaction among the layers are described below. Layers are further classified into sub layers based on the specific services offered by each one of them. Components of Big Data clouds in each of the layers are shown in Figure 3.2.

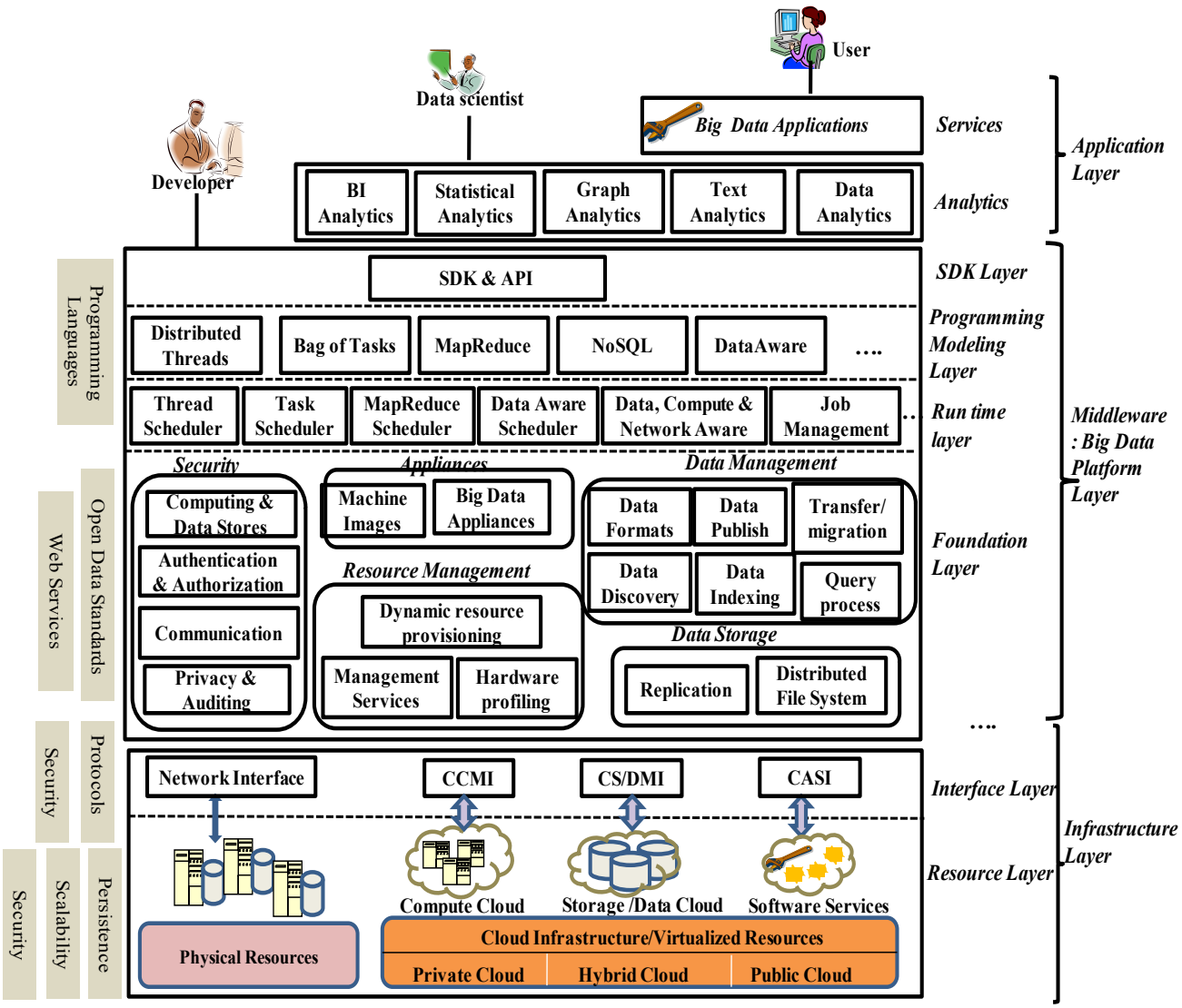


Figure 3.2. Big Data Cloud Layered Components.

3.2.1 Infrastructure layer

The functions offered by this layer are similar to the Infrastructure as a Service (IaaS) layer of the cloud computing model. The services offered by this layer are effective management and delivery of the compute, storage, data and networking infrastructure. This layer is further classified into two sub layers; resource and interface layers. Resource layer facilitates compute, storage and data services either on physical or virtual environments. Physical environment is similar to data centers without virtualization enforced and is similar to cluster setup in the local network. In the case of virtual environments, it could be a private/public/hybrid cloud provider who offer services based on the consumption. The functioning of resource layers over physical

and virtual environments is similar, however, virtual environments offer high utilization of the resources, on demand resource provisioning and highly scalable, however, endure performance degradations due to enforced virtualization technologies. Below, services offered by resource and interface layers are described in brief.

3.2.1.1 Resource layer: Resource layer handles both physical and cloud resources as discussed below.

a) Physical resources: non-virtualized compute and storage resources delivered via local data centers or in-house available. The resources may be accessed via standard protocol and networking interfaces.

b) **Virtualized/Cloud resources:** The resources are delivered by several cloud providers like compute, storage and application clouds. Compute clouds offers several scalable machine instances on demand, Storage / Data Clouds offers either storage repositories or data online, and sometimes both. Software services are similar to applications offered as services over the cloud. The cloud infrastructure may be either from private clouds or public clouds, and sometimes both. However, the access mechanisms and security implementations will differ depending on types of clouds were chosen. Below, we illustrate the functions of Compute, Storage/Data Cloud and Software Services.

i) Compute Cloud: large pool of compute machine instances to serve the demands. Compute machines could be created at run time and the data needed for analytics purpose may be made available dynamically.

ii) Storage/Data Cloud: Storage clouds offers a pool of the storage space where in files required for analytics could be placed. However, Data Cloud offers the storage space along with the Data necessary for compute. Such data or storage could be offered as Block Storage or Object Storage.

iii) Service Cloud: Several analytic tools which can be provisioned on demand.

3.2.1.2 Interface Layer

Interface layer facilitate open standards, protocols based on web and interoperable services. The major challenges include; interoperability between heterogeneous hardware and storage infrastructure, and migration/access across various cloud providers. Interface layer offers standard interfaces [65] to access Compute resources, Storage resources and application services. This layer could be classified into four components based on the services rendered, such as networking interface protocols, cloud compute management interface (CCMI), cloud Storage/Data management interface (CS/DMI) and Cloud application services interfaces (CASI). The detailed description for each of the components is given below.

a) Network Interface: This interface allows several physical devices access through standard networking interfaces and protocols. This includes accessing the compute instances via terminal services or web consoles. The storage devices can be mounted to the local compute machine instances or access via separate networks such as NFS protocols.

b) CCMI (Cloud Compute Management Interface): functional interfaces that applications will use to create virtual machine nodes on demand. As part this interface the client will be able put a request to the Compute Cloud Infrastructure and will be able to create or destroy the virtual nodes. There is a need to evolve interoperable system interfaces for accessing the compute machine instances across various public providers of compute machine instances.

c) CS/DMI (Cloud Storage / Data Management Interface): A functional interface that applications will use to create, retrieve, update and delete data elements from the Cloud. As part of this interface, the client will be able to discover the capabilities of the cloud storage offering and use this interface to manage container and the data that is placed in them. In addition, Metadata can be set on container and their contained data elements through this interface. This interface is also used by administrative and management applications to manage container, accounts, security access and monitoring/billing information, even for storage that is accessible by other protocols. The capabilities of the underlying storage and data services are exposed so that clients can understand the offering. Various CDMI interfaces are

- **Amazon S3**: Amazon S3 stands for Simple Storage Service, stores the data objects within the buckets, comprised of a file and optionally any metadata that describes that file. To store an object in Amazon S3, the file can be uploaded to the bucket and permissions can be set on the object as well as on the metadata. Buckets are the containers and there can be more than one bucket.
- **Open Stack Swift**: Object Based Data Storage system exposes the storage via REST API and stores a large amount of unstructured data at low cost.

d) CASI (Cloud Application Services Interface): set of Web Services that exposes the published applications through standard web protocols. This also involves application virtualization methodologies to serve only the needed applications as services from the cloud providers.

3.2.2. Big Data Platform layer

This is a middleware layer that is further categorized into four sub layers based on the functionality, they are, foundation layer, runtime layer, programming modeling layer and software development kit (SDK) layer. The Foundation layer offers mechanisms for resource management, data storage, data management, security and virtual appliance. The Runtime layer addresses several scheduling mechanisms and job management mechanisms. The Programming Modeling layer employs several programming standards; the SDK layer offers Application Programming Interfaces (APIs) for programming in several languages. The detailed description of the layers is given below.

3.2.2.1 Foundation layer: This is the core part of the middle ware layer, which interfaces with the resource layer. This layer mainly classified into components such as resource management, data management, appliances, data storage and security.

a) Resource management: Resource management consists of the following components.

- **Management services**: The services to manage the underlying physical resources. These can be middleware services to track of the available resources. The management services include applications to monitor the resource utilizations for data and compute such as computing resources availability, storage availability etc.

- **Hardware profiling**: The information services to retrieve the information regarding the available resources such as RAM, Network Bandwidth, compute load etc.
- **Dynamic resource provisioning**: Facilitates the resources at run time from the virtualized resources.

b) Data management: This mainly deals with data formats, discovery, and publishing mechanisms.

- **Data Formats:** Data formats service provides to store the data in various types of forms which include structured, unstructured and semi structured. Search mechanisms services offer various query mechanisms to search for the data of interest, Sharing allows various access privileges.

- **Data transfer/migration:** The mechanisms either pull or push the data for processing, automatic syncing of the files to the Big Data systems. It also contains tools that are necessary for migration of the existing structured/unstructured data to cloud Big Data workloads.

- **Data discovery mechanisms:** Several mechanisms to find the location of the data. This can be performed with the query mechanisms or looking for the meta data contents. Without data discovery mechanism, data infrastructures are merely a storage area, which would retrieve the data from the location you know, however searching for a data which you don't know the locations, would throw new challenges for designing search engines. Unlike Google like search engines where data is discovered in the web based on a key word, however, in scientific environment keyword-based search may be insufficient. Users may want to be able to search for number ranges in specific measurements, geospatial locations etc. For this reason, there could be a number of data discovery mechanisms in a single data infrastructure, and they may be driven by specialized user requirements. Discovery needs to be established as an application level mechanism to enable users to build their own discovery tool for search, analysis and visualization. Several filters could be required when choosing and describing data. Dedicated discovery mechanisms for specific communities need to be evolved. Technologies for data discovery might include visualization, structural query mechanisms, semantic query etc.

- **Data publication mechanisms:** Data Publication allows to publish the data and its characteristics via standard web interfaces and finally storage and indexing allows the storage and indexing mechanisms of the Big Data in the storage clouds.

- **Data indexing:** Indexing mechanisms are needed to speed up the process of accessing the data. Several Data Indexing mechanisms need to be explored for data redundancy, replication.
- **Query process:** Data Processing mechanisms and standard Query languages.

c) Appliances: As compared to "do it yourself" self-configuration, appliances eliminates the time consuming efforts of choosing and configuring hardware, determining the proper software components, integrating and tuning the overall configuration.

- **Machine image**: The repository of machine instances for creating the systems on demand. Virtual Machine Manager and Machine Image Instance, the former, is the scheduler; manages the life cycle of the virtual machines such creation and destroy and the later is Machine Image Instances are basically Image Templates for creation of virtual machines by virtual machine manager.
- **Big Data appliances**: The repository and management of Big Data Appliances for specific Big Data Analytics.

d) Data storage

- **Replication procedures**: Several replication procedures for duplicating the data onto multiple storage repositories for data redundancy, high availability and high performance Data Transfer. Data Replication allows Big Data Replication for efficient processing and backup mechanisms, Security component allows secured way of data access and transmission. Data Replication enables some of the most important functions like backup and restore, application performance and data integration. There is a need to replicate Big Data repositories that were previously confined to a single location. Tervela [69] accelerates Big Data Replication by efficiently duplicating to multiple sites with ease through one of two methods such as Changed data capture or parallel replication. Big Data Replication [61] address the issues like;

i. Providing extremely rapid access to data from multiple sources, even in a mixed workload.
ii. Reducing the drag on multi-way joins for complex queries.
iii. Accelerating reporting for faster analysis, review and decision making.
iv. Backup and Disaster Recovery.

- **Distributed File System:** File system which stores the data onto multiple distributed storage repositories, maintains the indexing of the data and offers various logical views of the entire data which is available within the system.

3.2.2.2 Runtime layer: This layer is concerned about workload handling with the help of several scheduling mechanisms. Examples include Thread, Task, Map Reduce, Data and network aware scheduling, batch job management based on the type of computation needed. Below, we briefly, discuss the functions and characteristics of several types of schedulers.

- **Thread scheduler**: Thread Scheduling exploits the available cores/processors effectively by utilizing either local system or remote system resources. Local threads execution could use shared memory, however, for remote execution; objects migration would take place. Thread scheduling, is applicable for problems that are recursive, multiple data streams but applied on a single instruction. Thread scheduler determines the best resources for running the several spawn independent threads on available resources/cores. Thread Scheduling addresses high performance computing problems.

- **Task scheduler:** Distributed processing of tasks on several computing nodes. Task Scheduling solves the high throughput problems by determining the best available resources for execution.

- **Map Reduce scheduler:** Type of Data Aware Scheduling which maps the compute process to the data nodes. After the completion of the process, the results are consolidated onto a single node for final result.

- **Data Aware scheduler:** Jobs execution, knowing the best available storage locations for execution or transfer the data to compute nodes from the best available store repositories. The process could depend on computing the best replicated site that

minimizes the compute time. This may apply Data Parallelism to pull/push the data to the compute nodes. Map Reduce is an example of Data Aware Scheduler.

- **Data Compute and Bandwidth aware Scheduler:** Considering compute, network and data to solve the data intensive scheduling. The techniques applied for this type of scheduling are

i. **Parallel Data Extractor**: Parallel Data Extractor is the high performance Data Transfer module. It enables extraction of transfer of data from storage clouds to the compute node. This module pulls the data from the storage repositories by establishing multiple parallel lines between storage clouds to compute resources. Parallel Data Extractor identifies possible data storage resources and identifies the amount of data to be pulled from each of the storage repositories.

ii. **Scheduler**: The scheduler which effectively maps a set of jobs to the computing nodes, the scheduling would depend on heuristic approaches. Big Data schedulers could pick up the best computing nodes or may quickly clone the virtual machines and perform the computation by applying effective data aware scheduling techniques.

3.2.2.3 Programming modeling Layer: Several programming models to solve the Big Data Problems. This may include; coarse/fain grain programming models for thread, tasks and Data Intensive. It also addresses data handling and query programming models for NoSQL databases.

SDK layer: The programming APIs to solve Big Data Problems. This could be Java, C, C++, and C# based APIs.

3.3 Application / Analytics layer: Analytics and Application services are the parts of this layer. Analytics layer offers several statistical, deterministic, probabilistic, machine learning techniques. Services layer, hosts the end user applications. The brief description of analytics and services layer is described below:

- **Analytics layer:** This layer is responsible for management of APIs and development of several data analytics specific to domain. This layer offers SDKs; APIs & Tools for the analytics development and also responsible for several management Interfaces development for monitoring the Big Data environments are Statistical Models, Graph Analytics, Business Analytics, Text Analytics and Data Analytics.

- **Services:** Application Services readily available for the end users. This could use underlying analytics and scheduling mechanisms to run on the infrastructure.
- **Users**: The several users of the system are Developer: Big Data general purpose application designer. Data Scientist: Data Analysts who design the Analytics applications. This could be Business Analytics, Scientific Explorations etc. End Users: Analytics users of the system.

3.4 Elements of Big Data Clouds

Big Data differs from the traditional data in several ways, in which the data is organized, processed, queried and delivered to the Data Scientists with a wide variety of analytic tools. In this section, we present Big Data Cloud shown in Figure 3.3 and describe several components of it.

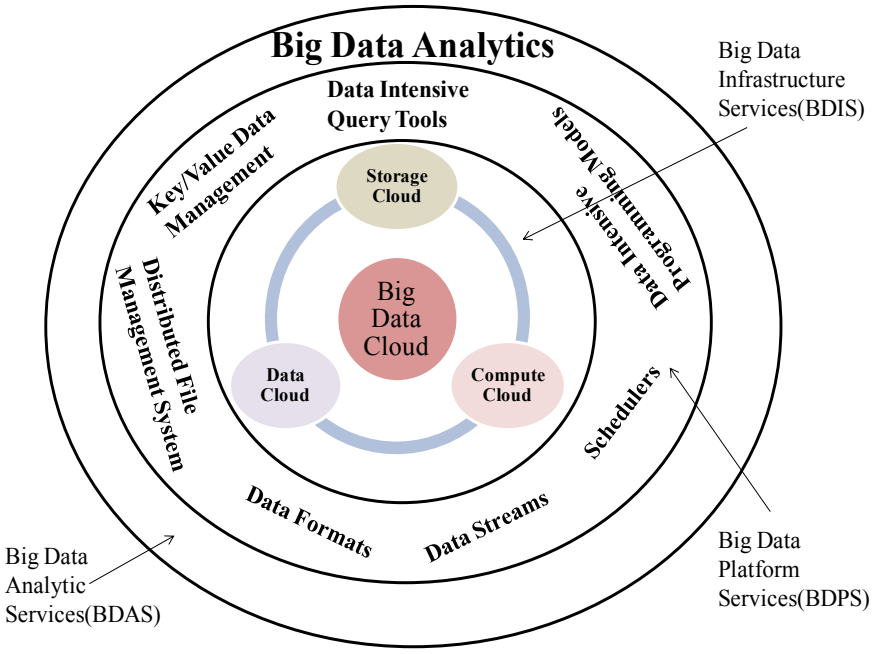


Figure 3.3. Big Data Cloud components

i) **Big Data Infrastructure Services (BDIS)**: This layer offers core services such as compute, storage and Data Services for Big Data Computing as described below. The several services offered by Infrastructure services are discussed below.

a) Basic Storage Service: Provide basis service for data over physical and virtual infrastructure, such as create, delete, modify and update. Provide unified data model for all kinds of data.

b) Data Organization and Access Service: Data Organization provides management and location of data resources for all kinds of data, and selection, query transformation, aggregation and representation of query results, which leads to exploitation of RDF-RDBMS semantic querying to select data of interest.

c) Processing Service: Mechanism to access the data of interest, transfer to the compute node, efficient scheduling mechanism to process the data, programming methodologies, various tools and techniques to handle the variety of Data Formats.

d) Data Formats Support: The service should provide mechanisms to handle various types of data for processing and aggregating the information from multiple sources. Types of data to support are

- Relational Data such as Tables/Transaction/Legacy Data
- Text Data: Web Documents such as HTML
- Semi-Structured Data: XML
- Graph Data: Social Network Semantic Web(RDF)
- Streaming Data: Video

The elements of BDIS are described below.

- **Compute Clouds:** On Demand provisioning of compute resources, which could expand or shrink based on the analytics requirements.
- **Storage Cloud:** Large volume of storage offered over the network. The storages offered include; file system, Block Storages and Object Based Storage. Storage clouds offer to create file system of choice and also elastically scalable. Storage Clouds can be accessed based on the pricing models which are usually based on data volumes, transactions/data transfer. The several services offered by Storage Clouds are described below.
  - **File Storages:** which were raw file systems similar to the direct disks attached to the storage. The examples include SkyDrive, Dropbox, Google Drive, and Amazon Cloud Drive.

These are useful to store the backup and useful to access the data from the local system from anywhere.

- **Block Storages:** These are the storages which can be mounted to the system and allows formatting the raw file system and creating a one of your own. These works through internet protocols with high bandwidth networks. Examples include Amazon EBS (Elastic Block Storage)], Open Stack Block Storage.

- **Object Based Storage:** The file system offered as container over the network. The files can be accessed only through web service calls such as HTTP/REST. Examples are Open Stack Swift Storage [68] , and Amazon S3 [67] .

- **Data Clouds:** Data Clouds are similar to Storage Clouds, however, unlike storage space delivery, they offer data as a service. Data Clouds offer tools and techniques to publish the data, tag the data, discovery the data and process the data of interest. Data Clouds operate on domain specific data leveraging the Storage Clouds to serve Data as a Service based on four step of "Standard Scientific Model" [64] such as data collection, analysis, Analyzed reports and long term preservation of the data.

ii) **Big Data Platform Services (BDPS):** This layer offers schedulers, Query mechanisms for Data retrieval, Data Intensive Programming models to address several Big Data Analytic problems.

iii) **Big Data Analytics Services (BDAS):** Big Data Analytics as Services over Big Data Cloud infrastructure.

Big Data cloud is an emerging technology to quickly build analytics over cloud infrastructure without worrying about the infrastructure setup and platforms to deploy Big Data workloads. Big Data clouds offers several tools and technologies to quickly transform the organizations data workloads into a Big Data computing workloads and offers platforms for statistical, predictive analytics tools as services helping organizations to understand and make use of the underlying data for intrinsic information extraction. Big Data clouds offer three major services; Big Data setup without investment on the infrastructure, on demand platforms tools to access the data of the organization or data services from several data providers and on demand data platform tools for the design and development of application specific analytics.

Big Data clouds could be classified into the following types based on the setup and the services they render to the users.

- **Public Big Data clouds:** offered by public providers to setup Big Data computing on the publicly available infrastructure and the cost is based on the consumption. The examples include Amazon Big Data clouds, Google Big Data clouds etc. This type of clouds enable to setup owns Big Data environment on demand based on pay-as-go models.
- **Private Big Data clouds:** Big Data clouds setup by the organizations for their own purpose. These are not accessible to the outside networks. Groups which are part of the private network only have an access to this setup.
- **Hybrid Big Data clouds:** Federation of public and private Big Data clouds either for data sharing or scalability.
- **On premise Big Data Clouds – Dedicated:** Big Data setup in the private/ local data centers such as Big Data setup over cluster of nodes.
- **Big Data access networks and computing platform:** These are the special type of Big Data setups where Data scientists could work on building the analytics by accessing the Big Data from several data providers using Data APIs offered, and computing infrastructure offered by compute providers.

Big Data computing examines the large scale data to extract useful information for decision making uncovering hidden patterns and unknown correlations, over a large scale infrastructure and storage resources for solving analytics. Big Data and Clouds **"Big Data Cloud"** offers an environment for Data Intensive application developments over a large scale, distributed compute and storage infrastructures. In this section Big Data Cloud characteristics and advantages, followed by how it is different from other Cloud technologies like Storage Cloud and Data Cloud are described. The major characteristics of Big Data Cloud are described below.

- **Large scale distributed compute and data storages:** wide range of computing facilities with seamless access to scalable storage repositories and Data Services.
- **Information defined data storage:** Meta data based data access instead of path and filenames.

- **Distributed virtual file system:** File system could be dynamically created and mapped to the compute node for data intensive computing.
- **Seamless access of computing and data:** Transparent access to large scale data and compute resources.
- **Dynamic creation of data storages and compute resources:** Able to handle dynamic creation of virtual machines and able to access large scale distributed data sources.
- **High performance data and computation:** Compute and data should be high performance driven.
- **Multi dimension data handling:** Support for several forms of data with necessary tools for processing.
- **Analytics platform services:** able to develop, deploy and usage Analytics over the environment.
- **High availability of computing and data:** Replication mechanisms for Compute and Data.
- **Platform for data intensive computing:** Support for both traditional and emerging Data Intensive computing models and scalable deployment and execution of applications.

Big Data Cloud is an infrastructure that unifies distributed compute and data sources in order to provide computing and data management support for a wide range of Data Analytics. The different types of computing support offered by Big Data Clouds are described below.

- **Data and compute intensive Platform:** offers high performance application delivery with large scale compute and data services, with several programming models.
- **On demand computing support:** Computing could be delivered as and when needed.
- **Data as a Service:** Data Services could be priced and delivered on demand.
- **On Demand Platforms for Analytic:** Analytics could be made available on line. Big Data users could use Analytics on demand basis.
- **Multi Dimension Data Addressing:** Supports several forms of data like text, audio, video including Relational data support.

- **On Demand Big Data Cloud Platform:** customize Big Data configuration with the choice of storage devices, platforms, easy pricing and simple scalability and setup your own Big Data setup: offers tools and platform to setup your own Big Data computing network in short time rapidly.

Big Data computing consists of applications that produce, manipulate or analyze variety forms of data in Peta bytes and beyond. The data is organized as either collections or data sets and are stored on mass storage systems and are offered via access networks. Big Data technology offers several data platform services offered by various service providers like social media, scientific experimental data, sensor networks, business studies. Figure 3.4 depicts Integrated Cloud and Big Data access networks for Big Data analytics which provide data platform and Cloud infrastructure integrated as unified Big Data computing Platform.

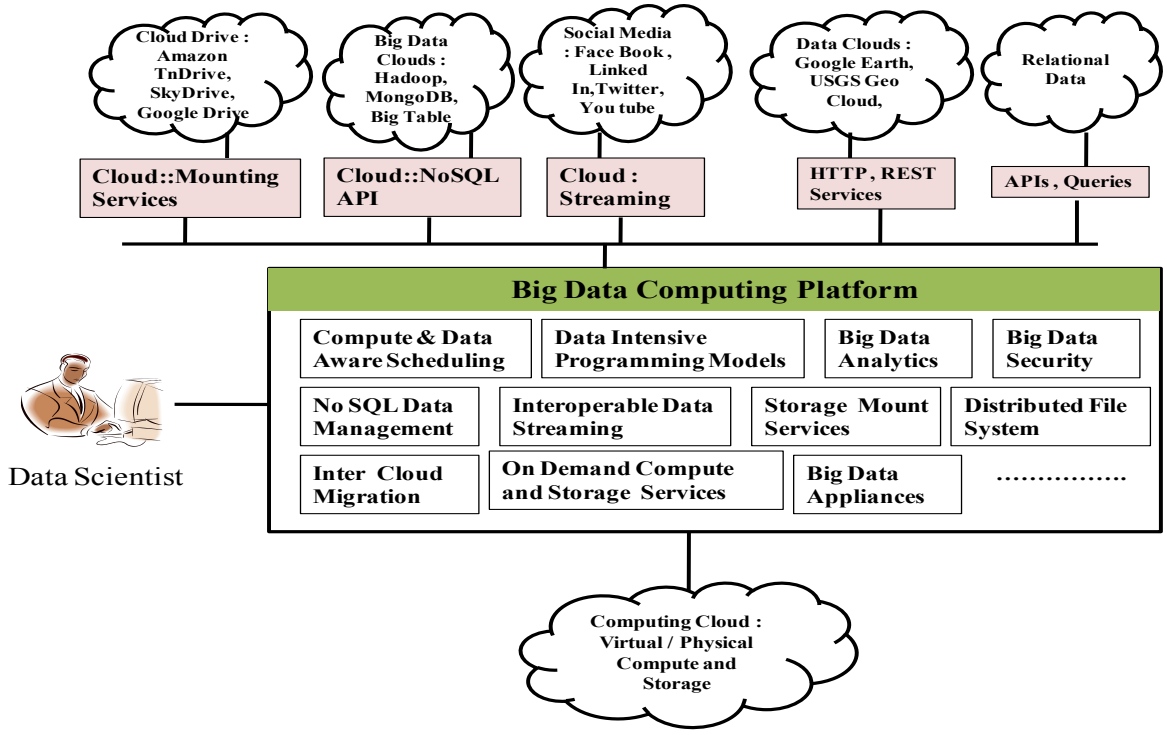


Figure 3.4. Integrated Cloud and Big Data Compute Network.

The several Data content from several sources like social media, web logs, scientific studies, sensor networks, business transactions etc. are growing rapidly. Deriving useful information for decision making from such large data, fusing the information from several sources would be a challenging task.

The elements of Big Data access network are; Data services, Big Data computing platform, Data scientist and Computing Cloud described below.

- **Data and Platform Services**: Several data providers who offer both data and platform services to access the data. For example; Google Data APIs (GData) [132] provide a simple protocol for reading and writing data on the web for several services like Content API for shopping, Google Analytics, Spreadsheets and YouTube.
- **Big Data Computing Platform:** Platform for Big Data managed services.
- **Data scientist:** Analytics developers.

➢ Storage Cloud Vs Data Cloud Vs Big Data Clouds

Big Data Clouds offer services for the development of analytics over a large scale dynamically scalable compute and storage infrastructures. As, there are already Data and Storage Clouds that offer services for data and storage related technologies over cloud infrastructures. To understand Big Data Clouds, and how this is different from Storage/Data Clouds, we briefly describe key properties of these technologies in Table 3-2.

Table 3-2. Comparison for Storage, Data and Big Data Clouds

S.no	Property	Storage Cloud	Data Cloud	Big Data Cloud	Remarks
1	Service offered	Storage as a Service	Data as a Service	Big Data as a Service; Compute, Data and Analytic services. Platform for Analytics development, access to Big Data networks	Storage clouds offers space to store the data mostly for backup purposes. Data Clouds are similar to Storage clouds and offer services to access the data through customized APIs. Big Data clouds provide a platform to data storage, access and application development for decision making on a large stream of rapidly increasing data.
2	Raw storage	Applicable	Applicable	Applicable	Storage offered for data archival purposes.
3	Object storage	Applicable	Applicable	Applicable	Object storage stores the data on a large cluster of storage devises. Object containers allow accessing the data through web services such as REST/HTTP and allow mounting as the disk to the clients. Automated synchronization methods are developed to sync the data to the cloud storages.
4	Block storage	Applicable	Applicable	Applicable	Large scale images repository.
5	File System for data organization	Not applicable	not applicable	Applicable	Storage clouds and Data clouds enable to access the data from remote repositories.

	and compute				However, Big Data clouds addresses distributed file systems for data organization, indexing and computing.
6	Data formats	Not applicable	Pre-defined data formats by the data providers	Structured , unstructured, semi structured data formats	Big Data computing addresses several platforms for processing several formats of data. Storage and data clouds may not address several forms. On the other side, Data clouds addresses only formats which are predefine.
7	Distributed storage	Applicable	Applicable	Applicable	Cluster of storage nodes for data organization and scalability.
8	Distributed file system	Optional	Specific to data providers	Applicable	Storage clouds and Data clouds offer data as a service but not computation, hence addressing file system is optional. However, Big Data systems should address the file system specific to data organization, indexing and syncing with the streaming data.
9	Authentication and authorization	Applicable	Applicable	Applicable	Several mechanisms such as user name, passwords and token based authorization mechanisms.
10	Pricing models	Applicable	Applicable	Applicable	Storage clouds pricing include the amount of storage space utilized or subscribed. Data clouds charge for the data services subscribed

					and how much utilized. Big Data clouds pricing include size of the Big Data setup, compute infrastructure needed for processing and analytics offered for data exploitation.
11	Query processing	Not applicable	Applicable	Applicable	Data clouds and Big Data clouds offer platforms to retrieve the data of interest with the several query processing mechanisms.
12	Domain specific	Not applicable	Applicable	Applicable	Data clouds offer specific services based on the user requirements. For example; Google earth services, Open Geo Spatial Services offer the services for geo spatial earth data. Big Data clouds may also offer specific services, for example, Google Big Query offers services to retrieve statistical information about the web documents.
13	Meta data organization	Applicable	Applicable	Applicable	meta data organization for storage clouds is about file organization, Data clouds works on several formats example; Open Geo Spatial formats, Big Data clouds needs to explore several meta data organization in terms of column oriented data bases.
14	Mash up services	Not	Applicable	Applicable	Storage clouds may not use mash up services,

		applicable			however, Data and Big Data clouds may use mash up services to offer new services by the selection from several services.
15	Security	Storage level	Data level	Storage, data, processing and analytics security	Big Data clouds needs to address security at several levels compared to storage and Data clouds.
16	Customized tools -application specific	Not applicable	Applicable	Applicable	Application specific customized tools such as web services, processing tools, data organization and retrieval tools needed for Data and Big Data clouds.
17	Customized web services	Optional	Applicable	Applicable	Storage clouds may or may not offer web services.
18	Replication	Applicable	Applicable	Applicable	Replication services ensure high availability and also effective retrieval of the data content by proper selection of replicated sites while minimizing the latency.
19	Data discovery mechanisms	Not applicable	Applicable	Applicable	Storage clouds need to offers any tools for data discovery, however Data clouds offers web services to find the data of interest, other side, Big Data clouds need to explore several data discovery mechanisms which could be much more complex than services offered by

					Data clouds. This could be due to large volume of data and variety forms of data formats and processing complexity.
20	Processing capability	Not applicable	Mandatory	Mandatory	Big Data clouds and Data clouds offer tools to process the data, however, storage clouds need not to have processing capabilities.
21	Data management	Only file level management	Web services for Data management	Data indexing, query mechanisms to handle several formats of data effectively.	Big Data clouds needs to explore several new technologies to manage the data.
22	Schedulers	Not applicable	Optional	Mandatory	Efficient Schedulers are needed for data processing for different formats on distributed compute nodes.
23	Data intensive computing	Not applicable	Optional	Mandatory	Big Data need to offer several programming tools for data intensive computing such as NoSQL programming models for unstructured data, map/reduce programming models for data centric operations.
24	Data management systems	Not applicable	Applicable	Applicable	Database systems could be offered as a Service over cloud e.g., Oracle Cloud, SQL Azure etc.,
25	Scalability	Applicable	Applicable	Applicable	All three systems should be scalable for

					storage and Big Data systems should be scalable for compute delivery.
26	Map Reduce	Not applicable	Not applicable	Applicable	Map Reduce is a new programming language for data intensive applications, which is an emerging programming model with in Big Data computing.
27	Data analytics	Not applicable	Not applicable	Platform for data analytics	Data analytics is a new generation of applications which combine the skills of computer science and mathematical domains to mine Big Data.
28	Time to deliver data	Optional	Optional	Mandatory	On time delivery of the information for decision making is an important factor for Big Data.
29	Column oriented database	Not applicable	Not applicable	Applicable	Big Data technologies need to address column oriented databases to handle the unstructured data formats.

