Dynamic and Scalable Data Access and Integration Services in Cloud Computing Environment

A Thesis submited for the award of degree of

Doctor of Philosophy

in

Computer Science and Engineering Department

by

Ajay Kumar
[Reg. No: 950903044]

Under the supervision of

Dr. Seema Bawa
Professor, CSED

Thapar Institute of Engineering and Technology
Patiala-147004, Punjab, India

June 2020

Certificate

I hereby certify that the work which is being presented in this thesis entitled **"Dynamic and Scalable Data Access and Integration Services in Cloud Computing Environment"**, for the award of degree of **"Doctor of Philosophy"** submitted to the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work, carried out under supervision of Dr. Seema Bawa, Professor, Computer Science and Engineering Department. It refers to the work done by other researchers which are duly listed in the reference section.

The matter presented in this thesis has not been submitted in part or full to any other institute or university, for the award of any other degree.

(Ajay Kumar)

Reg. No. 950903044

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

03/June/2025

(Dr. Seema Bawa)

Supervisor

Professor, Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

Punjab, INDIA

Acknowledgement

First of all, I express my gratitude to that almighty, who blessed me with the zeal and enthusiasm to complete this research work successfully in spite of hard commitments at my workplace. This thesis arose in part out of years of the research and I have interacted with a great number of people whose contribution in assorted ways to the research and the making of the thesis deserved special mention. It is a pleasure to convey my gratitude to them all in my humble acknowledgement.

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Seema Bawa, Professor, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology Patiala for the continuous support of my Ph. D. study and related researches, for his patience, motivation, and immense knowledge. Her guidance helped me in all the time of research. And during the most difficult times when writing this thesis, she gave me the moral support and the freedom I needed to move on. Her truly scientist intuition has made her as a constant oasis of ideas and passions in technology, which exceptionally inspire and enrich my growth as a student, a researcher and as a professional. I could not have imagined having a better supervisor and mentor for my Ph. D study.

I would like to thank to Dr. Maninder Singh, Head, Computer Science and Engineering Department for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I gratefully pay my sincere thanks to the members of Doctoral Committee, Dr. Inderveer Chana, Professor CSED, Dr. V. P. Singh, Associate Professor CSED and Dr. M. D. Singh, Associate Professor, EIED for their observations, constructive criticism and valuable comments which helped me a lot in presenting the results and shaping this thesis.

My sincere thanks also goes to Dr. Prashant Singh Rana, Assistant Professor, Computer Science and Engineering Department, TIET Patiala who provided me an opportunity to join their team as member of Thapar Summer School, and who gave access to the laboratory and research facilities. Without they precious support it would not be possible to conduct this research.

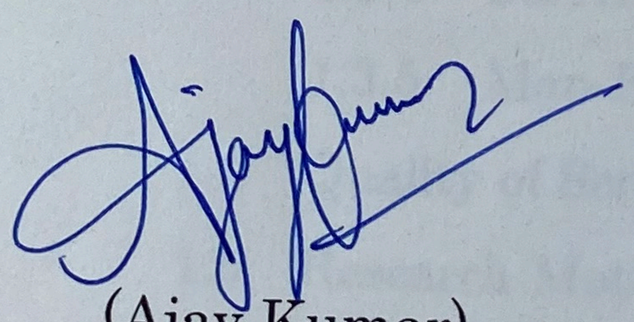
I thank my fellow labmates Mr. Vivek, Ms. Sumedha, Ms. Sawinder, Mr. Shubham, Ms. Priya for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had during Ph. D. tenure.

My Special thanks to Mr. Sandeep Saharan, Mr. Vijay Prakash, Mr. Sachin and Ms. Shukhandeep who always shows their willingness to support and helped me, with a smiling face, as and when required.

Last but not the least, I would like to thank my family. My parents deserve a special mention for their inseparable support and prayers. Thanks to my father Late Shashi Kant Prasad who always believed in my capabilities more than anyone else.

Finally, I would like to express my gratitude to the "**Heartfulness Organization**" and all the brothers and sisters associated with it who made me aware of the practical ways of learning to relax, meditate and discover unlimited resources of the heart. With practice, my inner peace has been very helpful in experiencing better concentration and inner balance.

With these words, I share the credit for my present humble achievement as well as the future successes with the persons mentioned above.



(Ajay Kumar)

Contents

List of Figures

List of Tables

List of Abbreviations

ABC	Artificial Bee Colony
ACOT	Adjacency COS Overlay Topology
APF	Apache Software Foundation
API	Application Programming Interface
ART	Average Response Time
AWS	Amazon Web Service
B2B	Business to Business
CaaS	Communication as a Service
CIS	COS Integration Service
COD	Cluster on Demand
COS	Cloud-Oriented Storage
COSaaS	Cloud-Oriented Storage as a Service
CRM	Customer Relationship Management
CSRD	Cloud Storage Resource Discovery
CUM	Cloud Usage Monitor
DaaS	Data as a Service
DAIS	Dynamic Access and Integration Services
DFS	Distributed File System
EC2	Amazon Elastic Compute Cloud
ETL	Extract Transform Load
FLOPS	Floating Point Operations Per Second
GA	Genetic Algorithm
GACO	Generalized Ant Colony Optimizer
HCI	Human Computer Interaction
HDFS	Hadoop Distributed File System
IaaS	Infrastructure as a Service
JIT	Just-in-Time
LAN	Local Area Network
MIPS	Million Instruction Per Second

MR	Map Reduce
NIST	National Institute of Standard and Technology
P2P	Peer to Peer
PaaS	Platform as a Service
PSO	Particle Swarm Optimization
QoS	Quality of Services
RDBMS	Relational Database Manegement System
S3	Simple Storage Service
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structure Query Language
TPC	Transaction Processing Performance Council
TT	Turnaround Time
UDDI	Universal Data Discovery Integration
VIOLIN	Virtual Inter-networking on Overlay Infrastructure
VM	Virtual Machine
WAN	Wide Area Network
WSDL	Web Service Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

Abstract

Cloud computing has been emerging paradigm which operates on large scales thus requires different approaches and application architectures. This is a suitable option for bulk storage and reporting on the Internet. Cloud computing provides cost-effective business solutions, as it allows hosting of various services including computational power, storage services and network services on shared physical resources. Cloud computing empowers organizations/ enterprises to host scientific, engineering and different business applications. It gives high availability and more accessibility across the globe. However, there is a need to draw attention to the growing trends and growing data of online application services, as well as problems related to accommodating the continued expansion of cloud data centers. Particularly huge storage space and high bandwidth usage resulting in ultimately high operating costs. Now, Cloud-Oriented Storage as a Service (COSaaS) became one of the major aspects of cloud service model in which user can get access storage space dynamically. Here, Service Level Agreement (SLA) violation is one of the key issues of cloud computing. In general, SLA is a contract between service providers and consumers with some conditions like, service availability, service deadline, consistency of service, network congestion etc. In cloud environment, each user has apparent definition of SLA constraints and flexible conciliation procedures that can increase the trustworthiness relationship between users and providers.

This thesis proposes a unique framework namely: Dynamic Access and Integration Service (DAIS) that achieve the on-demand and integrated data access in cloud computing environments. The DAIS framework presents unique approach in cloud environment for large-scale storage management. It introduces new concepts including dynamic and scalable data access, self-organizing capability and well balanced multi-way COS overlay topology. It removes the necessity of centralized approaches for traditional static storage management and adopts the dynamic and scalable data access in the cloud computing environment. DAIS framework has four major

components-Adjacency COS Overlay Topology (ACOT), Cloud Storage Resource Discovery (CSRD),COS Integration Services (CIS) and Cloud Usage Monitor (CUM). Each of these components have been explained thoroughly in this thesis. The proposed framework is tested in Hadoop 2.7 and Java 1.8.0 environments for all concerned experiments. WAN emulator has also been used to find the impact of dynamicity of DAIS framework. Several experiments including bandwidth evaluations (load vs. throughput and Delay vs. throughput), dynamicity and scalability evaluation, SLA optimization and energy efficiency evaluation have been conducted in this thesis. All test results are well explained and highlighted in different tables as well as visual graphs.

Apart from this an effort has been made to propose a swarm-based meta-heuristic algorithm named Generalized Ant Colony Optimizer (GACO) for dynamic execution of cloud services. GACO has been experimentally demonstrated and compared with three well-known algorithms including Particle Swarm Optimization, Genetic Algorithm and Artificial Bee colony. As validated by experimental results, the proposed algorithms perform better in most of the cases.

Chapter 1

Introduction

Cloud computing is an emerging paradigm which utilizes the central isolated servers via Internet. User can access data and applications at any server through Internet. The cloud paradigm permits highly proficient computing and storage system by consolidating processing power, bandwidth, and storage space. Cloud provides robust administration over sustain storage memory spaces. Today's, the increased deployment of data center to facilitate large scale data storage of different sources in cloud computing environment. Cloud computing is a cost effective solution, as it allows the hosting of computating It needs high performance computing and lead to huge consumption of energy.

This chapter presents the evolution of computing, cloud service model, cloud-oriented storage systems and their goals and challenges. Further, it highlights enabling technologies for cloud storage and service level agreement(SLA) optimization parameters. It culminates with elaboration of thesis organization and contribution.

1.1 Evolution of Computing

The distributed systems have been used widely for business-oriented applications and high performance scientific computations for a significant volume of data calculation in last three decades. In this regards, various distributed computing concepts have been evolved to solve the computational and storage problems like cluster comput-

ing, grid computing, P2P computing, market-oriented computing, service-oriented computing, etc. The cluster computing uses the processing capability within the organization itself [45]. The grid computing optimizes and shares the resources among geographically distributed systems [13]. P2P computing directly shares the capability to peer node without restriction [10]. Market-oriented computing uses both cluster and grid computing concepts to provide services over the Internet on payment basis. The service-oriented computing offers various business solutions in the form of web services. The concepts of cluster, grid, market-oriented and service-oriented computing combined together and working over the Internet is known as the cloud computing [3]. The fundamental goals of the cloud computing are availability, usability, scalability, reliability, security, flexibility, agility, serviceability and efficiency. Cloud system is operating at large scales that require different approaches to application architectures [3][4]. Figure 1.1 [168] illustrates the evolution of various computing paradigms.

1.2 Cloud Computing

Cloud compuitng is an emerging paradigm for convenient, ubiquitous, on-demand network access to a distributed shared pool of resources such as servers, networks, storage spaces, applications and services [90]. These resources can be frequently discovered and provisioned with minimal time and cost. This paradigm has five essential characteristics, three major service model and various QoS and SLA parameters which are explained in coming sections.

1.2.1 Essential Characteristics of Cloud Computing

As cloud computing services mature both commercially and technologically, it will be easier for any organizations to optimize its time and cost. Knowing what cloud computing is and what it does, however, is just as important. Specially in education system cloud computing became one of the most popular platform [151]. The National Institute of Standards and Technology (NIST) [106][116] defines cloud computing as

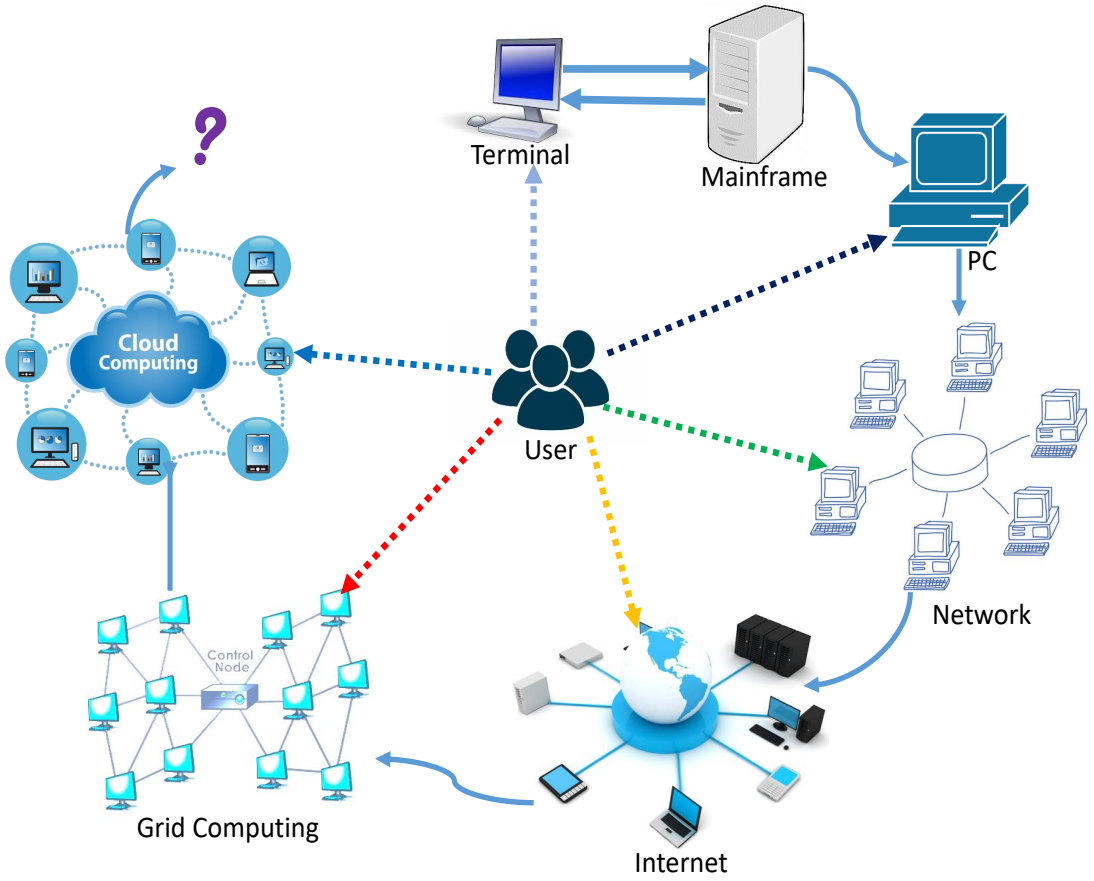


Figure (1.1) Evolution of Computing

it is known today through five particular characteristics. Figure 1.2 [164] illustrates the defining features of cloud computing.

a. **On-demand resources:** Cloud resources can be allocated to user without human interference from the provider. In other words, resources can be allocated on demand self-service. These resources may be Virtual Machine(VM) instances, storage space, applications services, etc. Most of the organizations can use a self-driven web portal to access their cloud services and also to activate and de-activate the services as they require. All resources can ne accessed dynamically and scale usage level.

b. **Broad network accessibility:** Cloud computing resources are made available on all networks and different types of consumers are accessed through various

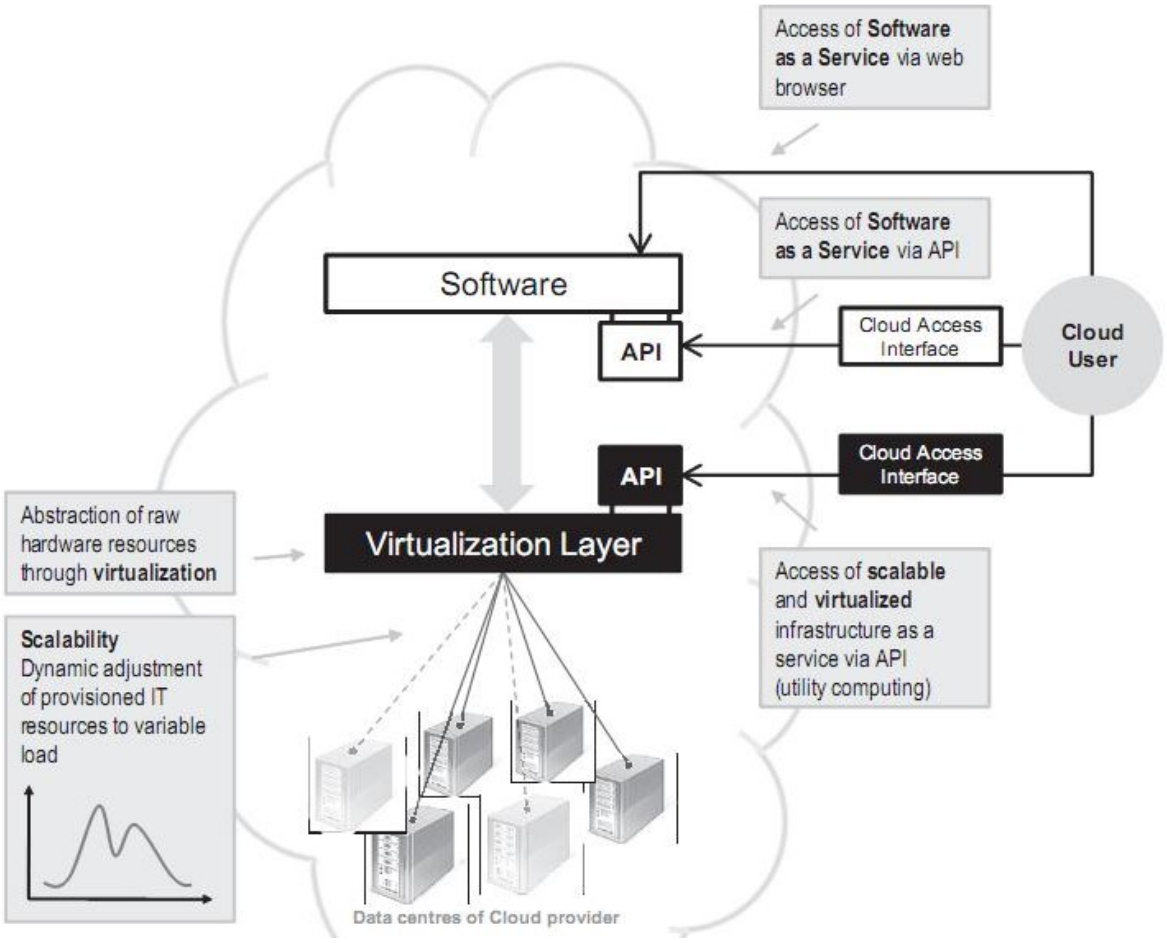


Figure (1.2) Defining features of cloud computing

platforms. These services are available via Internet with high bandwidth link. Sometime it could be a private cloud as Local Area Network (LAN). In general, the cloud networks are spread globally, it can be accessed via Internet. Latency and bandwidth are most important aspects of cloud and broad networks. It relates the Quality of Services (QoS) and user attraction.

c. **Resource Pooling and Multi-tenancy:** A single physical resource can be served to many users is known as resource pooling. The provider's computing resources are pooled together to serve multiple consumers using multiple-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The resources include among others storage, pro-

cessing, memory, network bandwidth, virtual machines and email services. In cloud computing resources are designed in such a way to support multi-tenant concept. It allows share same resource or infrastructure among many users without knowing the other users.

d. **High-level scalability and elasticity:** On demand resource provisioning is one of the great things in the cloud computing. Any user can activate and de-activate as per their choice. Resources can be scaled up/ down rapidly or sometime it may be automatically. Everything are scaled in cloud computing including usage of computing service, storage space, bandwidth capacity, data integration, etc without any additional cost. Elasticity is one of the important aspect of cloud computing that implies organizations which can quickly provision and de-provision of any cloud resources. Quick provisioning and de-provisioning might be applied to Virtual Machine (VM), data centers or business applications. Cloud elasticity requiring either to provision more resources or less in the cloud is known as Just-in-Time (JIT). Sometime an organization needs to test a virtual machine on Supervisory Control and Data Acquisition (SCADA)[27] system before going to next step.

e. **Measured service:** Usage of cloud computing resources of manufacturing organizations pay as per needs. Utilization of resource can be optimized by dynamic resource allocation strategy. It means whatever resources used by particular user that must be measured, monitored and reported to cloud service provider. Cloud computing resource usage can be measured, controlled, and reported providing transparency for both the provider and consumer of the utilised service. Cloud computing services use a metering capability which enables to control and optimise resource use. This implies that just like air time, electricity or municipality water IT services are charged per usage metrics – pay per use

1.2.2 Cloud Service Model

Cloud computing is a collection of dynamic and scalable virtualized resources in the form of services that are available over the Internet [5]. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. In General, this cloud model is composed of five essential characteristics, three service models, and four deployment models. Figure 2 shows the cloud service model along with their user characteristics.