3.5 Integrated Cloud and Big Data Platform - Convergence of Cloud and Big Data

The recent advancement in the service oriented technologies aka Cloud computing are delivering compute, storage and software applications as services over private or public networks based on pay-as-go delivery models [62] [19] . With Cloud computing paradigm becoming a reality, it is serving as a key enabler for Big Data to solve data intensive problems over a large scale infrastructure. The integration of Big Data technologies and Cloud computing read as - "**Big Data in Clouds**" is an emerging new generation data analytics platform for information mining, knowledge discovery and decision making.

With Big Data clouds, enterprises can save money, grow revenue and achieve many other business objectives in any vertical by quickly building their Big Data databases and writing analytics for mining the information. The benefits of Big Data clouds for the enterprises are mentioned below.

- **Build new applications**: Big Data clouds might allow a company to collect billions of real-time data points on its products, resources or customers and then repackage that instantaneously to optimize customer experience or resource utilization.
- **Improve the effectiveness and lower the cost of existing applications**: Big Data clouds offer services and pay as go consumption model similar to cloud services. This pricing model would effectively reduce both the cost of the applications development by minimizing the cost of development tools.
- **Realize new sources of information and build applications to gain competitive advantage**: The information could be quickly fused from several Big Data databases and rapidly build applications for several platforms like hand held and mobile devices.
- **Increase in customer loyalty**: Increase in the amount of data sharing within the organization and the speed with which it is updated allows businesses and other organizations to more rapidly and accurately respond to customer demand.

3.6 Big Data Computing Gap Analysis

Big Data computing can be classified into four technology segments denoted by 4D's (Depository, Devise, Domain and Determine) as depicted in Figure 3.5. Below, the detailed descriptions for several elements under each of the mentioned category are described.

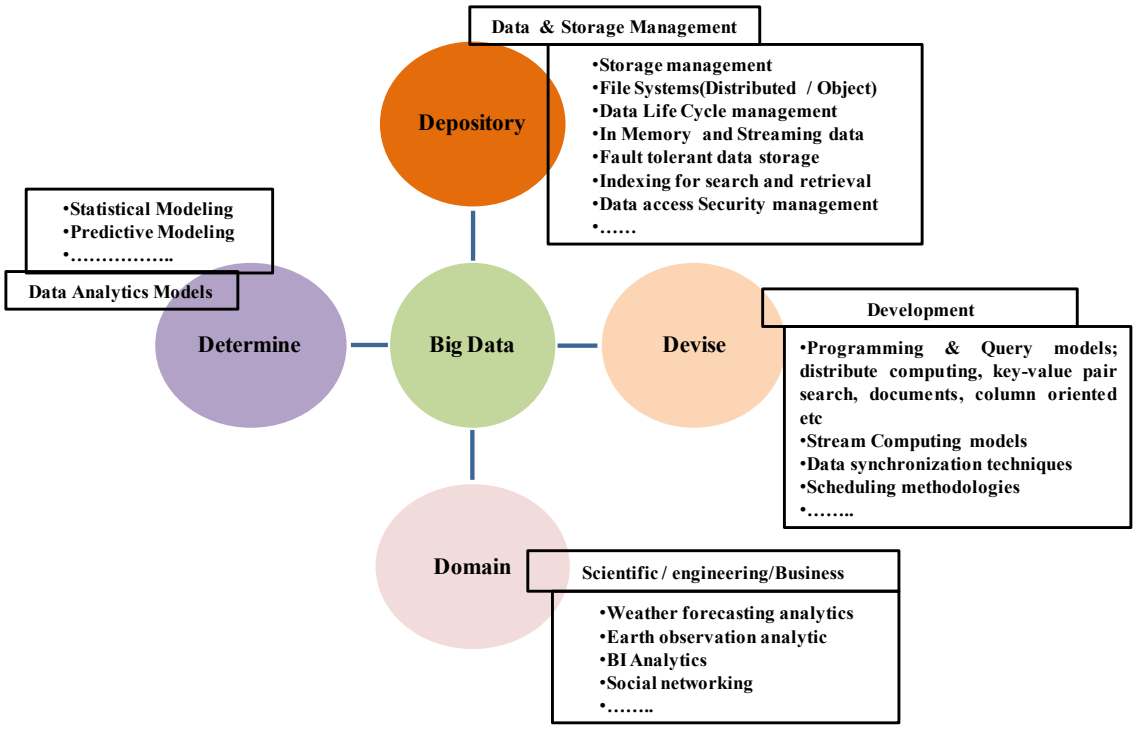


Figure 3.5. Big Data Segments.

A. Depository: Depository addresses issues related to data storage, organization and management of both structured and unstructured data from traditional storage like DAS, SAN/NAS to Cloud based storage architecture. Migrating from traditional file systems that maintain hierarchical organization using the file and folder analogy to object store which uses distributed data base model. Unlike traditional file system, big data storages addresses issues for a long-term storage system for more permanent type of static data can be retrieved, leveraged, and then updated if necessary. Several issues to be addressed are file systems for In memory databases, In memory computing, and Stream computing using distributed file systems over a cluster of nodes, fault tolerant data storages, indexing mechanisms for data retrieval, security and management issue for data access, data organization for several formats etc. as explained in Table 3-3.

Table 3-3. Depository: gap analysis and future directions.

Key element : Gap Analysis and future directions
1. Storage and network: • The unified storage systems need to be explored combining three layers of traditional storage architectures such as file system, volume manager and data protection into unified software layer, creating a single file system. These newer generation file systems should have support for wide range of industry standard protocols, including network file system (NFS), Server Message Block (SMB), Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), REST-based Object access etc. • Ethernet, today the mostly used network infrastructure, however, has limitation in the speeds due to its hierarchies of subnets connected by routers and network packets which takes exactly one path to traverse between any two points on the network. Newer technologies supporting multi paths for network packets to traverse between two points, such as InfiniBand [129] network need to be explored for Big Data HPC analytics. • Large scale data warehousing systems at present are hierarchical in nature with online, near online, and tape/offline storage support. These systems have limitations like storage scaling, data access from several devices, and large file name space creation. Apart, they use custom proprietary protocols that makes impossible for mounting and accessing the data from several devices. Hence, there is a need to explore on new generation of file systems for big data to support for scalable file systems that could be accessed, mounted and queried from several kinds of devices with large scale storage capability. This could be achieved by scaling the storage devices horizontal over a cluster of nodes, making the file system not limited by file name space and open standards for data access that mostly use REST based web services. New scale out storage architectures for object based storages need to be developed. These architectures should scale the storage devices in horizontal without limited by the file name space. The challenges involve in addressing the unified storage technologies

with object based file systems, where in user defined metadata could be attached to the file and NoSQL query engines could be developed for efficient query and processing.

- Existing cloud storage technologies support data archival for long term usage, however indexing and computations on the preserved data, is not supported, directly. Hence, there is a need to address better indexing support for cloud storage systems. This could be achieved Write Once and Read Many (WORM) object storage technologies for both long term archival and computation. The indexing techniques could be applied as metadata to the object or else separate data structures like NoSQL databases integrated with the storage could be used.
- New WAN-based protocols needs to be investigated as the traditional WAN-based transport methods cannot move terabytes of data, they use fraction of available bandwidth and achieve transfer speeds that are unsuitable for such volumes, introducing unacceptable delays in moving data into, out of and within the clouds.
- Big Data protection and data access at faster rate plays key role. Hence, new storage protocols need to be explored for on disk data encryption, privacy preserving, and query on encrypted data mechanisms with high performance rates.

2. **File system:** Present file systems have built-in namespace constraints for files and directories they can store and manage. Hierarchical directory structures can become unwieldy, performing poorly at navigating large number of users or files. These file systems are managed by the operating system and organized into folder hierarchy, which has limitations in searching with limited metadata which is defined by the operating system. To envisage the additional metadata searching, the data is organized by databases either into relational or any other relevant data. However, Big Data file system needs additional metadata to be required and have access from variety forms of devices. Hence, newer file systems needs to be explored, such as Object based storage technologies that allow the files to assign user define metadata to the files and are never confined to any operating system. These could be accessed via open standards API (Application Programming Interface) to interact for greater application data awareness. Object based storage frameworks providing native support of standard object protocols like REST while also supporting de-facto cloud protocols like Amazon S3,

Google Drive etc. needs to be evolved. Meter based billing usage and consumption of cloud based storage for Big Data needs to be evolved.

- Isolation of file systems from operating systems. This could be achieved by building file storage services for account management (for file access), container management (list of objects) and object management (files stored at physical devices).
- The file systems should support a large petabytes of storage with a single namespace supporting by user defined metadata to the files. This will enable the elimination for to eliminate the need for maintaining the separate database to describe the metadata of the files.
- Tools to transform the existing file systems to cloud based object storage structures needs to be developed. The challenges are; identifying the right tools required for data processing from a large set of available tools.
- Tools to automatically mount the object based storage to the virtual compute nodes for analytics execution needs to be developed. The object mount create a super logical view of the objects located on distributed data storages with the metadata constructed online.
- New data storage, access and retrieval mechanisms should enhance the current distributed file systems to retrieve the data based on their value, meaning and content. Technologies such as Information Defined Data Storage complement the existing distributed file system to derive the value out of the data by its content and meaning but not just with names.
- Object Based Storages for single flat name space, location independent addressing, per object user definable metadata, unlimited storage, user defined policies and random seek with high performance retrieval are to be developed.
- High performance parallel data transfer with data distribution, replication and redundant mechanisms. Big Data File System need to be supported with Object based storage and high performance file systems. However, the present HDFS supports only Distributed File System. Object based storage provides the location transparency with scalability features and block based retrieval. Object Storage can be considered as an alternative to store large volume of image, video and audio

data.

- Replication transparency needs to be developed which will ensure scalability. This could be achieved by replicating the big data file objects on multiple distributed storage systems and creating the clustered indexes.
- Migration transparency for files should be able to move around without the client's knowledge. This could be useful to move across the several storage providers over several geographical regions.
- Support fine-grained distribution of data to optimize performance by locating individual objects near the processes that use them.
- Benchmarking the several big data storage systems needs to be developed considering several QoS parameters like objects metadata, scalability of the file systems, replication and partition across the geographical boundaries, high performance delivery and computation.
- High performance, intelligent file syncing process, that could recognizes the changes and file operations (such as moves and renames), to avoid the unnecessary data copy over the network.
- On demand delivery of Big Data file systems as service models over cloud technologies needs to be explored. These systems would create the required file systems based on QoS and SLAs between users and providers. Also, necessary migration tools need to be addressed for moving the on premise big data files to the cloud based big data systems which could be relational data or unstructured like documents, texts, videos, audios etc.

3. Data Life Cycle Management (DLM)

- Data life cycle addresses the data management issues at several phases of data creation, usage, sharing, storing and eventually archiving or disposing automatically based on policies defined within the management. BDLM (Big Data Life Cycle Management) systems need to be developed incorporating several user defined policies, to enable the better data organization and to minimize the storage costs. For e.g., the policy could be data aging, addresses issues related to the data obsolete, something like, deleting the objects those are older than 365 days, deleting objects those are created before a date.

Another policy could be the version management, holding only the recent versions of each object in a bucket with a versioning enabled.

- There has been an explosion in the growth of data, and traditional approaches to scaling storage and processing, to reach computational, operational and economic limits. Hence, there is a need to intelligently manage and meet the performance and availability needs of rapidly growing data sets. Technologies such as automated N-tier storage migrations need to be investigated, that automatically and non-disruptively migrate data from one generation of a system to another to effectively address long-term archiving, also distributing storage among multiple data centers, either for effective recovery or to place content closer to the requesting users in order to keep latency at a minimum and improve response times. This N-tier storage architectures could organize the data which is mostly used in lower tiers (tier1) such as Flash/SSD and migrate the data moving down to other tiers such as flat disks, capacity disks to tier N such as Cloud storage for backup/archival.

4. In memory computing (IMC) systems

- In memory computing systems or in memory Data Grids (IMDG) technologies need to be investigated that could store terabytes of data completely in RAM, avoiding the need for mass storage media such as hard disks. The open challenges include, distributing the data structures among multiple servers, storing as key/value data structure, rather than a relation structure, providing flexibility for application developers.
- In-memory computing tools should work with Object storage file systems enabling the data computing faster and the results that could be stored making for longer duration. It also enables to query the data based on the metadata from the cloud storage pools and perform the object based data storage, query and object based data analysis.

5.High Availability(HA) and Fault Tolerant(FT) Data Storage

HA systems offer storage providing multiple internal components and multiple access points to storage resources. In other words, the system has a second critical component or path to data available in case something fails. This availability or single point of failure doesn't eliminate downtime. Instead, it minimizes it by restoring services behind the scenes, in most

cases before the user notices failures.
7. **Indexing for search and retrieval :** Indexing multidimensional data and enabling object based retrieval mechanisms instead of set based needs to be developed for efficient query processing.
7.Data access and Security : Data access and security mechanism in Big Data Clouds needs to be developed which set policies enabling which users get access to which original data, with protection of sensitive data that maintains usable, realistic values for accurate analytics and modeling on data.

B. Devise–Big Data Platform Services

This segment focused on design of new programming models for distributed computing, in-memory computing, stream computing, and tools to handle assorted data via query languages such as key-value pair, column oriented data, document databases etc, high performance data synchronization techniques, scheduling methodologies etc. This segment covers Big Data Platform tools such as programming models, Query Processing techniques, performance related issues, tools to process Big Data Types, schedulers, etc,. Currently, Hadoop and Map Reduce [28] , [70] have become an ubiquitous framework for large scale data processing, however, Map Reduce, has limitations both from theoretical perspective [71] , [72] and empirically by exploring classes of algorithms that cannot be efficiently implemented [73] , [74] , [75] , [76] several limitations of the Map Reduce Model over Hadoop File System are described below.

Table 3-4. Device: gap analysis and future directions

Key element :Gap Analysis and future directions
1.**Programming models :**Hadoop Distributed File System and Map Reduce programming models are amenable to the problems where the program is recursive, however, will not fit into to solve all classes of problems specifically which are iterative in nature [79] like Page Ranking Iterative Graph Algorithms, Gradient Descent and also few engineering and

scientific applications like Data Product Generation [80] , [81] , [82] and DEM [83] , [84] generation demands for a large data made available at the compute for processing leading data aggregation before the execution commences.

- New developments in the Hadoop project promise "meta-frameworks" such as Hadoop NextGen YARN [40] and other programming models like Spark [85] and MPI [86] , and MapReduce.
- There has already been plenty of works within Hadoop, from database perspective; integration with traditional RDBMS [87] , [88] , smarter task scheduling [89] , [90] , columnar layouts [91] , [92] , [93] , [94] , [95] embedded indexes [96] [97] , cube materialization [98] and efficient join algorithms [99] , [100] , [101] , [102] .
- HDFS address small compute and large data problems.
- NoSQL Data base Management System (NoSQL DBMS).

And NoSQL alternatives are in pre-production versions with many key features yet to be implemented.

- Proper indexing and efficient search tree mechanisms need to be explored to improve the response for queries.
- Debugging tools and profilers for Map Reduce Programming model required to be investigated for Map Reduce programming. Currently, there are batch based without user interaction.
- Domain specific languages need to evolve such data intensive, high performance, IOT programming etc. to solve specific problems in several fields of final services, business sectors, scientific explorations, sensors networks etc.
- The analytics executed on the cloud platform are currently, mostly batch oriented without user interaction. Tools need to be explored to make the user jobs more interactive and get the intermediate refined answers to the queries, instead waiting till completion of all the jobs.

2.**Unstructured data processing :**In the era of big data, good old RDBMS is no longer the right tool from many database jobs. NoSQL databases needs to be evolved to address several kinds of data.

- Document, text and graph based data processing mechanisms with better indexing mechanisms need to be explored. This could use key value pair mechanisms with schema less data bases and map reduce functions for effectively retrieving the data

by processing on the cluster of machines. • Unstructured database systems need to be developed to bride gap between traditional databases and key value pair databases. These data base systems should work on object notations and perform query on multiple nodes to improve the performance.
3. **Scheduling methodologies :** • Evolutions of new programming models for compute intensive Big Data Problem: programming models with the combination of Thread, Task and Map Reduce need to be devised. The current Map Reduce programming model will transfer the compute to the data node. Here, Compute is considered to be a small activity, when compared to Data. This model will not be amenable when Compute is as large as data. Hence new programming models needs to be exploited. • QoS based Resource Management scheduling methods needs to be developed which would work on parameters like time, budget, accuracy etc. • New HPC programming models such as high speed in memory computing, stream computing need to be worked for HPC Big Data Clouds for scientific applications to address data science problems with accuracy in real time.
4.Workspace Management : Creating workspaces for Big Data analytics development in collaborative environment. These workspaces need to organize the source code, data etc. in sharing mode, and allow the analysts to design and develop the applications over cloud infrastructure.

C. Domain - Scientific, Engineering and Business

Big Data Analytics extract information from large data for decision making. Examples include, earth observation systems, disaster management study, weather forecasting, simulations, engineering design problems, business intelligence applications. Apart, there is a need to evolve several complex applications such as monitoring historical data of a company with millions of rows of data for their business process improvement. Here, we say this new field as “Data Science” which incorporate mathematical model, simulations,

visualization, decision making needs to be done for the data. Specific Analytic tools need to be evolved in all domains of science and engineering, business intelligence. The researchers should focus on the following activities to address the domains of data science.

Table 3-5. Domain: gap analysis and future directions

Key element: Gap Analysis and future directions
1.Data Management and Supporting architectures : The current frameworks for data intensive computing such as Hadoop are good to solve when the compute is small, however, for larger computes, the current systems needs to be scaled. Several case studies need to be conducted to understand the performance of the existing frameworks, for several data science problems, like Genome Analysis, CFD, Earth Observation systems etc. These case studies need to address the data management and processing issues for unstructured data and investigate on several computing, data migration and indexing mechanisms specific to the domains.
2.Model development and scoring : Models need to be developed to work on the regions of data and assign the score based on the ranking assigned. This could enable the Analytics to pick up the most relevant data for analysis.
3. Visualization and interaction: Visualization tools need to be developed to view the large scale data and the analyzed/processed complex results. This could include building reports, charts, dashboards etc.
4.Domain specific models : • Domain specific analytics tools that would pick up the appropriate NoSQL databases necessary for the analytics needs to be explored. • Models to be investigated, for migration of the existing in house domain specific analytics to clouds. These models address the data management issues, extract, transformation and load tools for the in house data with effective indexing, processing and tools for analysis. • Open standards need to be evolved to publish analytics models as services. This would help the enterprises to quickly analyze the data for decision making without investing on infrastructure and analytics development.

D. Decision – Mining and Determining

Big Data processing is driven by statistical and analytical models to derive information for decision making. Big Data is not just about the data, however, ability to solve business and scientific exploration problems and provide new business opportunities and thoughts. Data Analytic play a major role in information mining and deriving a value from the data. Currently, analytic systems are evolving; however, the majority of them are based on the Hadoop framework. Big Data Analytics frameworks should evolve to solve specific domain problem issues for example in the areas of predictive analysis, behavior analysis, business intelligence etc. In Big Data mining, several open source initiatives tools are becoming popular, as mentioned below.

- Apache Mahout [111] : Scalable machine learning and data mining open source software which is a part of Hadoop framework. It has a wide range of machine learning and data mining algorithms: clustering, classification, collaborative filtering and frequent pattern mining.
- R [115] [116] : Open source programming language and software environment designed for statistical computing and visualization.
- MOA [117] [118] : Stream data mining open source software to perform data mining in real time. It has implementations of classification, regression; clustering and frequent item set mining and frequent graph mining.
- PEGASUS [119] : Big graph mining system built on top of Map Reduce. It allows finding patterns and anomalies in massive real-world graphs.
- GraphLab [120] : A high-level graph parallel system built without using Map Reduce. GraphLab computes over dependent records which are stored as vertices in a large distributed data graph. Algorithms in GraphLab are expressed as vertex-programs which are executed in parallel on each vertex and can interact with neighboring vertices.

A few new challenges researchers can investigate are:

- Evolve new architecture for analytics to deal with both historical and real time data at the same time. This could be achieved by organizing the unstructured data as N-tier system with effective indexing and performing the distributed queries and data intensive programming

techniques to analyze the historical data and compare it with the present data to derive the intrinsic information.

- Statistical significance tools need to be developed to determine maximum like hood (e.g. p-value [35]) to determine the evidence based on the probabilities and statistics, rather on randomness on the data distribution.

- Distributed Data Mining and Big Data Distributed parallel data mining algorithms and frameworks for unstructured large volume of data need to be investigated, to analyze the data quickly and provide the results summary. Time evolving data mining techniques need investigated for the evolving data sets such as words, Graph analytics for social networking, behavior analytics, predictive analytics, earth observation geo intelligence solutions, weather forecasting, etc. New techniques need to be evolved which could quickly identify the portion of the data needs to be mined rather as a whole to quickly deliver the analysis results. Big Data cloud analytics services need to be developed for domain specific applications meeting QoS, SLAs, and budget followed by deadline constraints. Distributed Real-time, predictive and prescriptive analytics tools need to be evolved that could provide the interactivity to the running jobs, apply statistical tools to determine the information and offer the results in the real time [36].

3.7 Discussion and Summary

In this chapter we have introduced technologies for data intensive computing in the domains of Big Data, and Cloud computing, followed by using Cloud computing as back end technologies for performing Big Data computing applications in several fields of science, engineering, business domains. Initially, we have discussed the differences between Traditional Data model followed by Big Data model, and comparisons between these two technologies in terms of data organization, and processing tools. Later, we have discussed, Big Data computing taxonomy and it's under pinning technologies such as Cloud computing. Followed by Cloud computing related technologies and its deployment models, and how these Clouds can deliver the backend infrastructure for Big Data problem solving. This chapter also introduced, the Integrated Cloud and Big Data Access Network for which Clouds would work as back end computing technologies for the development of Big Data Applications. Later, we have introduced several applications that would fall into scientific and Business intelligence applications, followed by

Big Data computing segments and gap Analysis. This chapter identifies two key element of the framework such as data aware scheduling, and domain specific programming models for scientific big data computing applications over Clouds infrastructure. Scheduling is presented in following chapters 4 and 5. The scheduling discusses about the challenges in bringing the data and computing elements together so as to solve large scale data intensive applications for the decoupled applications, data and computing resources using family scheduling model and genetic approaches, followed by simulation of the experiments using CloudSim and results. Chapter 6 presents one of the domains specific big data processing for scientific imaging applications for data organization, processing using extended Hadoop Distributed File System, and MapReduce computing. This chapter discusses, the extended Application Programming Interfaces (API) for data organization over Hadoop HDFS, followed by several high level APIs for data processing using MapReduce for image processing filters.

Chapter 4

Data Aware Scheduling in Big Data Clouds – A Mathematical model

In this chapter, we discuss the problem statement for large scale data intensive scientific problem solving in the Cloud computing environments, and present the several applications in earth observation systems which are considered as Big Data applications. We discuss, remote sensing earth observation system, followed by several applications in remote sensing data processing, those motivates the design of Data Aware Scheduler for addressing large scale data intensive applications in Cloud computing environments. Here, we discuss the overall requirements for data intensive scheduling, system architecture for the data aware scheduling, and present how the system can be viewed as problem of Big Data computing using Clouds infrastructure as back end technologies. Here, illustrate several challenges in handling the data intensive applications, and present a scheduling approach based on data availability, computing resources and network availability called as Data Aware Scheduling model, for addressing data intensive applications over large scale distributed data and computing resources. Here present, mathematical model, and discuss, how the several parameters such as job characteristics, data volumes, data replica locations, computing resources availability, and underlying network resources like bandwidth and latency are considered for scheduling the batch of job applications. Here, we address the problem using a grouping mechanism called as family grouping, which is based on grouping the jobs for which data required is nearly similar, followed by significance of optimization methods to address the scheduling problem. The proposed scheduling approach does the minimization of the turnaround times so as to achieve higher throughputs for group of jobs.

4.1 Problem statement

Data collected/produced from several fields of science, engineering, business intelligence, social networking is growing tremendously and such data volumes are expected to reach to several Zetta bytes in near future. Such collected data often require tools to facilitate efficient data management, analysis, validation, visualization, and dissemination by preserving the intrinsic value of the data [3] . Experiments in the fields of science and engineering such as remote sensing, high energy physics, molecular docking, computer micro-tomography and many other fields gather the data from several instruments place at distant apart and collect the data from such instruments, and analyze them from intrinsic information extraction. Such gathered data from several scientific instruments is organized over geographical distributed storage repositories, for further processing. The data would be replicated to ensure high availability, and also to enhance the location proximity while processing the data over several computing resources. Below, we describe one such large scale data gathering and distributed system in geo science of remote sensing data processing from space earth observation system.

- **Remote Sensing earth observation and data processing system**

Remote sensing is a broad term used to describe acquiring information about an object by means of "remote" examination; that is, with no direct contact of the object. There are several ways of sensing the data remotely such as ground based, airborne and space borne (satellite). In space borne remote sensing, sensors are mounted on-board a space craft (space shuttle or satellite) orbiting the earth as shown in Figure 4.1. There are several remote sensing satellites providing imagery for research and operational applications [130] . Space borne remote sensing provides the following advantages.

- Large area with wide swath, and systematic coverage.
- Repetitive coverage of an area of interest with frequent revisits for object change detections.
- Ground features using radio-metrically calibrated sensors for Quantitative measurement.
- Semi / fully automated computerized processing and analysis of objects.
- Relatively lower cost per unit area of coverage for earth based explorations.

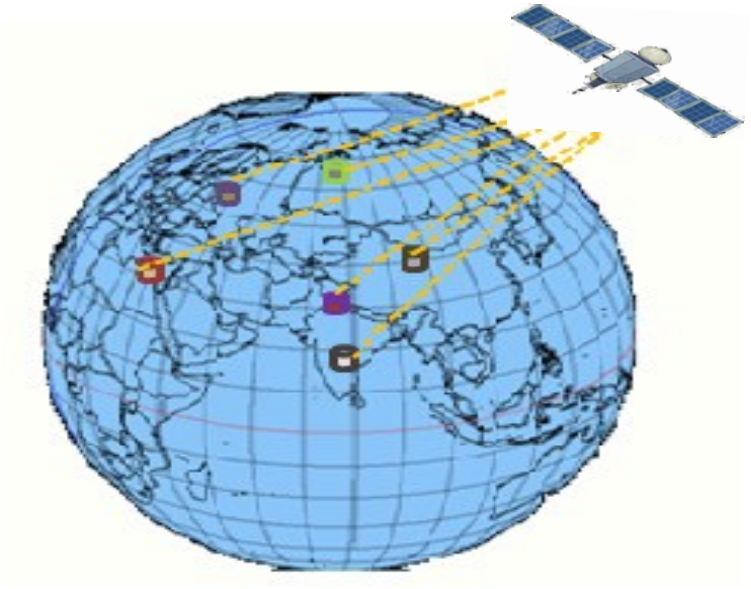


Figure 4.1. Space shuttle orbiting the earth and receiving ground stations

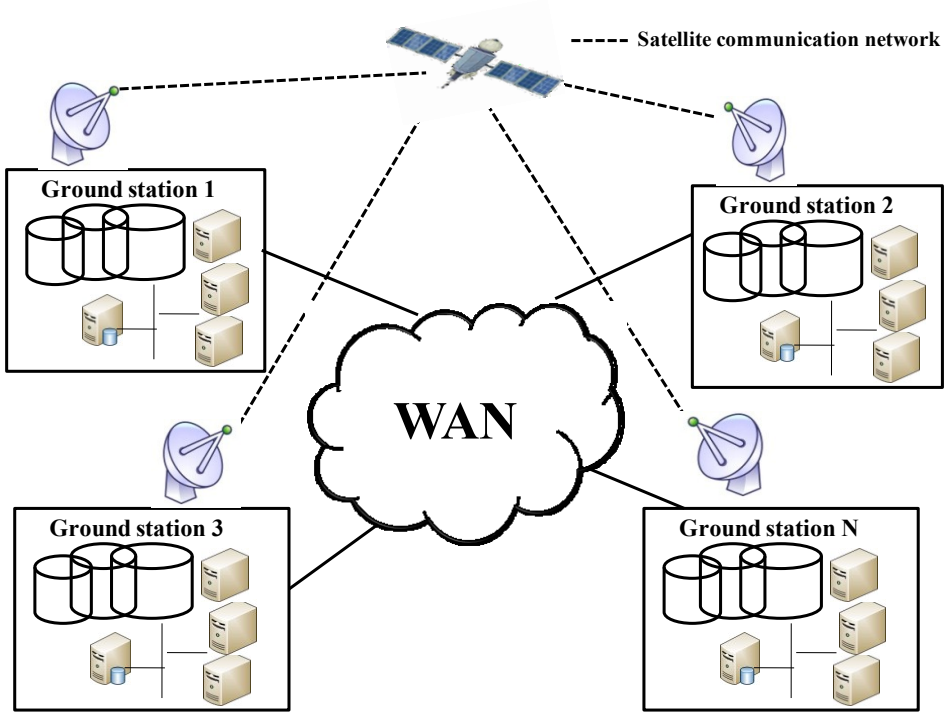


Figure 4.2. Remote sensing space craft for earth observation systems and their inter connectivity with ground stations

A ground station or earth station, or earth terminal is a terrestrial radio station designed for extra planetary telecommunication with space craft, or reception of radio waves from space craft. Ground stations are located either on the surface of the earth or in its atmosphere. Earth stations

communicate with spacecraft by transmitting and receiving radio waves, when ground station successfully transmits radio waves to a space craft or vice versa, it establishes a telecommunication link and receives the data from the space craft sensors. One such earth/ground stations and interconnectivity with space craft is depicted in Figure 4.2. Ground stations receives the data from the space craft, and such collected data is distributed across several storage repositories for further study by several researchers using several data processing techniques. The papers [80] [81] had discussed, one such satellite data processing system for precision data product generation in the private cloud environment over a distributed computing and storage repositories. Such collected data gets distributed and often replicated to several storage locations for efficient access and can be retried for processing by several research communities. And such storage repositories called as Storage Clouds where the digital data is stored in pools, the physical storage spans multiple servers and often locations, and are accessed on demand as several service oriented architectures.

Remote sensing data processing system involves a set of procedures starting from data collection, organization, processing, analyzing, extraction and dissemination of information to the end users of the systems. Several applications for remote sensing data processing are discussed below.

- **Applications of Remote sensing data processing**

In the data processing system a group of researchers' dispersed geographically conduct data exploitation experiments routinely which involves various activities, such experiments are of Object detection, pattern matching, Image matching, DEM generation [149] [84] etc. Below we will introduce such activities and also the nature of the data processing system:

a) Large volume of data is collected from various sensors, stored in the data repositories on the daily basis; also a large number of experiments could generate the data which in turn may be stored at these repositories. These repositories named as data hosts which are geographically distributed across the country/globe.

b) Large number of jobs from various research groups arrives for processing; these jobs need the computational power and the data for execution. In some of the cases data and compute may be collocated, and few cases distant apart. In the case of distant apart

proper selection for computation and data is needed along with the application service migration.

c) The services that are used for exploitation of the data are published by the researchers. These services can be part of data hosts, compute providers or may be located on other host providers.

d) The jobs are to be processed in minimal time by reducing the overall make span. Make span is the minimum completion time of all the jobs.

e) In many cases the data being used by the users is common while the application services are different. For example user A is interested in extracting objects in a specific Region of Interest, similarly user B is interested analyzing the patterns of object in the same or sub set of the region. The jobs which fall under this category are termed as the family job.

f) The pre processing stage is needed to identify the family jobs among the arrived jobs.

g) After the jobs and the families are identified, the next activity is to construct the schedule for the jobs for minimizing the overall completion time of all the jobs.

h) The jobs are to be processed in minimal time by reducing the overall make span. Make span is the minimum completion time of all the jobs.

i) After the jobs and the families are identified, the next activity is to construct the schedule for the jobs for minimizing the overall completion time of all the jobs.

Processing and analyzing large volumes of data plays an increasingly important role in many domains of scientific research. Here, we focus on data intensive applications with two important properties: (i) data elements have spatial coordinates associated with them and the distribution of the data is not regular with respect to these coordinates, and (ii) the application processes only a subset of the available data on the basis of spatial coordinates. These applications arise in many domains like satellite data processing and medical imaging. Here, we present a satellite data processing example give below.

Another computation on this remote sensing is as follows- portion of earth is specified through latitudes and longitudes end points with the time range. For any point on the earth within the specified area, all available pixels within that time-period are scanned and the best value is

determined. A typical criterion for finding the best value is cloudiness on that day, with the least cloudy image being the best. The best pixel over each point within the area is used to produce a composite image. This composite image is used by researchers to study a number of - properties like deforestation, pollution over different areas, etc A typical scenario and problem in one of scientific community is discussed below.

4.2 Remote Sensing Big Data Clouds

The recent advances in remote sensing and computer techniques are generating huge volumes of data, and are given an explosive growth of remote sensing digital data technologies, which is the earth observing data continuously obtaining from space, and airborne sensors, as well as some other data acquisition sensors. With the exponential growth and increasing degree of diversity and complexity, the remotely sensed data are regarded as Remote Sensing Big Data. Big Data occurs when large collection of data sets whose volume and rate of data is at scale that is far beyond the state-of-the art system and revolutionize the way of seeking solutions. This is also the case for the remote sensing and earth sciences domain that falls into Big Data domain.

With the recent advances in sensors and earth observation techniques, high resolution sensors are placed in the orbit employed to seek shorter re-visit cycle and larger ground coverage. Applications and experiments in all areas of earth observation systems are becoming increasingly complex and more demanding resources in terms of their computational and data requirements. Some applications generate data volumes reaching hundreds to terabytes even petabytes. As scientific applications become more data intensive, the management of data resources and dataflow between the storage and compute resources is becoming the main bottleneck. Analyzing, visualizing, and disseminating these large data sets has become a major challenge and Big Data is considered as the next generation data intensive computing model using after empirical, theoretical, and computational scientific approaches.

However, the transformation of Remote Sensing Data as Remote Sensing Big Data is due to the scale at which the data is acquired from a large collection of sensors, and whose volume and rate of data is at a scale that is far beyond the conventional systems and revolutionize new tools and techniques for handling and processing. Remote Sensing Big Data, is not just merely refers to the volume of and velocity of data that but also the storage and computing capacity, including the variety and complexity of the data to be handled, and processed.

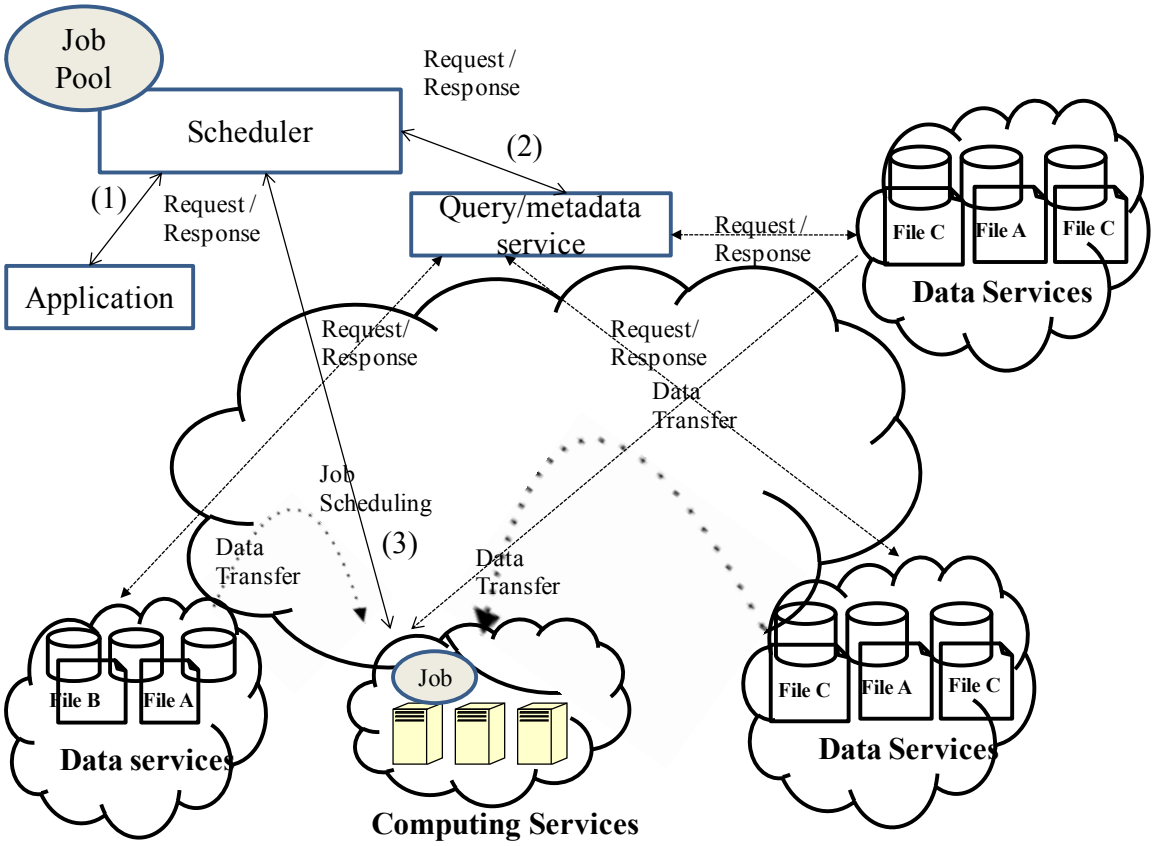


Figure 4.3. RS Big Data Cloud setup – Distributed data and computing services

An overview setup of Remote Sensing Big Data Cloud setup, with the distributed and dispersed Data/Storage services, and Computing services are depicted in Figure 4.3. The data collected/at the earth/ground stations will be pushed on to RS Big Data Cloud for further usage by several research analysts for necessary applications developments in the several areas of as spatial and temporal analysis which will be described in the next section 4.4. Below, we describe major elements of the RS Big Data Cloud.

- **Data services:** Large scale, secure, durable and highly scalable storages for data organization and retrieval on demand for the data of interest. The storage services include file system, block storages, and object storages for organization of the data based on the users and data specific requirements. For example, the data can be organized as Object Storage files, and such objects can be retried using web service calls such as Representation State Transfer (REST) APIs. Object storage also enables to attach the

user specific metadata to the file, which is could have wide usage for unstructured data organization and processing. Data services to provide data on demand to the users regardless of geographic or organizational separation of provider and consumer. Data services are similar to Software as a Service (SaaS) in that the information is stored in the Cloud storages and is accessible by a wide range of systems and devices.

Data services can eliminate redundancy and streamline costs by housing critical data in one location, enabling the data to be accessed and/or updated by multiple users while ensuring a single point for updates. Potential drawbacks to data services include server downtime from the data service provider, data loss in the event of a disaster, and the security of the data, both in its stored location and in the transmission of the data among the users. To overcome the limitations of the above said drawbacks, the data would be distributed and replicated onto the multiple storage locations along with the data services for accessing the data. In Figure 4.3, the shown data service providers indicate that, data gets replicated via distributed synchronization mechanism for ensuring the high availability and proximity as and when the data is required.

- **Computing services:** Large scale, elastically scalable computing resources which can be provisioned on demand either as a single individual systems or a group of systems like cluster. These computing services can be offered in three modes based on organizational requirements such as i) publicly available giving access to other organizations or individuals ii) owned and operated by a single organization, controlling the way the resources and automated services are customized and used by various constituent groups iii) with a combination of single owning, and public resources federation to meet the organizational demands.

The computing services offered can be categorized into two parts such i) resources which can be created on demand based on the application requirements. This could be achieved using the virtualization techniques and the web service API calls offered by the respective virtualization technology, and ii) already setup pool of computing resources similar to cluster setups which could be used by the application developers for performing the jobs execution. Below we describe how EOS and RS Big Data could offer services for large community of researchers for geo science applications.

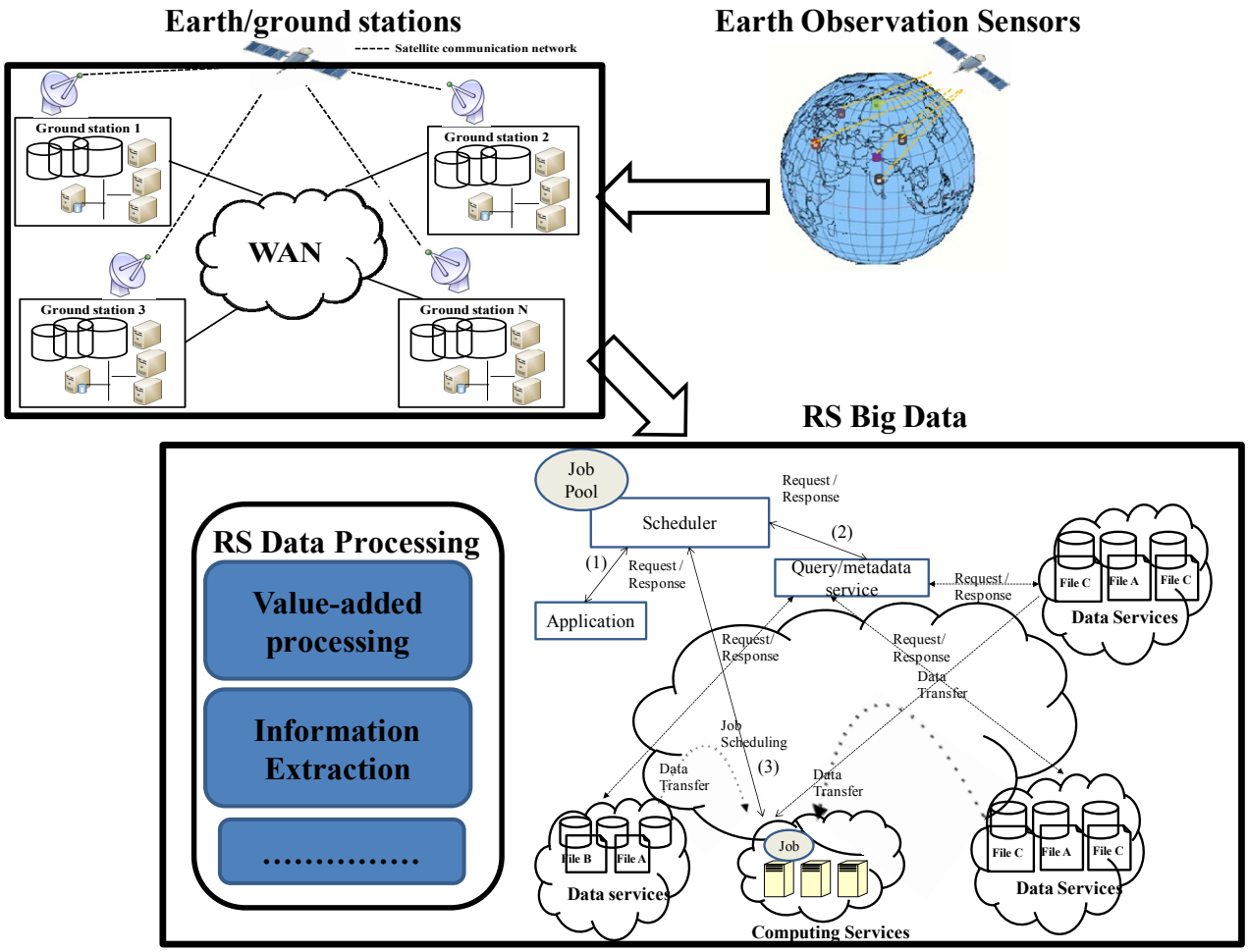


Figure 4.4 EOS and RS Big Data

4.3 RS Big Data Clouds for EOS

The data collected from Earth Observation Sensors at ground stations is pushed to Remote Sensing Big Data setup for further usage. RS Big Data is continually acquired for processing and knowledge discovering using several processing techniques. Communities groups can make use of the RS Big Data for their applications development. In recent years, spatial applications have become more and more important in both scientific and research industries. Geospatial analyses of distributed data from surveys and sensors are often stored and managed in diverse regional, national and global repositories. The nature of scientific processes requires composition of these resources in a meaningful order to solve a specific geosciences problem. Spatial data processing can be very resource intensive and complex, since it usually involves the analysis of large amounts of spatial and non spatial data from various diverse applications. The

topologies such as Euclidean, metric, topological, Set-Oriented are used to find the relationship between spatial objects. The amount of information in spatial databases is growing as more data is available from several experiments, and such data intensive computations from a number of domains share two important characteristics such as associating of ground coordinate value with the pixel, and followed spatial features/gray value with the pixel. Figure 4.4 describes Remote Sensing Big Data and its integration with EOS, where Computing and Data Storage resources are geographically dispersed. The data acquired from several space borne sensors are distributed and replicated over the storage repositories. These storage repositories can scale on demand and offer the data through metadata and query services on demand to the several application developers, research scientists on demand. Computing resources are dispersed from the storage repositories, and which are scalable and are offered as services for the jobs to be processed. The sequence of steps followed for processing the jobs are as follows – the application developers place the request either to process the job, or for retrieving the data of their interest. The request is processed by the scheduler, interacting with Query/meta data service. This would return the data sets to be processed or served. Later, the scheduler initiates the process for processing the jobs.

The execution of distributed data-intensive applications involve requirements for discovering, migration/transfer, processing, storing and managing large scale distributed data sets and is guided by speed and processing of the data. There may be multiple datasets involved in a computation, each replicated at multiple locations that are connected to one another, apart from the computing resources connected with a network of varying capabilities. Consequently, this kind of scenario makes it difficult to identify appropriate resources for retrieving and performing the required computation on the selected data sets. This thesis, therefore, develops and presents a scheduling model for applications that require massive data addressing in RS Big Data.

The researchers/users in the satellite Data processing run the large number of data intensive jobs which demand for more computation also data, and need to be served by reducing the overall completion time of all the jobs. Hence, there is a need of a system for effectively processing jobs. In this section various such issues in solving the data intensive jobs of the system along with the specific characteristics of the jobs are mentioned. Jobs need the data, compute and application services for processing; in some cases Jobs may need the same data, while the application services may differ. These application services are published by the application

providers, and may be collocated with the data, compute or may not be. In case of the data being used by the jobs are same, a pre processing stage can be applied to group them as a single Entity, enabling in reducing the data transfer time in the system for the group rather than individual job The data is replicated on the multiple data hosts, hence the mechanism can be devised to construct the multiple channels for the data transfer enabling the minimal data consolidation time. To reduce the data consolidation times, the job grouping is applied; in the job grouping the jobs which need similar data are grouped to one family. The above mentioned points enable us to devise a methodology to construct the optimal group scheduling for the jobs, enabling effective utilization of the available communication and computational resources of the system.

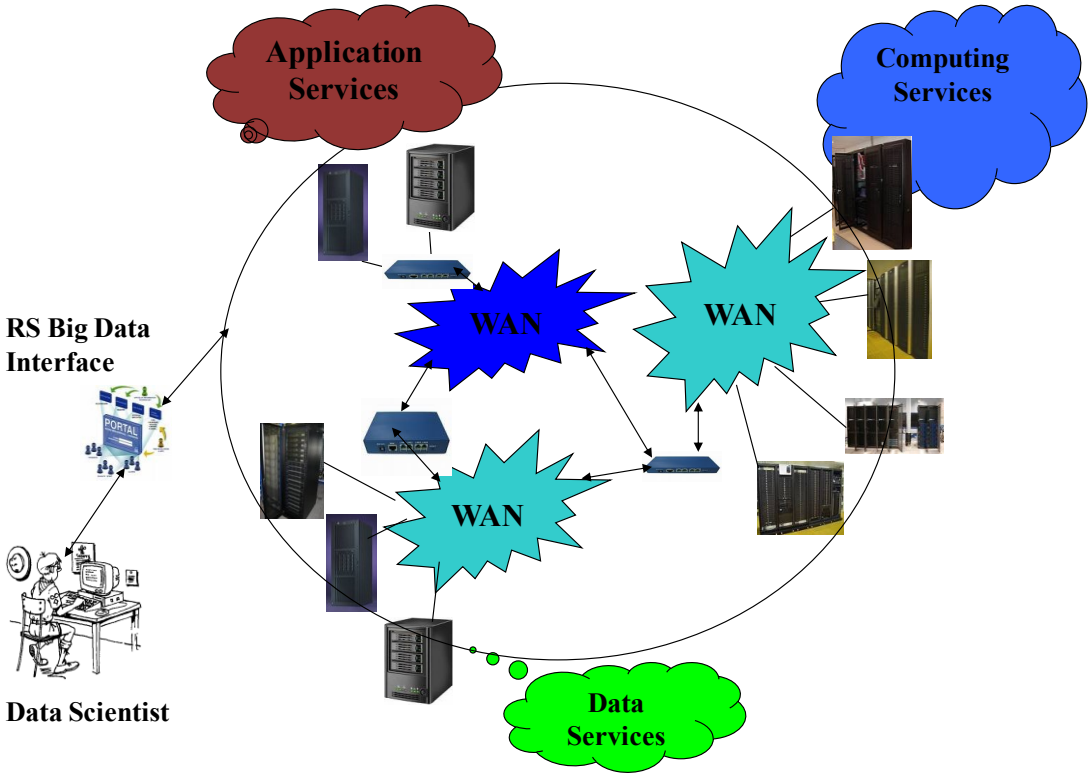


Figure 4.5. Remote Sensing Big Data elements

The typical structure of the model for data processing system is as follows - the applications/services are hosted as services, computational services by the compute providers and the data services by the data providers as depicted in Figure 4.5. From the diagram it is known that these resources (services, data and compute) are dispersed (in some scenarios may be collocated).

The processing of these massive RS data is quite a challenging issue. The difficulty lies in the data storing, organization, processing and analyzing. To meet these challenges, the researchers may need to concentrate on the below said points.

- Efficiently managing the massive amounts of variety forms of RS data
- The loading the transmission of data
- Handling large scale of data dependent tasks using schedulers
- The efficient programming models for data handling and processing.

Scheduling plays a major role in efficiently handling the large scale RS data intensive jobs. In the below section we discuss one such scheduler for addressing Data intensive jobs in RS Big Data Clouds.

4.4 Data intensive scheduling in Big Data Clouds

Big Data applications demand more computation power and large volume of data for processing, there is a need to access, transfer, and modify massive data sets stored in distributed storage resources as shown in Figure 4.4 RS Big Data setup. Data-Intensive computing is a class of distributed computing adopts data parallel approach to process the large volumes of data typically in size from terabytes to peta bytes with the primary objective to maximize the resource utilization and minimize the turnaround time of the jobs. Completion time of the job is dependent on the data make span, which involves the communication band width and the compute time which includes applications execution time on computational resources and migration time in staging the input and output files. Due to the advancement in the science and technology massive amounts of scientific data in the areas such as earth observation, climate analysis is collected from the sensors and stored on multiple storage servers for further analysis by the scientific community. While technology has made massive data storage cheap and bandwidth abundant, the ability to extract valuable knowledge from multiple types of data obtained from multiple sources in real time is a grand challenge problem and poses the need for better data intensive scheduling techniques.

For example considering the earth observation sensors generating the datasets contain sensors for different bands like Panchromatic and Multispectral. Satellite orbits the earth; the sensors sweep the surface building scan lines. The typical computation of this satellite data can be data product generation [80] [81] [82] is follows: A portion of earth is specified through latitudes

and longitudes end points. A time (typically 10 days to one year) is also specified. For any portion of the earth within the specified area, data product generation such as geocode, ortho and DEM may be generated by picking the images. Large number of data intensive jobs which demand for more computation also data, and need to be served by reducing the overall completion time of all the jobs. Hence, there is a need of a system for effectively processing jobs. In this section various such issues in solving the data intensive jobs of the system along with the specific characteristics of the jobs are mentioned.

Jobs need the data, compute and application services for processing; in some cases Jobs may need the same data, while the application services may differ. These application services are published by the application providers, and may be collocated with the data, compute or may not be. In case of the data being used by the jobs are same, a pre processing stage can be applied to group them as a single Entity, enabling in reducing the data transfer time in the system for the group rather than individual job The data is replicated on the multiple data hosts, hence the mechanism can be devised to construct the multiple channels for the data transfer enabling the minimal data consolidation time. To reduce the data consolidation times, the job grouping is applied; in the job grouping the jobs which need similar data are grouped to one family.

To address the data intensive scheduling we present Data Aware Scheduler in Big Data setup which effectively considers both data and computational requirements for large jobs handling.

4.4.1 Data Aware Scheduler (DAS) characteristics

Providing timely results in the face of rapid growth in data volumes has become an important for many scientific and business intelligence analytics. As more and more data is generated at faster-than-ever rate, processing these large volumes of data is becoming challenge for several scientific analytics. The importance of data aware scheduling is increasing with rapid growth in data volumes, and to cope with this data and yet provide timely results. A brief overview of Big Data Clouds is depicted in Figure 4.5, it shows a scenario where an application requires data and computing resources from the providers who are decoupled and geographically dispersed.
Several characteristics of Data Aware Schedulers are described below.

- Seam less access to data and computing resources.
- Ability to search the data set and identify the appropriate data sets those are required
- Transfer the data from the data storages to computing resources for process and analysis

- Select the best suitable computational resources for processing the data.
- Effective utilization of underlying band width between data storage services and computing resources.
- Effectively map the jobs to the best computing resources so as to achieve high performance and higher throughputs.
- Grouping the jobs based on the data awareness so as to minimize the data transfer migrations from data storages to computing resources.
- Make use of the data replica locations so as to achieve better location proximities.
- Reducing the communication and computation delays, thus increasing the total throughput of the system.
- Hide the complexities of underlying infrastructure by transforming user requirements into operations, which are then carried out without user intervention.
- Provide high level programming model APIs, write the logic and submit the jobs to the system.
- The system would run the jobs and the output is presented at minimal time as possible.
- Providing the grouping mechanism wherever is possible.
- Apply both data push and pull models whichever is necessary.

In many scientific workflows, particularly those that operate on spatially oriented data, jobs that process adjacent regions of space often reference large numbers of files in common. Such, workflows, when processed using workflow planning algorithms that are unaware of the application's file reference pattern, result in a huge number of redundant file transfers between data storage repositories and consequently perform poorly. Hence, the scheduling aspects should have a planning for spatial workflows for execution based on the spatial proximity of files and the spatial range of jobs. Overall, Data Aware Scheduler has to maximize the resource utilization, by minimizing the turnaround times of the job executions thus achieving higher throughputs. In the next section we describe the architecture of Data Aware Scheduling followed by its workflow.

4.4.2 Data Aware Scheduling (DAS) Architecture and Workflow

Data Aware Scheduling architecture is depicted in Figure 4.6, with four basic elements, computing infrastructure providers, data providers, analytics/applications developers/users, and

Data Aware Scheduler. Compute providers offers a large scale computing infrastructure, data providers service the data on demand, scheduler broker periodically collects the jobs from the pool and determines the effective schedule to increase system throughput.

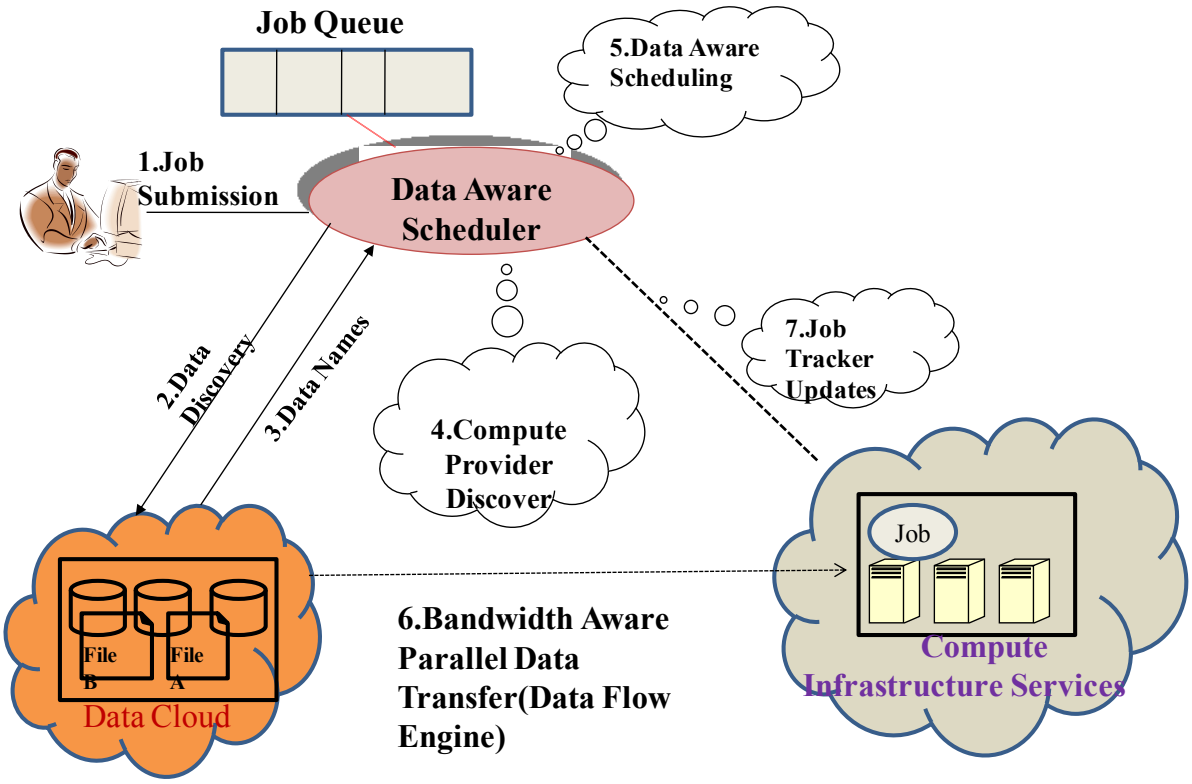


Figure 4.6. Data Aware Scheduler interfacing with Data and Computing resources in Big Data Clouds

The workflow is depicted in Figure 4.7. The several activities in Data Aware Scheduling are described below.

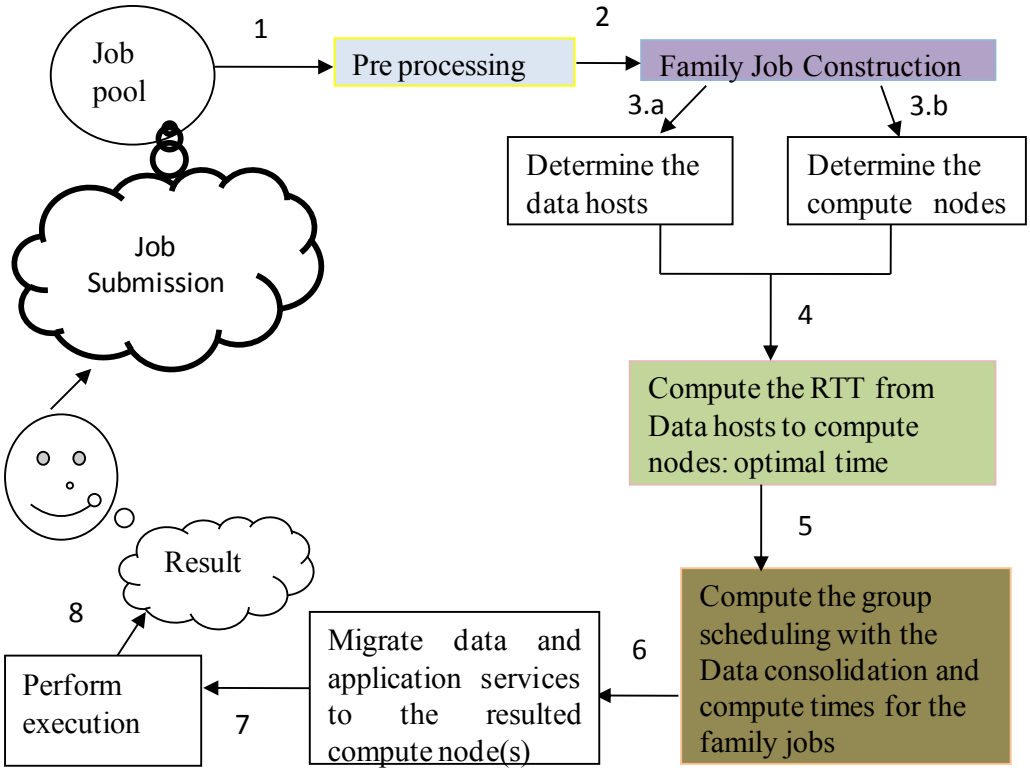


Figure 4.7. Workflow in Data Aware Scheduling

i. **Pre processing Stage:** Jobs in the pool are periodically collected and preprocessing has done for determining the jobs for which data is common. Here, each data is identified by the name and each job has unique job id. The data/file names required for the jobs are queried, and based on the query results, a set with the of data/file names is constructed. The data/filenames along with the job set is passed onto the next stage for grouping the jobs based on the data commonality the jobs are having.

ii. **Grouping the job:** Based on the preprocessing output stage, the jobs are categorized into several groups. Each group may consist of one or more number of jobs based on the common data requirements they have, as discussed in section A data aware modeling.

iii. Determining the data hosts and computing hosts: data replica locations and computing resources are selected, based on the data they have received from the previous step.

iv. Computing Round Trip Time (RTT), data chunk volumes for data migration to computing resources: Based on the data replica locations the data hosts are selected, and

similarly the amount of data to transfer from each of the data hosts to the computing resources would be determined. This is discussed in section 4.5.3.

v. **Estimation of data consolidation and computing times:** Times are estimated based on the consolidation of data and computing times between the data hosts and computing resources. This model is discussed in the next chapter using genetic approach for mapping the jobs to the computing resources.

vi. **High Throughput data transfers:** Based on the evolutionary genetic model the jobs would be mapped to the computing resources and data transfers would take place from the data resources to the computing nodes.

vii. **Jobs execution:** the jobs are run and the end results are sent to the users.

Below the model for the Data Aware Scheduling is discussed.