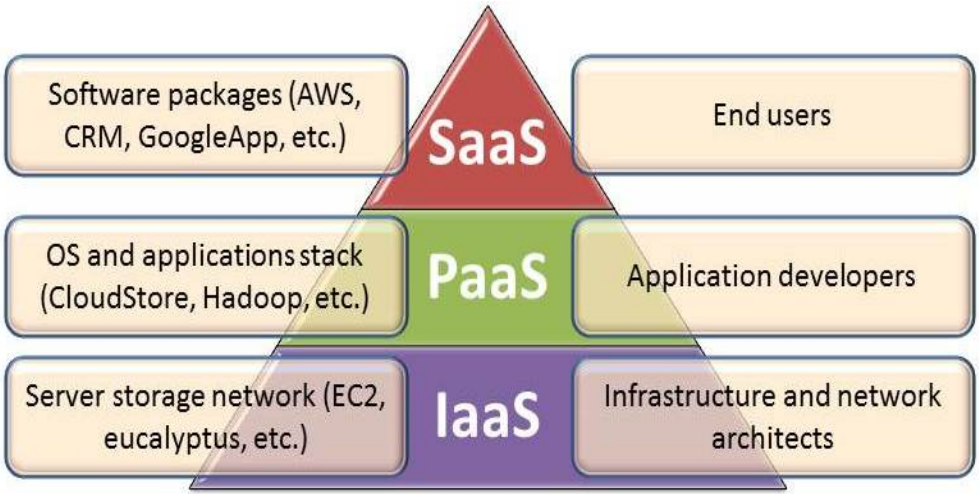


Figure (1.3) Cloud Service Model

Now, above service model can be extended in several types services as follows:

i Infrastructure as a Service (IasS): rent processing, processing resources, provides on-demand virtual images, such as EC2, Eucalyptus, etc.

ii Platform as a Service (PaaS): It specifies the various service capabilities including data processing, computing services, storage capacity, servers, operating system, etc. These services do not managed or controlled by any customer. But has control over the deployed applications and possibly configuration settings for the application-hosting environment. For examples- CloudStore, S3, Sector/Sphere, Hadoop, etc.

iii Software as a Service (SaaS): Software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of software maintenance, and reduces the expense of software purchases by on-demand pricing. Any software or application that can be delivered as a service through the Internet such as Salesforce.com, Amazon Web Service, CRM, LotusLive, GoogleApps, etc.

iv Data as a Service (DaaS): its concerns with various data qualities that offer in a centralized place, enriching data and providing it to different systems [40].

v Database as a Service (DBaaS): the cloud platforms offer options for using a database as a service, without physically launch. Users can independently use the cloud database through a virtual machine image.

vi Communication as a Service (CaaS):it supports to network security, dedicated bandwidth, dynamic provisioning and performance monitoring.

vii Security as a service (SECaaS): it is a business model for the safety concern issues. Typically, it involves applications such as anti-virus packages that delivered via the Internet [15]. These security services include intrusion detection, Internet security, firewall, authentication and authorization, anti-virus, anti-malware/spyware, etc[45].

1.2.3 Cloud-Oriented Data Storage Management

Cloud-oriented data storage is a type of service in which data is retained, managed, and backed up. This service is available to users on a network, which is usually the Internet. This allows the user to store the files online so that users can access them from any location through the Internet. Service provider company makes the user available online by placing uploaded files on the external server. This gives ease and convenience to companies using cloud storage services, but can be potentially costly for storage auditing process [61][152]. Users should also know that it is important to

back up their data when using cloud storage services because recovery of data from cloud storage is very slow for local backups [174].

1.2.4 Cloud Storage Management: Challenges and Goals

This section highlights some major challenges and goals for deploying the ultra-large cloud-oriented storage systems.

Challenges: Deploying ultra-large cloud-oriented data storage systems are not trivial or straightforward task. Following are list of obstacles that affect our system:

- Data Confidentiality
- Data Lock-in
- Data Transfer Bottlenecks
- Application Parallelism
- Shared-Nothing
- Performance Unpredictability
- Application Debugging in ultra-large distributed system
- Network Congestion
- Strong Consistency
- Fair Transaction guarantee
- Interoperability
- Unavailability of Services
- Data Integrity

Goals: In general, successful cloud-oriented data storage system has been designed to satisfy the following wish-list:

- Data Availability - It refers to the ability to ensure that necessary data is accessible within an organization's IT infrastructure even when there is a disruption.
- Elasticity - It defines the degree for which a system is capable of making provisions for changes in workload and favoring the resources by de-provisioning, such that the available resources at all times match the current demand as closely possible.
- Performance - It confines the bandwidth, Service Level Agreement (SLA) and well-organized cloud environment.
- Multi-tenancy - It provides a cloud computing architecture that allows customers to share computing resources in a public or private cloud. Each tenant's data is separated and is invisible to other tenants.
- Load Balancing - It enables the organization to manage workload demands or applications needs by distributing resources on different networks or servers.
- Fault Tolerance - It provides the continuity of service after failure of one system or any part of system.
- Flexible - Increase the level of customization and change as per needs any time.
- High data confidentiality - It refers to protecting information from being accessed by unauthorized users.
- Scalability - It refers to handled the increased loads.
- High availability- It refers to dynamic resource provisioning as per user needs.
- High durability - It confines the long life survival of the system.
- High data integrity - I maintains the consistency level of data and any other information.
- Cost saving

1.3 Enabling Technologies for Cloud Computing

This subsection explains existing cloud-based storage technologies such as virtualization, web services, Web 1.0/ 2.0/ 3.0, Service Oriented Architecture (SOA) and Map-Reduce (MR) model.

1.3.1 Virtualization Technology

It enables to dynamically adjust the underutilized resources deployed in IT infrastructures as per the need of the organization. The resources can be aggregated, expanded or pulled back as the application requirements for computations, storage or bandwidth change. It implies a level of automation with respect to resource sharing, scaling and new deployments [12]. Organizations can dynamically create copies of existing environments in very short time. This results in significant cost savings as these environments can coexist on same server utilizing very few resources. There are two approaches to virtualization in the Cloud Computing environment [4]. Hardware Virtualization- It allows users to manage hardware equipments in a plug and play manner. Hardware equipments could be added or removed without affecting the normal operations of other equipments in the system. Software Virtualization - It specify the software profile of a system, which is stored as a unique image file. This image file consists of software systems including operating system, middleware and applications resulting in software reusability. It also allows users to dynamically assemble and execute code. Code elements are dynamically copied from repositories and pasted in right places as per the business logic. With virtualization, only costs involved are of the computing and memory overhead to run the operating systems as a guest process and of the bandwidth consumed to download the image file or code elements [22]. It is envisioned in future, as the bandwidth growth curve increase; a complete virtual cloud operating system can be downloaded and emulated [5].

1.3.2 Web Services

Web services provide developers methods of integrating Web applications over the Internet. XML, SOAP, WSDL and UDDI open standards are used to tag data, transfer data, describe and list services available. More specifically, a Web service is a software application with a standardized way of providing interoperability between disparate applications. Web services allow organizations to share data without having direct access or knowledge of systems beyond the firewall. They're essentially communicators for the organizations' Web applications. The communicator is the API, which allows applications to interface with each other and link to GUIs to provide end-user applications. Web and cloud services have vastly improved the quality and functionality of applications for end users. The main component of a web service is the data that is transferred between the client and server. Every framework requires some kind of architecture to ensure that the entire structure as desired. Similarly, in web services, there is an architecture that has three different roles as given below:

i **Service provider** creates web services and make it available for clients as per their needs.

ii **Service requester** is the client application that communicate with web services.

iii **Service registry** enables the client application to locate the web services.

A provider informs the service mediator (service registry) about the existence of web service by using the mediator's publish interface to make the service accessible to consumers. Requestor communicate with service registry and find the appropriate published services. After collecting the information from service registry about the published web services, the requestor is able to invoke or bind the web services. Figure 1.4 shows the how all three are interact to each others.

1.3.3 Web 1.0/2.0/3.0

Web is a hypertext system that works on the Internet. The Internet world is constantly evolving with new technologies in order to enhance the user experience. This subsection highlights the differences between Web 1.0, Web 2.0 and Web 3.0.

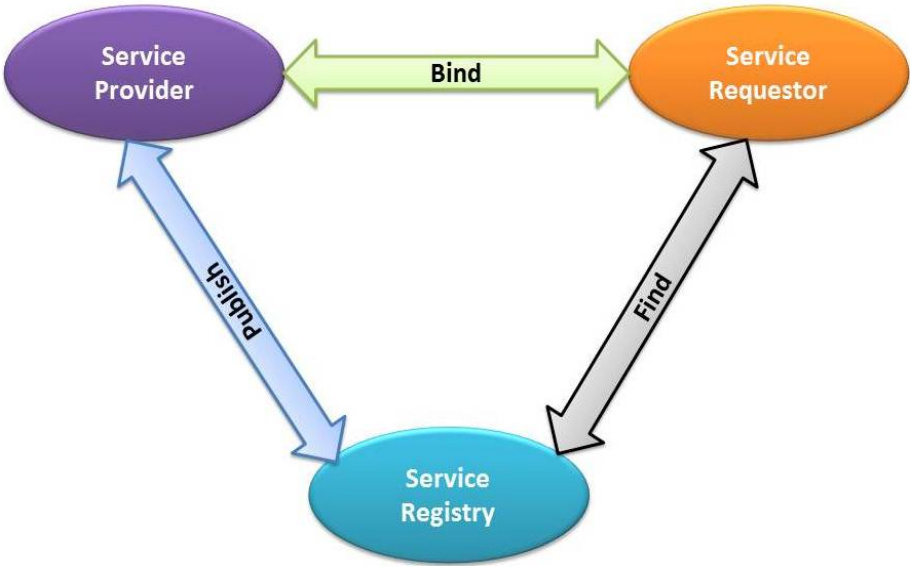


Figure (1.4) Web service architecture

a. **Web 1.0:** It is a initial stage of the World Wide Web(WWW) evolution. Web 1.0 consists very few content creators with the huge majority of the content consumers. It contains only static web page and mainly it is free hosted on web server os ISP-running server. Some main features of Web 1.0 are as follows:

   - It contains only static web pages.
   - Contents are accessed from server's file-system.
   - Supports Common Gateway Interface (CGI)
   - Create frames and table on web page.

b. **Web 2.0:** It refers to wide variety of web applications like web site, web blog, wiki, web page, etc which highlight user-oriented contents[20] . Web 2.0 [155] is supports readable as well as writable contents. It does not support to a modify any technical specification. Any collaboration and interaction with each other is permitted by Web 2.0 in the social media. It is the enhancement of Web 1.0. It supports AJAX and JavaScript features in web development. Following are some important features of Web 2.0:

   - Users can store and retrieve information from the web.

- Support dynamic contents.
- Evaluation and online commenting of data.
- Support APIs programming concepts .
- Allow to read and write the web page.

c. Web 3.0: It refers the evolution of web-oriented interaction and increase the utilization of web. It includes online data manipulation concept. It enables both front-end and back-end features on web applications. Web 3.0 [20] describes evolution of web site utilization and interaction among different locations. Today's implicit feedback of various users and their activities like clicks, visits, likes, comments, searching, browsing, etc are most important for future business prospective. After sometime these data takes in the form of large data-sets. Web 3.0 usages web analytics concept along with various machine learning techniques to find the appropriate business solutions [151]. Some important features of web 3.0 are:

   - Semantic Web
   - Artificial Intelligence
   - 3D graphics
   - Ubiquity computing

1.3.4 Service Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) establishes an architectural design for globally distributed application components which includes search, data mapping, access control, and security aspects [145]. The main purpose these model, to increase the status, agility, efficiency and productivity of an organization [9]. SOA consists two main functions. The first is to create a comprehensive architectural model that defines the goals and approaches of applications that will help them meet those goals. The second task is to define specific implementation specifications, which are usually associated with the formal Web Service Description Language (WSDL) and Simple

Table (1.1) Difference between Web 1.0, Web 2.0 and Web 3.0

Features	**Web 1.0**	**Web 2.0**	**Web 3.0**
Accessibility	Readable	Readable and writable	Executable and portable
Services	Web Forms	Web services	Intelligent services
Protocols	HTML/ Portals	XML/ XSLT/ WSDL	RDF/ RDFS/ OWL
Contents	Owning Content	Sharing Content	Consolidating Content
Focus	Organization	Community	Individual
Data mode	Static data	Interactive data	Dynamic data
Sources	Web forms/ pages	Web blogs/ sites/ wiki	Search engine/ personalized/ intelligent search

Object Access Protocol (SOAP) specifications. It has three main objectives, which is a separate part of the application life-cycle.

i To make these services loosely for applications. They are designed to be easily used by software developers who have built applications in a consistent way.

ii To provide a mechanism to publish the available services, including their functionality and input/output (I/O) requirements. The services are published in such a way that allows developers to engage in applications easily.

iii To prevent the problems of security and governance. The security architecture in SOA revolves around protecting individual components related to those components within the identity and authentication processes securing real connections between architectural components.

1.3.5 Map-Reduce (MR) Model

Map-Reduce [43] is a programming model using which is used to process large-scale data-set parallel on different clusters. It is a processing technique and a program

framework based on Java programming language. Map-reduce model contains two important tasks, namely Mapping and Reducing. Mapping process takes a bunch of data and converts it in another data sets, where each element divided into different key/value pairs. Secondly, reducing process gets output of mapping as input and combines those key/ value pairs into a smaller sets. There are following design principles of Map-reduce:

- Parallel task execution
- Easy fault tolerance
- Extremely scalable
- High level abstraction
- Low cost
- Load balancing

The main advantage of Map-reduce is that it is very easy to scale all the operational activities during data processing over multiple nodes. Decomposing a data processing service for mappers and reducers are sometimes nontrivial. But, once we write a program in Map-reduce form then scaling the data-sets and run over thousands of clusters. Map-reduce model execute the program in three stages- i) map stage, ii) shuffle stage and iii) reduce stage.

Map stage: The mapping task is to process the input data-sets. In general, input data-sets are in the form of files and is stored in the Hadoop File System(HDFS) [156]. Mapper function gets inputs line by line and create various small chunks of data-sets.

Reduce stage: It is the shuffling and combine stage of all the data-sets according to key value. Mainly, it takes input from Mapping function. After this reducer produces a new output sets and will be stored in the Hadoop File System(HDFS). Figure 3 illustrates the Map-reduce model.

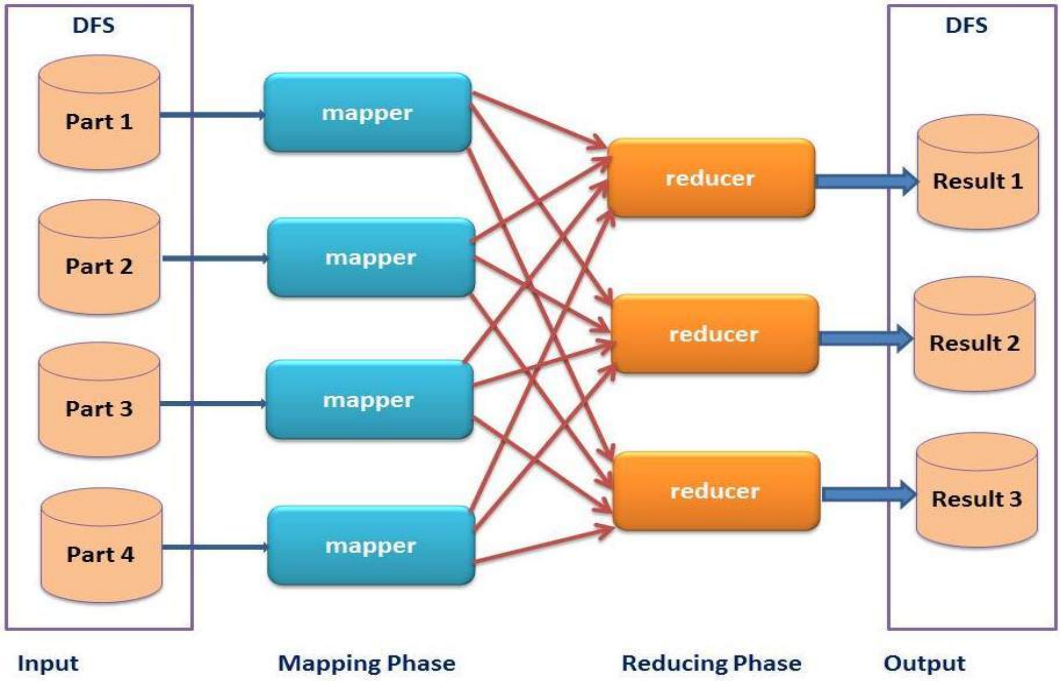


Figure (1.5) Map-Reduce Model

1.4 Quality of Service (QoS) and Service Level Agreement (SLA) parameters

Cloud computing is a large collection of on-demand and scalable virtualized resources that can access through Internet. Cloud supports virtualization technology and facilitates computing resources to be deployed as a service[95]. In general, resources are presented in the form of Virtual Machine (VM) instances. All these resources are dynamically allocated as per their Service Level Agreement(SLA)[96].In general, SLA is a contract between the user and the cloud service provider through negotiation in the following conditions [101]:

i Service availability: It refers to the ability to ensure that necessary resource must be accessible within an organization's IT infrastructure even when there is a disruption.

ii Service deadline: It ensures the penalty free service delivery to the user.

iii Consistency Service: It refers to stability of the service is expected of all customers at all times. Consumer always wants peace of mind and no unpleasant surprises.

iv Security Service: It evaluates the several factors like, data encryption capability, privacy controls, maintenance and management controls etc.

v Network congestion: It is a biggest problem that reduce the Quality of Service (QoS) agreement. It effects transfer time, packet lost and blocking of new connections.

vi Performance limit: Decrease of work permeability.

In the cloud environment, each user has apparent definition of SLA constraints and flexible conciliation procedures that can increase the trustworthiness relationship between user and provider. There are various issues and challenges in providing services to users in the cloud computing system. Deploying cloud services are not trivial or straightforward task. Following are list of obstacles that affect QoS:

- ◇ Unavailability of Services
- ◇ Fair Transaction guarantee
- ◇ Poor resource allocation
- ◇ Performance Unpredictability
- ◇ Lack of data integration services
- ◇ Limited bandwidth
- ◇ Security and migration concerns

1.5 Research Motivation

- ⋆ A novel framework
- ⋆ Well balanced self-organizing approach

- ⋆ Efficient resource discovery and invocation
- ⋆ Increased dynamicity
- ⋆ No need of extra control and external help
- ⋆ Ability to reconstruct / rejoin and remove parts of the system without human intervention
- ⋆ Reasonable computational complexity as compared to existing approaches

1.6 Thesis Organization

Chapter 1 presents the evolution of various computing paradigms, cloud service model, cloud-oriented storage systems and their challenges and goals. Further, it describes enabling technologies for cloud storage and service level agreement(SLA) optimization parameters. The chapter culminates with elaboration of thesis organization and contribution. The rest of thesis is driven based on objectives.