4.5 Data Aware scheduling model

Data Aware scheduling model is built on top of the three key factors, such as data dependency among the jobs, the information of the data required by the jobs and which is multiple replica locations at various geographically distributed locations, and the available bandwidth from such replicated storage data servers to the computing nodes. Looking into those three key factors, below we describe the several steps by which the model is derived.

In this section we describe the several steps for Data Aware scheduling model derivation, objective function for scheduling, and determine the data consolidation timings. Below we describe several notations used for deriving the model, followed by derivations for objective function for minimizing the scheduling makespan, and data consolidation timings for minimizing the data migration timings for job scheduling.

A. Notations used

Mathematical notations for the problem formulation are described in Table 4-1.

Table 4-1.Mathematical Notations

J = Total number of jobs
N= Total number of computing nodes in the grid
H= Total number of data service/providers
F= Total number of families
w_f= Total number of jobs in the family f.
TD_{fi} = Data Make Span(Consolidation time) in minutes of the family f∈F on Node i.
r_{hi} =Estimated packet transmission time in seconds between data provider h∈H to compute node i∈N.
X_f= Amount of data required in GB for the family f∈F.
x_{fhi} = Data chunk in GB from the data provider h∈H to compute node i∈N for the family f∈F.
ρ_{hi}= Weight assignment to the channel from data provider h∈H to compute node i∈N.
TR_{ji} = Turnaround time of the job j on node i.
TS_{ji}= Setup time of the job j on node i.
TA_j= Arrival Time of the job j.
Δ_{fi} = Decision variable.
δ_{fi} = Assignment variable.

B. Objective Function

The objective is to minimize the turnaround time of the jobs over the computing nodes.

$$\text{Minimize} \sum_{j=1}^{J}\sum_{i=1}^{N}\sum_{f=1}^{F} TR_{ji}\Delta_{fi}\delta^{f}_{j}$$

$$\delta^{f}_{j} = \begin{cases} 0 & \text{if } j \notin f \\ 1 & \text{if } j \in f \end{cases}$$

$$TR_{ji} = TD_{fi}/w_f + TS_{ji} + TL_{ji} - TA_j$$

Where

- TR_{ji}: Turnaround time of the job jЄf on computing node i.
- TD_{fi}: Data consolidation of the family f on computing node i.
- TS_{ji}: Setup time of the job jЄf, on computing node i.
- TL_{ji}: Length of the job j∈f, on computing node i.
- TA_j: Arrival /Submission time of the job j∈f.

Subject to the following constraints

- A family is assigned to either one compute node or none.

$$\sum_{i=1}^{N} \Delta_{fi} \leq 1 \text{ for } f = 1,2, \ldots . F$$

$$\Delta_{fi} \in \{0,1\}$$

- Compute node can have either none or many families assigned to it

$$\sum_{f=1}^{F} \Delta_{fi} \geq 0 \text{ for } f = 1,2, \ldots . N$$

$$\Delta_{fi} \in \{0,1\}$$

C. Determining the data consolidation time

Network traces between the computing resources and data hosts/providers are used to estimate the channel bandwidth availability. Such stored network traces over a time period for example 15 minutes are used as parameter to estimate the data quantity to migrate from each of the data providers to compute node. Below we will discuss the procedure for computing the percentage of the data to be obtained from each of the data providers to the compute nodes. Let us denote the compute node by *i*, data provider by h, job by j, and the family by f.

The families are constructed using the family graph as discussed above. The family may have one or more jobs if they have common data. If each family consist of only one job, then F=J, otherwise F<J. Let w_f be the number of jobs in the family f also called weight of the family, then $w_1+w_2+\ldots..+w_F=J$. In the family job scheduling problem, data consolidation time is same for all the jobs that belong to the same family. Data Consolidation time is defined as the maximum time to consolidate the data from the identified data providers to the compute node. Data Consolidation time TD_{fi}, is the maximum time required to bring the data from the data provider(s) to the compute node *i* for the family f, and is defined as

$$TD_{fi} = \max_{h=1,H} (x^f_{hi} * r_{hi}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where x^f_{hi} is the chunk size and r_{hi} is the estimated time from the previous historical traces, for the family f from data provider h to compute node i. x^f_{hi} can be computed as below.

$$x^f_{hi} = \rho_{hi} * X_f \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Where X_f is total data size required for the family f, and ρ_{hi} denotes the weight assigned to the channel from host h to compute node i, is defined as

$$\rho_{hi} = (1/r_{hi})/\sum_{l=1}^{H}(1/r_{li}) \dots\dots\dots\dots(3)$$

The time is estimated using the previous history of packet transmissions over a time period. Simplifying equation (1),using equations (2) and (3), we get

$$TD_{fi} = (1/r_{hi})\Big/\sum_{l=1}^{H}(1/r_{li}) * X_f * r_{hi}$$

$$= X_f \,/\, \sum_{l=1}^{H}(1/r_{li})$$

4.5.1 Data dependency jobs

In a pool of jobs, there are could be jobs for which the data required is common. In general, it happens for the jobs which are highly dependent on the data sets, like applications in domains like image processing, earth observation systems, bio medical applications etc. To minimize the data transfer issues, such data dependency jobs can be grouped as sub groups for further processing, so as to achieve the higher throughputs, and minimizing the data consolidation timings.

4.5.2 Data replicated storage servers

The jobs with common/overlap data are grouped together, called "**family**". To discover the grouping, metadata attributes such as object identifiers, and key/value descriptions are used as parameters. Object identifiers uniquely identify the objects in a bucket, and the object metadata is a set of name-value pairs for describing the data content. Object based storage mechanisms such as Openstack Swift [131], and Amazon S3 [67] offers object keys and the associated metadata tagged with the files/objects. Object metadata is of two types- system metadata and user-defined metadata. System metadata describes the object creation dates, storage class information etc., and the user-defined metadata tags the additional information for the objects. First, we apply the query to discover the jobs with similar object identifier tags, followed by key/value pair combination for finding the common/overlap data. However, the discussed methods are limited, but, these could be extended to other data overlap/commonality computing techniques.

Graph data structure, we call here as family graph is used for grouping the jobs. In the family graph, job is represented as node and the data required by both the jobs (adjacent nodes) is represented by the edge. A sample family graph with 7 jobs numbered from 1 to 7, and three data sets named from X_1 to X_3 are shown in Figure 4.8.

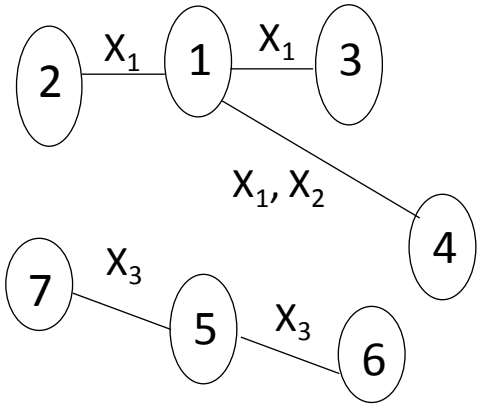


Figure 4.8. Family graph

The graph indicates that the jobs 1, 2, 3, and 4 require the data with id X_1, the jobs 1 and 4 require the data with id X_2, and jobs 5, 6 and 7 require the data with id X3 for processing. The families are formed by computing the connected components of the graph. The graph in Figure 4.8, results in two connected components with the nodes 1, 2, 3, and 4 for the first component, the nodes with 5, 6, and 7 for the second component. The resultant connected components form two groups or two family jobs which are to be processed further.

As mentioned in section 4.2 of RS Big Data Clouds, the data would get replicated onto multiple data storage repositories. Hence, instead of transferring the data from a single data storage provider, the information from multiple storage repositories could be used. This would increase the transfer of multiple data sets in parallel simultaneously from several data locations, as discussed below.

4.5.3 Bandwidth aware modeling

This model performs the data consolidations, followed by high throughput data transfers. By identifying the data hosts for each family job If the data hosts identified are more than one (it may occur due to replication of the data among the data host), then data transfer can be initiated from the identified multiple hosts. RTT (Round Trip Time) can be used to compute the amount of the data to be migrated from each of the data host to compute node. Below we will discuss the

procedure for computing the percentage of the data to be obtained from each of the data hosts to the compute nodes. Below we will discuss the procedure for computing the percentage of the data to be obtained from each of the data hosts to the compute nodes. Figure 4.9 depicts D_1 to D_H are the Data hosts and C_1 to C_N are the compute nodes. Let denote the compute node by *i*, data host *h*, job by *j* , and family by *f*.

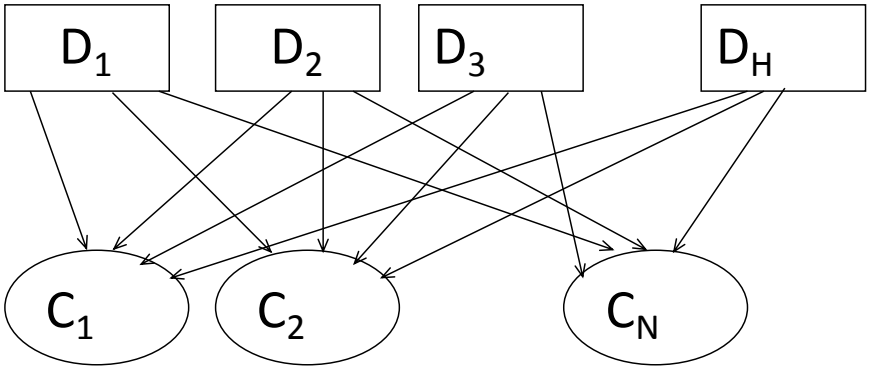


Figure 4.9. Data host to compute node mapping

The families are constructed at the pre processing stage mentioned above; the family may have single job or more jobs depending on the data dependency between the jobs. If each family has single job then then $F=J$,i.e number of Families is equal to number of jobs else $F<J$. Let w_f be the number of jobs in the family *f also* called weight of the family, then $w_1+w_2+.....+w_F=J$.

4.5.4 Scheduling methodology

The several steps in scheduling methodology is discussed below. pre processing, Family job construction followed by the optimal scheduling. Data and application services are migrated from data provider(s) and application service(s) respectively to the compute provider(s). The following steps are followed in designing the optimal schedule for the system

- **i.** Job pool: Queue of jobs to be processed.
- **ii.** Pre processing stage: grouping the jobs based on metadata attributes to construct the family jobs.
- **iii.** Family Construction: Using spatial graph and connected components to form the family job. Family job is defined as group of jobs for which the input data is same or nearly same.
- **iv.** Determine Data providers: Discovering the data providers who can supply the data of interest with the replicated sites.

v. Determine Compute nodes/providers: Locating the compute providers to process the family jobs.

vi. Round Trip Time (RTT) estimations: Estimation of RTT delay between data providers and compute providers.

vii. Data and Compute Time consolidation: Process of consolidating the computing and data migration times. This will enable to find the better schedule map to minimize the schedule makespan.

viii. Migration: Based on the schedule map, migration of the data and applications services to the compute nodes for execution.

ix. Execution: Initiating the execution on the compute nodes once the schedule is ready.

4.6 Discussion and Summary

The proposed data aware scheduling model addresses the data intensive applications on the decoupled computing, and storage repositories, by effectively scheduling the applications by grouping them based on the data needed for processing, by making use of the available bandwidth between the computing and storage repositories effectively. Here, the resources such as computational, and data files required are geographically dispersed, and the data is replicated over several storage distributed repositories. Here, we have discussed the problem statement, discussed several remote sensing earth observation system applications where the data and computing resources, are decoupled and offered as services for further explorations by several scientific researchers. We have discussed several motivating applications, where the computing elements and storage resources are decoupled, and the significance of brining the data to the compute for effectively processing such large scale applications. Later, we have discussed how these Remote Sensing turns into Big Data problems processing over Clouds as back end infrastructures.

Here, discussed several elements in Remote Sensing Big Data, followed by need for addressing Data Aware Scheduler, its architecture and workflow. We have discussed several steps in the workflow, and presented the need for grouping the jobs into a family; where the jobs require some similar kind of data for processing. For job grouping, we have presented a graph connected components approach, for computing the families. Based on the families, we have presented a

mathematical model for effectively consolidating the data so as to minimize the data migration times between the data storage repositories, and computing elements. Here, we have discussed the model, presented the mathematical notations, and derived the objective functions, which minimizes the Makespan of the jobs. Here, we have illustrated the derivations for data consolidation from replicated storage repositories, and presented a bandwidth aware model, which would help in building the high speed channels for transferring the large data volumes as chunks from multiple storage repositories to the computing nodes, so as to achieve higher throughputs. The scheduling methodology identifies the several steps in grouping the jobs for which data is common, and employs the steps for both data consolidation and computing times.

In the next chapter, we discuss Data Aware Scheduling model using evolutionary genetic approach, implementation of the Data Aware Scheduling model using genetic approach, fitness function for the genetic operation, crossover and mutation probabilities derivation, followed by experiments using Cloudsim Toolkit, and comparison of the model with non evolutionary models like Compute First, Data First, and Simulated Annealing.

Chapter 5

Genetic Approach for Data Aware Scheduling

In this chapter we discuss Data Aware Scheduling using evolutionary genetic approach, and graph theoretic connected components approach for job grouping and the effective mapping of the grouped jobs to distributed computational resources. Here, we present a group scheduling called as family scheduling, and genetic algorithm to map such constructed families of jobs onto the computational resources. While scheduling we apply the optimization techniques, so as to maximize the utilization of underlying computing resources, and minimize the turnaround time of the jobs. The proposed approach for data aware scheduling is twofold; first we propose a family spatial graph, which classify the jobs into family jobs using connected components of the graph, followed by finding a schedule to map such constructed family jobs onto the computing nodes. And the second is using data parallel approach, which uses Data-compute graph resulting in high compute-communication ratio to minimize the turnaround time of the job schedule. Although, the experiments which are conducted are specific to the Remote Sensing data, but the proposed scheduler is generic, such that it can be applied for job scheduling purposes, wherever the jobs have some common data requirements.

5.1 Introduction – Genetic Algorithm

Genetic Algorithm (GA) is a method for solving optimization problems based on a natural selection process for both constrained and unconstrained problems. GA mimics biological evolution, and repeatedly modifies a population of individual solutions, using genetic operators. Genetic Algorithms(GA) have shown to be well suited for high-quality solutions to larger NP problems and currently they are most efficient ways for finding an approximately optimal solution for optimization problems [138] [139] [141] . GA developed by J.H.Holland [142] is a model of machine learning, which derives its behavior from a metaphor of the processes of evolution in nature. GAs is executed iteratively on a set of coded chromosomes, called a *population*, with three basic genetic operators: *Selection, crossover* and *mutation.* Each member of the population is called a chromosome (or individual) and is represented by a string. GA uses only the objective function information and probabilistic transition rules for genetic operations. The primary operator of GAs is the *crossover.* GAs will not involve extensive search and not try to find the best solution, but they simply generate a candidate for a solution, check in polynomial time whether it is solution or not and how good a solution it is. GAs will not always give the optimal solution, but a solution that is close enough to the optimal one. GAs begins with a set of candidate solutions (chromosomes) called population. A new population is created from solutions of an old population in hope of getting a better population. Solutions which are chosen to form new solutions (offspring) are selected according to their fitness. The more suitable the solutions are the bigger chances they have to produce. This process is repeated until some condition is satisfied. Genetic Algorithms are classified into several algorithms based on the process of carrying out the population of chromosomes from one generation to the next generation such as Simple GA, Steady State GA etc. In case of Simple GA, the first

5.2 GA Problem Formulation – Data Aware Scheduling

Steady State GA [139] is used by replacing percentage of chromosomes across the generations until the solution converges or till the maximum number of generation is reached. Genetic algorithm library (GA Lib) [139] is used for implementation.

Proposed Scheduling Algorithm Genetic approach GA

1. Initial Runs if any

a. Randomly generate **populationSize** chromosomes. Select **populationSize/InitialRuns** chromosomes fittest at each run and copy to the Initial population.

b. Repeat each run for the maximum preliminary generations.

otherwise

Generate the **populationSize** chromosomes at random for the Initialpopulation

2. Fitness: Calculate the fitness of all chromosomes of the Initial population. This initial population is carried to the generation for further operations.

3. Perform the below steps in the current generation:

a. Selection: Select two fittest chromosomes and send them to the next generation

b. Repeat the below steps for the remaining chromosomes of the population

i. Select two chromosomes with randomSelectionChance probability chromosome.

ii. Crossover: perform crossover for the above two chromosomes selected. Each crossover results in the two off springs. Apply roulette wheel selection

c. Mutation: Perform mutation on the chromosome having the probability below $1/10^{th}$ of percent. Apply roulette wheel selection.

4. Fitness Function

a. Fitness value: Calculate the fitness value for the chromosomes. The chromosomes that won't satisfy the constraints are added a graded penalty value to the fitness value.

b. Computing the rankings: The rankings are computed based the statistical methods.

c. The chromosomes are ordered rankings in the descending order of ranks.i.e fittest chromosomes are first worst are at end.

d. Carry the new population to the next generation

5. Replace : Replace the current population with the new population

6. Test: Test whether current generation crossed the maximum number of generations. If so, stop. If not go to step 2 with the new population.

Genetic algorithm library (GALib) is used for implementation. The basic constructor call of the GA is shown below.

GA(families,// Chromosome dimension

populationSize, crossover,

randomSelectionChance,

maxgenerations,

preliminaryruns,
max prelim generations,
mutation,
decimal,
possibleGeneValues,
crossover type,
true);

families, PopulationSize, maxgenerations, preliminaryruns, randomSelectionChance represent the chromosome length, size of the population in each generation, maxgenerations to be conducted, preliminary runs to be conducted and probability of random selection of chromosomes to be selected for the next generation respectively. Crossover, mutation, possibleGeneValues represents the crossover probability, mutation probability and the total compute nodes respectively. The last two parameters represent the cross over type and statistics like average deviation, average fitness to compute. Here we used randomselectionChance of two percent, maxgeneration of two hundred, crossover probability of 65% , mutation of one percent and cross over is experimented for single, two point and uniform. Below, the data structures, cross over types and mutation are illustrated.

- **Data Structure**

The data structure of the chromosome is shown in Figure 5.1. Here, the index represents gene is the family number and the value; gene code is the compute node.

Index	0	1	2	3	4	5	6	7	8
value	2	4	1	0	4	3	2	9	7

Figure 5.1. Chromosome representation

The above data structure indicates a chromosome of length with the possible gene space of 10(means integer representation). In the above data structure, we can observe that family0 is assigned to node2, family1 to node4, family2 to node1, family3 to node0 respectively. The above data structure indicates that node5,6 and 8 have no family assigned and for node4 both family 1&4 are assigned , for node2 both family0&6 are assigned.

a) Evolve Method

For the initial runs, population is selected at random; fittest chromosomes are carried to the next generation. The process is repeated for the maximum preliminary runs. In order to choose the fittest chromosomes for the next generation the operators like crossover and mutations are used.

b) Elitism

The next generation chromosomes are created by genetically mating fitter individuals of the current generation. Also we apply the elitism for the first two fittest chromosomes; these two will always survive to the next generation. This way an extremely fit chromosome is never lost from our chromosome pool.

c) Genetic Mating

Crossover and mutations are applied in the current generation to generate the population for the next generation.

d) Crossover

The crossover operation carried out the GALib under GASequence is as follows.

Select two chromosome at random from the population leaving the two fittest chromosomes. if the probability of selection is below the random selection chance or else if the fitness rank is high these two chromosomes are selected for the crossover operation. The crossover type can be one point, two point , roulette wheel. Generally the crossover probability applied is in the range 60% to 70%.

e) One/single point Crossover

The two genes of the selected parents are swapped resulting in the two off springs for the next generation. The two genes are swapped to the left side of the crossover point. The resultant two off springs listed below are one point cross over, are shown in Figure 5.2.

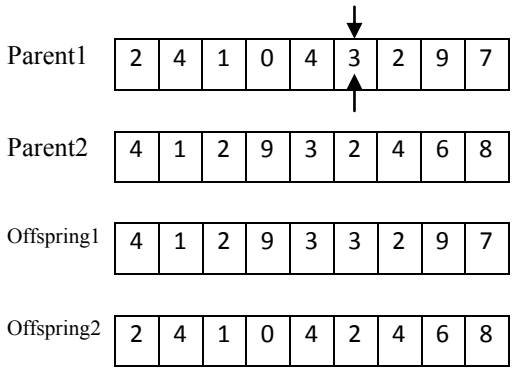


Figure 5.2. One/single point crossover

f) Two point Crossover

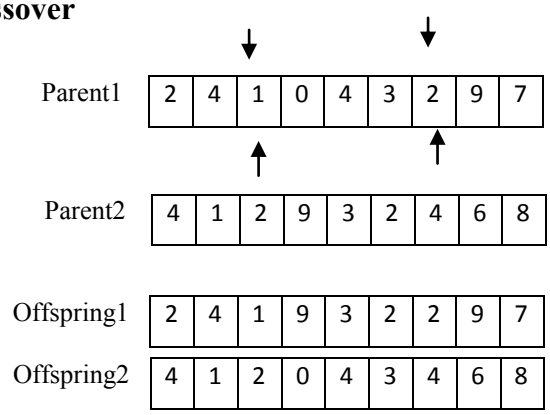


Figure 5.3. Two point crossover

The two genes of the selected parents are swapped resulting in the two off springs for the next generation. The two genes are swapped in between the crossover points are shown in Figure 5.3.

g) Uniform crossover

In uniform cross over the two genes at the crossover point are swapped, the sample is shown in Figure 5.4.

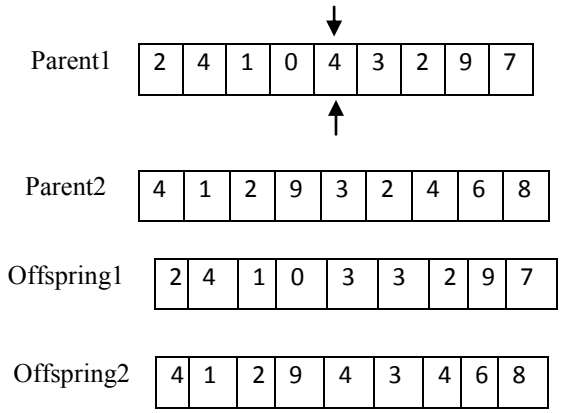


Figure 5.4. Uniform crossover

h) Mutation

The mutation probability chosen is $1/10^{th}$ of percent. The worst fit chromosomes with this probability in the current generation are mutated at random gene point and carried to the next generation. The operation of the mutation is shown below, gene3 at index 5(indexing is from 0) is mutated to gene 8.

Gene	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Gene Index	1	2	3	4	5	6	7	8	9

Node Id	2	4	3	2	4	5	3	2	1
Family id	1	2	3	4	5	6	7	8	9

Figure 5.5.Sample Chromosome for family scheduling

5.4.1 Chromosome representation

We choose integer based data structure for genomes representation. The gene index represents the familyid, and the genome indicates the compute nodeid to which the family is mapped. The data structure of the chromosome for family scheduling is shown Figure 5.5.

Consider F families to be mapped onto N computing nodes. Each family f will go to only one computing node, whereas a computing node may or may not be assigned with families. This yields a possible assignment size of N^F which is an exponential large value. Let us assume that there are 9 families and 10 computing nodes. Figure 5.5 represents that familyid=2 is assigned to nodeid=4, familyid=1 is assigned to nodeid=2, and so on. From the above figure it is also seen that nodeid=4 has been assigned with familyid=2 and 5; nodeid=2 has been assigned with familyid=1,4 and 8 and nodeid=6, 7, 8, 9 and 10 have not been assigned to any family.

5.4.2 Evolve methods

i. **Determining data workloads:** The data workload is determined based on the available bandwidth between the replicated sites and the compute nodes. Network traces based on round trip time over a time period is used as parameter (weight factor for the data channel) in our model to estimate the amount of data to be transferred from each of the replicated sites.

ii. **Optimal Schedule:** The schedule is based on steady state genetic approach using turnaround time minimization as fitness value.

iii. **Data and applications migration:** Based on the schedule map, data and application services would be migrated to the compute nodes.

iv. **Execution:** Jobs execution on the compute nodes, and the deletion of the temporary and migrated data sets from the computing nodes.

v. **Result:** Final result sent to the end user.

- **Methodology**

Here, we discuss the methodology for family construction, notations and problem formulation. Family to node mapping is represented as the weight matrix. The problem can be solved as the bipartite assignment problem, but the limitations are, a node can have maximum of one family assigned, although it has enough processing elements for handling more than one family. Hence, this problem reduces to 0/1 knapsack which can be solved using greedy, dynamic programming or evolutionary techniques like genetic algorithms. In the proposed group scheduling, three possible schedules may occur for the computing nodes during execution, such as:

(i) No family assigned.

(ii) With exactly one family assigned.

(iii)More than one family assigned.

Based on the schedules described as above for a compute node, we discuss below the GA based scheduling, chromosome representation, scheduling algorithm and the results obtained with the simulated data.

5.3 GA based scheduling

Below we describe the derivation of fitness function, followed by scheduling algorithm using Genetic approach.

5.3.1 Fitness Function

Initial populations of chromosomes are selected at random and the fitness function is applied. The fittest chromosomes are carried to the next generation based on the Steady State overlap percentage of chromosomes. The process is repeated either till the maximum preliminary runs are completed or the convergence of the objective value is achieved. In order to choose the fittest chromosomes for the next generation the operators like crossover and mutations are used. The next generation chromosomes are created by genetically mating fitter individuals of the current generation.

5.3.2 Scheduling Algorithm

Table 5-1 describes the fitness function pseudo code and Table 5-2 discusses the proposed GA for scheduling the family jobs using the fitness function. We have used CloudSim toolkit [135] with its new capabilities for file replication, simulated object storage identifiers for the data sets to simulate the Big Data Clouds environment. We use a simulated network with computation and data storage nodes spread at several locations as shown in Table 4, depicting: (a) 4 locations CHYD, CBGL, CMUB, CDEL having 7, 6, 7 and 8 virtual computing resources. These 28 virtual compute resources provide an aggregate of 1400 processing elements.(b) 4 locations that provide 40 data storage nodes with corresponding simulated bandwidths.

Table 5-1. Fitness Function pseudo code

Algorithm. Fitness function F pseudo code
Input: Chromosome C
Output: Turnaround time of the schedule

1 Turnaround time T:= 0
2 For all genes C perform the following steps do
3 read gene index f and genome value i
4 Compute the jobs $\{J\} \in$ to f.
5 for all $j \in J$ do
6 Estimate data consolidation time TD_{fi}, compute total jobs W_f in family f.
7 Compute setup time TS_{fi}, estimated job length on node i, TL_{ji} and arrival time TA_j.
8 Compute the turnaround time TR_{ji} of job $j \in J$ on computing node i.
9 $TR_{ji} = TD_{fi}/W_f + TS_{fi} + TL_{ji} - TA_j$
10 $T := T + TR_{ji}$
11 end for
12 until end of chromosome
13 return T

Table 5-2. GA for schedule discovery

Algorithm: Scheduling Algorithm pseudo code
Input: Population, Generations, Crossover percent, Mutation percent, Gene length, Percentage of chromosome to carry forward(P,g,c,m,l,r)
Output: A Schedule for all the jobs

7.Initial Run: Randomly generate population P chromosomes.
8.Repeat
9.Calculate the fitness of all chromosomes using **Fitness function F**
10. Arrange the population in the ascending order of fitness value
11. Copy the r best chromosomes to new population.
12. for the remaining chromosomes; perform the crossover with percent c and mutation with percent m. Copy the new off springs to new population.
13. Replace the current population with the new population
14. Until maximum generations or convergence.

5.4 Simulation and Results

In this section we describe the simulations performed using Cloudsim tool kit, followed by several experimental results. The basic features of Cloudsim does not have the components such such as job grouping, and data awareness features. So, in order to test the model, we have extended the basic features of Cloudsim tool kit, and later the experiments are conducted, as discussed below.

5.4.1 CloudSim Tool kit

CloudSim goal is to provide a generalized and extensible simulation framework that enables modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services, allowing its users to focus on specific system design issues that they want to investigate, without getting concerned about the low level details related to Cloud-based infrastructures and services. Main features of CloudSim are

- support for modeling and simulation of large scale Cloud computing data centers.
- support for modeling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines.
- modeling and simulation of energy-aware computational resources
- modeling and simulation of data center network topologies and message-passing applications.
- modeling and simulation of federated clouds.
- dynamic insertion of simulation elements, stop and resume of simulation.
- support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines.

Below we describe the extensions made over Cloudsim toolkit for the simulated experiments.

5.4. 1.1 **Extensions for Big Data computing in Clouds**

The extensions for incorporated for several virtual data centers and their management using the Application programming interfaces. The files are organized in the virtual data centers, as distributed files. The virtual data centers are managed as geographically distributed repositories, with varying bandwidth and latency speeds.

5.4. 1.2 **Data Replication**

The files which are organized on the virtual data centers are replicated at several sites. The indexing mechanism is organized to know the files, at what are all the places it is being replicated. The APIs are extended to incorporate the replication locations and the respective data centers. The index query mechanisms, list of the virtual data centers, the data locations, for the files supplied as input arguments.

5.4. 1.3 **Data and Bandwidth Awareness Parallel Data Transfer**

With the extended functionality of data replication over distributed virtual data center storage repositories, the models are built, which would understand the data locations, and gives the details such as available network bandwidth, and latencies among the computing nodes. Virtual computing nodes are built within the model, with several resources parameters attached to the virtual computing nodes. Based on the bandwidth and latencies between the virtual computing nodes, and the replicated storage repositories, the parallel data transfers with several simulated threads were extended.

5.4. 1.4 **Family construction - Job grouping**

Graph connected components based package is added Cloudsim for finding the connected components of the graph, which would result in the jobs for which data is common.

5.4.2 Simulated Data for experiments

The following experiments are conducted in the order described below.

Table 5-3. Simulated configuration of Big Data Clouds

Resource Name	Type	Virtual processing elements/ Bandwidth
CHYD1-7*	Virtual Compute Provider	200/100Mbps-200Mbps#
CBGL1-6	-do-	500/50 Mbps-100Mbps
CMUB1-7	-do-	400/200Mbps-500Mbps
CDEL1-8	-do-	300/20Mbps-200Mbps
HYDS1-5	Virtual Data provider	0/100 Mbps to 200Mbps##
BGLS1-10	-do-	0/10 Mbps to 100 Mbps
MBS1-10	-do-	0/256 to 512 Mbps
RJPS1-15	-do-	0/56 to 128 Mbps

**1-7 indicates a total of 7 numbers.*

#200/100Mbps-200Mbps indicate randomly generated PEs with a maximum of 200 for each compute provider, with the network channel bandwidth randomly simulated between 100Mbps to 200 Mbps.

##0/100 Mbps to 200Mbps represents the randomly generated data source without computing elements, the bandwidth varying from 100 to 200 Mbps for a total of five data providers.

- ***Data consolidation Analysis:*** Experiments are conducted to analyze the data transfer and consolidation timings from single site vs. the multiple replicated data sites considering the network traces over a time period.
- ***Determining optimal probabilistic values of genetic operators:*** Experiments are conducted to derive the optimal values of Genetic Algorithm (GA) operators for cross over, mutation and types of crossovers.
- ***Comparing with match making and heuristic techniques:*** The algorithm is compared with match making techniques like Data First, Compute First, and heuristics such as Simulated Annealing (SA).
- ***Comparison with Non family scheduling:*** The algorithm is compared with Non family i.e. without grouping the jobs.

5.4.3 Experiments

Below, we describe the several experiments performed for data aware scheduling for family and non family scheduling. First we analyze the data transfer and consolidation performance

measurements from a single site Vs replicated sites. Followed by, experiments to derive the probabilities operators for cross over and mutation operations. Later, we present the results for family Vs Non family scheduling, and comparison of the proposed GA technique with other heuristic and match making techniques.

5.4.3.1 Data Consolidation from Single Vs. Replicated Sites

Figure 5.6 depicts the data consolidation timings for the jobs obtained using single data storage Vs multiple replicated storage repositories. The results indicate that, data migration time from replicated sites is better as compared to the data migration timings from a single site. In case of a single site, the data is transferred from a single storage location as a single channel. However, in the case of replicated sites, the data available in multiple storages can be retrieved simultaneously, over multiple channels, as piece wise using the information of available bandwidth between the storage locations to the end client system. This would effectively utilize the channel available bandwidth effectively, the amount of the data to be retrieved from these storage repositories, is computed over a time period. Here for simulation a total 8 storage repositories are used to transfer the files for a total of 30 randomly simulated jobs, along with the necessary data sets required.