Chapter 2 elaborates the extensive literature reviews based on prevailing models and applications. Comparative analysis of 20 well-known cloud-oriented storage systems and 6 most popular service providers have been explained in this chapter. It presents existing meta-heuristic-based optimization approaches, self-organizing storage models and data integration services for cloud-oriented storage management. Research gaps, problem formulation and research objectives have also been identified in this chapter.

Chapter 3 explains the proposed DAIS framework. DAIS framework has four major components- Adjacency COS Overlay Topology (ACOT), Cloud Storage Resource Discovery (CSRD), COS Integration Service (CIS) and Cloud Usage Monitor (CUM). Each of these four components have been explained thoroughly in this chapter. The various architectural diagrams of DAIS framework and its components has also been visualized in well manner.

Chapter 4 describes overall design and implementation of DAIS framework. Various algorithms have been analyzed and designed in this chapter. Each algorithm has been thoroughly computationally analyzed and evaluated its searching cost and time. A novel swarm-based meta-heuristic algorithm named Generalized Ant Colony Optimizer (GACO) has been proposed and well designed in this chapter. GACO has tested on 17 well-known benchmarks test functions and compared with three most popular algorithms. Apart from this, the integration of data from different sources has been done using this Map-Reduce (MR) algorithm in this chapter.

Chapter 5 exhibits the testing benchmark and results of DAIS framework. The proposed framework is tested in Hadoop 2.7 and Java 1.8.0 environments for all concerned experiments. WAN emulator has also been used to find the impact of dynamicity of DAIS framework. Several experiments including bandwidth evaluations (load vs. throughput and Delay vs. throughput), dynamicity and scalability evaluation, SLA optimization and energy efficiency evaluation have been conducted in this chapter. All test results are well explained and highlighted in different tables as well as visual graphs.

Chapter 6 provides concluding remarks on the thesis; at the end future scope of work has been discussed.

1.7 Thesis Contributions

This thesis makes following research contributions:

i This thesis presents extensive literature review on cloud-based storage system, large-scale storage capability, meta-heuristic optimization approaches, self-organizing data storage models and integration services. It highlights the existing techniques/tools, key challenges, comparative analysis.

ii This thesis proposes a novel framework namely Dynamic Access and Integration Services (DAIS) framework in cloud computing environment. Its main aim to

achieve high level dynamicity, efficient data integration from heterogeneous platforms and enhance the performance of cloud-based storage systems.

iii Adjacency COS Overlay Topology (ACOT) has been proposed to achieve the self-organizing capability for large-scale cloud-oriented storage nodes. Four algorithms have been designed for ACOT: *Routing and naming mechanism, Joining new COS, Leaving COS, Overlay COS node traversal*.

iv Design and development of a discovery module named Cloud Storage Resource Discovery (CSRD) for DAIS framework.

v A novel Cloud-Oriented Storage Integration Service (CIS) model has been designed in this thesis. This model can integrate structured,semi-structured and unstructured data that are generated from heterogeneous platform.

vi Highlights a swarm-based meta-heuristic algorithm namely GACO to optimize the SLA violation cost for dynamic execution of cloud services.

vii Implementation and result analysis of proposed DAIS framework in Hadoop 2.7 running on Java 1.8.0. The experimental results show the proposed framework perform better in most cases.

viii The statistical analysis (trial and error approach) of experimental results of GACO optimization algorithm has been conducted in order to assess the 0.05 level of significance.

Chapter 2

Literature Review

Cloud Computing has emerged as an incredible platform which provides Internet computing services. Cloud-Oriented Storage as a Service (COSaaS), a cloud storage service model is one of the significant emerging field in which user can get access storage space in the form of Virtual Machines (VMs). Efficient storage management is a challenging task to optimize the available resources and enhance better performance for COSaaS. The purpose of this chapter is to present comprehensive literature review on cloud-oriented storage systems, large scale data management capabilities and meta-heuristic approaches for storage management. Comparative analysis of various cloud-oriented storage system's features and their capabilities have also been done in this chapter. Comparison of different storage types like cloud-oriented, object-based, hard drive, optical disk, etc have been highlighted. At the end, this chapter conclude with research gaps, problem formulation and research objectives.

2.1 Cloud-Oriented Storage (COS) Systems

Cloud storage is a cloud computing model in which data is stored on remote servers accessed from the Internet, or "cloud." It is maintained, operated and managed by a cloud storage service provider on a storage servers that are built on virtualization techniques. This subsection highlights some major cloud-based storage systems and theirs features. Comparatively analysis have also been done at the end of this

subsection.

2.1.1 Neo4j

Neo4j [56] is a first graph-based cloud platform database. It discovers how graph database can help to manage and query highly connected data. Neo4j's graph platform creates a bridge between application analytics and enterprises [77]. It is one of the first approach to reduce the gaps between application and business users. Following are some important features:

- Graph-based database
- High level data integration
- Graph analytics
- Supports graph query language
- Visualization
- Supports enterprises architecture

2.1.2 Virtual Inter-networking on Overlay Infrastructure (VIOLIN)

VIOLIN [82] is an isolated virtual networks created on top of an overlay infrastructure (e.g., PlanetLab). Entities in a VIOLIN include virtual end-hosts, routers, and switches implemented by software and hosted by physical overlay hosts. The features of VIOLIN include: (1) having own IP address and creates a "virtual world". (2) more flexible for data manipulation like addition, migration, modification, configuration or deletion. (3) provides value added network services. (4) support legacy applications. Jiang X and Xu D [82] have designed and implemented a prototype of VIOLIN in Planet-Lab. Due to its excel properties, VIOLIN became more popular platform for the execution of legacy applications APIs and high-risk network experiments.

2.1.3 Cluster-on-Demand (COD)

Demand for data-intensive applications is putting pressure on the enterprise to give more storage capacity at a lower cost [125]. In general, the cost, time and effort required for creating a new software or replicating errors. Sometime these may be too much burden. A better way that would be the ability to create a cluster quickly from available resources, configure it as per requirement, and use it for particular time duration and lastly release the resources back if task is performed. Several types of cluster can be implemented, including Hadoop, HPC, Spark etc. It provides better support and quality assurance.

2.1.4 Amazon Elastic Compute Cloud (EC2)

EC2 [185] is a cloud computing platform developed by Amazon.com Inc. It provides scalable application deployment on rent via web service. EC2 has capability to control over geographical locations of service instances and optimize the high level of redundancy. There are following type of service instances offered: general purpose, on demand, compute and storage optimized etc. It has many features: automated scaling, elastic ip address, tier free, reserved instances, dynamic instances etc.

2.1.5 Eucalyptus

Eucalyptus is an open source cloud platform to make compatible hybrid cloud computing environment [133]. It was originally developed by Eucalyptus Systems, Inc. Eucalyptus stands for Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems. Some major components of eucalyptus are- cloud controller, storage controller, cluster controller, measurement console, AWS compatible APIs, VM broker, node controller and scalable object storage. Instances, images, IP-addressing, access control are important terminologies. Auto-scaling and elastic load balancing features have added in eucalyptus 3.3 which is knows as new feature of AWS compatible tool [123]. Following are benefits of eucalyptus:

- Operational efficiency

- Increase storage reliability
- Flexible infrastructure environment
- Support hybrid capability
- Improve performance
- Enhance dynamicity and scalability
- Reduce unnecessary expenditure

2.1.6 Comparative Analysis

This subsection highlights comparative analysis of 20 most popular cloud-oriented storage systems and their capability and development environment. It also considers some important features like scalability, durability, consistency and storage types. Table 2.1 lists the salient features of all 20 well-known cloud-oriented storage systems. Today's, there are many organizations that are offering cloud storage service. Most of the providers have used common platform and approaches. Table 2.2 highlights some well-known cloud storage providers and their features.

Cloud computing and large scale storage systems are conjoined. In general, large scale data refers as "big data" which provides commodity computing to execution of multiple data-sets in cloud computing environment. The usage of large scale cloud oriented shown in Figure 3.6. Cloud computing also serves as a storage service model in various aspects. A cloud-oriented database provides the mechanism to store and access large volume data of distributed environment. The features of cloud-oriented storage system include self-organizing, durability, consistency, storage type and developer. There are various storage mediums are integrated with Network Attached Storage (NAT) [65][28] with cloud cloud computing . Table 2.3 shows the comparison among various storage mediums in cloud environment.

Scalability is a process of measuring the storage to handle increasing data volume in a systematic way. Scaling is one of the critical part of cloud storage systems. Table 2.4 shows the comparison of different storage characteristics of cloud environment.

Table (2.1) Comparison of cloud-oriented storage systems

Cloud Database	Storage capabilities							
	Self-organizing	Scalability	High-level availability	Durability	Consistency	Storage Type	Developer	Ref.
Neo4j	Yes	High	Yes	-	Yes	Graph-based	Neo4j, Inc.	[56]
Hbase	Yes	Medium	Yes	Yes	Yes	Column	ASF*	[169]
Eucalyptus	-	High	Yes	Yes	Yes	Document	Eucalyptus Sys, Inc.	[133]
BigTable	Yes	High	Yes	Yes	Yes	Column	Google	[31]
SimpleDB	Yes	High	-	-	Yes	Document	Amazon	[12][45]
DynamoDB	No	High	Yes	Yes	Yes	Key-value	Amazon	[44]
MongoDB	Yes	High	Yes	Yes	Yes	Document	MongoDB Inc.	[37]
Cassandra	Yes	High	Yes	-	Yes	Column	ASF*	[32][169]
Redis	No	Medium	Yes	-	Yes	Key-value	Redis Labs	[37]
PNUTS	No	Medium	-	Yes	-	Column	Yahoo!	[40]
CouchDB	Yes	High	Yes	Yes	Yes	Document	ASF*	[40]
FusionTable	No	High	-	Yes	-	Column	Google	[66]
Xeround	No	High	Yes	No	-	MySQL	Xeround	[13]
Cloudant	Yes	High	Yes	Yes	-	Key-value	IBM	[21]
AWS	Yes	High	Yes	Yes	Yes	Document	Amazon	[12]
ClearDB	No	High	Yes	-	Yes	Column	ClearDB Inc.	[45]
SQL Azure	No	Medium	Yes	Yes	Yes	NoSQL	Microsoft	[94]
Hadoop	Yes	High	-	Yes	Yes	DFS†	AFS*	[169][43]
Voldemort	No	High	Yes	Yes	Yes	Key-value	LinkedIn	[59]
Apache Hive	Yes	High	Yes	Yes	Yes	Column	ASF*	[78]

* Apache Software Foundation

† Distributed File System

Table (2.2) Comparison of cloud storage features of some popular providers

Provider	Storage features							
	RDBMS	NoSQL	Data Sources	Data import environment	Streaming processing	Data processing	Large data storage	Map Reduce (MR)
Microsoft[1]	SQL Azure	Table storage	Windows Azure	Network	Streaminsight	Hadoop	MSSQL Azure	Hadoop on Azure
Amazon[2]	MySQL or Oracle	DynamoDB	Public Datasets	Network	-	Hadoop	Simple Storage Service (S3)	Elastic MR
Google[3]	Cloud SQL	AppEngine	Sample datasets	Network	Search API	Prediction API	Google Cloud Platform	AppEngine
Cloudera[4]	PostgreSQL	Apache Accumulo	Public Datasets	Network	Apache Spark	Hadoop and Oryx	Apache Kudu	Mr Yarn
Yahoo![5]	MySQL	DynamoDB	Public Datasets	Network	-	Hadoop	Pnuts	Hadoop
Oracle[6]	Oracle RDBMS	on-Cloud	Public database	Network	RESTful API	IoT	Oracle Cloud Platform	HDFS

[1] "https://azure.microsoft.com/en-in/services/storage"

[2] "https://aws.amazon.com/products/storage"

[3] "https://cloud.google.com/storage"

[4] "https://www.cloudera.com/products/open-source/apache-hadoop/apache-kudu.html"

[5] "https://www.yahoo.com/tech/best-cloud-storage-services-191500344.html"

[6] "https://cloud.oracle.com/storage"

Table (2.3) Comparison of different storage mediums

Storage medium	Objectives	Advantages	Disadvantages	Ref.
Cloud-oriented storage	To provide as a service model and on-demand access	More flexible storage system; useful for small company that don't have sufficient storage infrastructure	Lack of trust and security	[174]
Object-based storage	To store data in the form of variables and objects	Easy to access and scale the data; highly secure because physical location cannot track easily	Tracking indices are more complex	[119]
Optical storage	To store data in different angles	High speed and cost effective	More complex ability to re-produce in different optical disk	[70]
Solid-state storage	To store data in various peripheral devices; to store upto 2TBs data	Frequent access; fast movement; start-up time is very less	More expensive for large-scale storage	[143]
Hard Disk storage	To store data upto 10 TBs	high speed start-up; storage cost per bit	High latency time for reading data	[91]
Flash Storage Drive	It stores data in flash memory with integrated USB interface. Size of this storage device from 8GBs to 2TBs	It is faster as compare to optical storage	Limited number of write and erase cycles; no facility to write protection	[33]

Table (2.4) Comparison of scalable data storage characteristics in cloud computing environment

Storage characteristic	Advantages	Disadvantages	Ref.
Key-Value	Scale ultra-large storage systems; no limit; simple data format; flexible schema	Optimization problem for single key value; needed parser for multi values storage	[99]
DBMS	Fast data processing system; query-based data manipulation; deal with structured and semi-structured data	Not suitable for ultra-large scale data processing; difficult to store unstructured data	[144] [114]
Document	Store structured, semi-structured and unstructured data; highly scalable	Less reliable; Not more secure	[146]
Column	Store each database table column separately; fast reading and updating; support large-scale data intensive applications	Low performance; data integration problem from heterogeneous platforms	[1]
Graph	High level data integration; supports graph query and analytics; deals with all types of data	Low performance; more complex and security concerns	[56]

2.2 Scalable and Self-Organizing Cloud-Oriented Storage Model

Cloud computing must be highly scaled data storage systems and it should transparent to all users also. Scalability can be achieved by two important aspects:

i Horizontal scaling - it connects multiple clouds to integrate and creates single logical cloud. A cloud with a computational affinity might be accessed data from heterogeneous platform. The relationship between cloud data integrates with compute-data is more crucial.

ii Vertical scaling: It improves the cloud storage and computing capacity by magnifying the existing systems and enhancing the communication bandwidth [16]. It has ability to increase the current resources by adding a new database, processing power, etc. But, it is limit by the fact that can only get as big as the server size.

Self-organizing is an autonomous concept designed to make more efficient resource invocation and run-time estimation in distributed environments [97]. It describes the system's ability to organize its components in the work structure without the need of external help. Self-organizing systems have the ability to manipulate system components without the need for human intervention. In general, self-organizing system has been classified into different overlay structures- 1)Fully decentralized, 2) Partially centralized 3) Mess Overlay 4) Tree Overlay and 5) Hybrid Overlay [8] [7]. Each structure having its own rules and regulations. Now, self-organizing becomes one of the key features for cloud-oriented storage systems. This thesis has summarized some literature review based on self-organizing storage models which are highlighted in Table 2.5.

2.3 Large Scale Storage Management Capabilities

There are many challenges faced by large scale data analysis, including data transfer, integrating data, managing data, controlling data, standardizing data, etc [150].

Table (2.5) Self-Organizing storage model in distributed/ cloud environment

Prevailing Model	Approaches	Objectives	Time Complexity	Environment	Ref.
Proactive Dynamic Secure Data Scheme (P2DS)	Attribute-based Semantic Access Control Algorithm	Constraints data access and deal with dynamic threats.	–	Cloud Computing	[141]
SOPSys	M-ways balanced tree	Reduce the network join and discovery cost.	$O(logN)$	Decentralized Peer-to-Peer Network	[79]
SSA_G	Election algorithm	Resource optimization in dynamic environment and improve the system performance	$O(N^{1/2})$	Cloud Computing	[104]
S-TREE	Supervised learning	Illustrate the data clustering and compression	$O(logN)$	Cluster Computing	[29]
Adaptive Resource Allocation Model	Ant-based control algorithm	Adaptive resource allocation in dynamic environment	$O(logN^2)$	Distributed Computing	[14]
SOS Cloud	Bio-inspired algorithm	Reduce the Service Level Agreement (SLA) violation cost.	–	Cloud Computing	[30]
Safari	Hybrid routing algorithm	Scalable ad hoc network routing and integration services.	$O(logN)$	Ad Hoc Network	[57]
Cloud Cell	Fuzzy-logic-based decision framework	Improve the storage capacity and mobility performance.	$O(2^{N+1})$	Cloud Computing	[11]
ecoCloud	Bernoulli trials	Virtual Machine (VM) consolidation Assignment and migration of VMs.	–	Cloud Computing	[112]
CCPS	M/M/1 queuing model	PC cluster based storage system.	–	Cloud Computing	[184]
AETOS	Tree Overlay topology	Improve the performance and enhance runtime estimation strategies.	$O(logN)$	Distributed Computing	[140]

Definitely, some new solutions that are readily available to address all such kind of challenges. Such solution includes their large scale data management capabilities. Cloud computing enables businesses to combine and match services from various cloud providers and chooses which is best suited to the requirements of a particular use case or workload. Today's many cloud storage management systems are available. Some major cloud storage management capabilities are explained bellow:

a. **Hadoop Distributed File System (HDFS):** The Hadoop framework is the most popular open source developed by Apache and Yahoo! foundation. It provides a framework to manage distributed storage systems and large-scale data processing. It is designed to provide local computing and storage from single servers to thousands of machines. A wide variety of organizations use for research, analyzing data and production. It enables the parallel workloads on same data at same movement. Hadoop is more flexible framework to analyze and store unlimited data. Store, process, discover, model and serve are some important terms associated with Hadoop. It frequently integrates with other applications or platforms and easily transfer/ load data. Hadoop has ability to process any kind of data like structured, semi-structured and unstructured. As per literature survey with some researchers and users of Map-reduce framework were given their opinion. Dean and Ghemawat [43] have noted that Map-reduce is one of the better tool for handling large scale data storage in cloud environment. Moreover, it gives the better solution for solving a complicated storage problems which are globally distributed. Blanas et al. [23] have given a performance report of various distributed join algorithms using Map-reduce framework. Mainly authors have used two join algorithms- broadcast join and re-partition join. User can easily map and reduce process over large-scale data set. Gu and Grossman [71] have performed an experiment over Map-reduce and some important results were reported in four aspects. These aspects are- load balancing, data locality, fault tolerance and data streaming.

b. **Cloudera manager:** It is one of the most popular distribution of Hadoop which makes faster processing streaming large data [117]. It can easily allocate the resources and reduce the SLA violation cost. The sole motivation of cloudera manager is making Hadoop easy. It consists following features:

   - Customizable monitoring and reporting
   - Automated configuration and deployment
   - Zero downtime maintenance
   - Effortless and robust troubleshooting
   - Multi-tenant management and visibility
   - Extensible integration
   - Built-in predictive and proactive support

c. **NoSQL:** It stands for "not only SQL" which is working with large set of geographically distributed data[74]. It accommodates all types of scalable storage models including key-value, column, document, graph-based. It is specially design for the purpose of large-scale storage clustering in cloud computing environment. In general, NoSQL avoids the fixed relational schema. Some of most popular NoSQL databases already explained in previous section and their comparison shown in Table 2.1. Table 2.4 has highlighted advantages and disadvantages of different scalable storage models.

d. **Workflow:** Any organization can reduce their cost with cloud data integration ability to extract, combine, transfer data frequently [170]. A workflow can arrange all the task in a particular order and define some objectives in cloud environment. It improves the job execution process and increases the performance. There are thousands of applications and users are running on different cloud servers. Obviously, these servers must be designed to working fast, reliable and secure. The cloud computing is a type of service model approach. Workflow gives solutions within the cloud that come with this service model.