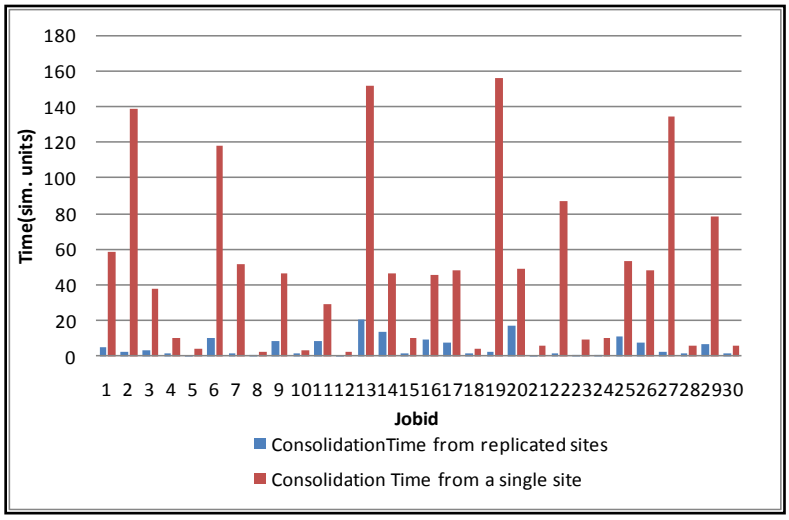


Figure 5.6 Transfer Times in replicated vs. Single

5.4.3.2 Genetic operation and probability derivation – Cross over and mutation

In this section, we conduct experiments to determine the probabilities parameters for cross over and mutation operations. We conduct 12 experiments, with each experiment, consisting of

population length of 100 chromosomes, and the experiments are repeated for a maximum of 100 to 200 generations. The experimental results are depicted in Table 5-4. As roulette wheel operator with the combination of one point, two point or uniform cross over, hence, we choose the roulette wheel as cross over operator, and conduct the probability derivation. Similarly, we also analyze the optimal probability value for the mutation operator.

Table 5-4.Experiments to determine genetic operators

Exp. no	Pop. Length	Total no. of generations
1,2	100	100
3,4	100	200
5,6,7	200	100
8	200	100
9,10,11,12	200	200

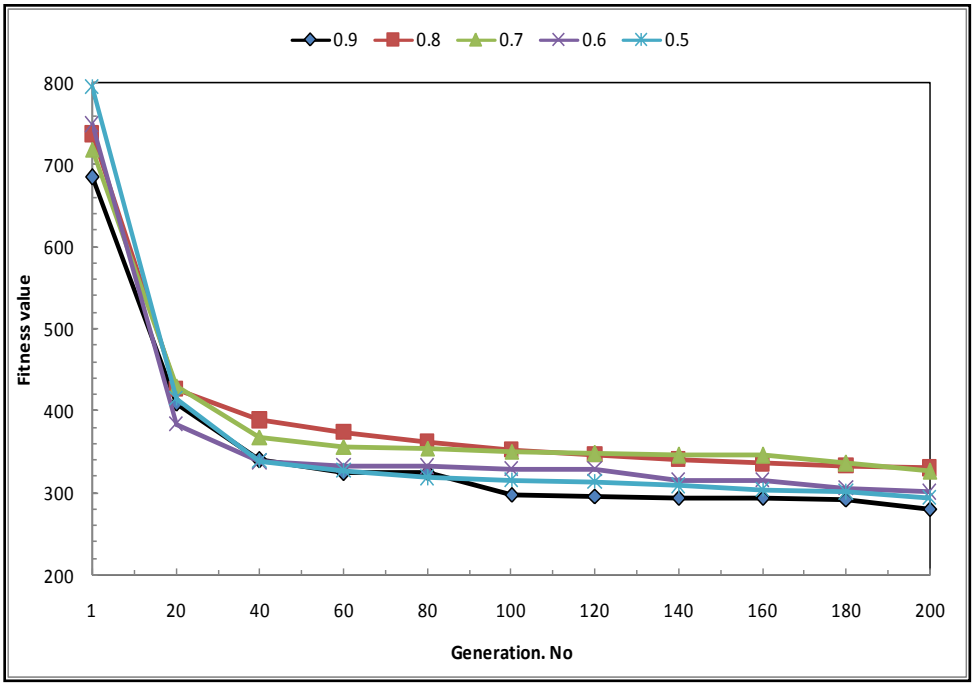


Figure 5.7. Fitness value convergence across generations

The results are shown in depicted in Figure 5.7 to determine the roulette wheel has better convergence across the generations while compared to the other genetic operators. Hence, the roulette wheel operator is selected for cross over operations for the next level experiments.

Experiments are performed with varying mutation probabilities to determine the appropriate mutation ratio by fixing cross over operator of 0.9 and roulette wheel, the experiments shown in Figure 5.8 indicates with mutation probability of 0.1 has the better convergence fitness value.

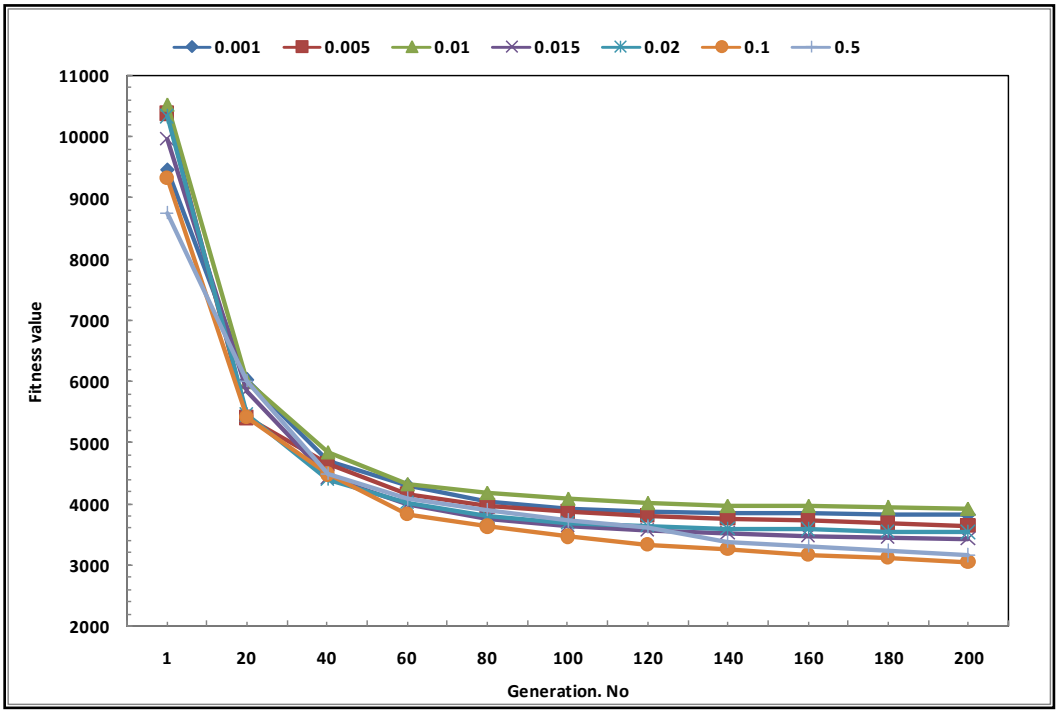


Figure 5.8. Mutation probability

5.4.3.3 Performance of Family Vs Non Family

Experiments are conducted for analyzing the turnaround times for the families and non-families from a single storage vs. replicated data storages. The results depicted in Figure 5.9, illustrate that turnaround time is lesser from the replicated sites while compared to the results obtained from the single site. This is due to, transfer of chunk of the data from the whole data, simultaneously from several storage repositories to the minimal data consolidations from replicated sites.

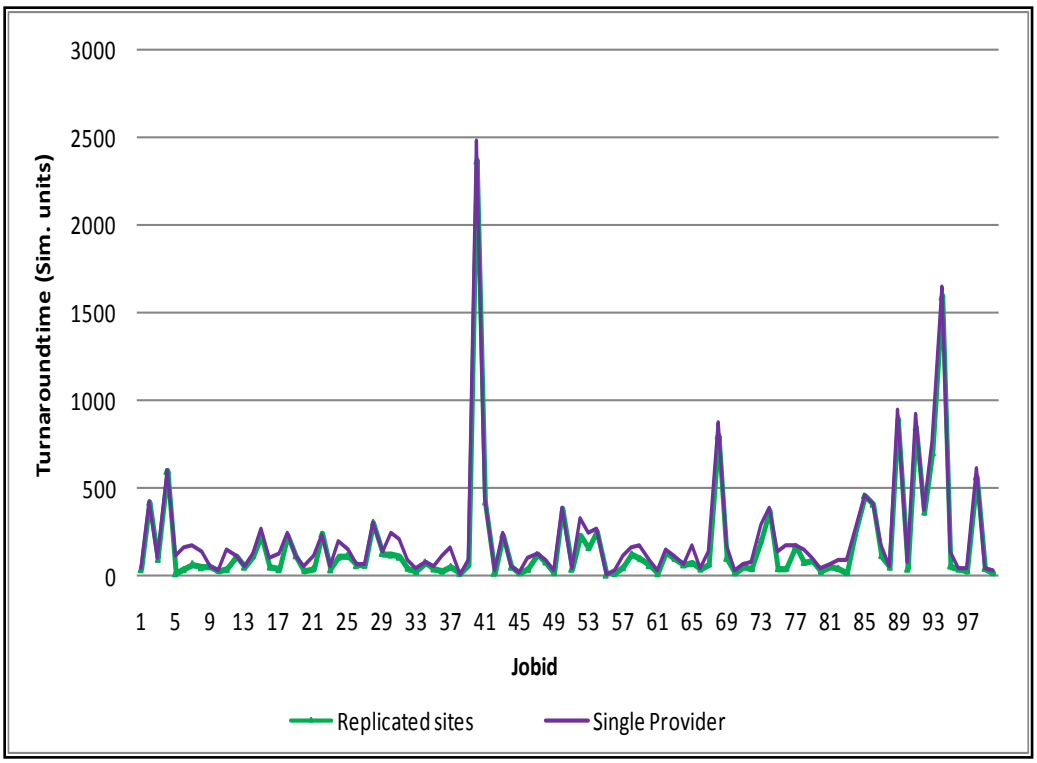


Figure 5.9. Turnaround times (data consolidation) of jobs (non family) from single Vs. replicated sites

Another set of experiments are performed for analyzing the turnaround times of family vs. non family scheduling is depicted in Figure 5.10. This is carried out by applying the data migration/consolidation from the replicated sites to the selected computing nodes. Illustrates, the jobs with the family grouping has resulted in minimal turnaround time while compared to the non-family. This is due to the data consolidation carried out one time for the entire family job. This would reduce the data migration overheads for each job and reduce the network bandwidth consumptions. However, for few jobs the resultant turnaround time is more while compared to non-family scheduling, which could be due to the grouping that has resulted in longer data consolidations and computing times for the jobs. The longer data consolidations is due to more numbers of jobs in the family, and the longer computing times is due to the availability of minimal computing elements at the selected compute node than actually demanded for processing. However, the overall makespan of the family scheduling is less than the non family scheduling.

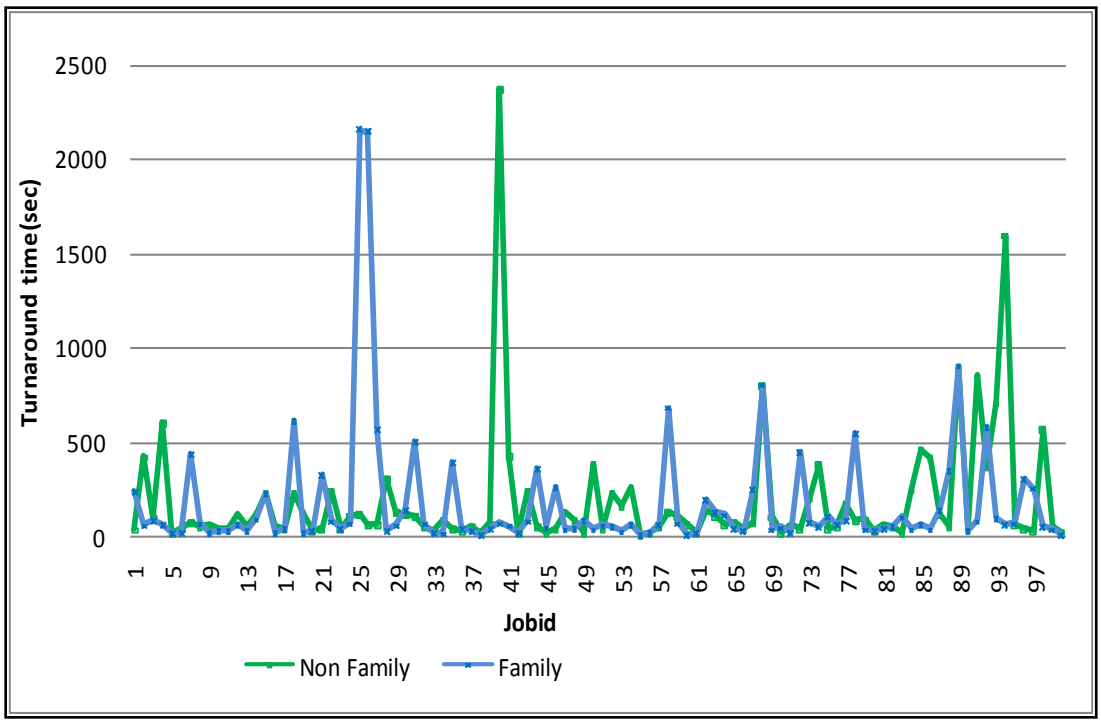


Figure 5.10. Turnaround Times for Family vs. Non Family

5.4.3.4 Comparison with other techniques – Match Making & Heuristics

The proposed GA is compared with match making and heuristic techniques discussed below. Here, two types of match making techniques such as Minimum Data consolidation First and Minimum Compute First are used. Later, heuristic technique such as Simulated Annealing techniques is discussed. The obtained results are compared with the proposed GA approach which is depicted in Figure 5.11.

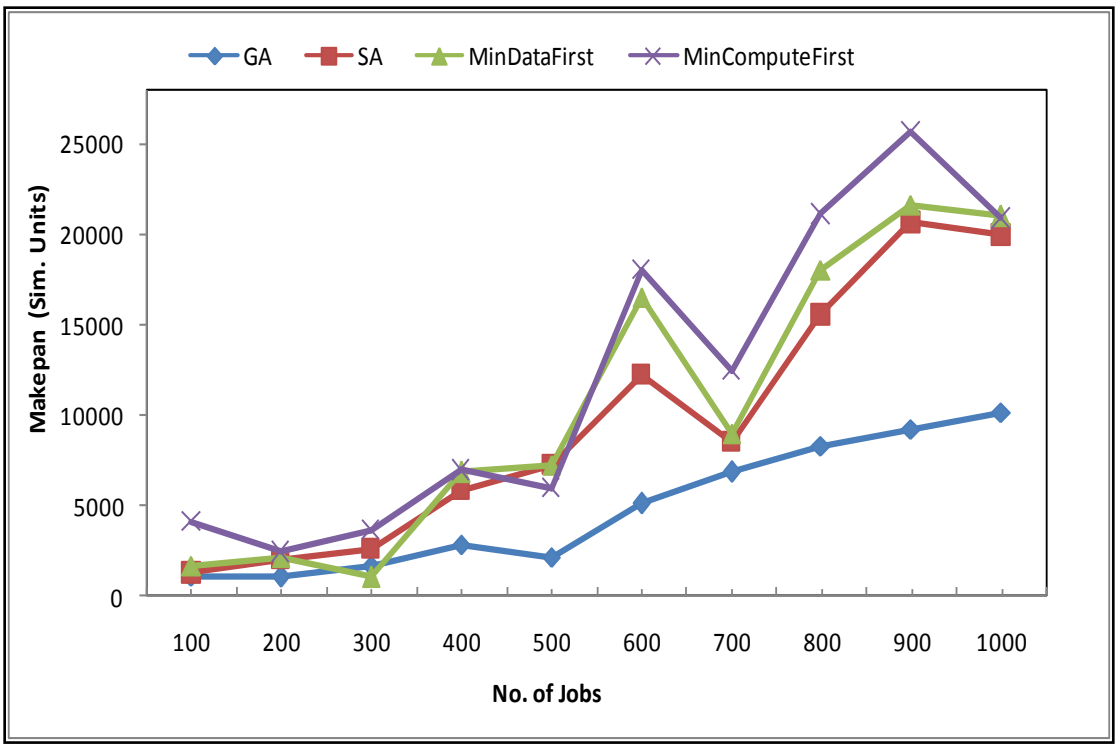


Figure 5.11. Comparing GA with other techniques

• ***Minimum Data consolidation First at node*** – In this mapping, a compute resource that ensures minimum data consolidation time is selected for the family.

• ***Minimum Compute First*** – In this mapping, a compute resource that ensures the minimum computation time is selected for the family.

• ***Simulated Annealing-*** In this mapping heuristics are applied by discarding the worst fit values from the current state to the next state and moving towards the best selection.

The results indicate that Minimum Compute First technique has resulted in larger makespan while compared to Minimum Data First and SA techniques. Minimum Data First and Simulated Annealing techniques have almost the same makespan value with performance better than Minimum Computer Fist technique. However, the proposed GA has resulted in minimal makespan while compared to matchmaking techniques and SA. This could be due to the natural evolution procedures of GA and fitness functions used to obtain the near optimal solution.

5.5 Discussions and Summary

The proposed family/group scheduling model addresses the data intensive problems to minimize the turnaround time of the jobs where the computing and data resources are decoupled. Here, we have introduced a group based scheduling model, called as family (the jobs for which the data required is common) schedule, and processed these jobs as a group. This would minimize the data migration overheads, and increases the throughput of the jobs.

The jobs with common data are grouped together, using the family graph, and connected components are used to compute the family group. Later, parallel data approach is applied for data migration based on the available bandwidth from the computing nodes to the storage repositories. The family schedule problem is addressed using Genetic Algorithm (GA), for which a schedule is computed by effectively mapping the data to the computing nodes. Based on high throughput model for data transfers, and graph connected components for family grouping, a chromosome data structure is defined, and fitness functions are derived, and Steady State GA is applied is applied to discover the optimal schedule.

CloudSim tool kit is used to simulate the environment. Here, both Computing Clouds, and Data Storage providers are simulated with varying bandwidths, and different communication links. The available bandwidth is computed using Round Trip Time (RTT), by sending the dummy packets from computing nodes to the data storages. The performance comparisons of the schedules are illustrated for the both family and non-family schedules from a single site and multiple replicated sites. The results indicate that, data migration from replicated sites show performance improvement over a single site. The experiments show that family schedule performs better over the non-family schedule, by minimizing the data migration overheads using the data parallel approach and computes the schedule, which would minimize both computational and data migration overheads. Finally, the results obtained from GA based data aware scheduling is compared with other scheduling mechanisms such as Computer First, Data First, and Simulated Annealing techniques.

Chapter 6

Extended Hadoop and MapReduce Models for Big Data Computing in Clouds

Hadoop Distributed File System (HDFS) and MapReduce model have become popular technologies for large scale data organization and analysis. Existing model of data organization and processing in Hadoop using HDFS and MapReduce are ideally tailored for search and data parallel applications, for which there is no need of data dependency with its neighboring/adjacent data. However, many scientific applications such as image mining, data mining, knowledge data mining, and satellite image processing are dependent on adjacent data for processing and analysis. In this chapter, we discuss the requirements of the overlapped data organization and propose a two phase extensions to HDFS and MapReduce programming model, called XHAMI, to address them. The extended interfaces are presented as APIs and implemented in the context of Image Processing (IP) application domain. We demonstrated effectiveness of XHAMI through case studies of image processing functions along with the results. Although XHAMI has little overhead in data storage and input/output operations, it greatly enhances the system performance and simplifies the application development process. Our proposed system, XHAMI, works without any changes for the existing MapReduce models, and can be utilized by many applications where there is a requirement of overlapped data.

6.1 Introduction – Extended HDFS and MapReduce

The amount of textual and multimedia data has grown considerably in recent years due to the growth of social networking, healthcare applications, surveillance systems, earth observation sensors etc. This huge volume of data in the world has created a new field in data processing called as Big Data [4] , which refers to an emerging data science paradigm of multi-dimensional information mining for scientific discovery and business analytics over large scale scalable infrastructure. Big Data handles massive amounts of data collected over time, which is an otherwise difficult task to analyze and handle using common database management tools [158] . Big Data can yield extremely useful information; however apart, urges new challenges both in data organization and processing [159] .

Hadoop is an open source framework for storing, processing, and analysis of large amounts of distributed semi structured/unstructured data [13] . The origin of this framework comes from internet search companies like Yahoo and Google, who needed new processing tools and models for web page indexing and searching. This framework is designed for data parallel processing at Petabyte and Exabyte scales distributed on the commodity computing nodes. Hadoop cluster is a highly scalable architecture, that spawns both compute and data storage nodes horizontally for preserving and processing large scale data to achieve high reliability and high throughput. Therefore, Hadoop framework and its core sub components i.e. HDFS and MapReduce are gaining popularity in addressing several large scale applications of data intensive computing in several domain specific areas like social networking, business intelligence, and scientific analytics, etc. for analyzing large scale, rapidly growing, and variety structures of data.

The advantages of HDFS and MapReduce in Hadoop eco system are – horizontal scalability, low cost setup with commodity hardware, ability to process semi-structured/ unstructured data, and simplicity in programming. However, HDFS and MapReduce, though offer tremendous potential for gaining maximum performance, but due to its certain inherent limiting features, does not confine to be used for all areas. Below we describe one such domain specific applications in remote sensing image processing.

6.2 Extended MapReduce Vs Existing Models

Below we discuss the comparisons of existing models Vs proposed XHAMI model for large scale data processing on Hadoop clusters based on the extensions of Hadoop Distributed File System (HDFS) and MapReduce models. Below we describe several data intensive models and their comparisons with the proposed XHAMI model.

Image processing and computer vision algorithms can be applied as multiple independent tasks on large scale data sets simultaneously in parallel on a distributed system to achieve higher throughputs. Hadoop is an open source framework for addressing large scale data analytics using HDFS and MapReduce programming models. In addition to Hadoop, there are several other frameworks like Twister [168] for iterative computing of streaming text analytics, and Phoenix [169] used for map and reduce functions for distributed data intensive Message Passing Interface (MPI) kind of applications.

Kennedy et al. [170] demonstrated the use of MapReduce for labeling 19.6 million images using nearest neighbor method. Shi et al. [171] presented use of MapReduce for Content Based Image Retrieval (CBIR), and discussed the results obtained by using around 400,000 images approximately. Yang et al. [172] presented a system MIFAS for fast and efficient access to medical images using Hadoop and Cloud computing. Kocalkulak et al. [173] proposed a Hadoop based system for pattern image processing of intercontinental missiles for finding the bullet patterns. Almeer et al. [174] designed and implemented a system for remote sensing image processing with the help of Hadoop and Cloud computing systems for small scale images. Demir [175] et al. discussed the usage of Hadoop for small size face detection images. All these systems describe the bulk processing of small size images in batch mode over HDFS, where each map function processes the complete image.

White et al. [176] discussed the overheads that can be caused due to small size files, which are considerably smaller than the block size in HDFS. A similar approach is presented by Sweeney et al. [177] and presented Hadoop Image Processing Interface (HIPI) as an extension of MapReduce APIs for image processing applications. HIPI operates on the smaller image files, which are bundled into a large block called Hadoop Image Bundle (HIB). In HIPI each image is

applied to only one map function, which has limitation in dividing the data into smaller file sets. All these said methods discussed aggregation of smaller images and mapping each image within the bundle as a whole to one single map function.

Potisepp et al. [178] discussed the processing small/regular images of total 48675 by aggregating them into large data set, and processed them on Hadoop using MapReduce as sequential files, similar to the one addressed by HIPI. Also, presented feasibility study as a proof-of-concept test for a single large image as blocks and overlapping pixels for non-iterative algorithms image processing. However, no design, or solution, or methodology has been suggested to either to Hadoop or MapReduce for either Image Processing applications or for any other domain, so that the methodology works for existing as well as new models under consideration. Table 6-1 illustrate the differences between XHAMI and other MapReduce models.

Table 6-1. XHAMI Vs Other MapReduce models

S. no	Model name	Underlying file system	Programming model	Remarks
1	Apache Hadoop	Hadoop distributed file system	MapReduce	Works on large scale clusters of commodity hardware based on Hadoop distributed file system. Generic file system and MapReduce, requiring customizations for several domain specific applications.
2	Twister	HDFS and In Memory	Twister MapReduce	Runs on the large scale cluster of machines with distributed shared file repository. Does the processing of text related applications for iterative applications.
3	Aneka	Windows	MapReduce	Runs on large cluster of

		distributed file system		desktop machines using windows distributed file system. Good for text related MapReduce processing, requires several customizations for domain specific models.
4	Dryad LINQ	Windows distributed file system	MapReduce	Runs on the cluster of windows servers using distributed file system of windows for large scale data parallel applications running on PC clusters.
5	Apache Hadoop Image Processing Interface (HIPI)	Hadoop distributed file system	MapReduce	Runs on large scale clusters of commodity hardware based on Hadoop distributed file system. This avoids the small files problem by combining the smaller image files into larger images. A single image is given to one Map function for processing.
6	XHAMI	Hadoop distributed file system	MapReduce	High level APIs for data abstraction and Map Reduce computations for a single large image volumes, where overlapped data among the blocks is the essential requirements.

6.3 XHAMI – Extended Hadoop and MapReduce Interface

Earth observation satellite sensors provide high-resolution satellite imagery having image scene sizes from several megabytes to gigabytes. High resolution satellite imagery for example Quick

Bird, IKONOS, Worldview, IRS Cartosat etc. [164] are used in various applications of analysis and information extraction like oil/gas mining, engineering construction like 3D urban/terrain mapping, GIS developments, defense and security, environmental monitoring, media and entertainment, agricultural and natural resource exploration etc. Due to increase in the numbers of satellites and technology advancements in the remote sensing, both the data sizes and their volumes are increasing on a daily basis. Hence, organization and analysis of such data for intrinsic information is a major challenge.

Ma et al. [165] have discussed challenges and opportunities in Remote Sensing (RS) Big Data computing, focused on RS data intensive problems, analysis of RS Big Data, and several techniques for processing RS Big Data. Two dimensional structured representation of images, and majority of the functions in image processing being highly parallelizable, the HDFS way of organizing the data as blocks and usage of MapReduce functions for processing each block as independent map function, makes Hadoop a suitable platform for large scale high volume image processing applications.

An image is a two-dimensional function f(x,y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x,y) is called intensity or gray level of the image at that point [166] . Image data mining is a technology that aims in finding useful information and knowledge from large scale image data [167] . This involves use of several image processing techniques such as enhancement, classification, segmentation, object detection etc. which use many combinations of linear/morphological spatial filters [166]

Many of the linear/morphological spatial filters demand use of adjacent pixels for processing the current pixel. For example, as shown in Figure 6.1, a smoothening operation performs weighted average of a 3X3 kernel window. The output of pixel *X* depends on the values of *X1,X2,X3,X4, X5,X6,X7,and X8*. Therefore due to the dependency, these types of operations cannot be performed on the edge pixels. Hadoop and many of the implementations discussed in Section 2, split the data based on a fixed size, which results in partitioning of data as shown in Figure 1. Each of the blocks is written to different data nodes.

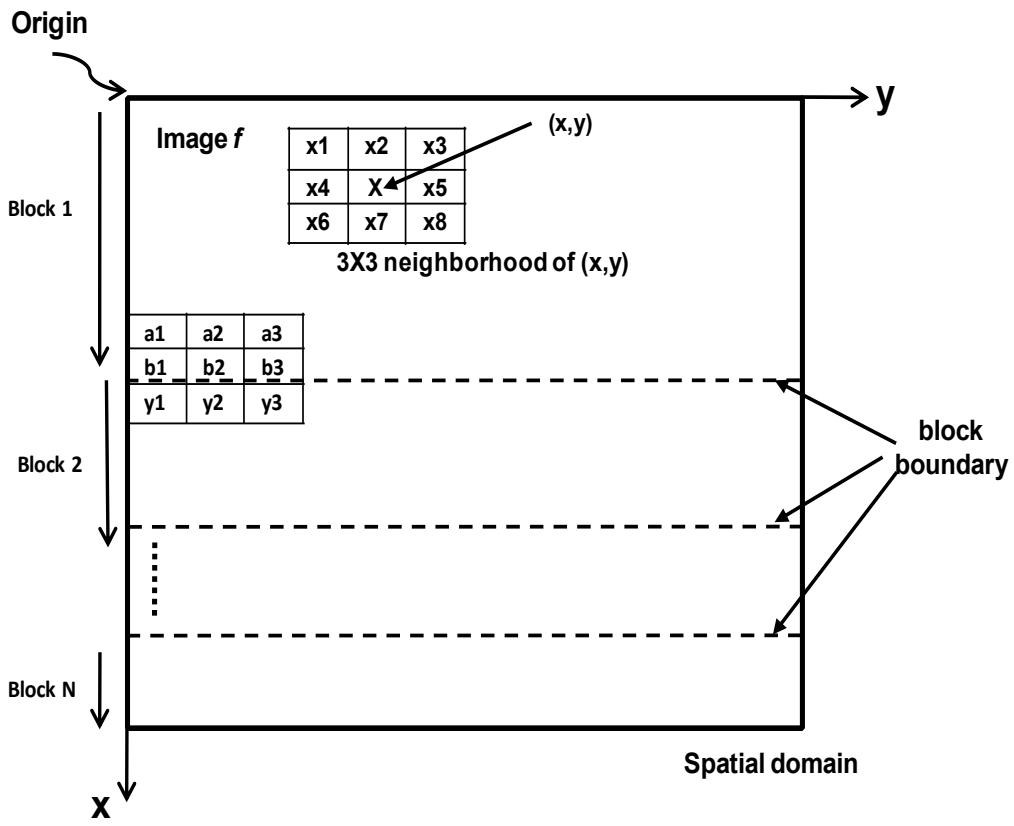


Figure 6.1. Image representation with segmented blocks

Therefore the boundary pixels of entire line *b1, b2, b3,..* in each block cannot be processed, as the adjacent pixels are not available at the respective data nodes. Similarly for the pixels marked as *y1,y2,y3,y4*,… also IP operations cannot be performed straight away. To process these boundary pixels i.e., the start line and end line in each block a customized map function to read additional pixels from a different data node is essential, otherwise the output would be incorrect. This additional read operations for each block increase the overhead significantly.

To meet the requirements of applications with overlapped data, we propose an Extended HDFS and MapReduce Interface, called XHAMI, which offers ***a two phase extensions to HDFS and MapReduce programming model.*** The extended interfaces are presented as APIs and implemented in the context of Image Processing (IP) application domain. We demonstrated effectiveness of XHAMI through case studies of image processing functions along with the results. Our experimental results reveal that XHAMI greatly enhances the system performance and simplifies the application development process. It works without any changes for the

existing MapReduce models, and hence it can be utilized by many applications where there is a requirement of overlapped data. Here, we addresses the issues related to processing large remote sensing images which run into several Megabytes to Gigabytes, addressing several issues related to data organization over HDFS, and processing them using XHAMI. The proposed extensions are applied for image processing applications, but the same can be extended to other domains also where such similar data dependency exists.

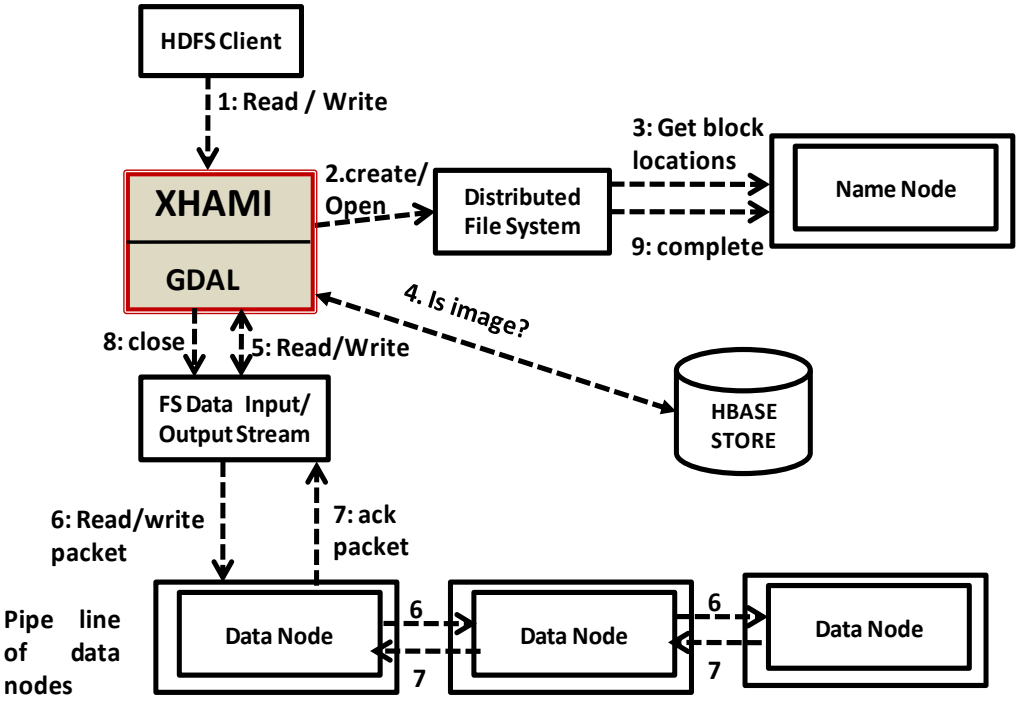


Figure 6.2. XHAMI for read/write operations

6.3.1 **Architecture**

Figure 6.2 depicts the sequence of steps in reading/writing the images using XHAMI software library over Hadoop framework. Initially, client uses XHAMI I/O functions (step 1) for reading or writing the data. The client request is translated into create () or open () by XHAMI, and sent to DistributedFileSystem (step 2). Distributed File System instance calls the namenode to determine the data block locations (step 3). For each block, the namenode returns the addresses of the datanodes for writing or reading the data. DistributedFileSystem returns FSDataInput/Output Stream, which in turn will be used by XHAMI to read/write the data

to/from the datanodes. XHAMI checks file format, if the format is in image type (step 4), then metadata information such as file name, total scans, total pixels, total numbers of bands in the image, and the number of bytes per pixel are stored in HBASE, this simplifies header information reading as and when required through HBASE queries, otherwise reading the header block by block is tedious and time consuming process.

Later on XHAMI calls FSDataInput/Output Stream either to read/write the data to/from the respective data nodes (step 5). Steps 6 and 7 are based on standard HDFS data reading/writing in the pipelining way. Each block is written with the header information corresponding to the blocks i.e. blockid, start scan, end scan, overlap scan lines in the block, scan length, and size of the block. Finally, after the read/write operation the request is made for closing the file (step 8), and the status (step 9) is forwarded to the namenode.

Below we describe techniques for data organization followed by extended HDFS and MapReduce APIs.

6.3.2 **Data organization**

The image is organized as blocks in HDFS along with overlap among the subsequent blocks. The image as a one directional sequence of bytes is shown in Figure 6.3a, the construction of the blocks, where the image is single dimension represented as blocks and the two ways of block arrangement is described in Figure 6.3.b, and Figure 6.3.c, depicts the blocks are constructed in two ways i.e. **(i) uni-directional**: partitioning across the scan line direction as shown in Figure 6.3.a, and **(ii) bidirectional**: partitioning both horizontal and vertical directions as shown in Figure 6.3.b. while construction, it is essential to ensure that, no split take place within the pixel byte boundaries. The methods are described below.

i) Unidirectional split: blocks are constructed by segmenting the data in across scan line (horizontal) direction. Each block is written with the additional lines at the end of the block.

ii) Bi-directional split: splitting the file into blocks in both horizontal and vertical directions. The split results in the blocks, for which, the first and last blocks have overlap with their adjacent two blocks, and all the remaining blocks have overlap with their adjacent four blocks. This type of segmentation results in large storage overhead which is approximately double the size of the

unidirectional segment construction. This type of organization is preferred while images have larger scan line lengths.

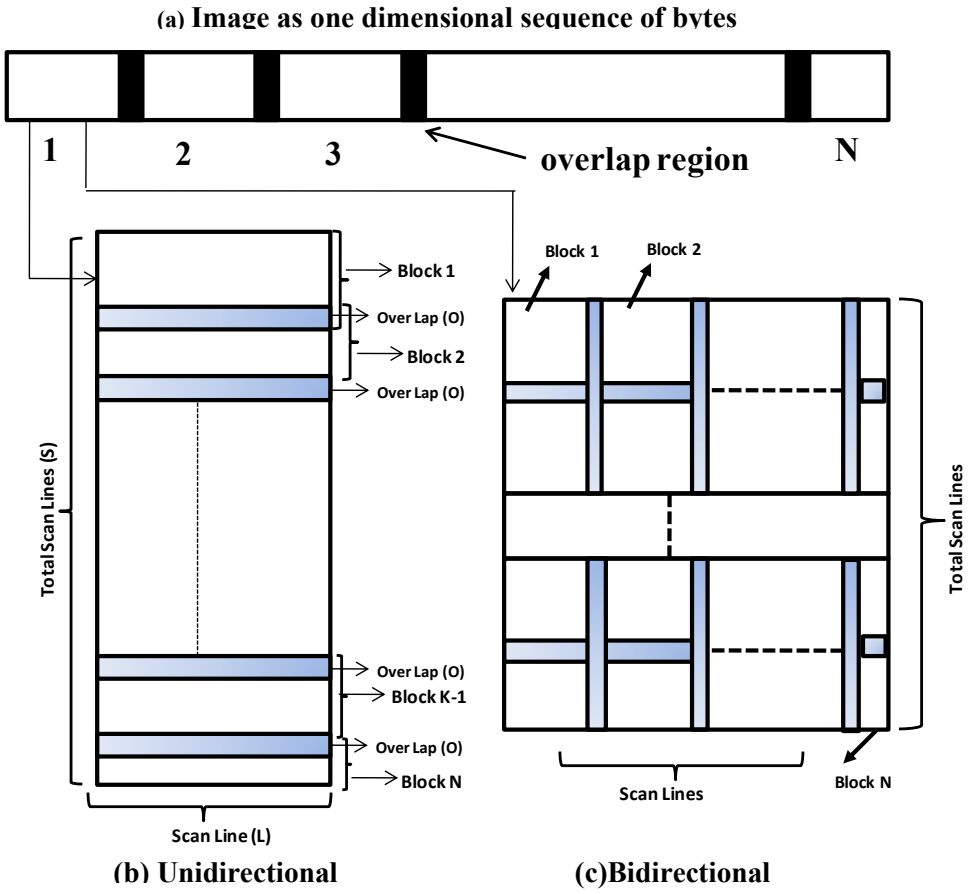


Figure 6.3. Block construction methods

Scan lines for each block *Sb* computed as

$$Sb = \left\lceil {}^{H}/_{(L * P)} \right\rceil$$

H = HDFS Default block length in Mbytes.
L = length of scan line i.e. total pixels in the scan line.
P = pixel length in bytes.
S = total number of scan lines.

Total number of blocks *T,* having overlap of α number of scan lines is

$$T = \left\lceil {}^{S}/_{Sb} \right\rceil$$

If $T^* \alpha > Sb$ *then* $T = T+1$.

The start and end scan lines $B_{i,s}$ and $B_{i,e}$ in each block is given below; N representing total scans in the image.

$$B_{i,s} = \begin{cases} 1, & i = 1 \\ B_{i-1,e-\alpha+1}, & 1 < i < T \\ B_{N-1,e-\alpha+1} & i = T \end{cases}$$

$$B_{i,e} = \begin{cases} B_{i,s} + S_b - 1 & 1 \leq i < T \\ S_b\, i = T \end{cases}$$

Block length is computed as below.

$$R_i = (B_{i,e} - B_{i,s} + 1) * L * P\ ,\ 1 \leq i \leq T$$

The blocks are constructed with metadata information in the header, such as blockid, start scan, end scan, overlap scan lines in the block, scan length, block length. Though, metadata adds some additional storage overhead, but, simplifies the processing activity during Map phase, for obtaining the total number of pixels, number of bands, bytes per pixel etc, and also helps to organize the blocks in the order during the combine/merge phase using blockid.

6.3.3 **XHAMI package description**

XHAMI offers Software Development Kit (SDK) for Hadoop based large scale domain specific data intensive applications designing. It provides high level packages for data organization and for MapReduce based processing simplifying the development and quick application designing by hiding several low level details of image organization and processing. XHAMI package description is as follows- XhamiFileIOFormat is the base class for all domain specific applications which is placed under the package xhami.io, as shown in Figure 6.4.

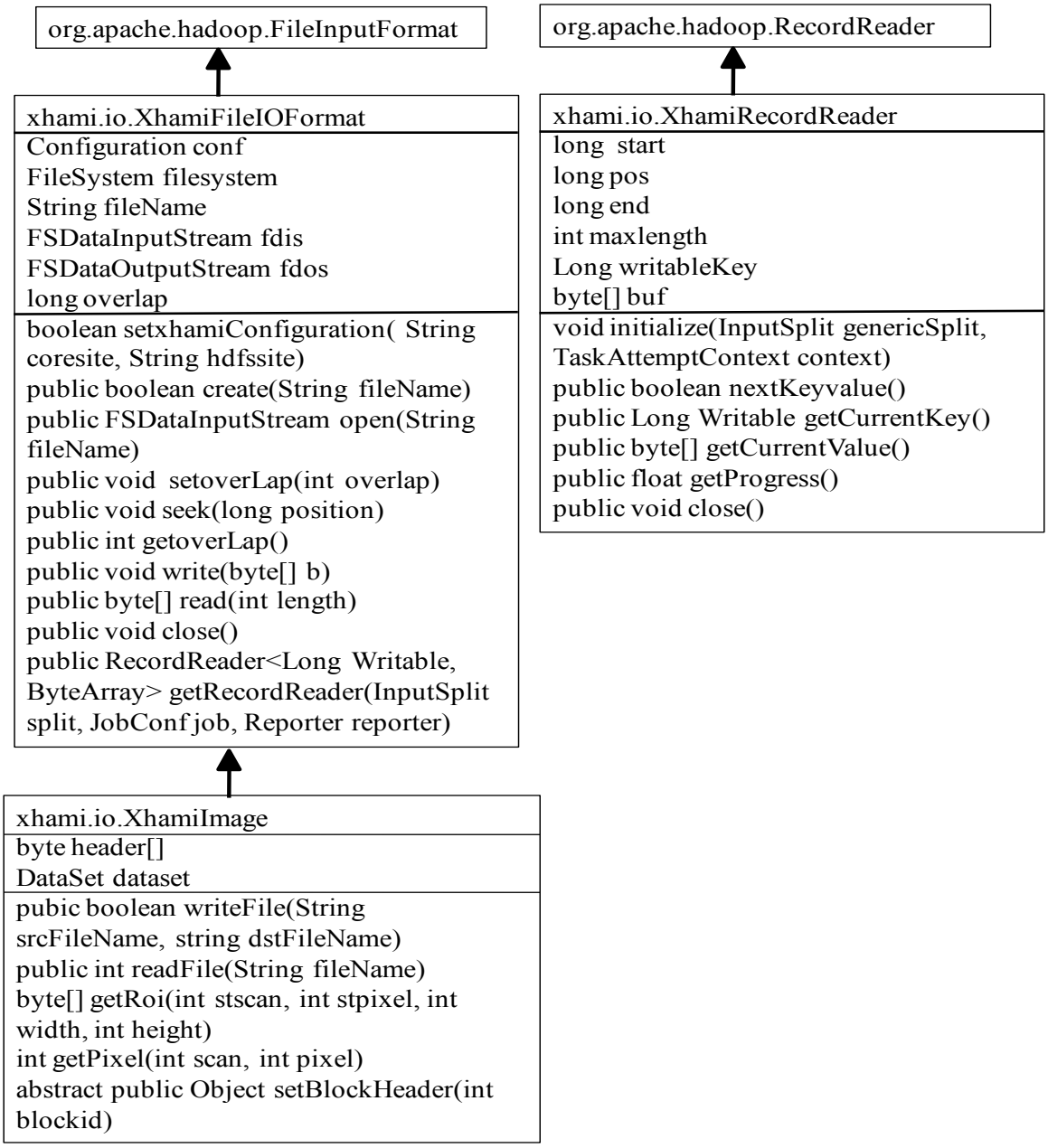


Figure 6.4. XHAMI I/O package

XhamiFileIOFormat is the base class for all domain specific applications offered under the package xhami.io, as depicted in Figure 6.4. XhamiFileIOFormat extends FileInputFormat of standard Hadoop package, and does the implementation of several methods, which hide the low level handling of data for HDFS data organization. The major methods offered are for setting the overlap, getting the overlap, reading, writing, seeking the data without knowing/knowledge of the lower level details of the data.

XhamiFile is used as base class for several application developers for file I/O functionality and implement further for domain specific operations. Xhami.io.XhamiImage is one such image processing domain specific functionality implemented using XhamiFileIOFormat. XhamiImage extends the base class and offers several image processing methods for read/write the image blocks into the similar format of that original file, using Geographical Data Abstraction Layer (GDAL) library [179] and offers several methods such as reading the region of interest, getting the scan lines, pixel, reading the pixel grey vale etc., by hiding the low level details of file I/O.

Several methods in XhamiFileIOFormat and XhamiImage classes are described in Figure 6.4. XhamiFileIOFormat extends FileInputFormat from the standard Hadoop package, and the implementation of several methods, for hiding the low level handling of data for HDFS data organization is handled by this package. The major methods offered under XhamiFileIOFormat class are i) setting the overlap, ii) getting the overlap, iii) reading, writing, and seeking the data without knowing/knowledge of the low level details of the data.

XhamiFileIOFormat is used as base class for several application developers for file I/O functionality and implement further for domain specific operations. Xhami.io.XhamiImage is an abstract class which provides the methods for the implementation of image processing domain specific functionality by extending XhamiFileIOFormat. XhamiImage extends XhamiFileIOFormat class and implements several image processing methods for setting the header/metadata information of the image, block wise metadata information, and for read/write the image blocks into the similar format of that original file, using Geographical Data Abstraction Layer (GDAL) library [179] and offers several methods such as reading the region of interest, getting the scan lines, pixel, reading the pixel grey vale etc., by hiding the low level details of file I/O .Several methods in XhamiFileIOFormat, XhamiImage classes, and XhamiRecordReader classes are described in Table 6-2, Table 6-3 and Table 6-4 respectively. XhamiImage implements Geographic Data Abstraction Layer (GDAL) functionality using the DataSet object for image related operations. XhamiImage class can be extended by the Hadoop based Image processing application developer by setting up their own implementation of setBlockHeader method. XhamiRecordReader reads the buffer data and sends to the Map function for processing.

Table 6-2. Description of Methods in XhamiFileIOFormat class

Method name	Method description	Return value
boolean setxhamiConfiguration(String coresite, String hdfssite)	Sets the HDFS configuration parameters such as coresite and hdfssite. This function in turn uses the Configuration object, and calls addResource methods of its base class, to set establish the connectivity to HDFS site.	If the configuration parameters are correct, then boolean value true is returned. In case wrong supply of arguments, or if the parameter files are not available, or due to invalid credentials, or else HDFS site may be down, false will be returned.
boolean create(String fileName)	Create the file with name filename to write to HDFS. Checks if the file already exists. This function is used before the file is to be written to HDFS.	Returns true if the file is not present in HDFS, or else returns false.
FSDataInputStream open(String filename)	Checks if the file is present in the HDFS.	If the file is present FSDataInputStream having the object value is returned, otherwise FSDataInputStream with value having null is returned.
void setoverLap(int overlap)	Used to set the overlap across the segmented blocks. The supplied overlap value is an integer value corresponding to the overlap size in bytes.	-
void seek(long position)	Moves the file pointer to the location position in the file.	-
int getoverLap()	Reads the overlap value set for the file while writing to HDFS.	Return the overlap value if set for the file or else returns -1.

void write(byte[] b)	Writes b number of bytes to the file.	-
byte[] read(int length)	Reads length number of bytes from the file.	Returns the data in the byte array format.
void close()	Closes the data pointers those are opened for reading / writing the data.	-
RecordReader<Long Writable, ByteArray>getRecordReader(InputSplit split, JobConf job, Reporter reporter	Reads the Xhami compatible record reader in bytes, for MapReduce computing by overriding the RecordReader method of FileInputFormat class. The compatible here means the window size to be read for processing the binary image for processing. This would be supplied as argument value to the Image Processing MapReduce function. If the value is of type fixed, then the entire Block is read during processing.	Returns the Default Hadoop RecordReader Object for processing by the MapReduce job.

Table 6-3.XhamiImage class description

Method name	Method description	Return value
boolean writeFile(String srcFileName, String dstFileName)	Used for writing the contents of the file srcFileName, to the destination dstFileName.	Boolean value true is returned if the writing is successful, or else false is returned.
int readFile(String fileName)	Set the file fileName to read from the HDFS. Returns the total numbers of blocks, that the file is organized in HDFS.	Number of blocks that the file is stored in HDFS. If the file does not exist -1 is returned.
byte[] getRoi(intstscan, intstpixel, int width, int height)	Reads the array of bytes from the file already set, starting at stscan and pixel, with a block of size width and height bytes.	If successful returns byte array read, or else returns NULL object.

intgetPixel(int scan, int pixel)	Reads the pixel value at the location scan and pixel.	Returns pixel(gray) value as integer.
abstract public Object setBlockHeader(int blockid)	Abstract method which would be overwritten by XHAMI application developers for image processing domain applications.	Header information type casted to Object data type.

Package hierarchy of MapReduce for Image processing domain is shown in **Figure 6.5** and methods are described in Table 6-5.

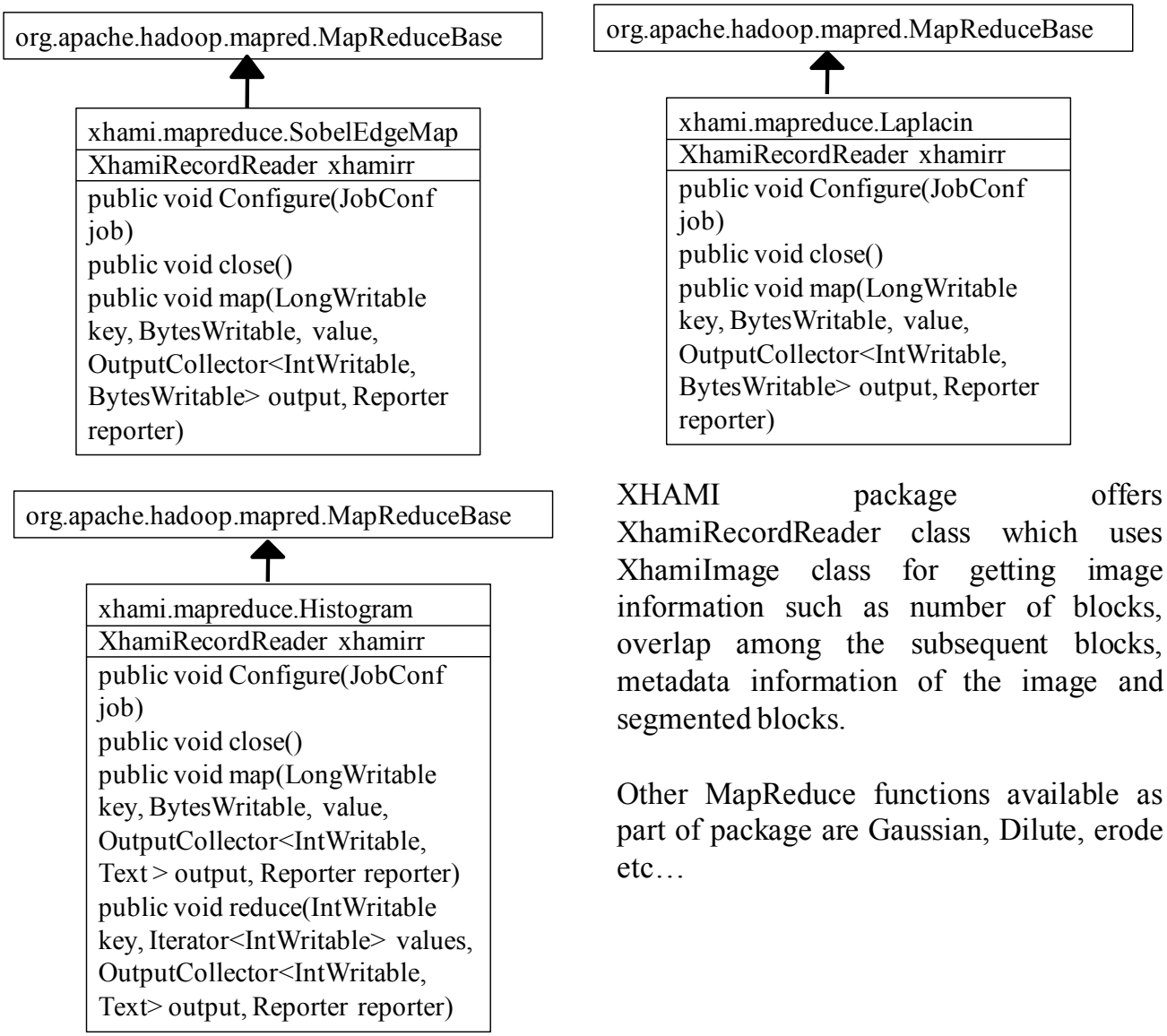
Table 6-4. Methods in XhamiRecordReader

Method name	Method description	Return value
Initialize(InputSplitgenericSplit, TaskAttemptContext context)	Overrides the Initialize method of standard Hadoop Record reader method. Implements the own split method, which reads the content which is compatible with the Xhami File data format, hiding the overlap block size.	-
booleannextKeyvalue()	Overrides the nextKeyvalue method of its base class RecorReader.	Returns true if it can read the next record for the file, or else return false.
Long Writable getCurrentKey()	getCurrentKey method of its base class is overridden.	Return Writable Object of the record recorder method.
float getProgress()	Overriding method, to send the progress of the data read.	Return float false representing the percentage of data read so far from the XhamiRecordRecorder, corresponding to the InputSplit.
byte[] getCurrentValue()	Reads the bytes array to be sent for computing for Map	Return byte array if true, else returns

	function.	NULL object.
void close()	Closes the record reader object.	-

Table 6-5. Description of XHAMI MapReduce classes for Image Processing domain

MapReduce Class	Description	Return value
Sobel	Implementation of Sobel spatial edge detection filter. It has map function implementation only, and the Reduce is not required, as the output of the map itself is directly written, as it does not required any collection of the map inputs for processing further. This implementation hides the several details such as overlapping pixels across the blocks, and the kernel window size to be read for processing. Output is written to the HDFS file system.	Output Images with the detected edges.
Laplacian	Implementation of Laplacian differential edge detection filter. It has map function implementation but not reduce method. Reduce is not required, as the output of the map itself is directly written, as it does not required any collection of the map inputs for processing further. This implementation hides the several details such as overlapping pixels across the blocks, and the kernel window size to be read for processing. Output is written to the HDFS file system.	Output Images with the detected edges.
Histogram	Implements both Map and Reduce functions. Map collects the count of the pixel (gray) value, and reduce does the aggregation of the collected numbers from the map functions. While processing it does not consider the overlapping pixels across the blocks.	Histogram of the image.



XHAMI package offers XhamiRecordReader class which uses XhamiImage class for getting image information such as number of blocks, overlap among the subsequent blocks, metadata information of the image and segmented blocks.

Other MapReduce functions available as part of package are Gaussian, Dilute, erode etc…

Figure 6.5. XHAMI MapReduce Package for Image processing domain

Below we describe sample implementation of XHAMI MapReduce functions for image processing domain.

6.3.4 XHAMI – MapReduce functions

In this section we describe the extensions for Map and Reduce functions for image processing applications. The implementation of Map and Reduce function depends on the image processing operation to be performed on the image. For example, in the case of edge detection, there is no need of reducer function implementation, as the resultant output of the map function is straightway can be written to the storage. Each map function reads the block numbers and

metadata of the corresponding blocks. The sample job configuration for MapReduce operation in java language is shown Table 6-6

Read operations can be implemented in two ways in HDFS, one way is to implement own split function, ensuring the split does not happen across the boundaries, and other one is to use FIXED LENGTH RECORD of FixedLengthInputFormat class. The package offers the implementations for both FIXED LENGTH RECORD and custom formatter XhamiRecordRecorder.

Table 6-6.Sample job configuration for MapReduce

1.JobConfconf = new JobConf(XhamImageMROperations.class); //setting MapReduce job configuration
2.conf.setWorkingDirectory(new Path("hdfs://namenode/user/hduser"));//setting configuration working directory.
3.conf.addResource(newPath("/home/hduser/hadoop-2.7.0/etc/hadoop/core-site.xml"));// adding the core site file.
4.conf.addResource(newPath("/home/hduser/hadoop-2.7.0/etc/hadoop/hdfs-site.xml"));// adding the hdfs site information file.
5.conf.setInt(XhamiFileIOFormat.RecordReader);//setting XhamiFileIOFormat recorder class
6. conf.setInputFormat(XhamiFileIOFormat.class);//setting XhamiFileIOFormat class

(1) MapReduce Sobel edge detection sample implementation

Edges characterize boundaries in images are areas with strong intensity contrasts- a jump in intensity from one pixel to the next. There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories, gradient, and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for zero crossings in the second derivative of the image to find the edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. In the map function, for edge detection, the combiner and reduce functions are not performed, as there is no need of

aggregation of the individual map functions. A sample map function implementation of XHAMI for Sobel edge detection using 3X3 image gradient operator is discussed in Table 6-7.

Table 6-7. Sample map function of Sobel gradient edge operator

```
int[][]  GX={
   {-1,0,1},{-2,0,2},{-1,0,1}
};
int[][]  GY={
   {1,2,1},{0,0,1},{-1,-2,-11}
};
public void map(LongWritable key, BytesWritable value,
OutputCollector<IntWritable, BytesWritable> output,
   Reporter reporter) throws IOException {//key contains the value of the block number to be processed and value is the buffer to be processed
    byte[] data = value.getBytes(); //data array to process
   //apply the kernel on the variable value using GX and GY operator on data array. The resultant output written to a file from output data.
byte[]  output = new byte[data.length]; //output array to be written.
   //output buffer is written to individual files, with the sequence number as name. Once all the map functions are completed, they would be merged to generate a single large output.
}
```

The differences between conventional Hadoop implementation and the XHAMI implementations are as follows- in the former, the data is organized as segmented as blocks, and there is no overlap of the line pixels across the blocks. Hence, it would be difficult to process the edge pixels of the blocks, and to process the edge pixels, one should get the two blocks, and compute the overlap pixels before it is sent to the map function for processing. Also, it would be difficult

to ensure that the split does not happen within the pixel while reading. But, XHAMI hides all such low level details of data organization, lines or pixels overlap, no split within the pixels, number of blocks the image is organized as blocks, header information of the blocks etc.

(2) MapReduce Histogram sample implementation

Histogram operation computes frequency count of the pixel in the image. The histogram is computed as follows, first, the block and length of the block is read, and each blockis mapped to one map function. Sample code for histogram of map and reduce function is described in Table 6-8 and Table 6-9 respectively. The difference between the conventional implementation and the XHAMI MapReduce Histogram implementation are – in the former, it is necessary to ensure that the split does not happen within the pixels. The later overcomes this problem by using XHAMI Image implementation for data organization, and ensures that overlap lines (pixels) are vomited during processing by the Map function.

Table 6-8.Histogram map function

```
public void map(LongWritable key, BytesWritable value,
OutputCollector<IntWritable, Text> output,
Reporter reporter) throws IOException {
   byte[] data = value.getBytes();
   byte pixelValue=0; //skip overlap scan lines
   for(inti=0;i<data.length-(overLapScanLines*scanLineLength);i++){
      pixelValue= data[i];
      output.collect(new IntWritable(pixelValue), new Text(""+1));
   }
}
```

Table 6-9.Histogram reduce function

```
public void reduce(IntWritable key, Iterator<IntWritable> values,
OutputCollector<IntWritable, Text> output,
Reporter reporter) throws IOException {
    int sum=0;
    while (values.hasNext()){
        sum+=Integer.parseInt(""+values.next());
    }
    byte b = (byte)key.get();
    int v = (int)b;
    key  = new IntWritable(new Integer(v));
    output.collect(key,new Text(""+sum));
}
```

6.3.5 Writing domain specific applications by extending XHAMI package

XhamiFileIOFormat class is the base class, which hides the low level details of data organization and processing for the several applications of binary data handling. This class offers several methods for reading, writing, seeking to the particular block of the image, getting the overlap information among the subsequent blocks etc. XhamiImage class is an extended class of XhamiFileIOFormat, which offers several methods for handling the data for several applications in image processing domain. XhamiImage could be used for development of HDFS based data organization readily, or else, one can extend the class XhamiFileIOFormat for handling similar kind of image processing domain applications of their own interest. Below, we describe the procedure for writing and reading the images in HDFS format using by extending XhamiImage for writing XHAMI based applications.

- Extend XhamiImage class and implement setBlockHeader method.

- Define header class and the necessary data types for implementation of the Image processing application.
- Implement setBlockHeader method using the FSDataInputStream available in XhamiImage class as member variable.
- Set the overlap required among the adjacent blocks using setoverLap method.
- Assign the source file and destination files, using the writeFile method. Internally, this method computes the total file size, by computing the total numbers of blocks that the image gets divided into and writes the corresponding block header.
- The contents of the file using getBlockCount and getBlockData methods.

Table 6-10 shows a sample code describing how to extend XhamiImage class for writing the images along with the image specific header while storing the image into HDFS.

Table 6-10. XhamiImage extension for writing block header

```
class ImageHeader implements Serializable{
    int blockid, startscan, endscan, overlapscan, scanlength, blocklength, bytesperpixel;
}
public class XhamiImageIOOperations extends XhamiImage{
  //other implementation specific to the application
@override
public object setBlockHeader(int blockid){
 ImageHeader ih = new ImageHeader();
 //set the details and write to FSDataOutputStream
}
```

6.3.6 **Performance evaluation**

In this section we present the experiments conducted for large size images of remote sensing data having different dimensions (scans, pixels) and sizes varying approximately from 288 Megabytes to 9.1 Gigabytes. First we discuss the read and write performance, storage overheads of the

conventional system, both with and without overlapping scan lines, followed by performance comparison of histogram and sobel edge detection filter operations. We conduct the experiments both on conventional APIs and XHAMI libraries, and discuss how XHAMI simplifies the programming complexity and also increases the performance when applied to a large scale image over Hadoop framework.

Table 6-11.System configuration

Type	Processor type	hostname	RAM (GB)	Disk (GB)
Name node	Intel Xeon 64 bit , 4 vCpus, 2.2 GHz	namenode	4	100
Job tracker	-do-	jobtracker	2	80
Data node 1	-do-	datanode1	2	140
Data node 2	Intel Xeon 64 bit , 4 vCpus, 2.2 GHz	datanode2	2	140
Data node 3	Intel Xeon 64 bit , 2 vCpus, 2.2 GHz	datanode3	2	140
Data node 4	-do-	datanode4	2	100

For the experimental study, we have used virtualized environment running on Xen hypervisor with a pool of four servers of Intel Xeon 64 bit architecture, with 2TB internal storage. Hadoop version 2.7 is configured in the fully distributed mode, running on the server pool of four virtual machines with 64 bit 'Cent OS', the nodes configuration is shown in Table 6-11. A sample screen from HDFS with four node configuration is depicted in Figure 6.6.