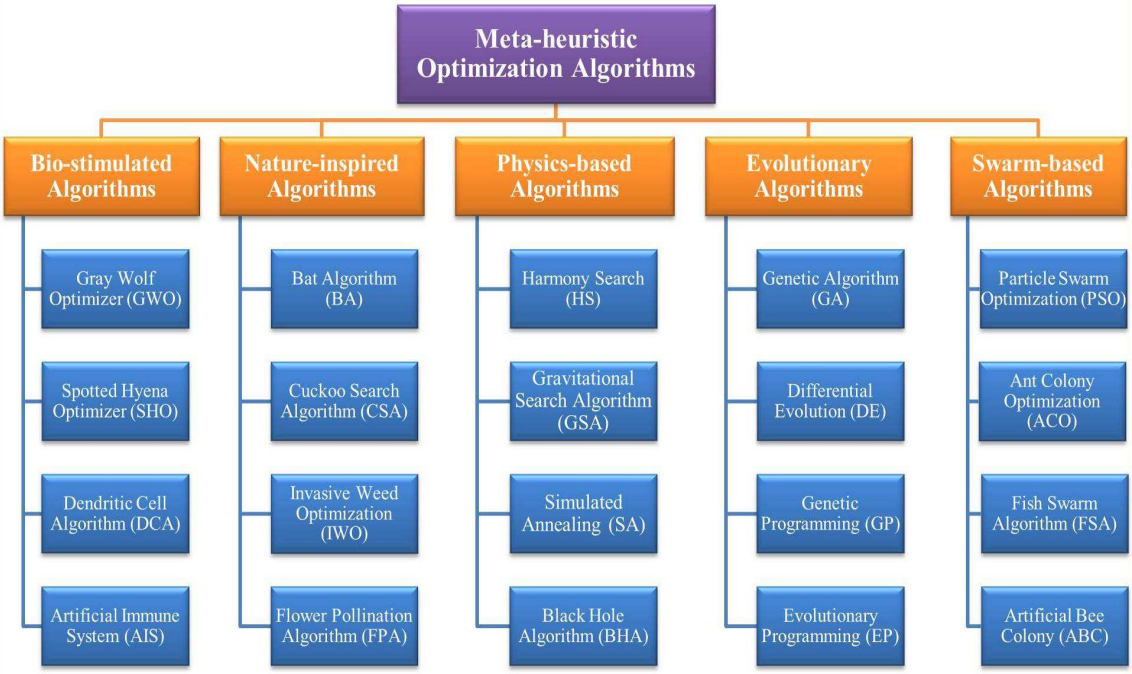


Figure (2.1) Taxonomy of metaheuristic optimization algorithms

2.4 Meta-heuristic-based Optimization Approaches for Cloud Storage Management

This section presents the comprehensive literature review about meta-heuristic taxonomy, meta-heuristic algorithms in cloud storage management. In last four decades, the increase in complexity of large scale computing problems has given arisen the need of better meta-heuristic approaches. Generally. meta-heuristic algorithms have been used for finding optimal solution for large scale computing problems and became more popular due to their reduced complexity and increased efficiency as compared to other traditional approaches [129] [135][176]. Mostly, meta-heuristic algorithms are inspired by animal/ insect behavior, nature, evolutionary concepts and natural phenomena types. Based on inspiration of meta-heuristic techniques, it can be classified into five different categories: Bio-stimulated, Nature-inspired, Physics-based, Evolutionary and Swarm-based. Figure 2.1 shows the taxonomy of various meta-heuristic optimization algorithms.

The first category is bio-stimulated meta-heuristic optimization algorithms. It

is based on foraging and hunting behavior of wild-life animals and sea creatures. Some recently developed and popular bio-inspired algorithms are Grey Wolf Optimizer (GWO)[121], Spotted Henya Optimizer (SHO)[47], Dendritic Cell Algorithm (DCA)[68] and Artificial Immune System(AIS)[88]. All these meta-heuristic algorithms have their own substantial ability to apply the preeminent efforts and find the optimal solution.

The second category of meta-heuristic is nature-inspired optimization algorithms. Such metaheuristic algorithms include Bat Algorithm (BA)[180], Cuckoo Search Algorithm (CSA)[62], Invasive Weed Optimization (IWO)[86] and Flower Pollination Algorithm (FPA)[182]. These algorithms are based on natural system[60].

The third category of metaheuristic technique is physics-based. It is typically inspired by physical rules. We can also say this category is different form of swarm-based algorithms. Undefined usual of search agents are communicating and stirring from one point to another due to physical rules. Some popular algorithms are Harmony Search (HS)[64], Black Hole Algorithm (BHO)[75], Simulated Annealing (SA)[167], Gravitational Search Algorithm (GSA)[179].

The fourth one is evolutionary type meta-heuristic algorithms. It is based on natural selection process concepts. Here, the population tries to continue building on the fitness measurement in their environment and applied best effort to get best optimal solution [69] in the search spaces. This method has no idea about their adaptive nature and fitness. Some of well-known and recently developed evolutionary-based algorithms are like, Deferential Evolution (DE)[177][190], Genetic Algorithm (GA)[171], Evolutionary Programming (EP)[183], and Genetic Programming [92].

The last category of metaheuristic algorithms is swarm-based which is based on mutual behavior of insects, particles and various social-creatures. Artificial Bee Colony (ABC)[84] [85][93], Fish Swarm Algorithm (FSA)[132], Particle Swarm Optimization (PSO) [87][18][153][58], and Ant Colony Optimization (ACO) [50][52][188][96] are some popular swarm-based metaheuristic optimization algorithms. Some major advantages of swarm-based algorithms are:

☐ Swarm-based optimization technique has less number of parameters to adjust

them-self.

☐ It does not have a substantial number of operative agent.

☐ It collects different facts about their search spaces till last iterations

☐ Easy to implement and simulate

The taxonomy of meta-heuristic mentioned in Figure 2.1 is not unique. There may be possible, that some of algorithms have been included in other category.It completely depends on what the emphasis is and what can be the perspective. In this context, we are describing some special remarks which reveal the diversity in metaheuristic algorithmic design. For example, ant has more than 12,000 known species and each has unique features that help them to survive their environment. This does not mean that the researcher will be developed 12,000 different algorithms for ant. Therefore, it should not be fitting that algorithms begin to develop by red ant algorithm,fired ant algorithm, harvest ant algorithm, hony-pot ant algorithm etc. A formula for an efficient metaheuristic optimization algorithm schematic design as proposed by Yang [181] is given below:

$$z^{t+1} = A\left\{z^t, p(t), r(t)\right\} \tag{2.1}$$

Here, algorithm A is nonlinear mapping from a current solution z^t to better solution z^{t+1}. In this process, uses some dependent parameters $(p_1, p_2, p_3, ..., p_k)$ and some random variables $(r_1, r_2, r_3, ..., r_m)$. Both may be time dependent and can be adjusted if needed.

This section has summarized some related work done using meta-heuristic techniques for dynamic execution of cloud services. Over the last three decades, metaheuristic algorithms have been developed as a dominant technique to solve the various optimization problems [49][55][154]. The Ant Colony Optimization (ACO) metaheuristic algorithms [53][55] have applied in different domains such as supply chain management [157], a dynamic scheduling problem [108][158], communication network design [48], multi-objective problem [10][63], clustering problem [149], optimal truss

design [110] etc. Table 2.6 shows the details of recent applications including objectives, algorithms, simulation environment and references. In this perspective, IT experts are concentrated on the exploration technique. Thus, metaheuristic algorithms which have global impact, because they don't fall into the trap of local optima and ensure the convergence of solution.

2.5 Cloud-Oriented Storage Integration Services

Cloud computing services have gained a place in the enterprise based on its low cost and unlimited virtual scaled data to meet the user needs. It is completely natural for a new technology; Since enterprises has serious or important responsibilities on the architecture to prove significant resources into real resources. But, now days many organizations have gained experience in the cloud, therefore industries are ready to take the next step. A legacy enterprise has integrated cloud storage services with the various infrastructure. There are many such organizations available, who have spent just getting someone else's port in the data center [17].

One of the company named as "Cast Iron Systems", that is to reveal the "Omni Connect system", and work as an interface between internal systems and different cloud service models. It is a unique approach which provides cloud-oriented storage systems. In contrast to local management and data integration have greater insight into organization data structures. The service provider wants to integrate the applications, if the vendor break their lock-in platforms[12].

Another company growing towards cloud integration is "Infomatica" [19]. With cloud integration, the complexity of adding data sources and goals has been overcome. Consequently, when cloud-based Big Data Deployment combines with the Cloud Integration solution, it can result in more time and cost savings and projects can be acquired faster than the ground. Company has released the cloud services assortment, which includes a large scale data replication and synchronization processes. It enables a multi-phase data integration service from internal as well as external sources.

In [17], authors have determined the topological relationship between two fuzzy spatio-temporal data integration tree. In this model, XML twig pattern approach has

Table (2.6) Meta-heuristic algorithms in cloud computing

Applications	Objectives	Algorithms	Simulation environment	Ref.
Dynamic resource allocation	To achieve maximum resource efficiency and scalability in speedy manner. Effective distribution of tasks among virtual machines.	Grey Wolves Optimizor	Cloud	[127]
Partly dependent tasks (PDT) scheduling	Find the optimal value for PDTs scheduling and reduce the unnecessary SLA penalty cost.	Genetic Algorithm	Cloud	[175]
Workflow scheduling	To meet the quality of service and minimize the SLA cost.	Genetic Algorithm	Cloud	[36]
Task scheduling	Optimized the scheduling algorithm for computation grid	Ant Colony Optimization	Grid	[159]
Load balancing	Improve the overall response time and reduce the queuing wait time.	Honey Bee Behavior	Cloud	[93]
Energy-aware scheduling	Reduce the energy consumption and dynamic scaling during execution of application.	Evolutionary Algorithm	Distributed	[186]
Load balancing	To achieve dynamicity and scalability during task distribution.	Hill Climbing	Cloud	[124]
Job scheduling	Reduce the cost and long termination time.	Simulated Annealing	Cloud	[2]
Energy-aware resource scheduling	Improve the response time and energy consumption. Reduce the SLA resource cost.	Genetic Algorithm	Cloud	[120]
Load balancing	Optimize the resource cost related to load balancing.	Particle Swarm Optimization	Cloud	[136]
Tasks distribution	Improve the average response time and enhance the performance and efficiency of nodes.	Particle Swarm Optimization	Distributed	[187]
Text document clustering	To find the best features, to enhance the effectiveness of the text document clustering approach and also improve global search ability and convergence rate	Krill Herd Algorithm	Distributed	[5]
Information retrieval systems	Explore the problems embedded to information retrieval, Find the solutions to increase the efficiency of the optimal information retrieval as per user needs.	Genetic Algorithm	Distributed	[6]
Text clustering analysis	Hybrid text clustering analysis using improved krill herd algorithm to obtain promising and precise results.	Krill Herd Algorithm	Distributed	[3][4]

been used to find the relationship consistently and reduce the execution time.

The data integration service is important if the technologies are to conform and its promise as geographically distributed resources pool which capable of high scalability and proper load balancing. On-demand cloud services are definitely on the downside, but it can be changed the track if it is found. Innovative is definitely prevailing on the cloud, but if it does not show up in some old-fashioned practical functionality it will come to nothing. Cloud application integration is dependent on synchronization, data migration, quality and replication. Most of the cloud integration tools are based on Pass of SaaS providers. Some state-of-the-art cloud data integration tools have been summarized in Table 2.7.

2.6 Recent Research Work on Cloud Storage

Currently, there are so many researchers are working on cloud-oriented storage systems in order to solve the various issues related to large-scale storage. In this section, thesis has summarized some recent researches on cloud storage systems which are reflected in Table 2.8.

2.7 Research Issues

Cloud computing plays an important role in sharing data or moving information from one place to another. Storage of user's data in the cloud server which provides high quality and more enjoyable services. It shares data to the users in efficient way and gives quick response to the users. Despite this, there are many flaws in cloud computing. All these flaws move towards research itself. In this section, some major research issues has been addressed related to cloud storage systems in cloud computing.

Table (2.7) Cloud-oriented storage integration tools

Tools	Services	Application model	Objectives	Ref.
CloudFuice	SaaS	Web engineering model and mashup development	Task-based execution approach that allows parallel execution of of data-flows in the cloud	[165]
iProX	IaaS	Hyper-converged architecture	To enable the computing, storage requirements using unified and fully-virtualized resources pool. To achieve high scalability	[111]
SnapLogic	PaaS	HR, CRM	Provide both data integration as well as application multi-point connection strategies. Handle a more multifarious set of use-cases	[138]
WorkDay	SaaS	Legacy application of HR, CRM	To provide pre-built connectors, intuitive user interface and reporting structure	[73]
Adeptia	SaaS	B2B, SOA, Social Network	To provides application and data integration services that include connections, data mapping, conversion and extractions	[83]
Scribe	PaaS	Mobile applications	To improve application life span. To increase the cloud adaption in the market segment.	
Nginx [105]	PaaS	Microservice model	Integrate API broker and API getways and provides microservice users	[100]

Table (2.8) Recent Research Work on Cloud Storage

Author(s)	Prevailing Model	Objectives	Target Metrics	Ref.
E. Torres et al. (2018)	Hierarchical model-based strategy	evaluating availability and performance-related metrics for private cloud storage services.	Performance and Availability	[166]
Suji Gopinath and Elizabeth Sherly (2018)	Dynamic Replica Management	To reduce the unnecessary storage wastage; To maintain an optimal number of replicas for each node based on its popularity.	Dynamicity and Availability	[67]
Y. Yan et al. (2018)	Lattice and Bloom filter methods	To improve the privacy of signature information; to establish a safe, efficient and cost effective cloud storage system.	Data Security and Integration Services	[178]
E Lopez-Falcon et al. (2018)	Adaptive Encrypted Cloud Storage Model	To minimize the data access and reduction of data redundancy.	Reliability, Integrity and Dynamicity	[107]
W. Chenggang et al. (2019)	ANNA	It is designed to overcome the narrow cost-performance limitations typical of current cloud storage systems; to balance the service level objectives around the cost.	cost-effective, latency and fault tolerance	[173]
E. Mugisha et al. (2019)	Multi-Cloud Storage Architecture (MCSA)	To ensure the stability in storage costs with best practice to guarantee data accessibility failure recovery	Storage Security, Reliability and Failure recovery.	[128]
C. Yong et al. (2019)	TAILCUTTER	to optimize the tail latency while meeting cost constraints given by application providers. Extensive evaluation of large data sets across the globe.	Latency minimization under cost constraints,	[42]

2.7.1 Dynamicity and Scalability Issues

A large number of organizational units that share a shared computing infrastructure implement their special blend of each task, and their overall demand and patterns of arrival are more dynamic. Such environment must be characterized by a large number of on-demand requested resources within short intervals with high variation. There will be a number of scheduling results, including the dynamic Virtual Network Function (VNF) [142] rapid task scheduling decision, the amendment of the previous assignment decisions and the long-running jobs include the resident of Dynamic Resource Request which, with the help of time, predicts the intervention of resources difficult to predict. Some issues include: SLA violation, poor prediction, resources constraints and workload heterogeneity and variability. All these issues must be taken care.

2.7.2 Data Integration Issues

Cloud data integration provides very crucial role if we deliver cloud services to business [26]. It is more challenging and time consuming process to coordinate different resources, infrastructures and applications. Before migrating a particular cloud domain to other environment; it creates a plan for how data integration service will be assessed the right technology for the cloud storage systems. There are some critical issues related to data integration include: Moving data tracking [16], data migrating, lack of performance, firewall mediation, maintenance and up-gradation [109]. With little planning and some good techniques, data integration issues get resolved quickly, and the cloud storage system becomes so obvious.

2.7.3 High Energy Consumption Issues

Scientific computing, business applications and domains are continuous improvement demands in performance which must be taken care during deployment of cloud storage systems [163]. Reasons of high energy consumption includes equipment's cooling purpose and transferring power to whole infrastructure [160]. The energy consumed

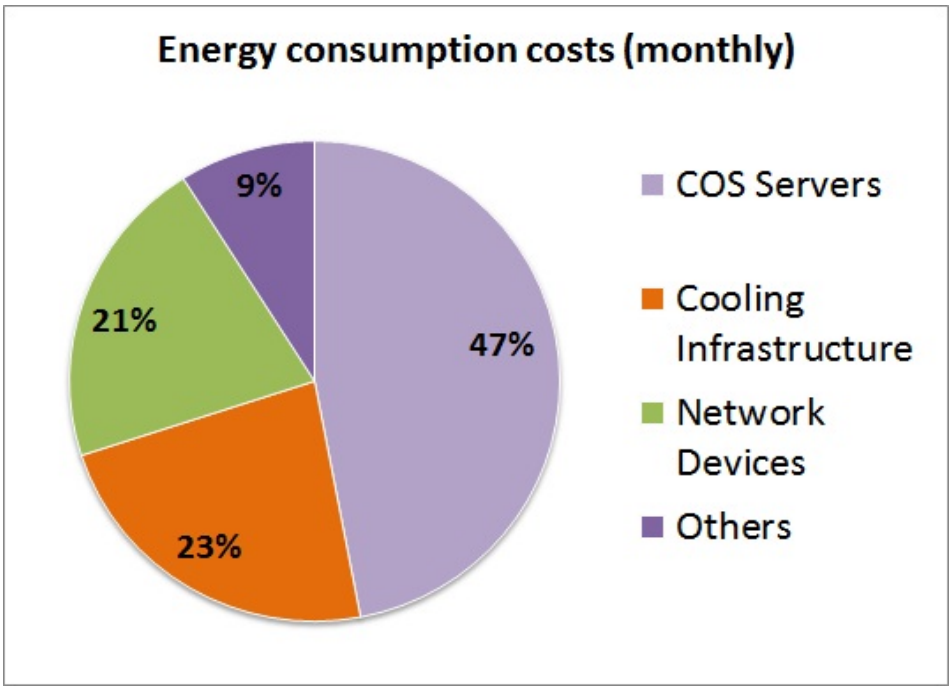


Figure (2.2) Energy consumption costs distribution

by the servers contributes to 70% of the total consumption of a cloud data center. However, such improvement has increased the energy consumption limit. According to Amazon energy consumption cost (as shown in Figure 2.2) for cloud-oriented storage server is 47%, cooling infrastructure is 23%, network devices is 21% and remaining 9% of others. Thus, the objective is to minimize the total power consumption of all the cloud-oriented storage systems. Energy consumption in storage systems are mainly calculated from server, network devices and disk storage. Nowadays even IoT devices are made in such a way that energy consumption is minimized. In [126] researchers have been developed an energy efficient multi-objective scheduling model for monitoring of IoT devices which will be optimized energy consumption. Therefore, today highly needs to develop energy efficient cloud-oriented storage models which can monitor and save excessive energy expenditure [25].

2.8 Research Gaps

Based on the findings of the literature available and trends of dynamic and scalable data access and integration service in cloud computing. Following research gaps have been observed:

i Cloud applications only focus on the processing data stored in files. However, several applications require accessing the data stored in sources other than the files [76].

ii Parallel Data Access technique does not exist much in Cloud environments [137].

iii In Cloud data is relocated into cloud services, either through database-as-a-services or applications-as-a-service. In both instances there is no any strategy for syncing that data back into core enterprise systems [168][95].

iv There are some problems in data management in Cloud environments such as, data inconsistency and integrity problems [168][113][115].

v Lack of dynamicity and scalability in data management for Cloud environment is reported. [115][46][22].

vi The problem becomes more complex when the data sources are diverse as in case of Cluster, Grid and Cloud computing environments. Because, such applications require the data from the diverse sources, demanding the effective data integration techniques, preferably in the form of web services due to the diversity in their resources [148].

vii The patterns of data integration for Clouds are more complicated within an enterprise as an enterprise needs to leverage interfaces that do not directly control [81].

viii In current scenario of web service architecture for cloud, the service discovery is simple; it is not very efficient since it requires prior knowledge of the web service, as well as the contact information for the service provider [147].

ix Many enterprises are not accounting for the costs and complexities of doing data integration between the clouds and the enterprise [147].