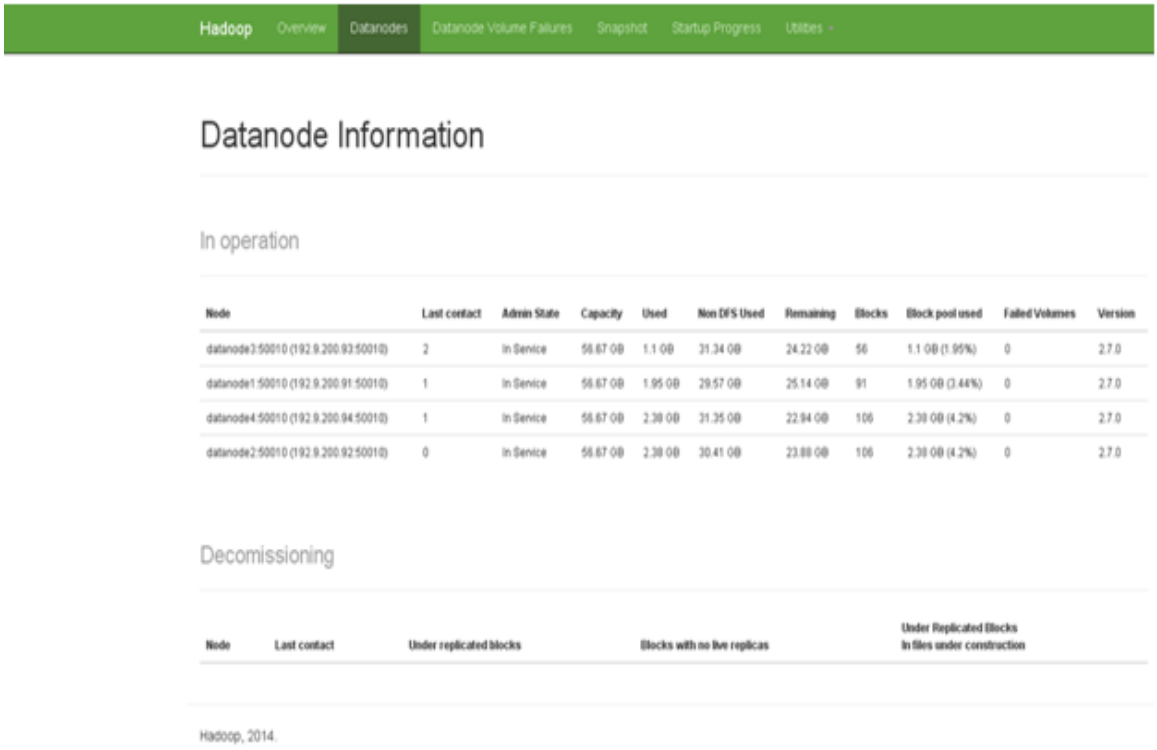


Figure 6.6. HDFS data nodes configuration

6.3.6. 1 Comparisons of XHAMI Vs Hadoop and Similar systems

Sample data sets used for experiments are from the Indian Remote Sensing (IRS) satellite series i.e. CARTOSAT-1, and CARTOSAT-2Aare shown in Table 6-12, the columns in the table, *Image size* represents the original image size in bytes in regular file system, and the *resulted image size* indicates the size in bytes in HDFS with overlapping of 5 scan lines using the block length computation algorithm presented in section 6.3.2. A sample image with overlap of 5 scan lines shown in red color is depicted in Figure 6.7. The results shown in Table 6-12 , illustrate that a maximum of 0.25% increase in the image size, which is negligible.

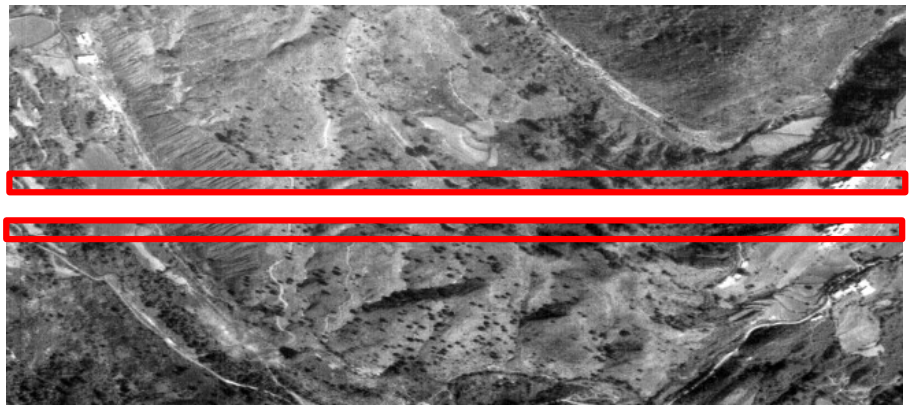


Figure 6.7.Image blocks with overlap highlighted in rectangular box

Table 6-12. Sample data sets used and the resultant image size

S.No	Image size (in bytes)	Scan line length	Total Scan lines	Resulted Image size (in bytes)
1	288000000	12000	12000	288480000
2	470400000	12000	19600	471240000
3	839976000	12000	34999	841416000
4	1324661556	17103	38726	1327911126
5	3355344000	12000	139806	3361224000
6	9194543112	6026	762906	9202738472

Here, we discuss the performance comparisons of HDFS data organization both in XHAMI and default Hadoop. Followed by, performance comparisons of MapReduce based image processing applications in XHAMI, Hadoop- HDFS and MapReduce, and HIPI. The data sets used for the experiments shown in Table 6-12. Here, Hadoop is, as directly in its native form cannot be used for MapReduce Image processing, and such customized Hadoop here called as customized Hadoop MR. In this customized Hadoop MR performs the Map Reduce is computed on the data organized as non overlapping blocks. We present the results for image processing operations such as histogram and Sobel filter, followed by read and write overheads while the data blocks are written to HDFS. The I/O overhead comparisons are presented for both Hadoop, and XHAMI. The advantage of HIPI over customized Hadoop is the use of Java Image Processing Library. HIPI comes with processing the image formats like jpg, and png files, hence do not require the additional coding. As HIPI uses HIB formats to create a single bundle for smaller

image files, but, here, the image sizes are large, hence, we have used only one single image is supplied to create the HIB format.

A sample single large binary image with approximately around one GB in volume, uploaded to the HDFS file system with the uni-directional scheme is shown in Figure 6.8.

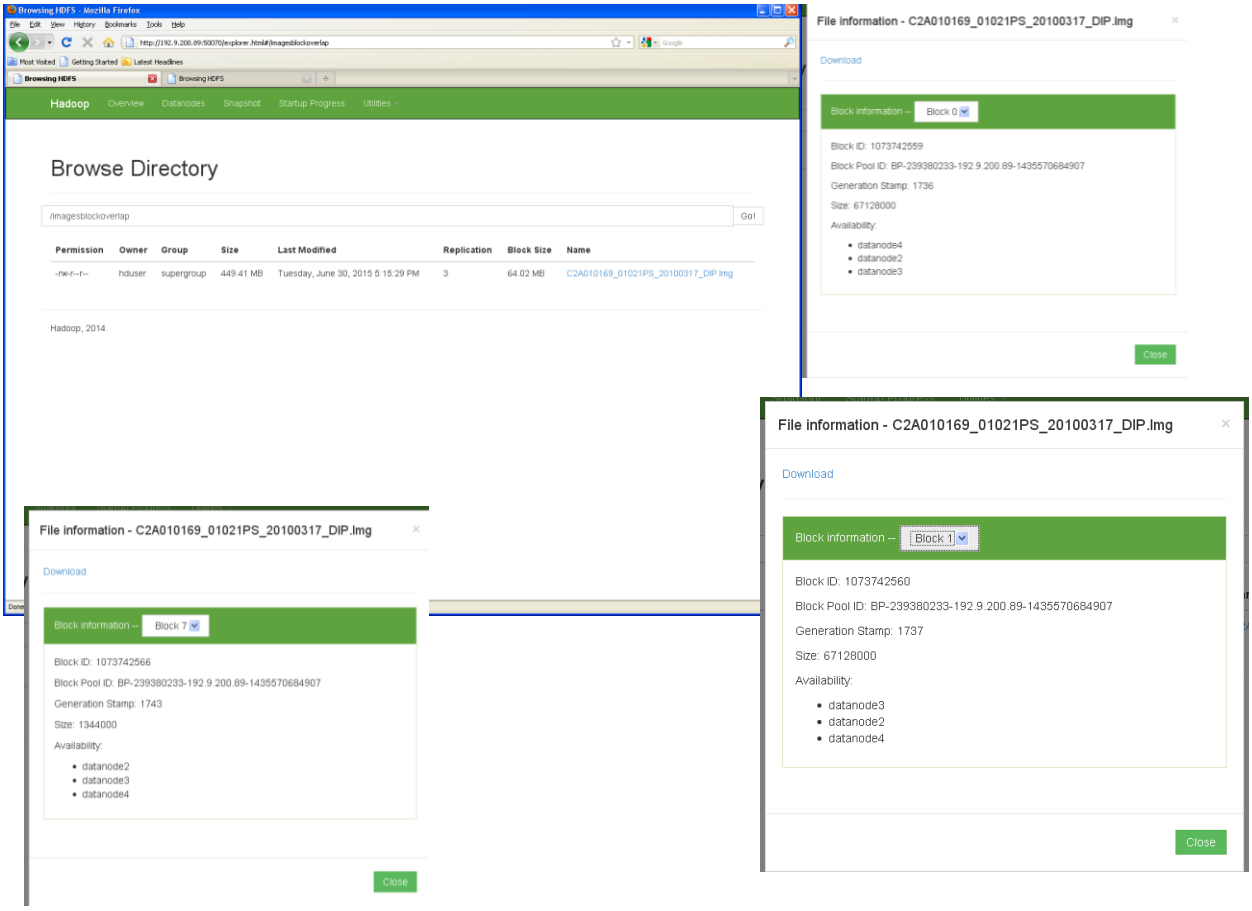


Figure 6.8. HDFS browse with a single large image and its related block information

(a) Histogram operation

Histogram operation counts the frequency of the pixel intensity in the entire image, which is similar to counting the words in the file. The performance results of histogram operation for customized Hadoop MapReduce (MR), HIPI and XHAMI are shown in Figure 6.9 In the case of, customized Hadoop MR, the HDFS data blocks are organized as non overlapped blocks. For HIPI, the data blocks are retrieved using HIB files, and XHAMI uses data blocks that are with overlap partitioned HDFS blocks for processing. The results show that, performance of

Customized Hadoop, HIPI, and XHAMI are similar, and XHAMI has little overhead over Customized Hadoop, which is less than 0.8%, this is due to overheads accounted for processing the overlapped scan lines.

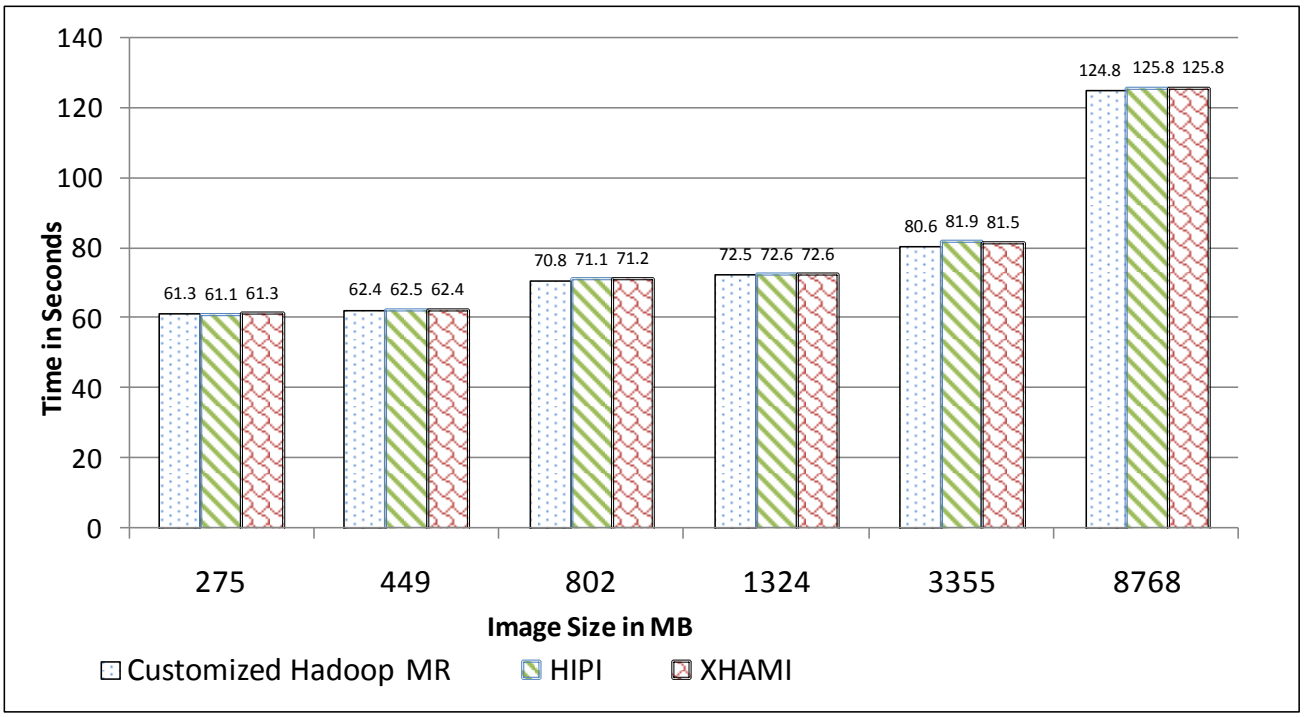


Figure 6.9. Histogram performance

(b) Fixed mask convolution operation

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the position where the kernel fits entirely within the boundaries of image. Convolution methods are the most common operations used in image processing which uses the mask operator i.e. kernel for performing windowing operations on the images. Sobel operator is one of the commonly used methods for detecting edges in the image using convolution methods. In case if the image is organized as physical partitioned distributed blocks, then, convolution operations cannot process the edge pixels of such blocks, due to the non availability of the adjacent blocks data on the same node. In conventional Hadoop based HDFS and MapReduce processing, the data is organized as

physical partitioned blocks, hence, the edge pixels cannot be processed directly, and demands the additional I/O overheads for processing the edge pixels of each block.

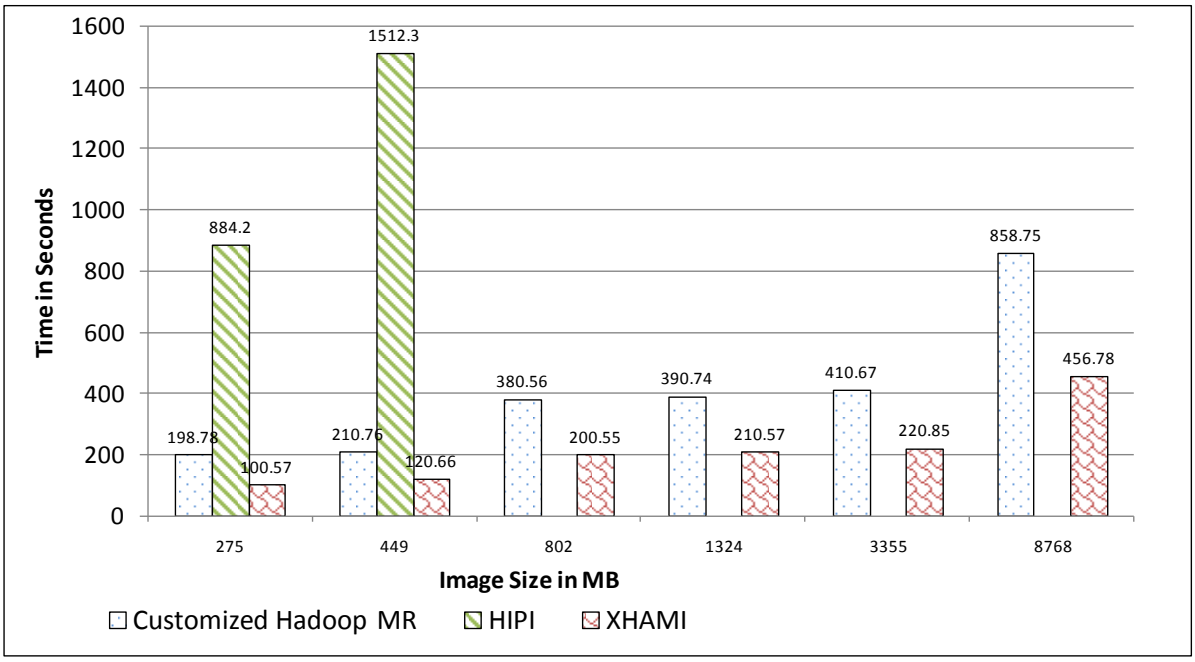


Figure 6.10.Sobel filter performance

Here, we present the performance of the Sobel edge detection implementation in XHAMI, and compare it with customized Hadoop MapReduce (MR) and HIP. In customized Hadoop MR; data is physically partitioned as non overlapping data blocks, and for HIPI data is organized as single large block stored in HIB format for a single file. For MapReduce computing in customized Hadoop MR, an additional functionality is included in Map function for retrieving the adjacent block information corresponding to the block to be processed. In the case of HIPI such logic additional is not possible, due to the non availability of HIPI APIs to know corresponding adjacent block information. The results are depicted in Figure 6.10, compare the performance of XHAMI with customized Hadoop MR and HIPI. The results indicate that, the performance of XHAMI is much better, and which is nearly half of the time taken by Customized Hadoop MR, and it is extremely better over HIPI. Customized Hadoop MR is implemented with standard Java Image Processing Library, with few customized features over default Hadoop, like retrieving the adjacent blocks information in Map functions for processing the edge pixels of the blocks. This customization requires additional overheads, increasing both

the programming and computational complexities. The additional overheads are mainly due to the transfer of whole data block which is located in different data nodes, than the one where Map function to be processed.

HIPI has in built Java Processing APIs for processing the jpg and png image formats. For HIPI, the data is organized a single large block equivalent to the size of the image, as there is no functionality readily available for retrieving the adjacent blocks information. Due to this reason, the experiments for the larger image size data sets starting from Serial numbers 3 and above mentioned in the Table 6-13, could not be conducted. XHAMI overcomes the limitations of both Hadoop, and HIPI, by extending the Hadoop HDFS and MapReduce functionalities with the overlapped data partitioned approach, and MR processing using high level APIs with the integrated open source GDAL package for handling several types of image formats. XHAMI not only simplifies the programming complexity, but also allows the development of image processing applications quickly over Hadoop framework using HDFS and MapReduce.

Table 6-13. Read/write performance overheads

S.No	Image size (MB)	Write (Sec)		Read (Sec)	
		Default Hadoop	XHAMI	Default Hadoop	XHAMI
1	275	5.865	5.958	10.86	10.92
2	449	14.301	14.365	19.32	19.45
3	802	30.417	30.502	40.2	40.28
4	1324	44.406	77.153	50.28	50.95
5	3355	81.353	88.867	90.3	90.6
6	6768	520.172	693.268	550.14	551.6

6.3.6. 2 Read / Write overheads

Performance of read and write function in default Hadoop and XHAMI with overlap of 5 scan lines in horizontal partition direction is shown in Figure 6.12 and Figure 6.11 respectively. The results shown in Figure 6.12 represents a negligible read overhead, as the scan lines to be skipped are very few, and also the position of those lines to skip are known prior. Write performance overheads are shown in Figure 6.11, indicate that, XHAMI has minimal overheads compared with default Hadoop, and it is observed that data sets in serial nos. 1, 2, 3 and 5 is less than 5%, and for other data sets it is 33%.

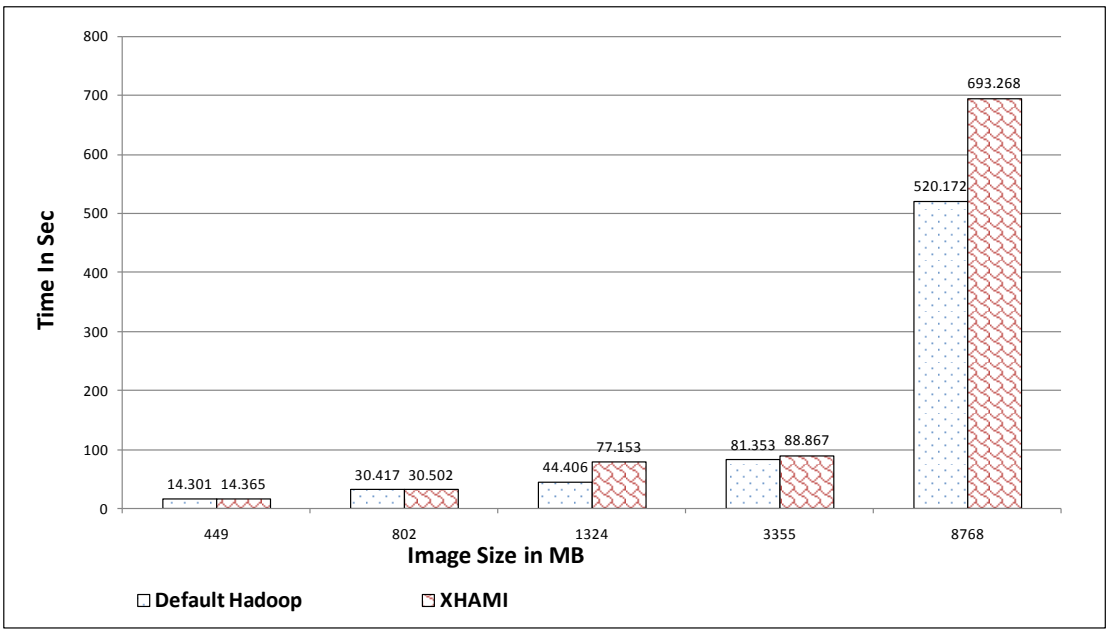


Figure 6.11. Image write performance

Writing overheads for data sets 4 and 6 is more, this is due to more numbers of pixels in scan direction i.e. large scan line lengths, which has resulted in additional storage disks space during HDFS block construction, which is in turn has resulted in more read and writes overheads. For these data sets, even the vertical direction partition also has similar write overheads, as the number of pixels in pixel direction is also more as compared to the scan line direction. Hence, the method for choosing the partitioning approach is use to compute the number of blocks during data partition either in horizontal or vertical direction, and subsequently compute the storage overhead for each blocks and accumulate the overheads for all the blocks. Based on the storage overheads, data partitioning approach can be chosen the one which has resulted in minimal write overheads. However, this is to be noted that, for image processing applications, in general the images are written to disk once and read operation is performed several times, hence, the as writing overhead is one time activity, which is minimal, while compared with the overall performance achieved while processing. Figure 6.12 depicts the result of read performance for the data sets, is less than 0.2% which is very negligible.

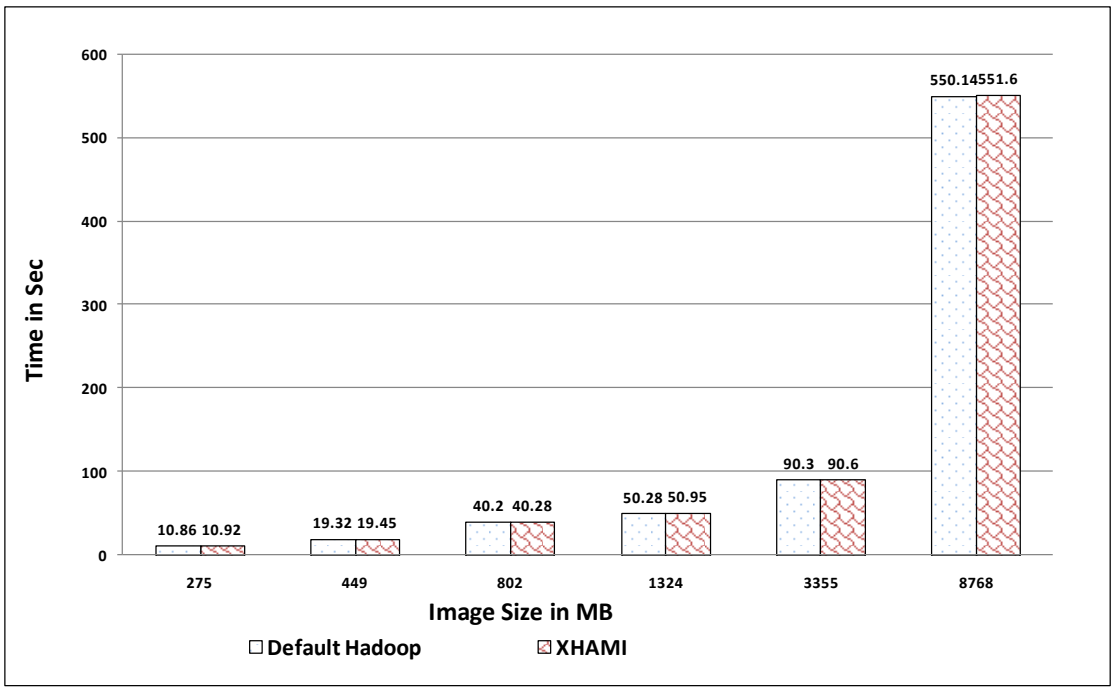


Figure 6.12. Image read performance

6.4 Discussion and Summary

Image processing applications deal with processing of pixels in parallel, for which Hadoop and MapReduce can be effectively used to obtain higher throughputs. However many of the algorithms in Image Processing and other scientific computing, require use of neighborhood data, for which the existing methods of data organization and processing are not suitable. We presented an extended HDFS and MapReduce interface, called XHAMI, for image processing applications. XHAMI offers extended library of HDFS and MapReduce to process the single large scale images with high level of abstraction over writing and reading the images. APIs are offered for all the basic forms Read/Write and Query of images. Several experiments are conducted on sample of six data sets with a single large size image varying from approximately 288 MB to 9.1 GB.

Several experiments are conducted for reading and writing the images with and without overlap using XHAMI. The experimental results are compared with the conventional Hadoop system, the experimental results show that, though the proposed methodology incurs marginal read and write

overheads, due to overlapping of data, the performance has scaled linearly and also programming complexity is reduced significantly.

The system is implemented with both the fixed length record and the customized split function which hides the low level details of handling and processing, spawning more map functions for processing. However, challenges involved in organizing the sequence of executed map functions for aggregations need to be addressed. We plan to implement the bi-directional split also in the proposed system, which would be the requirement for large scale canvas images. The proposed MapReduce APIs could be extended for many more Image processing and Computer vision modules. It is also proposed to extend the same to multiple image formats in the native format itself. Currently, image files are transferred one at a time from the local storage to Hadoop cluster, hence XHAMI integration with Data Aware Scheduler, would lead to a system, which would effectively migrate the data files required for processing to the computing nodes.

Chapter 7

Conclusions and Future Directions

7.1 Summary

Big Data computing is an emerging platform for data analytics to address large scale multidimensional data for knowledge discovery and decision making. In this thesis, we have studied, characterized and categorized several aspects of Big Data computing systems. Big Data technology is evolving and changing the present traditional data bases with effective data organization, large computing and data workloads processing with new innovative analytics tools bundled with statistical and machine learning techniques. With the maturity of Cloud computing technologies, Big Data technologies are accelerating in several areas of business, science and engineering to solve data intensive problems. We have enumerated several case studies of Big Data technologies in the areas of health care studies, business intelligence, social networking, and scientific explorations. Further, we focus on illustrating how Big Data databases differ from traditional data base and discuss BASE properties supported by them.

To understand Big Data paradigm, we presented taxonomy of Big Data computing along with discussion on characteristics, technologies, tools, security mechanisms, data organization, scheduling approaches, etc. along with relevant paradigms and technologies. Later we presented under pinning technologies for the evolution of Big Data and discussed how cloud computing technologies would be utilized for infrastructure services delivery for the analytics development. Later, we discussed an emerging Big Data computing platforms over Clouds, Big Data Clouds, an integrated technology from Big Data and Cloud computing, delivering Big Data computing as a service over large scale clouds. The thesis also discussed types of Big Data clouds and illustrated Big Data access networks, an emerging data platform services for Big Data analytics.

Further on, we presented a layered architecture, components under each of the layers followed by technologies to be addressed under each of the layers. We then compare some of the existing

systems in each of the areas and categorize them based on the tools and services rendered to the users. In doing so, we have gained an insight into the architecture, strategies and practices that are currently followed in Big Data computing. Also, through our characterization and detailed study, we are able to discover some of the short comings and identify gaps in the current architectures and systems. These represent some of the directions that could be followed in future. Thus, this chapter lays down a comprehensive classification framework that not only serves as a tool to understanding this emerging area but also presents a reference to which future efforts can be mapped. Based on the layered architectural framework, we have identified two key elements which are to critical elements to be addressed such as scheduling the data intensive jobs using Data Aware scheduler, and data organization and processing using new programming approaches. These were addressed by Data Aware Scheduling with family grouping using genetic approach, and XHAMI respectively.

The proposed family/group scheduling model addresses the data intensive problems to minimize the turnaround time of the jobs where the computing and data resources are decoupled. The jobs with common data are grouped together, based on the family graph and connected components to which a parallel data approach is applied. Steady state GA is applied to discover the optimal schedule. The results are illustrated for the both family and non-family schedules from a single site and multiple replicated sites. The results indicate that, data migration from replicated sites show performance improvement over a single site. The experiments also show that family schedule performs better over the non-family schedule, whenever the grouped jobs do not exceed the available node capacity.

We presented an extended HDFS and MapReduce interface, called XHAMI, for image processing applications for both data organization and large scale data processing on a cluster of machines. XHAMI offers extended library of HDFS and MapReduce to process the single large scale images with high level of abstraction over writing and reading the images. APIs are offered for all the basic forms Read/Write and Query of images. Several experiments are conducted on sample of six data sets with a single large size image with varying data volumes. The experimental results are compared with the conventional Hadoop system, the experimental results show that, though the proposed methodology incurs marginal read and write overheads,

due to overlapping of data, the performance has scaled linearly and also programming complexity is reduced significantly.

7.2 Conclusions

Big Data technologies and Clouds are needed to be addressed so as to address large scale problems on the scalable infrastructures effectively. At the same time, the frameworks addressing these two technologies need to provide scalable, inter operable, easy tools and techniques for management and solving the large scale problems. To address these requirements, a layered framework is developed, which addresses the integration of Big Data computing over Clouds infrastructures. In the framework, we have identified two key elements such as data aware scheduling using data pull models, and processing using XHAMI.

Data aware scheduling discussed the scheduling model in Big Data over Clouds infrastructure, where the data is replicated over several storage repositories for further access and exploitation. The scheduling model presented the problem solving approach using family grouping techniques using connected components, and optimization using genetic approach. The thesis discussed bandwidth aware model by pulling the large chunks of data from storage repositories to the Computing nodes. The algorithm compared the results, with the non grouping data mechanisms with the family grouping based on the data dependencies. The results of genetic approach are compared with the other approaches like Min compute first, Min data consolidation first node, and simulated annealing. The results indicated that Min compute first has resulted in larger makespan while compared with Min data consolidation first node and Simulated Annealing. Min data consolidation and Simulated Annealing techniques have almost the similar makespan value with performance better than Min compute first technique. However, the proposed Genetic approach and family grouping technique has resulted in minimal makespan, this is due to bandwidth aware data model, and natural evolution procedures of GA and fitness function used to obtain the near optimal solution.

To address the large scale data organization, and processing we have developed a system called XHAMI, which addresses the data organization and processing over Hadoop Distributed File System (HDFS) and MapReduce models. We have demonstrated the importance of the data

overlapping in several scientific applications such as image processing, and extended the APIs of HDFS and MapReduce. The thesis developed the data organization models in uni-directional and bi-directional split methods, and presented the high level APIs for MapReduce. Sample image processing experiments are conducted such as image histogram, and spatial filters such as Sobel edge detection were performed, and the results are compared with the customized Hadoop MapReduce (MR), and Hadoop Image Processing Interface (HIPI). XHAMI performance is much better over customized Hadoop MR, and HIPI for several image processing spatial filter applications where data dependency with the adjacent blocks is necessary.