2.9 Problem Formulation

For this thesis the research problems have been formulated mainly based on the following points:

i Parallel data access system in cloud environment needs to be developed. The techniques to resolve the data management issues like data inconsistency and data integrity problems need to be developed.

ii Dynamicity: Cloud computing leverage Internet or Intranet to provide the users with resources they require in an abstract way as per user demands. It means whenever user demands storage space as per their requirements it should be provided. Therefore, existing technologies can be used with modifications as required.

iii Scalability: Cloud should be highly scalable and this scalability should be fully transparent to users. It allows multiple clouds to connect, integrate and work as one logical cloud. It allows improving the cloud capacity or bandwidth connecting two nodes.

iv Integration Service: There should be a uniform interface to integrate different sources of data as available in a cloud environment. The cloud computing begins to gain transaction within the enterprises; that should refocus on data integration.

The thesis targets to address all the aforementioned mentioned problems. In order to cloud-oriented storage systems all the targeted research issues, gaps and problem formulation has also been tabulated in Table 2.9. Logically, Table 2.9 pointing out major research issues, gaps and problem formulation that would lead to motivation of this research work.

Table (2.9) Research Issues, Gaps and Problem Formulation

Research Issues	Research Gaps	Problem Formulation
Dynamicity	In the cloud computing, data is relocated into cloud services, either through Database as a Service (DBaaS) or Application as a Service (AaaS). In both instances there is no strategy for syncing that data back frequently into core enterprise systems. Lack of dynamicity in the cloud service execution.	Cloud computing leverage Internet or Intranet to provide the users with resources they require in an abstract way as per user demands. It means whenever user demands storage space as per their requirements it should be provided. Therefore, existing technologies can be used with modifications as required. Cloud environment must be characterized by a large number of on-demand requested resources within short intervals with high variation.
Scalability	Lack of scalability in data management for Cloud environment. Currently, there are no proper strategy available to measure all types of cloud services.	Cloud should be highly scalable and this scalability should be fully transparent to users. It allows multiple clouds to connect, integrate and work as one logical cloud. It allows improving the cloud capacity or bandwidth connecting two nodes.
Integration Services	The problem becomes more complex when the data sources are diverse as in case of Cluster, Grid and Cloud computing environments. Because, such applications require the data from the diverse sources, demanding the effective data integration techniques, preferably in the form of web services due to the diversity in their resources. The patterns of data integration for Clouds are more complicated within an enterprise as an enterprise needs to leverage interfaces that do not directly control	There should be a uniform interface to integrate different sources of data as available in a cloud environment. The cloud computing begins to gain transaction within the enterprises; that should refocus on data integration. With little planning and some good techniques, data integration issues get resolved quickly, and the cloud storage system becomes so obvious.

2.10 Research Objectives

The following objectives of the propose works are:

i To study and analyze existing frameworks, models and policies for dynamicity and scalability of Data management system in Cloud Environments.

ii To propose a Dynamic and Scalable Data Access and Integration Service framework for Cloud environment.

iii To implement the proposed framework in Cloud environment.

iv To test and demonstrate the proposed framework in Cloud environment.

The followings are mentioned of how all the objectives of this thesis have been achieved (the publication number is in accordance with the order given in **"List of Publications"**):

i Objective 1 has been achieved and outcomes are reported mainly in Chapter 2: Literature Review. **The work of objective 1 has been partially published in Publications 1, 2, 5 and 6.**

ii Objective 2 which is for proposed DAIS framework. In Chapter 3, all the information related to the DAIS framework have been explained properly. **The work of objective 2 has been fully published in Publication 1.**

iii Objective 3 has been achieved in Chapter 4: Design and Implementation of proposed DAIS Framework. **This work has been partially published in Publications 1, 3 and 4.**

iv Objective 4 is fully concerned with testing and results analysis of proposed work. The test benchmarks and result analysis have been reported in Chapter 5: Testing and Result Analysis. **Publications 1, 3 and 4 have been partially covered the work of Objective 4.**

2.11 Conclusion

This chapter has been conducted comprehensive literature survey in the field of cloud storage systems. This survey has been conducted on different aspects including cloud-oriented storage systems, self-organizing and scalable storage models and storage integration approaches in cloud computing environment. Despite above, this chapter has also been consolidated some recent researches in the context of dynamic and scalable storage approaches in cloud environment. The major research issues, gaps, and problem formulation are also highlighted on the basis of literature review. The section 2.10 mentions all four objectives of this thesis.

The chapter 3 has proposed a framework named Dynamic Access and Integration Services (DAIS) and explained all its components systematically.

Chapter 3

Proposed Dynamic Access and Integration Services (DAIS) Framework

This chapter explains the proposed DAIS framework. DAIS framework has four major components-Adjacency COS Overlay Topology (ACOT), Cloud Storage Resource Discovery (CSRD),COS Integration Services (CIS) and Cloud Usage Monitor (CUM). Each of these components have been explained thoroughly in this chapter. Section 3.1 explains the architecture of DAIS framework. Section 3.2 elaborates the component Adjacency COS Overlay Topology (ACOT)[97] and its functionalities. Further, Section 3.3 explains Cloud Storage Resource Discovery (CSRD) component and its sub-parts. Finally, fourth component CIS module has been explained in Section 3.4. The chapter culminates with Cloud Usage Monitor (CUM) module

3.1 Architecture of DAIS Framework

The DAIS[191] framework presents unique approach in cloud environment for storage management. It introduces new concepts including dynamic and scalable data access, self-organizing capability and well balanced multi-way COS overlay topology. It removes the necessity of centralized approaches for traditional static storage man-

agement and adopts the dynamic and scalable data access in the cloud computing environment. It is quite challenging to meet the requirements requested by different applications among thousands of COS nodes. The proposed framework consists of multiple geographically distributed domains of different locations. It divides COS nodes into multiple geographically distributed domains to facilitate the dynamic data access in cloud computing environment. DAIS framework has mainly two layers: Interior Layer and Exterior Layer.

Interior Layer: It provides dynamic, scalable and self-organized virtual COS nodes for efficient data integration and resource discovery. This layer has four sub components including Adjacency COS Overlay Topology (ACOT) [97], Cloud Storage Resource Discovery (CSRD), COS Integration Service (CIS) and Cloud Usage Monitor (CUM). Each of component has their identical functionalities and capabilities. All these components have been explained separately in next sections.

Exterior Layer: It creates multiple cloud domains that consists geographically distributed COS nodes and each cloud domain has one COS Representative (CR) node (also known as COS agent) [72] node). CR node coordinates with members nodes as well as other cloud domains with the help of ACOT algorithms. If for some reason the agent node stopped working, in that case the highest power node will automatically become an agent node. Mainly, each COS node has dynamically organized in different cloud domains. In this way, the popularity of data access from geographically distributed networks increases. All these multiple distributed domains are communicated with the help of CR nodes. These physical domains are geographically distributed on the basis of three criteria: a) distance, b) qualitative (performance-based) and c) quantitative (bandwidth-based). Figure 3.1 shows the architecture of proposed DAIS framework. All COS node can join or leave the domain dynamically. Any COS node can directly communicate with other nodes.

Virtual Organization: It refers to a dynamic and scalable set of individual and organizational defined set of resource-sharing rules and conditions. Possibly, it may be volatile interconnection between individual and organization. In general, it comprises a complex storage network of smaller enterprises which each contribute a part of storage system. The virtual organization is often expressed as its functional benefits as

given below:

- ☐ Large scale data storage system across the globe.
- ☐ Improve the Quality of Service (QoS) in term of instance, performance, flexibility and collaboration.
- ☐ Increase the on-demand data accessibility level.
- ☐ Easy to upgrade with new technology.
- ☐ Infinite availability of resources.
- ☐ Users will typically belong to more than one virtual organization.

3.2 The Proposed Adjacency COS Overlay Topology (ACOT)

ACOT is a flexible and self-organizing overlay topology, introducing a well-balanced m-way tree structure. It is based on balanced multi-ways (m-way) tree structure. In general, m-way tree has been conveniently defined with small value of m. But in our case, m is usually very large. Each node corresponds to a physical COS device and m denotes the highest number of COS devices that can be linked with a cloud domain. The value of m could be maximized as per demand of storage service in the particular cloud domain. ACOT is divided into two tiers: First Tier and Second Tier. First tier focuses on self-organized virtual COS which involves in overall routing process. It is a balanced m-way tree structure that manipulates COS nodes in flexible manner. In this context, four different algorithms are designed, which are described in the next chapter. Second tier maps the COS nodes and creates n numbers of cloud domains. The parent node will be assigned as COS Representative (CR) node that cooperates with other COS node. The sole motivation of Second Tier is to increase the fault tolerance and optimize the storage network. Two tiers representation of ACOT has been shown in Figure 3.2a and 3.2b . ACOT is a unique and flexible topological

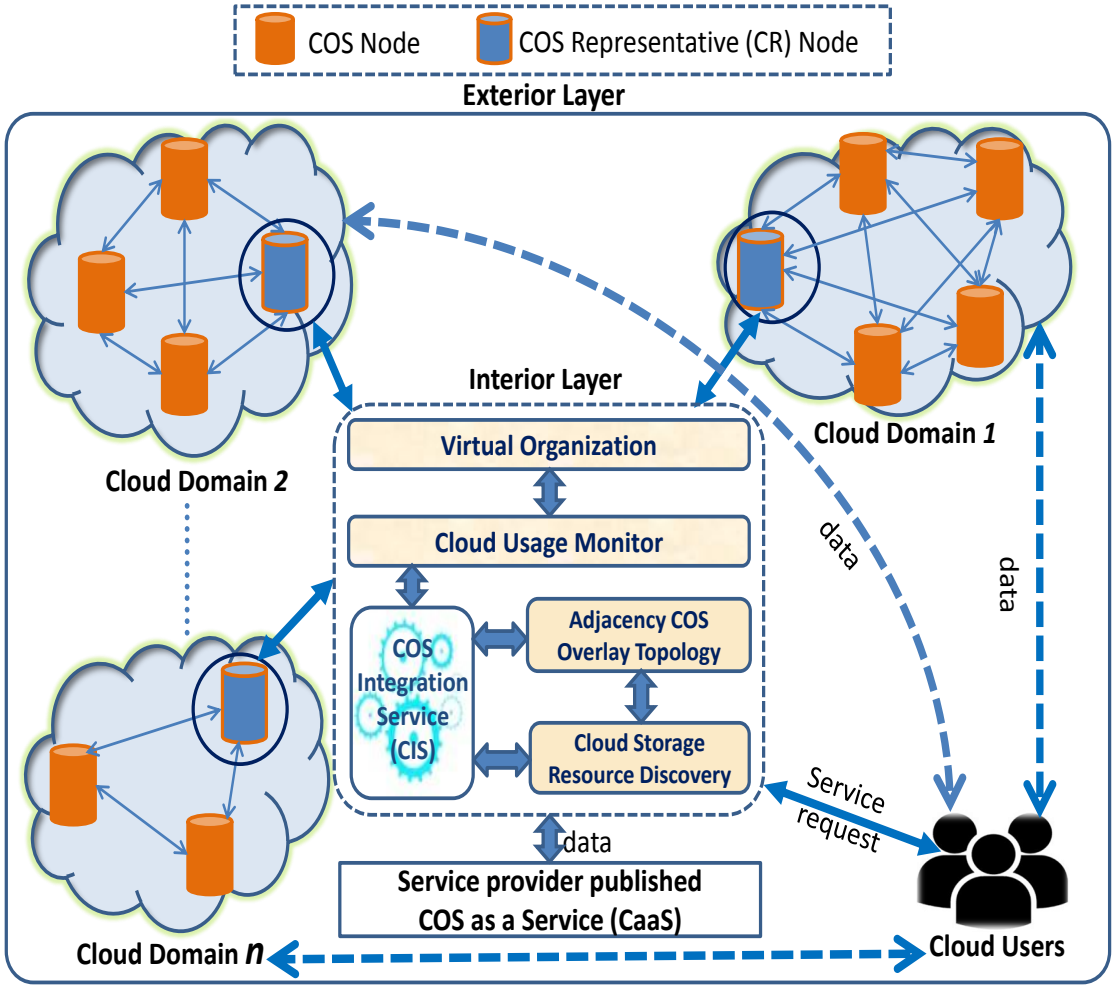


Figure (3.1) Architecture of proposed DAIS framework

designed for self-organizing cloud-oriented storage systems. Analysis and design of ACOT algorithms and their computational cost have been presented in the Chapter 4.

3.3 The proposed Cloud Storage Resource Discovery (CSRD)

Efficient resource discovery is one of the challenging task for cloud computing environment. The fundamental requirements of resource discovery to enhanced the high

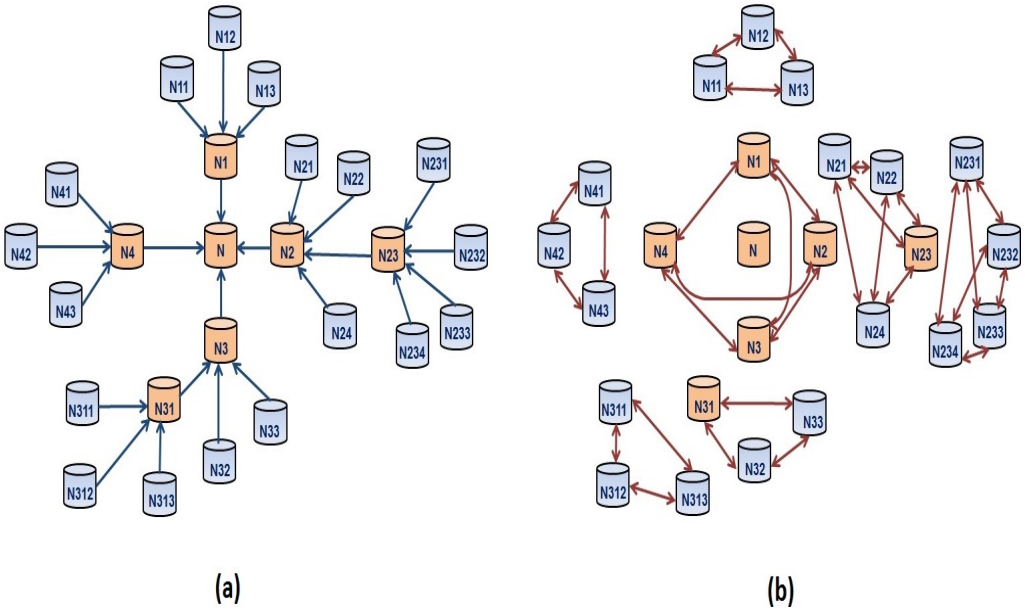


Figure (3.2) Two tiers representation of ACOT. Figure 2a shows the first tier using balanced multi-way tree organization.Each node having unique prefix id. Figure 2b represents the second tier and highlights the storage domain formation algorithms.

performance, distributed nature, Quality of Service (QoS), and global scope [89]. The CSRD module plays important role for DAIS framework. It contains various components like certificate authority, resource locator, computing utilities, storage clusters and various cloud services. Figure 3.3 shows the internal view of CSRD.

3.3.1 Certificate Authority

The certificate authority (CA) is a trusted unit which is used to authenticate and verify the users before joining and leaving the domain. It servers as a trusted parties to provide secure communication over Internet. It consists various components for ensuring the security of big data repositories and cloud storage infrastructures. It produces the login credentials and digital certificate to the users which maintain the confidentiality and integrity of cloud storage network. . Once and maintains the list of trusted and authorized cloud user list. The certificate given by CA to build trust among users and service providers because they can ensure the validity of the identity of each other and the authorities.

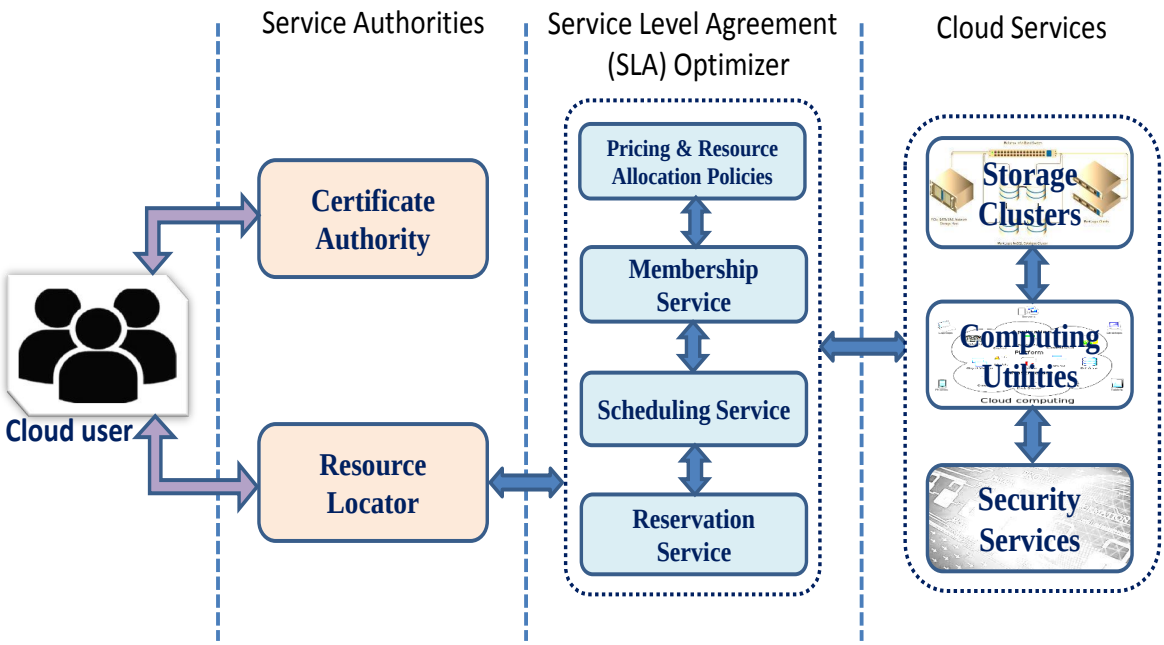


Figure (3.3) Internal view of CSRD

3.3.2 Resource Locator

Resource locator is a kind of log file that maintain the entire services path. It contains complete path of various services. Any user can submits their requests along with service queries to the resource locator and the resource locator will return a list of matched services which are linked with Universal Resource Identifier (URI). Table 3.1 shows the some important methods that will help in service invocation. To use the services, users will have to follow the steps given below:

Step 1: Discover the storage resource location (SRL) from existing URI list.

Step 2: The SRL request to *CreateResource()* method for creating and initializing a new storage service instance.

Step 3: SRL assigns each location with unique ID.

Step 4: Once the CreateResource() call finishes, then URI send to user.

Step 5: All storage service instances will be employed and find the corresponding physical storage location to operate the newly created service instance.

Figure (3.4) Resource locator format

3.3.3 Service Level Agreement (SLA) Optimizer

The main objective of optimized resource allocation strategy must be satisfied the Quality of Service (QoS) [98] constraints and profit of cloud provider must be maximized [38]. Generally, service level agreement violations affect the overall quality service guarantees and it may be possible to suffered some loss due to certain penalty cost. Most of the cloud service provider have been adopting the same traditional approached and auction mechanism so far to allocate the resources. Due to the diversity and different characteristics of resources, existing approaches are not much effective. The dynamic resource allocation is NP-hard problem by nature. Therefore, an efficient and customized resource allocation techniques should be provided by the cloud service provider, in which the various issues such as the specified penalty cost, time, and other SLA parameters must be followed by the users [134][103]. Many researchers are working on SLA constraints and try to reduce unnecessary penalty costs.

i.) Pricing and Resource Allocation Policies: This thesis has proposed an SLA optimization model with the help of service delivery response time and demonstrated their different aspects in Figure 3.5 [98]. The Figure 3.5 (a) denotes that if the response time of the service is less than S_d, then the provider get a revenue R. If

Table (3.1) List of service invocation methods

Method	Description
Subscribe()	Associate as a new member with the required storage facility
Unsubscribe()	Leave the membership along-with all facilities
Lst_Dir()	Display list of all the directories/path information
Clear_Space()	Delete data from storage capacity
Grap_Info()	Search data as per given pattern
Upload()	Transform data in remote location
Download()	Retrieve data from remote location

the service response time crosses the S_d, then the penalty must be paid to the consumer in this situation as shown in Figure 3.5 (b). Here, consumer has already made advance payment R and if the service R_{time} (Response Time) crosses the S_d then in such a situation, the R will also be returned with the penalty cost. The both aspects are shown in Figure 3.5(c) and Figure 3.5(d) respectively. For better understanding, Let, $R + P = 1$ for the moment and SLA penalty cost P for service S can define as:

$$P\,(S) = \begin{cases} 0 & if\ S_{time} \leq S_d \\ 1 & otherwise \end{cases} \tag{3.1}$$

Consequently, it has defined the average SLA penalty cost for service S is the sum of penalty over the total number of services N, i. e.,

$$P_{avg} = \frac{1}{N} \sum P(S) \tag{3.2}$$

As we are inspired by the above problem, the goal of the service provider is to maximize profits and reduce unnecessary SLA infringement costs.

The mathematical formulation of SLA optimization is based on biding concepts [96]. Let, cloud service provider offers various services to the user on t types of Virtual Machine (VM) instances such as VM $= \{\mathrm{vm}_1, \mathrm{vm}_2, \ldots\ldots\ldots \mathrm{vm}_\mathrm{t}\}$. Each VM instance has its own identical service capacity w. Let, Z $= \{\mathrm{z}_1,\ \mathrm{z}_2,\ \mathrm{z}_3 \ldots\ldots\ldots \mathrm{z}_\mathrm{i}\}$ is a set of different bid providers. Let, A $= \{\mathrm{a}_1,\ \mathrm{a}_2,\ \mathrm{a}_3 \ldots\ldots\ldots \mathrm{a}_\mathrm{m}\}$ is a set bids submitted by

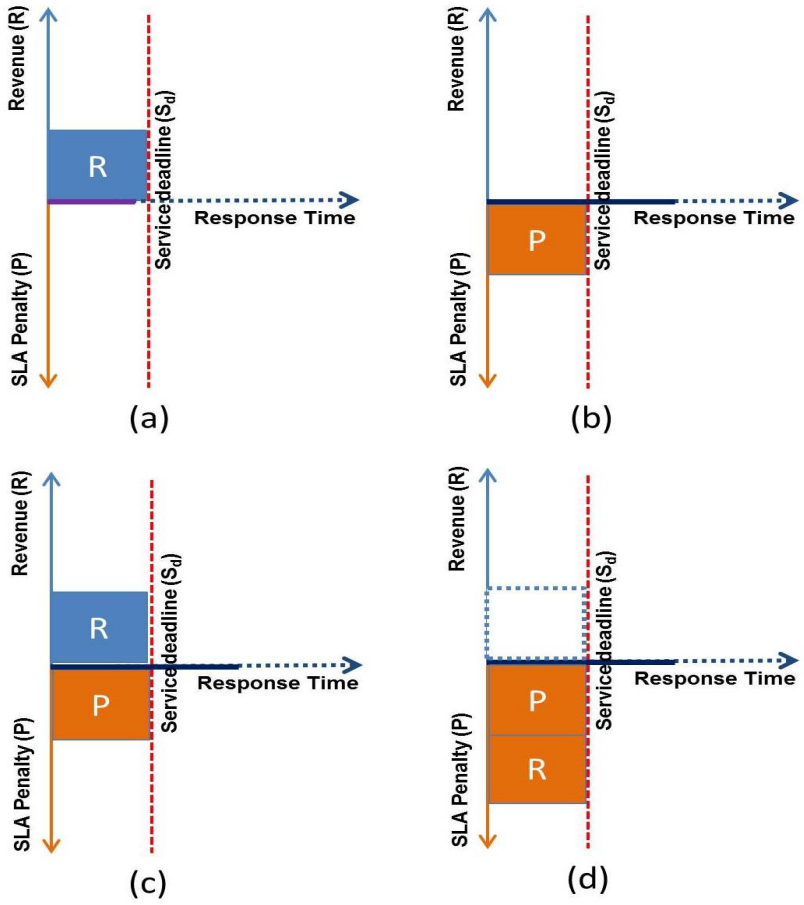


Figure (3.5) SLA revenue and penalty cost model

different bid providers. Here, we assume some providers are not participating in the biding. Each bid has been defined in different pairs like $a_i = \{R_n^i, c_i\}$ where, R_i is a set of requested VM instances and c_i indicates the particular bid cost. Each requested VM instances further divided in order pairs along with respective units/capacity of specific resources. It is defined as: $R^i_{\{1,\ 2,\ \ldots\ldots n\}} = \left\{ \left(r_1^i, u_1^i\right),\ \left(r_2^i, u_2^i\right), \ldots\ldots \left(r_n^i, u_n^i\right) \right\}$. All the bids are published on the cloud as per given below template:

$A_1 = \{[(Intel/I3/2.0GHz, 30), (Memory, 2GB), (Storage, 250GB)], \$1500\}$

$A_2 = \{[(AMD/x4/4.3GHz, 20), (Memory, 16GB), (Storage, 320GB)], \$2500\}$

.....

$A_{m-1} = \{[(Xeon/e7/4.2GHz, 10), (Memory, 16GB), (Storage, 1TB)], \$3000\}$

$A_m = \{[(AMD/Ryzen/4.6GHz, 20), (Memory, 32GB), (HDD, 2TB)], \$10500\}$

The satisfaction level of all resource sets are conditionally defined as:

$$R^i_n = \begin{cases} \left\{ \left(r^i_1, u^i_1\right) \cap \left(r^i_2, u^i_2\right) \cap \ldots \cap \left(r^i_n, u^i_n\right) \right\} \neq \emptyset, \text{satisfiable for } \forall R_i \\ \\ \left\{ \left(r^i_1, u^i_1\right) \cup \left(r^i_2, u^i_2\right) \cup \ldots \cup \left(r^i_n, u^i_n\right) \right\} \neq \emptyset, \text{satisfiable for } \exists R_i \end{cases}$$

The set P(VM) ={p_1,p_2, ...p_t} specify the SLA violation penalty cost. The simple penalty cost and average penalty cost for each VM has been determined according to Equation 3.1 and 3.2 respectively. The formulation of above problem has been used linear programming concepts using three independent variables (α,β and γ). Description of all the symbols shown in Table 3.2. The mathematical formulation of aforementioned problems as follows:

$$Maximize \sum_{a_i \in A} c_i \alpha_i - \sum_{a_j \in A} p_j \gamma_j \tag{3.3}$$

Subject to

$$\sum_{a_i \in A \ \wedge (r^i_n, u^i_n) \in R^i_{\{1,\ 2, \ldots\ldots n\}}} \beta^j_{i,n} + \gamma_j = \ w_j \quad (vm_j \in VM) \tag{3.4}$$

$$\sum_{VM_j \in VM} \beta^j_{i,n} - u^i_n \alpha_i = 0 \qquad \left(a_i \in A \wedge \left(r^i_n, u^i_n\right) \in R^i_{\{1,\ 2, \ldots\ldots n\}}\right) \tag{3.5}$$

$$\beta^j_{i,n} = 0 \quad \left(\left[a_i \in A \wedge \left(r^i_n, u^i_n\right) \in R^i_{\{1,\ 2, \ldots n\}}\right] \wedge [VM_j \in VM] \wedge [VM_j \notin r^i_n]\right) \tag{3.6}$$

$$\alpha_i = \{0,\ 1\} \tag{3.7}$$

$$\beta^j_{i,n}, \gamma_j \in Z^+ \tag{3.8}$$

The main purpose of the above formulation is to maximize the profit of service provider and to satisfy all constraints. The Equation 3.3 represents our objective that maximizes the profit. Equation 3.4 specifies the no. of allocated VM instances of each vm_j that should not be exceeded from total no. of available VM instances. The successful bids of each pair is recognized by Equation 3.5. This model have All those resources which have been underestimated, Equation 3.6 has assigned all

Table (3.2) Notation table

Symbol	Description
VM	Set of Virtual Machines
SLA	Service Level Agreement
w	Service capacity
Z	List of cloud service providers
A	Set of bids submitted by particular cloud service providers
R	Set of bids which are requested by consumers
c	Cost of particular requested bid
u	Requested unit/capacity of resources for respective bid
P	Simple SLA violation penalty cost
$P(VM)$	Service Level Agreement violation penalty cost applicable on particular VM instance
P_{avg}	Average SLA penalty cost
S_d	Service deadline
S_{time}	Service time
R_{time}	Response time
α	Boolean variable which identify the bid is success or not. If bid is success then store '1' otherwise '0'
β	Identify the number of successful allocated resources
γ	Specify the un-allocated number of resources or not assigned in biding process
Z^+	Identifies the positive integer
n	Specifies the natural number

such resources to particular bid. Equation 3.7 and 3.8 are independent variables which contains Boolean value and positive integer value respectively. These variables have explained in Table 3.2. In Chapter 4, a new algorithm named GACO has been proposed that can solve the aforementioned problem efficiently. The rest of details includes test benchmark functions and implementation are also presented in

well manner.

ii.) Membership Service: It manages all the authenticated cloud users who uses the resources. A part from this, all users are categories into different groups so that resource can be easily allocated. It provides the user credentials to particular user. Each user has its own characteristics for usage of resources.

iii.) Scheduling Service: Scheduling service determines how to schedule all the requests. In most cases, many users simultaneously demand resources, which makes it difficult to negotiate. Better way to keep all such users in same time slot and assign some priority.

iv) Reservation Service: During the reservation phase, the cloud user or provider submits the reservation requests through the Reservation Service. It discovers all the available COS node and interact with requested cloud user. It performs COS node selection by choosing the required number of available time slots from execution COS nodes and manages admission control by accommodating or denying a reservation request. It also estimates the price for a confirmed reservation based on the implemented pricing policy. Available time slots have been selected keeping in mind the user's application requirement.

3.3.4 Cloud Services

A cloud service is any service made available to users on demand through the Internet. In general, cloud services are classified into three categories- Storage clusters, Computing service and Security service. All these services are designed to provide dynamic, scalable access to software, resources, and security services that are fully managed and controlled by a Cloud Services Provider (CSP). Cloud service can dynamically scale to meet the user's needs, and because the CSP provides the necessary infrastructure for the service, there's no need for a company to provision or deploy its own resources or allocate IT staff to manage the service. Examples of cloud services include online data storage and backup solutions, Web-based e-mail services, hosted office suites and document collaboration services, managed technical support services, database processing, and so on.

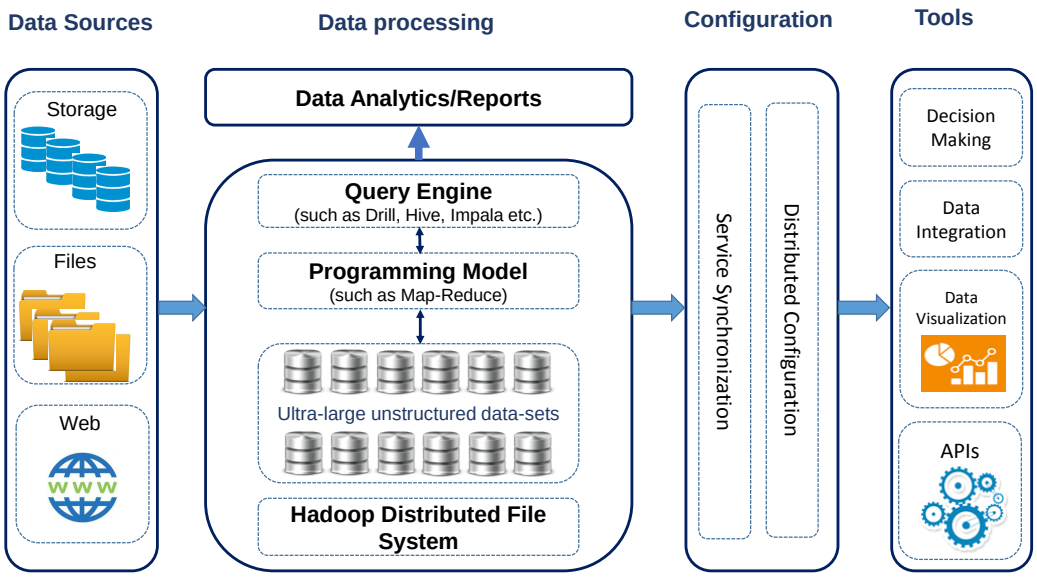


Figure (3.6) Working model of CIS

3.4 The proposed COS Integration Services (CIS) Model

Access to multiple data sources and integration of such data is usually expensive in cloud environment. In general, data integration can be particularly beneficial for cloud computing. This thesis has designed a novel model for CIS. This model has been diveded into four different levels- Data sources, Data processing, Configuration and Tools. Each level has performed specific set of tasks. Figure 3.6 shows the working model of CIS.

3.4.1 Data Sources

In general, the data source is the name assigned to the connection to a server from a database. The data may be in form like structured, semi-structured and unstructured. There are so many sources of generating large amount of data such as social media, sensors, business transaction, news media, etc. Social media is one of the

major sources of generating huge amount of unstructured data. It shares of transfer information within virtual communities such as Facebook, Twitter, web-blogs, etc. Transnational data such as business data, financial data, ERP data that are generally in structured form. However, sometime it becomes in very large data-set form and it can not be easily processed. A large number of IoT devices like, sensors, cell phone, cameras, etc. are connected to Internet that generate large amounts of data every movement. An integration of such huge amounts of different forms data is very challenging task and need some better approaches. Therefore, CIS model proceeds to second level of tasks i.e. data processing.

3.4.2 Data Processing

Data processing have four different components- data analytic, query engine, programming model and Hadoop Distributed File System (HDFS). Query engine fetches data from distributed servers with the help of specific data connection query and directly send to programming model. CIS uses Map-Reduce algorithm for further processing of large amounts of data and store in HDFS. Data analytics module measures all the activities and generate various log file and reports. Basic details for Map-Reduce already explained in Chapter 1. Analysis and design of Map-Reduce algorithm has been elaborated in Chapter 4.

3.4.3 Configuration and Tools

CIS model has used two types of configuration approaches- service synchronization and distributed configuration. Synchronization refers to keep files in different locations and make accessible for multiple users. Another aspects of synchronization services are designed to distribute single set of data among two or more users. The sole motivation of synchronization is to control the data access from multiple threads. In general, Hadoop can be installed in three ways-Standalone, Semi-distributed and Fully-distributed. There are lots of configuration parameters have to setup during data processing. Some basic configuration parameters are given below:

i. core-site.xml(set the HDFS IP address and port number)

```
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:8000</value>
</property>
```

ii. hdfs-site.xml (If value is 1 then backup data from HDFS to another node)

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

iii. mapred-site.xml (configure service negotiator as TIET)

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

iv. TIET-site.xml (configure service provider TIET as map-reduce)

```
<property>
        <name>TIET.nodemanager.aux-services</name>
        <value>map-reduce</value>
</property>
```

Apart from the above, there are so many more configuration parameters which have been mentioned in Table 4.6.

Decision making, data integration, data visualization and APIs are some important tools of CIS model.

3.5 Cloud Usage Monitor (CUM)

Monitoring of infrastructural resources in the cloud computing plays an important role in providing application surety, such as availability, performance and security.

Cloud Usage Monitor (CUM) [15] is an autonomous and lightweight open source application program responsible for collecting, analyzing and processing data usage in cloud environment. Firstly, a company named CloudOYE [39] used it very easily in cloud environment. CUM is one of the most important component of DAIS framework that measures all the activities happening inside cloud domain. It works within interior layer of DAIS framework. It scales and reviews all the operational tasks of cloud domains. It also finds the operational status of various cloud components. CUM has three agents- Monitoring agent, polling agent and resource agent. A monitoring agent is transitional, task-driven programs that work as a service mediator and transparently analyzes and monitor data-flows. A resource agent collects usage data by having task-driven communications with specified resources. It is used to monitor usage metrics based on predefined, apparent events at the resource application level, like initiating, resuming, suspending, and scaling the resources. A polling agent is a processing segment that collects cloud service usage data periodically and monitor resource status like uptime and downtime. It can be easily implemented with the help of automated software and access control of cloud domains. It is the use of manual as well as automated IT management and monitoring approaches to ensure that cloud infrastructure, platform or application performs better. CUM provides following functionalities:

i Virtual machine monitoring: It manages the overall virtualization infrastructure and keeps records of all Virtual Machines (VM) configurations.

ii Cloud storage monitoring: It monitors the Virtual Cloud-Oriented Storage (vCOS) systems, databases and data services.

iii Web portal monitoring: It analyzes the web related services including storage utilization, data processing, web hosting, etc.

iv Network monitoring: It ensures the bandwidth, throughput and overall network performance.

v Data transfer monitoring: It maintains the various log files of source and destination IP address, amount of data, speed etc.

vi Bandwidth monitoring: It specifies the amount of data transfer in a particular time movement.

vii Performance monitoring: It monitors the dynamicity, availability and processing speed of storage systems.

Some silent features of Cloud Usage Monitor are:

i High capability to deal with large amounts of data

ii Increase the transparency level between users and service provider

iii Reporting high accuracy data

iv Continuous scaling and collecting data

v Generate analytical reports

vi Easily identify the user satisfaction level

Here, the cloud user sends a message to the cloud service provider to provide the service. The monitoring agent captures the request to collect appropriate usage data. Before, allowing it to continue to the cloud service the monitoring agent stores the usage data in a log file. After this, the cloud service provider answers back with a response message without any intervention to the cloud user.

3.6 Conclusion

This chapter has been proposed a novel framework named Dynamic Access and Integration Services (DAIS). The architectural view of DAIS framework has been exhibited in Figure 3.1. The DAIS framework consists four major components. Each one component has been designed and thoroughly explained in separate section. Adjacency COS Overlay Topology (ACOT), Cloud Storage Resource Discovery (CSRD), COS Integration Services (CIS) are some major researches of this chapter. Apart from these, some scalability issues and mathematical formulation of SLA optimization have been explained properly.

Chapter 4

Design and Implementation of Proposed DAIS Framework

This chapter describes overall design and implementation of DAIS framework. Four algorithms have been analyzed and designed in this chapter. Each algorithm has been thoroughly computationally analyzed and evaluated its searching cost and time. A novel swarm-based meta-heuristic algorithm named Generalized Ant Colony Optimizer (GACO) has been designed for execution of cloud services. GACO has tested on 17 well-known benchmarks test functions and compared with three most popular algorithms-Particle Swarm Optimization, Genetic Algorithm and Artificial Bee Colony. Apart from this, a Map-Reduce model has been implemented for cloud-oriented storage integration services using two data-sets.