7.3 Future Directions

The thesis formulated a comprehensive framework for Big Data computing over Cloud computing platforms, and addressed the data aware scheduling approach considering both computation and data awareness into the account, followed by a system XHAMI for data organization and processing for Big Data scientific applications over large scale Cloud platforms. This work laid a foundation for Big Data computing in Clouds, and it opens up several avenues for future work in Big Data based Cloud computing, scheduling and Programming models.

7.3.1 Rough set approach for family construction

The proposed Data aware scheduling does the grouping of the jobs for the data files which are common among the jobs. However, the percentage of common data among the jobs is not considered among the jobs. This sometimes may result in a large super job, which perhaps may need to schedule a node, which should have larger enough computing resources for processing such jobs. Hence, in order to avoid such large set of the jobs, a rough set rules can be applied, which would group the job, based on the data commonality and percentage of overlap among the data sets. This approach also would eliminate connected components graph traversal which would a time consuming process for family job construction.

7.3.2 Supporting quality and budget aware scheduling

Data aware scheduling addresses the job scheduling for minimizing the makespan for the jobs using both data consolidation and computing times. However, other QoS parameters such as

deadline, budget and on demand varying bandwidth network speeds are not considered. Hence, the proposed model can be extended for several QoS parameters. Presently, the model executes the jobs after the data is consolidated for the family jobs, however, the studies can be conducted for the execution soon after the data for the job is made available.

7.3.3 Space shared scheduling

At present, the scheduling model supports time shared scheduling approach. This would share the resources within the node while running. However, a dedicated processor or resource scheduling approach such as Space shared scheduling mechanisms could be developed as an extension to the model. Also, other mechanisms such as buddy systems, Distributed Hierarchical Control (DHC), Ouster out matrix, and bin packing algorithms can be tested with the models for minimizing the job make spans, depending on the job application requirements.

7.3.4 Extended MapReduce for other application domains

XHAMI offers package for image processing domains and other domains where the overlapped data among the blocks is the primary requirements. However, the system can be used for other domain too, with a small pre processing overheads. The offered package can be extended for other domains like business intelligence, social networking, and financial domains, and high level data organization and MapReduce APIs can be developed.

7.3.5 Data organization extensions for XHAMI

XHAMI discusses the data organization model for both uni-directional and bi-directional models, with the custom split methods for data organization and processing. Experiments are conducted on the unidirectional data organization, and the model needs to be extensively studied for bi-directional data organization, with the overheads of pre processing and MapReduce computing. Presently, XHAMI discusses the data organization of uncompressed data, the model and package needs to be extended for compressed data as well, with its custom splitter functions.

7.3.6 XHAMI and Data Aware Scheduler integration

Currently, image files are transferred one at a time from the local storage to Hadoop cluster to be performed by XHAMI. The proposed Data aware scheduling can be integrated with XHAMI, for effective bulk data transfers and identifying the better computing resources for processing, later XHAMI could do the scheduling on those identified nodes, for achieving higher throughputs.

References

[1] A. Janaki, T. Kubach, M. Loffer, and U. Schmid, *Data Driven Management: Bringing more science into management*, White Paper, McKinsey Technology Initiative Perspective, 2008, http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation.

[2] Challenges and Opportunities with Big Data, http://www.cra.org/ccc/files/docs/init/bigdatawhitepaper.pdf.

[3] Advancing Discovery in Science and Engineering, The Role of Basic Computing Research, http://www.cra.org/ccc/files/docs/Natl_Priorities/web_data_spring.pdf.

[4] IDC IVIEW, The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, www.emc.com/leadership/digital-universe/index.htm.

[5] C.L.P. Chen, and C.Y. Zhang, Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, Inform.Sci (2014), http://dx.doi.org/10.1016/j.ins.2014.01.015

[6] V.V. Mayer, and K. Cukier, Big Data: A Revolution that will transform How we Live, Work and Think, John Murray Press, 2013.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications 23 (3) (2000) 187-200.

[8] J. Dean, and S. Ghemat, MapReduce: Simplified Data Processing on Large Cluster, Communications of the ACM, 51(1) (2008): 107-113.

[9] W. Hoschek, J. JaenMatrinez, A. Samar, H. Stockinger, and K. Stockinger, Data Management in International Data Grid Project, Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing, 2000, Springer Verlag Press.

[10] H. Nakada, M. Sato, and S. Sekiguchi, Design and Implementation of Ninf, Towards a Global Computing Infrastructure, 15(5-6):649–658, 1999.

[11] T. H. Davenport, At the Big Data Crossroads: turning towards a smarter travel experience, 2013 Amadeus IT Group, http://www.amadeus.com/blog/26/06/big-data (20.03.2014).

[12] Ruchi Verma and Sathyan Ramakrishna Mani, Using Analytics for insurance fraud detection, FINsights, Infosys, India, http://www.infosys.com/FINsights/Documents/pdf/issue10/insurance-fraud-detection.pdf (30.4.14)

[13] W. H. Inmon, Building the Data Warehouse, 3rd edition, John Wiley & Sons, 2002.

[14] Eric A. Brewer, Towards robust distributed systems, keynote speech in 19th ACM Symposium on Principles of Distributed Computing (PODC 2000), Portland, Oregon, July 2000.

[15] J. Gray, The Transaction Concept: Virtues and Limitations, Proceedings of the 7th International Conference on Very Large Databases (VLDB 1981), Cannes, France, Sept'1981.

[16] D.Pritchett, BASE: An ACID Alternative, article on Object Relational Mapping, ACM queue, Vol 6 No.3, May/June 2008.

[17] M.Cooper, and P. Mell, Tackling Big Data, NIST Information Technology Laboratory, Computer Security Division.

[18] F. Magoules, J. Pan, F. Tenq (Eds), Cloud computing: Data Intensive Computing and Scheduling, Chapman & Hall / CRC Numerical Analysis and Scientific Computing Series, September 2012.

[19] J. Broberg, S. Venugopal, and R. Buyya, Market oriented grids and utility computing: The state-of-the art and future directions, Journal of Grid Computing 6 (3) (2008) 255-276.

[20] R. Buyya, C. Vecchiola, and T. Selvi (Eds.), Mastering in Cloud computing – Foundations and Applications Programming, Morgan Kaufman Publishers (2013).

[21] CloudStor- Data Intensive Computing in the Cloud, http://acid.sdsc.edu/projects/cloud.

[22] Open Topography Facility, http://acid.sdsc.edu/projects/opentopo.

[23] LHCGrid, http://www.cern.ch/lcg.

[24] Biogrid Project, http://www.ncbiogrid.org.

[25] International Virtual Observatory Alliance, http:/www.ivoa.net.

[26] S. Venugopal, R. Buyya, K. R. Rao, A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing, ACM Computing Surveys, 38 (1) (2006) 1-53.

[27] K. Ranganathan, and I. Foster, Decoupling Computation and Data scheduling in Distributed Data-Intensive Applications, Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing (HPDC 2012), Edinburgh, Scotland, July 2002.

[28] C. Hu, C. Ouyang, and Z. Liu, MatDC: A Data Cloud Model For management and Sharing of Material Science Data, Proceedings of the 8th International Conference on Grid and Cooperative Computing (GCC2009), Lanzhou, Gansu, August 2009.

[29] D. Tracey, and C. Sreena, A Holistic Architecture for the Internet of Things, Sensing Services and Big Data, Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013), Delft, Netherlands, May 2013.

[30] Apache HBase, http://hbase.apache.org.

[31] Big Data on AWS, http://aws.amazon.com/big-data.

[32] Big Data Hadoop, www.rackspace.com/big-data.

[33] A. M. Middleton, HPCC Systems: Introduction to HPCC (High Performance Computing Cluster), White paper, LexisNexis, May 24,2011, http://cdn.hpccsystems.com/whitepapers/wp_introduction_HPCC.pdf.

[34] ECL Programmers Guide, HPCC Systems, Boca Raton Documentation Team, 2013 Version 3.10.4.1, http://cdn.hpccsystems.com/releases/CE-Candidate-3.10.4/docs/ECLProgrammersGuide-3.10.4-1.pdf.

[35] A. John, R.A. Fisher and the making of maximum likelihood 1912-1922, Statistical Science 12 (3) (1997), 162-176.

[36] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, R. Buyya, Big Data Computing and Clouds: Challenges, Solutions, and Future Directions,, http://arxiv-web3.library.cornell.edu/pdf/1312.4722.pdf (12.06.2014).

[37] T. Phan, K. Ranganathan, and R.Sion, “Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm”, Proc. 11th Workshop on Job scheduling Strategies for Parallel Processing. Cambridge MA: Springer-Verlag, Berlin, Germany, June 2005.

[38] H. Mohamed, and D. Epema, “An evaluation of the close-to-files processor and data co-allocation policy in multi-clusters”, in Proc. 2004 IEEE International Conference on Cluster Computing, San Diego, CA, USA, Sept. 2004.

[39] S.Venugopal, Scheduling Distributed Applications on Global Grids,Ph.D. Thesis, University of Melbourne,Australia , July 2006.

[40] Apache Hadoop, http://hadoop.apache.org.

[41] Capacity Scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.pdf.

[42] S. Gupta, C. Fritz, R. Price, J. D. Kleer, and C. Witteveen, Throughput Scheduler: learning to schedule on heterogeneous Hadoop clusters, Proceedings of the International Conference on Autonomic Computing, ICAC 2013, June, 2013, San Jose, CA, USA.

[43] Fair Scheduler , http://hadoop.apache.org/docs/r1.2.1/fair_scheduler.pdf

[44] L. Shi, X. Li , and K.L. Tan, S3: An efficient Shared Scan Scheduler On MapReduce Framework, International Conference on Parallel Processing, ICPP 2011, Taipei, Taiwan, September 2011.

[45] A Very Short History of Data Science, http://whatsthebigdata.com/2012/04/26/a-very-short-history-of-data-science (02.01.2014).

[46] In-Memory Analytics; Leveraging Emerging Technologies for Business Intelligence, Gartner Report, 2009.

[47] C. C. Aggarwal, and C. Zhai, Probabilistic Models for Text Mining, in Mining Text Data (eds), Kluwer Academic Publishers, 2012, pp.257-294.

[48] C. Nyce, Predictive Analytics White Paper, American Institute for CPCPU/Insurance Institute of America, 2007, http://www.theinstitutes.org/doc/predictivemodelingwhitepaper.pdf.

[49] Google Big Query , https://cloud.google.com/bigquery-tour.

[50] G. Decandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P.Vosshall, and W.Vogels, Dynamo: Amazon's Highly Available Key-value Store, Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP 2007), Stevenson, Washington, USA, 2007.

[51] Oracle Berkeley DB, Oracle Data Sheet, http://www.oracle.com/technetwork/products/berkeleydb/berkeley-db-datasheet-132390.pdf.

[52] Apache Couch DB , A Database for the Web , www.couchdb.apache.org.

[53] MongoDB Operations Best Practices, http://info.10gen.com/rs/10gen/images/10gen-MongoDB_Operations_Best_Practices.pdf

[54] InfiniteGraph: The Distributed Graph Database, White Paper from Objectivity, www.objectivity.com.

[55] Apache MapReduce, http://hadoop.apache.org/docs/stable/mapred_tutorial.html.

[56] Amazon Elastic Map Reduce, http://aws.amazon.com/elasticmapreduce.

[57] Nancy Lynch, and Seth Gilbert, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT News, 33(2), 2002, pp. 51-59.

[58] J. Kelly, Big Data: Hadoop, Business Analytics and Beyond, Wikibon White paper, 27th August 2012, http://wikibon.org/wiki/v/Big Data: Hadoop, Business Analytics and Beyond.

[59] Neo4j Graph Database, http://www.neo4j.org.

[60] J. P. Collins, Sailing on an Ocean of 0s and 1s, www.sciencemag.org.

[61] Minimizing Processing Bottlenecks with Data Replication, www.information-management.com/web_seminars/100223316-1.html.

[62] R. Buyya, C. Shin Yeo, S. Venugopal, J. Brobergand, and I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems, 25 (6) (2009) 599-616.

[63] M. Quartulli, and I. G. Olaizola, A review of EO image information mining, ISPRS journal of Photogrammetry and Remote Sensing 75 (2013), 11-28.

[64] P.T. Jaeger, J. Lin, and J.M. Grimes, Cloud Computing and Information Policy, Computing in a Policy Cloud, Journal of Information Technology and Politics 5 (3) (2008).

[65] Cloud Infrastructure Management Interface(CIMI) model and RESTful HTTP based protocol -An Interface for managing cloud infrastructure, http://dmtf.sites/default/files/standards/documents/DSP0263_1.0.1.pdf (02.01.2014).

[66] C. Jin, and R. Buyya, Map Reduce programming model for .NET based distributed computing, Proceedings of the 15th European Conference on Parallel Processing (Euro-Par 2009), Delft, Netherlands, August 2009.

[67] Amazon S3 : www.aws.amazon.com/s3.

[68] OpenStack Swift, Object Based Storage and REST Services, http://swiftstack.com/openstack-swift.

[69] Big Data Replication, www.tervela.com/big-data-replication.

[70] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, R. Sears, MapReduce online, Proceedings of the 7th USENIX on Networked systems design and implementation (NSDI 2010), San Jose, California, April 2010.

[71] H. Karlof, S. Suri, and S.Vassilvitskii, A model of computation for MapReduce, Proceedings of 21st ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), Austin, Texas, January 2010.

[72] F. Afrati, A. Sarma, S. Salihoglu, and J. Ullman, Vision paper: Towards an understanding of the limits of Map-Reduce computation, arxiv.org/abs/1204.1754.

[73] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. H. Bae, J. Qiu, and G. Fox, Twister: A runtime for iterative MapReduce. Proceedings of the ACM International Symposium on High Performance Distributed Computing (HPDC 2010) , Chicago, Illionios, June 2010.

[74] P. Bhatotia, A. Wieder, I. Akkus, R. Rodrigues, and U. Acar, Large-scale incremental data processing with change propagation, Proceedings of 3rd USENIX Workshop on Hot topics in Cloud computing (HotCloud 2011), Portland, June 2011.

[75] Y. Bu, B. Howe, M. Balazinska, and M. Ernst, HaLoop: Efficient iterative data processing on large cluster, Proceedings of the 36th International Conference on Very Large Databases (VLDB 2010), Singapore, September 2010.

[76] Y. Zhang, Q. Gao, L. Gao, and C. Wang, PrIter: A distributed framework for prioritized iterative computations, Proceedings of the ACM Symposium on Cloud Computing (SoCC 2011), Cascals, Portugal, October 2011.

[77] Apache HDFS, http://hadoop.apache.org.

[78] Apache Hive, http://hive.apache.org.

[79] J. Lin, MapReduce is good enough? If All You Have is a Hammer, Throw Away Everything That's not a Nail! , http://arxiv.org/pdf/1209.2191v1.pdf (03.12.2013).

[80] K. Raghavendra, A. Chaudhri, K.P. Kumar, and G. Varadan, High Performance Private Cloud for Satellite Data Processing – Engineering in Cloud, Proceedings of First International Conference on Advances in Cloud Computing (ACC 2012), Bangalore, India, 2012.

[81] K. Raghavendra, A. Akilan, K. Ravi, K.P. Kumar, and G. Varadan, Satellite Data Product Generation on Aneka .NET Cloud, Research Product Demonstration,

International Conference on Cluster Cloud and Grid Computing (CCGRID 2010), Melbourne, Australia.

[82] P.V. Radhadevi, and S.S. Solanki, In-flight Geometric Calibration of Different Cameras of IRS-P6 Using a Physical Sensor Model, The Photogrammetric Record 23(121) (2008), 69–89.

[83] P. Deepak, K. P. Kumar, and G. Varadan, A Service Oriented Utility Grid for Data Parallel Remote Sensing Applications, Proceedings of the High Performance Computing & Simulation (HPCS 2009), Kingston, Ontario, June 2009.

[84] P. Deepak, K. P. Kumar, and G. Varadan, Service Oriented Utility grid for 3-Dimensional Topographic Visualization from Satellite Images, Proceedings of the 4th International Conference on eScience, Indianapolis, Indiana, USA, Dec 2008.

[85] L. Bottou, and Y. Lecun, Large scale online learning, Proceedings of the 7th annual conference on Neural Information Processing Systems (NIPS 2003), Whistler, Columbia, CA, December 2003.

[86] Y. Bu, V. Borkar, M. Carey, J. Rosen, N. Polyzotis, T. Condie, M. Weimer, and R. Ramakrishnan, Scaling Datalog for Machine Learning on Big Data, Cornell University Library, vol. abs/1203.0160, 2012., http://arxiv.org/abs/1203.0160.

[87] A. Abouzeid, K. B. Pawlikowski, D.Abadi, A.Silberschatz, and A.Rasin, HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads, Proceedings of the Special Interest Group on Management of Data (SIGMOD 2009), Providence, Rhode Island, USA, June-July 2009.

[88] K. B. Pawlikowski, D. Abadi, A. Silberschatz, and E. Paulson, Efficient processing of data warehousing queries in a split execution environment, Proceedings of the Special Interest Group on Management of Data (SIGMOD2011), Athens, Greece, June 2011.

[89] M. Zaharia, A. Konwinski, A. Joseph, R. Katz, and I. Stocia, Improving MapReduce performance in heterogeneous environments, Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2008), San Diego, USA, December 2008.

[90] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, Job scheduling for multi-user MapReduce clusters, Technical report, Berkeley,2009.

[91] Y. He, R .Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu, RCFile: A fast and space efficient data placement structure in MapReduce based warehouse systems, Proceedings of the International Conference on Data Engineering (ICDE 2011), Hannover, April 2011.

[92] Y. Lin, D. Agarwal, C. Chen, B. Ooi, and S.Wu, Llama: Leveraging columnar storage for scalable join processing in the MapReduce framework, Proceedings of the Special Interest Group on Management of Data (SIGMOD2011), Athens, Greece, June 2011.

[93] A. Floratou, J.Patel, E.Shekita, and S.Tata, Column-oriented storage techniques for MapReduce, Proceedings of the 36th International Conference on Very Large Databases (VLDB 2010), Singapore, September 2010.

[94] A. Jindal, J. A, Q. Ruiz, and J. Dittrich, Trojan data layouts, Right shoes for a running elephant, Proceedings of ACM Symposium on Cloud Computing (SoCC 2011), Cascais, Portugal, October 2011.

[95] T. Kaldewey, E. Shekita, and S. Tata, Clydesdale: Structured data processing on MapReduce, Proceedings of the 15th International Conference on Extending Database Technology (EDBT 2012), Berlin, Germany, March 2012.

[96] J. Dittrich, J.A.Q. Ruiz, A. Jindal, Y. Kargin, V. Setty, J. Schad, Hadoop++: Making yellow elephant run like a cheetah (without even noticing), Proceedings of the 38th International Conference on Very Large Databases (VLDB 2012), Istanbul, August 2012.

[97] J. Dittrich, J.A. Quiance-Ruiz, S. Richter, S. Schuh, A. Jindal, and J. Schad, Only aggressive elephants are fast elephants, Proceedings of the 38th International Conference on Very Large Databases (VLDB 2012), Istanbul, August 2012.

[98] A. Nandi, C. Yu, P. Bohannon, R. Ramakrishnan, Distributed cube materialization on holistic measures, Proceedings of the International Conference on Data Engineering (ICDE 2011), Hannover, April 2011.

[99] S. Blanas, J. Patel, V. Ercegovac, J. Rao, E. Shekita, and Y. Tian, A comparison of join algorithms for log processing in MapReduce, Proceedings of the Special Interest Group on Management of Data (SIGMOD2010), Indiana, USA, June 2010.

[100] A. Okcan, and M. Riedewald. Processing theta-joins using MapReduce, Proceedings of the Special Interest Group on Management of Data (SIGMOD 2011), Athens, Greece, 2011.

[101] R. Lee, T. Luo, Y. Huai, F. Wang, Y. He, X. Zhang, and Y. Smart, Yet another SQL-to-MapReduce translator, Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS 2011), Minneapoli, USA, June 2011.

[102] Y. Kim, K. Shim, Parallel top-k similarity join algorithms using MapReduce, Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE 2012), Washington DC, USA, April 2012.

[103] Pig, http://pig.apache.org.

[104] C. Vecchiola, X. Chu, and R. Buyya, Aneka: A Software Platform for .NET-based Cloud Computing, Proceedings of the High Speed and Large Scale Scientific Computing, IOS Press, Amsterdam, Netherlands, 2009,p.p. 267-295.

[105] Gartner Hype Cycle for Emerging Technologies for 2013, www.gartner.com/newsroom/id/2575515.

[106] GFS: Evolution on Fast-forward; http://queue.acm.org/detail.cfm?id=1594206.

[107] Luster: A Scalable, High-Performance File System, http://www.cse.buffalo.edu/faculty/tkosar/cse710/papers/lustre-whitepaper.pdf.

[108] Exploring Distributed File Systems in Microsoft Windows Server, http://www.dell.com/downloads/global/power/ps2q06-20050301-Lee.pdf.

[109] IBM General Parallel File System, Intelligent Storage and big Data Management, IBM Technology Data Sheet, http://www-03.ibm.com/systems/software/gpfs (22.03.2014).

[110] J. H. Howard, An overview of the Andrew File System, http://ra.adm.cs.cmu.edu/anon/itc/CMU-ITC-062.pdf.

[111] Apache Mahout, Scalable machine learning library, http://mahout.apache.org.

[112] Rack Space, www.rackspace.com (22.03.2014).

[113] F. Chang, J. Dean, S. Ghemawat, W. C. Heish, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, Big Table: A Distributed Storage System for Structured Data, Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006), Seattle, WA, Nov 2006.

[114] The Apache Cassandra Project, http://cassandra.apache.org

[115] W.N. Venables, D.M. Smith, and R core Team, An Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics, http://cran.r-project.org/doc/manuals/R-intro.pdf.

[116] P. Spector, An Introduction to R Statistical Computing Facility, University of California, Berkeley, Sep'2004, http://www.stat.berkeley.edu/users/spector/R.pdf.

[117] A. Bifet, and R. Kirby, Data Stream Mining, A Practical Approach, August 2009, http://www.cs.waikato.ac.nz/~abifet/MOA/StreamMining.pdf.

[118] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, MOA: Massive Online Analysis, A Framework for Stream Classification and Clustering. Journal of Machine Learning Research (JMLR 2010), pages 44-50.

[119] U. Kang, C. E. Tsourakasis, and C. Faloutsos, PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations, Proceedings of the 9th IEEE International Conference on Data Mining (ICDM 2009), Miami, Florida, USA, December 2009.

[120] Graph Lab, A New Parallel Framework for Machine Learning, http://select.cs.cmu.edu/code/graphlab

[121] Apache Spark, https://spark.incubator.apache.org

[122] C. Gentry, A Fully Homomorphic Encryption Scheme, Ph.D thesis, http://crypto.stanford.edu/craig/craig-thesis.pdf.

[123] C. Gentry, Computing Arbitrary Functions of Encrypted Data, Communications of the ACM 53 (3) (2010) 97-105.

[124] D. Boneh, A. Sahai, B. Waters, Functional Encryption: Definitions and Challenges, Proceedings of 8th Theory of Cryptography Conference (TCC 2011), Providence, RI, USA, March 2011.

[125] G. Wroblewaki, General Method of Program Code Obfuscation, Ph.D thesis, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002.

[126] A.C.C Yao, Protocols for secure computations, Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS 1982), Chicago, Illinios, 1982.

[127] MongoDB, http://docs.mongodb.org/manual/faq/developers/#faq-developers-when-to-use-gridfs.

[128] Chang F, Dean J, Ghemawat S, Heish W. C., Wallach D. A, Burrows M, Chandra T, Fikes A, Gruber R. E. Big table: a distributed storage system for structured data, Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006), Seattle, WA, Nov 2006.

[129] O. Pentakalos, An Introduction to the InfiniBand Architecture, Oreilly, 2002.

[130] Space borne remote sensing. http://www.crisp.nus.edu.sg/~research/tutorial/spacebrn.htm.

[131] OpenStack Swift, Object Based Storage and REST Services, http://swiftstack.com/openstack-swift (5.3.2014).

[132] Google-gdata, .NET library for the Google Data API, http://code.google.com/p/google-gdata

[133] Apache storm, http://storm.incubator.apache.org/

[134] S4, distributed stream computing platform, http://incubator.apache.org/s4/

[135] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE), 41(1): 23-50, Wiley Press, New York, USA, January 2011.

[136] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, Big Data Computing and Clouds: Trends and Future Directions, Journal of Parallel and Distributed Computing, Available online 27 August 2014, DOI: 10.1016/j.jpdc.2014.08.003

[137] A. Chaudhri, R. Kune, K.P. Kumar, and G.Varadan, High Performance Private Cloud for Satellite Data Processing– Engineering in Cloud, Proceedings of International Conference on Advances in Cloud Computing , ACC 2012 , Bangalore , India.

[138] Z. Michalewicz: Genetic Algorithms + Data Structures =Evolution Programs, 1992, Springer.

[139] Java GA Lib, Genetic Algorithm Library: http://sourceforge.net/projects/java-galib.

[140] Goldberg D.E., Genetic Algorithms in search, Optimization and Machine Learning, Addison Wesley , Reading , MA,1989.

[141] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selections, MIT Press, Cambridge, MA, USA.

[142] J. H. Holland, Adaption in Natural and Artificial Systems, MIT Press, 2nd edition, 1992.

[143] T. Phan, K. Ranganathan, and R.Sion, Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm, Proceedings of 11th Workshop on Job scheduling Strategies for Parallel Processing. Cambridge MA: Springer-Verlag, Berlin, Germany, June 2005.

[144] H. Mohamed, and D. Epema, An evaluation of the close-to-files processor and data co-allocation policy in multi-clusters, Proceedings of the 2004 IEEE International Conference on Cluster Computing, San Diego, CA, USA, Sept. 2004.

[145] Capacity Scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.pdf

[146] L. Shi, X. Li , and K.L. Tan, S3: An efficient Shared Scan Scheduler On MapReduce Framework, Proceedings of the International Conference on Parallel Processing, ICPP 2011, Taipei, Taiwan, September 2011.

[147] Paul M.Mather, Computer processing of remotely sensed images- An introduction, John Wiley Sons, 1987.

[148] Radhadevi .P.V, S. S. Solanki, In-flight Geometric Calibration of Different Cameras of IRS-P6 Using a Physical Sensor Model, The Photogrammetric Record 23 (121) (2008): 69–89.

[149] P.Deepak,K.Pramod Kumar, and Geeta Varadan: A Service Oriented Utility Grid for Data Parallel Remote Sensing Applications. Proceedings of the High Performance Computing & Simulation (HPCS), 2009.

[150] George Joseph: Fundamentals of Remote Sensing, University Press, 2nd Edition.

[151] M. F. Worboys, and M. Duckham, GIS: a computing perspective, 2nd Edition, May 2004, CRC Press.

[152] M. D. Assuncao, R. N. Calheiros, S. Bianchi, M. A. S. Netto, and R. Buyya, Big Data Computing and Clouds: Trends and Future Directions, Journal of Parallel and Distributed Computing, Available online 27 August 2014, DOI: 10.1016/j.jpdc.2014.08.003

[153] T. Phan, K. Ranganathan, and R.Sion, Evolving toward the perfect schedule: Co-scheduling job assignments and data replication in wide-area systems using a genetic algorithm, Proceedings of the 11th Workshop on Job scheduling Strategies for Parallel Processing. Cambridge MA: Springer-Verlag, Berlin, Germany, June 2005.

[154] H. Mohamed, and D. Epema, An evaluation of the close-to-files processor and data co-allocation policy in multi-clusters, Proceedings of the 2004 IEEE International Conference on Cluster Computing, San Diego, CA, USA, Sept. 2004.

[155] S. Gupta, C. Fritz, R. Price, J. D. Kleer, C. Witteveen, Throughput Scheduler: learning to schedule on heterogeneous Hadoop clusters, Proceedings of the International Conference on Autonomic Computing, ICAC 2013, June, 2013, San Jose, CA, USA.

[156] OpenStack Swift, Object Based Storage and REST Services, http://swiftstack.com/openstack-swift.

[157] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE), 41(1): 23-50, Wiley Press, New York, USA, January 2011.

[158] K.Bakshi, Considerations for Big Data: Architecture and Approach, Proceedings of the 2012 IEEE Aerospace Conference- Big Sky, Montana, USA, March 2012.

[159] K. Michael, and K. W. Miller, Big Data: New opportunities and New Challenges, IEEE Computer, 46 (6) (2013), pp. 22-24.

[160] S. Ghemawat, H. Gobioff, and S. T. Leung, The Google File System, Proceedings of the 9th ACM symposium on Operating System Principles (SOSP 2003), NY, USA 2003, pp. 29-43.

[161] K. Schvachko, H. Kuang, S. Radia, and R. Chansler, The Hadoop Distributed File System, Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST 2010), Incline Village, Nevada, USA, May 2010.

[162] C. Jin, and R. Buyya, MapReduce Programming Model for .NET-Based Cloud Computing, Proceedings of the Euro-Par 2009 Parallel Processing, Lecture Notes in Computer Science, 5704 (2009), pp. 417- 428.

[163] J. Ekanayake, S. Pallickara, and G. Fox, MapReduce for Data Intensive Scientific Analyses, Proceedings of the IEEE fourth International Conference on e-Science, Indiana Police, Indiana, USA, Dec 2008, pp. 277-284.

[164] Satellite imaging corporation, http://www/satimagingcorp.com/satellite-sensors

[165] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie, Remote Sensing Big Data computing: Challenges and opportunities, Future Generation Computer Systems, Volume 51, October 2015, Pages 47–60.

[166] R. C. Gonzalez, and R. E. Woods, Digital Image Processing, 3rd Edition, 2007 (chapters 1, and 3).

[167] X. Cao, and S. Wang, Research about Image Mining Technique, Journal of Communications in Computer and Information Sciences, 288(2012), pp. 127-134.

[168] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, Twister: A Runtime for Iterative MapReduce, Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010), Chicago, Illinois, USA 2010, pp. 810-818.

[169] W. Jiang, V. T. Ravi, and G. Agarwal, A Map-Reduce System with an Alternate API for Multi-Core Environments, Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGRID 2010), Melbourne, May 2010,pp. 84-93.

[170] L. Kennedy, M. Slaney, and K. Weinberger, Reliable Tags using Image Similarity: Mining Specificity and Expertise from Large-Scale Multimedia Databases, Proceedings of the 1st workshop on Web-Scale Multimedia Corpus, Beijing, October 2009, pp. 17-24.

[171] L. L. Shi, B. Wu, B. Wang, and X. G. Yang, Map/Reduce in CBIR Applications, Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT), Harbin, December 2011, pp. 2465-2468.

[172] C. T. Yang, L. T. Chen, W. L. Chou, and K. C. Wang, Implementation on Cloud Computing, Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS 2011), Beijing, September 2011, pp. 482-485.

[173] H. Kocalkulak, and T. T. Temizel, A Hadoop Solution for Ballistic Image Analysis and Recognition, Proceedings of the International Conference on High Performance Computing and Simulation (HPCS 2011), Istanbul, July 2011, pp. 836-842.

[174] M. H. Almeer, Cloud Hadoop MapReduce For Remote Sensing Image Analysis, Journal of Emerging Trends in Computing and Information Sciences, vol. 3, 637-644.

[175] I. Demir, and A. Sayar, Hadoop Optimization for Massive Image Processing: Case Study of Face Detection, International Journal of Computers and Communications and Control, 9(6) (2014), 664-671.

[176] T. White, The Small files problem: 2009, http://www.cloudera.com/blog/2009/02/02/the-small-files-problem.

[177] C. Sweeney, L. Liu, S. Arietta, and J. Lawrence, HIPI: A Hadoop Image Processing Interface for Image-based MapReduce Tasks, undergraduate thesis, University of Virginia, USA.

[178] K. Potisepp, Large Scale Image Processing Using MapReduce, MSc. Thesis, Institute of Computer Science, Tartu University, 2013.

[179] GDAL, Geospatial data abstraction library.http://www.gdal.org/