4.1 Algorithm for Adjacency COS Overlay Topology (ACOT)

4.1.1 COS Routing and Naming Mechanism

The interior layer of ACOT is well balanced m-way tree organization that manipulates COS nodes effectively. The root node will be most responsible because it makes the list of most trusted COS node that are geographically distributed in public cloud

domain. Algorithm 1 illustrates the routing and naming mechanism for active COS nodes. All nodes are given a unique code that is in the form of a prefix id of whole tree structure. Line 2, finds the Longest Prefix Length (LCP) of the current tree structure and traverse the tree as per applicable conditions and find an appropriate NC_{id} . This prefix is send to join process. The main purpose of routing and naming algorithms is to keep the structure of the tree well balanced.

4.1.2 Joining, Leaving and Traversing Overlay

ACOT provides simple and flexible approach to join and leave the COS domain anytime. There is no any restriction for joining and leaving any node from ACOT. If any node wants to join ACOT then it has to make a request to main root node and receive the list of all active domains. After this, establish the connection with nearest domain and find the appropriate location with the help of Algorithm 1. Algorithm 2 focuses on joining the COS node and maintain the balancing factor of the storage topology. In Algorithm 2, first check the NC_{id} (founded by Algorithm 1) have empty slot or not. If NC_{id} have empty slot then join the network with valid ip address (generated by Hadoop Distributed File System). If NC_{id} does not have empty slot then redirect to neighbor node and check empty slot. In case of non availability, create new level of tree and adjust tree structure as per M-Way tree properties.

ACOT is based on trusted and stable COS nodes. Any COS node can delete from domain without any restriction. Before deleting any node, first find LCP and check the node is main root, internal root or leaf node. If LCP is 0 i.e. deleting node is main root. If LCP is equal to depth of deleting node then delete leaf node. Otherwise, deleting node is an internal root. The technical specifications of joining and leaving process are defined in Algorithm 2 and 3 respectively.

Algorithm 4 specifies the overlay COS traversal process and find the total combination of possible paths. All the decedent COS nodes of each path have been en-queued during overlay traversal.

The interior layer of ACOT is well balanced m-way tree organization that manipulates COS nodes effectively. The root node will be most responsible because it

Table (4.1) List of notations used

Notation	Description
COS	Cloud Oriented Storage
N	Total number of active COS nodes on ACOT
CR	COS Representative node
msg	Message
$depth$	Depth of the particular COS node
$path$	It shows the complete traversal path
id	It is unique and self-explainer id of COS node such as N232.
S_{id}	Source COS node ID
D_{id}	Destination COS node ID
N_{id}	Neighbor COS node ID
C_{id}	Child COS node ID
NC_{id}	It is a new COS node ID that want to join the network
NC_{level}	Current level of new COS node
L_{level}	Last level (leaf COS node level)
C_{max}	Maximum number of children COS node
C_{remain}	Shows the status of remaining child COS slots.
NC_{ip}	IP address of new COS node
N_{ip}	IP address of neighbor COS node
A_{ip}	IP address of ancestor COS node
C_{ip}	IP address of child COS node
C_{accept}	Shows the status of either child node is accepted or not.
$LCP(NC_{id}, D_{id})$	It is a method to find the Longest Common Prefix path between new COS node to destination COS node
$STA(NC_{id}, D_{id}, S_{id}, msg)$	Send to ancestor COS node
$STN(NC_{id}, D_{id}, S_{id}, msg)$	Send to neighbor COS node
$STD(NC_{id}, D_{id}, S_{id}, msg)$	Send to descendant/child COS node
$RJA(NC_{ip}, A_{ip})$	Redirect to join ancestor COS node
$RJD(NC_{ip}, C_{ip})$	Redirect to join descendant COS node
$RJN(NC_{ip}, N_{ip})$	Redirect to join neighbor COS node

Algorithm 1 : Routing and naming mechanism

Input: S_{id}, D_{id}, msg

Output: $depth, NC_{id}$

1: Procedure $Path(S_{id}, D_{id}, msg)$
2: **if** $D_{id} = NC_{id}$ **then**
3: $Set(msg)$
4: **end if**
5: $depth = LCP(NC_{id}, D_{id})$ ▷ find the depth with the help of node prefix id
6: **if** $depth = NC_{level}$ **then**
7: $N_{id} = D_{id}[NC_{level}]$
8: **if** $N_{id} = 0$ **then**
9: $STA(NC_{id}, D_{id}, S_{id}, msg)$ ▷ destination node is an ancestor node
10: **else**
11: $STN(NC_{id}, N_{id}, S_{id}, msg)$
12: **end if**
13: **else if** $depth < NC_{level}$ **then**
14: $STA(NC_{id}, D_{id}, S_{id}, msg)$ ▷ destination node is neither children nor adjacent node
15: **else**
16: $Cid = D_{id}[NC_{level} + 1]$
17: $STD(NC_{id}, D_{id}, S_{id}, msg)$ ▷ destination node is either descendant or child node
18: **end if**
19: End Procedure

makes the list of most trusted COS node that are geographically distributed in public cloud domain.

4.1.3 Computation Analysis

This section highlights computational analysis of ACOT operations. Some important properties of ACOT has been defined as follows:

Definition 1 *We assume ACOT has order of m and height h. Let N number of COS nodes are connected with ACOT. Then the following properties have been satisfied:*

1. $[2(\frac{m}{2})^{h-1} - 1] \leq N \leq m^h - 1$

2. $log_m(N+1) \leq h \leq log_{\frac{m}{2}}(\frac{n+1}{2}) + 1$

Definition 2 *For the height h, the maximum number of COS node is $m^h - 1$. Where, m denotes the branching factors of the tree.*

Definition 3 *To handle overflow at physical domain D_i; perform middle entry $m = \lceil \frac{N+1}{2} \rceil$ in D_i directly goes to the ancestor in right place.*

Definition 4 *Each node have a degree a or b. The number of nodes is between a^h–1 and b^h–1. Where h is the height. Hence, the height between $\lceil log_b(N+1) \rceil$ and $\lceil log_a(N+1) \rceil$ for n nodes.*

Definition 5 *Each node take $O(n)$ time to compute factorial for every COS at the level order traversal. Hence, total time is $O(N * k)$, where, N denotes total number of active COS and k indicates number of branching factors.*

Insertion cost: ACOT takes constant time i. e. $O(1)$ on each level to perform normal operations such as rotating and shifting nodes. There are $O(logN)$ different levels. Therefore, insertion takes $O(log_k N)$ time, where k indicates the branching factors and N specifies the all active nodes in a complete domain as shown in Figure

Algorithm 2 : Joining new COS node

Input: NC_{id}, msg

Output: NC_{ip}

Procedure $Join(NC_{ip}, msg)$

if $C_{accept} = 1$ **then**

if $C_{remain} > 0$ **then**

$NC_{id} = C_{max} - C_{remain}$

$STD(NC_{id}, D_{id}, S_{id}, msg)$ ▷ move on leaf node

else if then$C_{remain} = 1$

$RJN(NC_{ip}, N_{ip})$ ▷ join neighbor as ancestor

else

$RJA(NC_{ip}, A_{ip})$ ▷ join as ancestor node

end if

else

if $L_{level} = 1$ **then**

$RJA(NC_{ip}, A_{ip})$ ▷ join as ancestor

else if $L_{level} < NC_{level}$ **then**

$RJD(NC_{ip}, C_{ip})$ ▷ join as child node

else if $L_{level} < NC_{level}$ **then**

$RJN(NC_{ip}, N_{ip})$ ▷ join neighbor as child

else

$RJA(NC_{ip}, A_{ip})$

end if

end if

End Procedure

Algorithm 3 : Leaving COS node

Input: S_{id}, D_{id}, msg

Output: D_{id}, CN_{id}

1: Procedure $Leave(S_{id}, D_{id}, msg)$
2: $depth = LCP(S_{id}, D_{id})$
3: **if** $depth! = 0$ **then** ▷ not a leaf node
4: $Delete(D_{id})$
5: $NC_{ID} = C_{MAX} - depth$ ▷ create new COS id
6: $STN(NC_{id}, D_{id}, S_{id}, msg)$
7: $Update()$ ▷ update tree after deletion
8: **else**
9: $Delete(S_{id})$
10: $STA(NC_{id}, D_{id}, S_{id}, msg)$
11: $Update()$
12: **end if**
13: End Procedure

Algorithm 4 : Overlay COS node traversal

Input: $S_{id}, D_{id}, root$

Output: $path$

1: Procedure $Traversal(S_{id}, D_{id})$
2: $depth = LCP(S_{id}, D_{id})$
3: **if** $root = NULL$ **then** ▷ returning 0 if root node is NULL
4: $return(0)$
5: **else**
6: **while** $root! = NULL$ **do**
7: $path = path * factorial(depth)$ ▷ find the total combination of path and enqueue all the decendent COS nodes
8: **end while**
9: $return(path)$ ▷ return the traversal path
10: **end if**
11: End Procedure

4.1. In the worst case, one leaf node is placed to another sub domain and $2log_kN$–1 hopes will traverse.

Searching cost: If ACOT has height h has least number of nodes when all internal nodes are $m-1$. Since all leaf nodes must be at the same level, the ACOT is a balanced and the number of COS nodes are $n = m_{h+1} - 1 \implies h = \lfloor log_m n \rfloor$. If ACOT of height h has the largest number of COS node when all internal root nodes are m. Then the number of nodes $N = \sum_{i=0}^{h} m^i = \frac{m^{h+1}-1}{m-1}$ and the number of keys $n = m^{h+1} - 1 \implies h = \lfloor log_m n \rfloor$. Hence, the search time on ACOT overlay with m COS node is $O(log_{m-1} n)$.

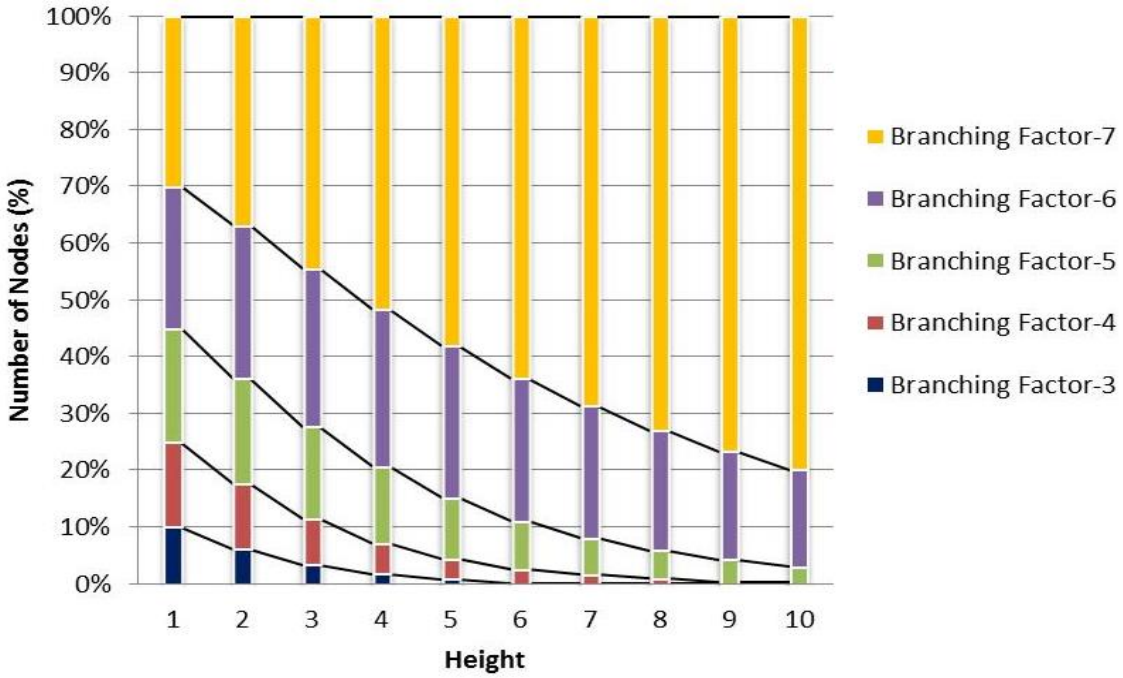


Figure (4.1) Number of active COS nodes involves in routing process

4.2 Generalized Ant Colony Optimizer (GACO) for SLA optimization

In this section thesis has proposed a swarm-based meta-heuristic algorithm named Generalized Ant Colony Optimizer (GACO) for SLA optimization which is a part of CSRD module (explained in Chapter 3). GACO is based on hybrid approach which consists of Simple Ant Colony Optimization (SACO)[51] and Global Colony Optimization (GCO)[162][118] concepts. The main concept behind GACO is the foraging behavior of ants. GACO operates in the following four phases: Creation of a new colony, search of nearest food location, balance the solution, and updating of pheromone. GACO has been tested on seventeen well recognized standard benchmark functions and its results have been compared with three different well-known meta-heuristic algorithms namely as Genetic Algorithm (GA) [171], Particle Swarm Optimization (PSO) [87] and Artificial Bee Colony (ABC)[85]. The performance metrics such as average (AVG) and standard deviation (STDEV) are computed and evaluated with respect to these metrics. The proposed GACO performs better in

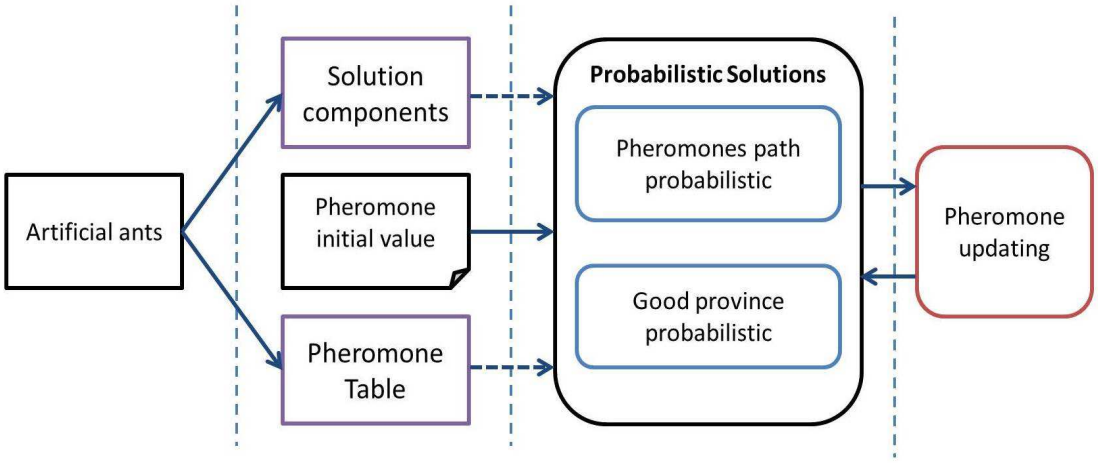


Figure (4.2) Working model of GACO algorithm

comparison to the aforementioned algorithms. The proposed algorithm optimizes the SLA problem and gives better results with unknown search spaces.

4.2.1 Working Model of GACO

The simple ACO algorithm was explained by Marco Dorigo et al. in the early 1990's. Artificial ants will find the nearest node randomly first time and the ants who have successfully reached the destination will update the total path length (L) as a pheromone trail of the link visited in pheromone table. Here, actually routing table is known as pheromone table that contains for each destination with real valued information, on for each known adjacency node. This information is a remedy of goodness of traversing over that adjacency node on the way to the destination. This table is continuously updated conferring to the quality of solution by the artificial ants. The next ant's group will explore the pheromone trails with the help of pheromone table to reach the destination. The working model of proposed algorithm graphically exhibited in Figure (4.2).

Let, number of artificial ant sets $m = \{1, 2, \ldots\ldots, n_m\}$ are located in the colony. We assume ant m is currently situated at node i then the transitive probability of

the ants to choose next node $j \epsilon v_i^m$ is as follows:

$$P_{ij}^m(t) = \begin{cases} \frac{z_{ij}^{\alpha}(t)}{\sum_{j \epsilon v_i^m} z_{ij}^{\alpha}(t)} & if\ j \in v_i^m \\ 0 & if\ j \notin v_i^m \end{cases} \tag{4.1}$$

where v_i^m is the set of feasible path linked with v_i with respect to m number of ants. Each ant has their own decision policy to choose the next node. The z_{ij} indicates the pheromone concentration on corresponding edge *(i, j)*. If any v_i and ant m does not has any adjacent $v_i^m = \emptyset$, then the predecessor to v_i is included in v_i^m. There is always a probability of getting a cyclic path. In this case we can remove the cycle once the destination node has been selected. The α denoted as a constant (positive) value to increase the impact of pheromone deposit. Once ants have built a complete path between source to destination nodes then entire cycle has been removed and individual path can be retraced to their source and retain a pheromone amount on each edge (i, j) of their respective path. The total deposited pheromone by ant m is $\delta z_{ij}^m(t)$ on path length $L^m(t)$. The inverse if the path length $L^m(t)$ consider as quality path as follows:

$$\delta z_{ij}^m(t) \propto \frac{1}{L^m(t)} \tag{4.2}$$

$$therefore, \qquad z_{ij}(t+1) = z_{ij}(t) + \sum_{m=1}^{n_m} \delta z_{ij}^m(t) \tag{4.3}$$

where n_m denotes the population. We assume if $O^m(t)$ denotes the optimal solution during time t then $f(O^m(t))$ depicts the quality solution. It is more important to emphasize on the optimal path selection process as a result of coordination which are evaluated from the modest behaviors of individual ants. In multiple global colony optimization [162] [161] [54] [102] for artificial ants n_m, we consider both local as well as global search spaces. We assume n_l and n_g ants perform local and global search respectively. Let r_i denotes the province and $f(r_i)$ is the fitness of province r_i. All provinces must be assigned with quality solution dynamically and z_i initializes the pheromone amount for each province. At starting point global search space is considered as weak because $\forall n_m$ unknown about global province to explore new path. Here, 95% of the n_g perform crossover to struggle for new province and remaining

5% involved in pheromone trail distribution. The probability of each m of the local ants n_l selects r_i province which partially towards the good province as follow:

$$p_i^m(t) = \frac{z_i^\alpha(t)\mu_i^\beta(t)}{\sum_{j \epsilon N_i^m} z_j^\alpha(t)\mu_j^\beta(t)} \tag{4.4}$$

where, $\mu_i^\beta(t)$ denotes the a priori attractiveness of the move from source to destination, N_i^m indicate the feasible nodes. If province find better fitness then ant moves in the same direction otherwise increase the age of province and choose new direction randomly. The age of province denotes the weakness of the particular solution. The pheromone is modified by adding the amount of each z_i proportionate to the fitness of relative province. Some of other techniques to solve continuous optimization are mentioned in [24][162][161]. The technical specification of GACO algorithm has been explained in Algorithm 5.

The main motivation of doing this is that GACO provides best solution in larger search space as compare to existing algorithms. The cloud resource allocation problem has been mathematically formulated which is based on biding concepts [134]. The main objective of this resource allocation process is the Quality of Service (QoS) constraints must be satisfied and maximized the service provider profit. The sole motivation behind this formulation is to reduced the unnecessary SLA violations penalty cost.

4.2.2 Implementation and Performance Evaluation of GACO

This section explains seventeen well-known benchmark functions that are used to evaluate the efficiency and performance of GACO algorithm. There are seventeen benchmark test functions have been applied on GACO algorithm and evaluate the performance. These test functions are categorized into two classes: a.) uni-modal, b.) multi-model. The modality of benchmark function defines the number of local minima and global minima locations. Table 4.2 shows the complete details of test functions along with search space range. These benchmark test functions evaluate the behavior of an algorithm in difficult situation and sometime diverse [130]. The seven functions (T1 – T7) are uni-modal and ten functions (T8 – T17) are included

Algorithm 5 : Generalized Ant Colony Optimizer

Input: Number of artificial ants $n_m(m = 1,\ 2, \ldots n_m)$, number of province $r_k(k = 1,\ 2, \ldots r_k)$

Output: Optimal path with quality solution, Province with best fitness value

Procedure GACO
Set $d = 0$, $z_{ij}(0) = 0.01$ (to small default values), $z_k(0) = 1$
do
    for each $m = 1,\ 2,\ 3, \ldots\ldots\ldots n_m$ **do**
        $O^m(t) = NULL$
        do
            Pick next node using Eq. 4.1
            Link edge (i, j) to path $O^m(d)$
        while till reach on destination node
    end for
    for each ant $m = 1,\ 2,\ 3, \ldots\ldots\ldots n_m$ **do**
        for each edge (i, j) of $O^m(d)$ **do**
            $\delta z^m = \frac{1}{f(O^m(d))}$
            Add pheromone z_{ij} using Eq. 4.3
        end for
    end for
    d=d+1
while exit condition is true
Update $O^m(d)$ as optimal with quality solution $f(O^m(d))$
do
    Calculate $f(r_i)$
    Sort fitness of $\forall r_i$ in decreasing order
    Post 95% of $n_g$for mutation and crossover
    Post 5% of $n_g$for trail diffusion
    Update pheromone and age of $n_g$week province
    Send n_l ants to picked good province by Eq. 4.4
    for each n_l **do**
        if improve fitness of r_i **then**
            Update pheromone and move to good province
        else
            Increase province age and select new direction randomly
        end if
    end for
while exit condition is true
End Procedure

in multi-modal test benchmark. The first group of uni-model benchmark functions is more relevant for evaluating the utilization capability and convergence rapidness of an algorithm. There is no local optima and having single global optimum. Another group of multi-domain benchmark functions provide multiple local solutions and test the finding ability of an algorithm.
The population size/search agent of each competitive algorithm is set to 20. We observed that 20 is an equitable value of population size for several optimization problems. Generally, large size is not suitable for determining the global optima because it reflects the higher probability.

The parameters description of proposed algorithm and competitive algorithms are mentioned in Table 4.3. These parameters are fixed on the basis of reported literature review. The algorithms illustrated in this thesis are implemented in MATLAB R2017b-9.3.0713579 version and then executed on MS Windows 7 Professional N with 64 bits. It is deployed on hardware configuration such as Intel Xeon e5 2650 2.6GHz and 8 GB memory. The performance metrics are computed as average (AVG) and standard deviation (STDEV) of best solution till last iteration. Table 4.4 & Table 4.5 show the evaluation results of test benchmark functions (T1 – T7) and (T8 – T17) respectively. As per reported results, the GACO is more efficient algorithm for test functions T1, T2, T5 –T8, T10 – T14 and T16. Remaining functions produce competitive results. Results of each benchmark has been visualized in two spaces-parameter space and objective space. Figure 4.3 and Figure 4.5 shows the results of uni-modal and multi-model test benchmarks functions respectively.

4.3 Implementation of Map-Reduce Model for CIS

Map-Reduce technique provides an innovative idea to the rapid usage of very large-scale and complex data-sets processing work. This section has been implemented highly parallel Map-Reduce model for four different queries. A join operation is used to match two large-scale data-sets in Map-Reduce. However, this process encompasses high code volume to accomplish the concrete operation. If a data-set is smaller than other data-sets, then small data-sets are distributed in all data node in the cluster.

Table (4.2) Uni-model (T1-T7) and Multi-model (T8-T17) Test benchmark functions [122] [80]

Test functions	Dim	Range	f_{min}
$T1(x) = \sum_{k=1}^{N} x_k^2$	20	[-100,100]	0
$T2(x) = \sum_{k=1}^{N} \lvert x_k \rvert + \prod_{k=1}^{N} \lvert x_k \rvert$	20	[-10,10]	0
$T3(x) = \sum_{k=1}^{N} \left(\sum_{l=1}^{k} x_l \right)^2$	20	[-100,100]	0
$T4(x) = max_k \{ \lvert x_k \rvert ,\ 1 \leq k \leq N \}$	20	[-100,100]	0
$T5(x) = \sum_{k=1}^{N-1} \left[100 \left(x_{k+1} - x_k^2 \right)^2 + (x_k - 1)^2 \right]$	20	[-30,30]	0
$T6(x) = \sum_{k=1}^{N} \left(\lvert x_k + 0.5 \rvert \right)^2$	20	[-100,100]	0
$T7(x) = \sum_{k=1}^{N} k x_k^4 + rand[0,1]$	20	[-1.28,1.28]	0
$T8(x) = \sum_{k=1}^{N} -x_k sin \left(\sqrt{\lvert x_k \rvert} \right)$	20	[-500,500]	-418.9829×5
$T9(x) = \sum_{k=1}^{N} \left[x_k^2 - 10\cos(2\pi x_k) \ + 10 \right]$	20	[-5.12,5.12]	0
$T10(x) = -20exp \left(-0.2 \sqrt{\frac{1}{N} \sum_{k=1}^{N} x_k^2} \right) - exp \left(\frac{1}{N} \sum_{k=1}^{N} cos(2\pi x_k) \right) + 20 + e$	20	[-32,32]	0
$T11(x) = \frac{1}{4000} \sum_{k=1}^{N} x_k^2 - \prod_{k=1}^{N} cos \left(\frac{x_k}{\sqrt{k}} \right) + 1$	20	[-600,600]	0
$T12(x) = \frac{\pi}{N} \left\{ 10sin(\pi z_1) + \sum_{k=1}^{N-1} (z_k - 1)^2 \left[1 + 10sin^2(\pi z_{k+1}) \right] + (z_N - 1)^2 \right\} + \sum_{k=1}^{N} u(x_k, 10, 100, 4)$	20	[-50,50]	0
$z_k = 1 + \frac{x_k + 1}{4} \qquad u(x_k,\ p,\ q, r) = \begin{cases} q(x_k - p)^r & x_k > p \\ 0 & -p < x_k < p \\ q(-x_k - p)^r & x_k < -p \end{cases}$			
$T13(x) = 0.1 \left\{ sin^2(3\pi x_1) + \sum_{k=1}^{N} (x_k - 1)^2 \ \left[1 + sin^2(3\pi x_k + 1) \right] + (x_k - 1)^2 \ \left[1 + sin^2(2\pi x_k) \right] \right\} + \sum_{k=1}^{N} u(x_k,\ 5,\ 100,\ 4)$	20	[-50,50]	0
$T14(x) = -\sum_{k=1}^{N} \sin(x_k) \cdot \left(sin \left(\frac{k x_k^2}{\pi} \right) \right)^{2r},\ r = 10$	4	[0,π]	-4.687
$T15(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	[-5,5]]	-1.0316
$T16(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) cos x_1 + 10$	2	[-5,10]	0.398
$T17(x) = \left[1 + (x_1 + x_2 + 1)^2 * \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2 \right) \right]$ $* \left[30 + (2x_1 - 3x_2)^2 * \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2 \right) \right]$	3	[-2,2]	2

Table (4.3) Test parameters setting of algorithms

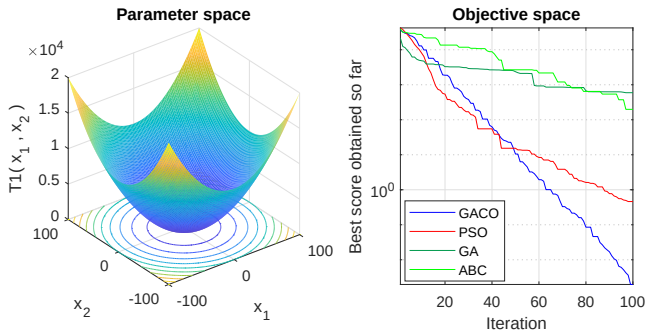
Algorithms	Parameters	Values
GACO	Number of Iteration	100
	Population Size	20
	Enforcement factor	0.1
PSO	Number of Iteration	100
	Number of Particles	20
	Inertia Coefficient	0.50
	Cognitive and Social Coefficient	1.7, 1.5
GA	Number of Iteration	100
	Population Size	20
	Mutation and Crossover	0.02, 0.8
ABC	Number of Iteration	100
	Population Size	20
	Acceleration Coefficient	1

Table (4.4) Results of uni-model test benchmark functions

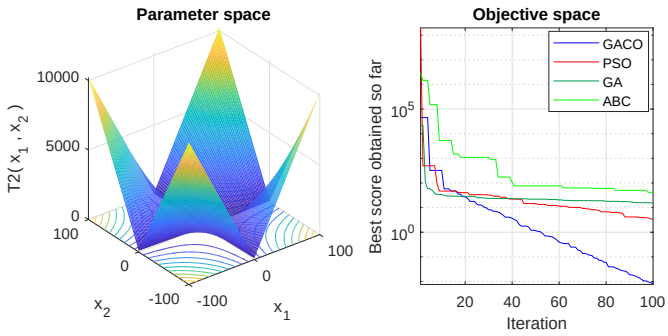
Performance Metrics		Test Benchmark Functions (Uni-model)						
		T1	T2	T3	T4	T5	T6	T7
AVG	GACO	**-0.62797019**	**0.000221190**	1.748032206	0.391470966	**0.455660092**	**-0.4997646**	**0.002744267**
	PSO	-0.02911386	0.364284527	**-0.13791292**	**-0.21619396**	0.637147652	-0.52166412	0.031880239
	GA	1.194915470	0.034654050	0.619005273	-3.924912955	1.289315143	1.62667031	0.018838751
	ABC	0.921683838	0.010837072	2.065509962	-4.078155224	1.932765879	-0.416151698	0.003724579
STDEV	GACO	**5.382147**	**0.000377**	20.67296	4.907896	**0.516724**	**0.010934**	**0.087702**
	PSO	0.18897	1.507345	**4.767657**	**2.910768**	0.868995	0.106138	0.222223
	GA	5.777799	0.356267	20.61802	22.43178	3.755351	6.344261	0.245649
	ABC	4.812106	0.991172	55.35147	34.39109	4.860732	5.04982	0.168650

Table (4.5) Results of Multi-model test benchmark functions

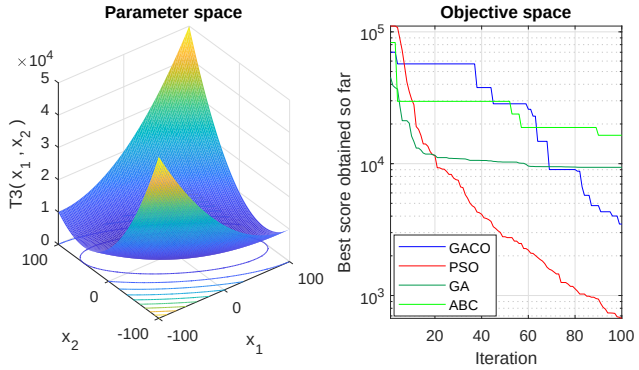
Performance Metrics		Test Benchmark Functions (Multi-model)									
		T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
AVG	GACO	**6.6361E+22**	0.07516	**0.00045**	**0.68084**	**-1.0027**	1.00593	-122610.8	2.26764	-43.38227	-0.41502
	PSO	22.62901	0.48929	0.04936	11.78129	-0.55434	**0.91878**	**-2.07970**	**0.03114**	0.27082	-0.05000
	GA	-16.19261	**0.18020**	-0.36855	0.80121	0.38154	1.26491	0.65199	0.45261	2.95325	0.08616
	ABC	-4.5215E+11	0.14332	-1.23789	-6.86156	1.82284	-1.00734	2.04154	-8.67197	**-10.8251**	**2.54838**
STDEV	GACO	**2.97E+23**	1.005825	**0.002141**	**3.85908**	**0.018907**	0.072945	969763.2	31.73859	254.8488	24.96636
	PSO	319.0099	1.279973	0.148833	43.55997	3.544811	**0.311512**	**5.449231**	**0.161662**	0.845367	0.223607
	GA	323.1886	**0.83332**	1.525809	19.47741	13.62174	4.460204	2.237045	2.940258	3.954134	0.940207
	ABC	2.02E+12	1.639552	2.609788	22.64074	7.838364	7.311913	54.12545	37.46899	**70.15135**	**12.33269**



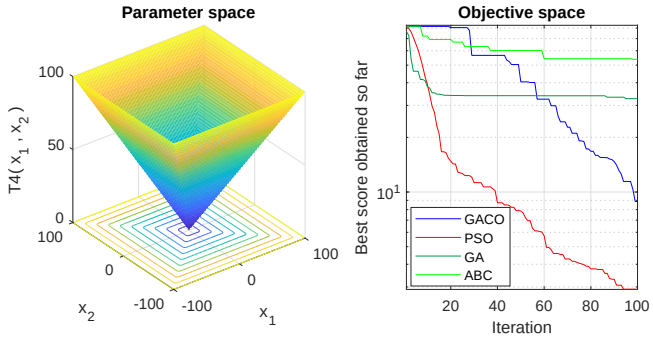
(a) T1



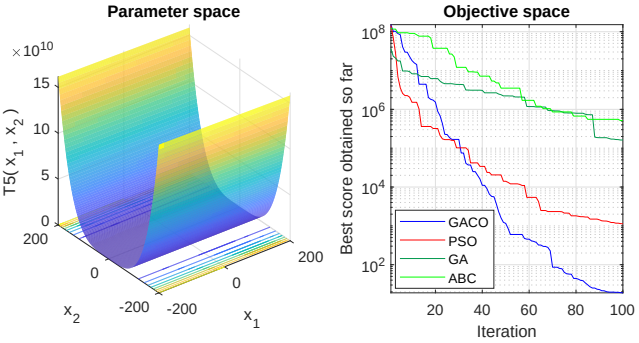
(b) T2



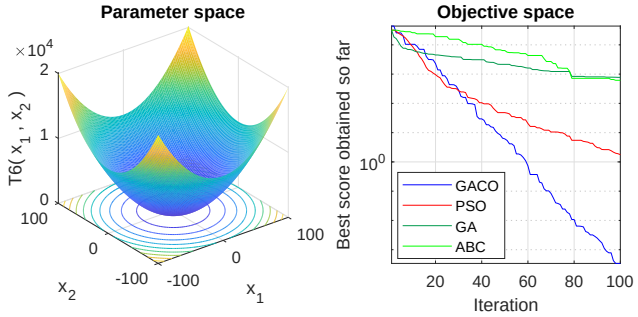
(c) T3



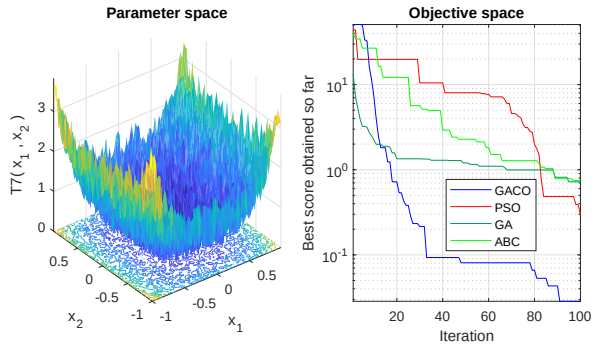
(d) T4



(e) T5

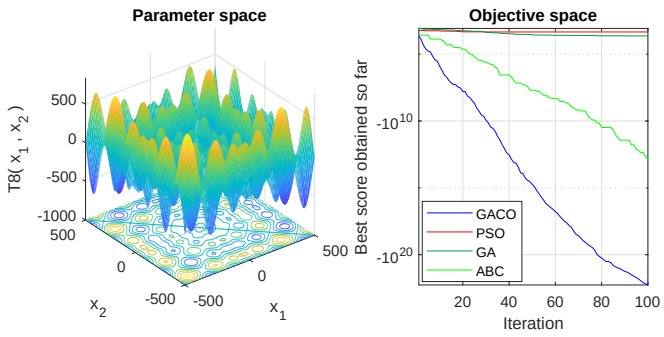


(f) T6

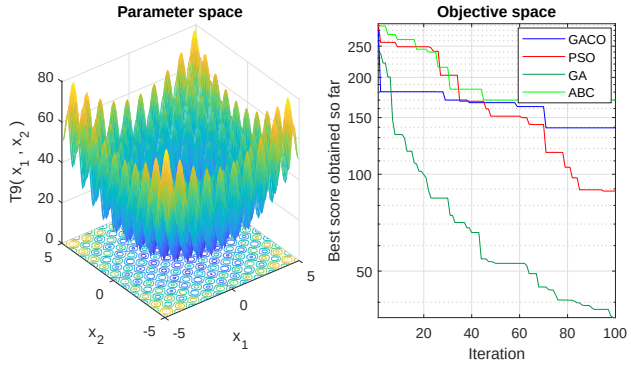


(g) T7

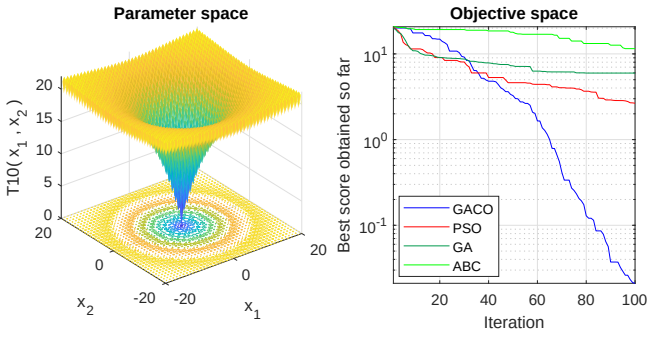
Figure (4.3) Results of uni-model test benchmark functions



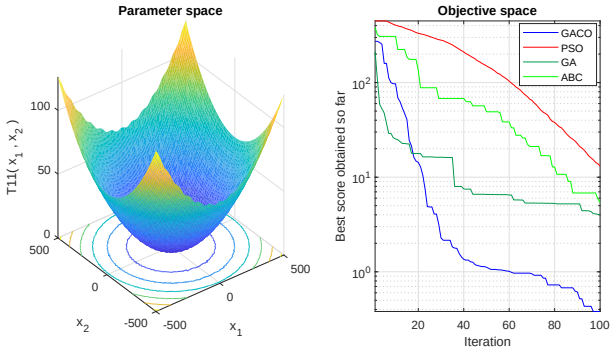
(a) T8



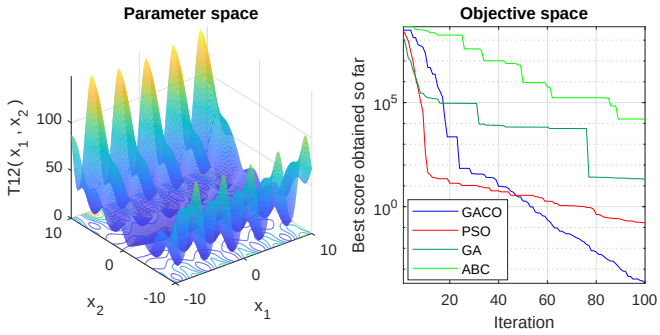
(b) T9



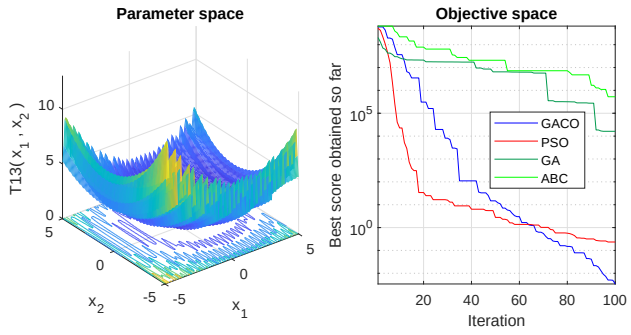
(c) T10



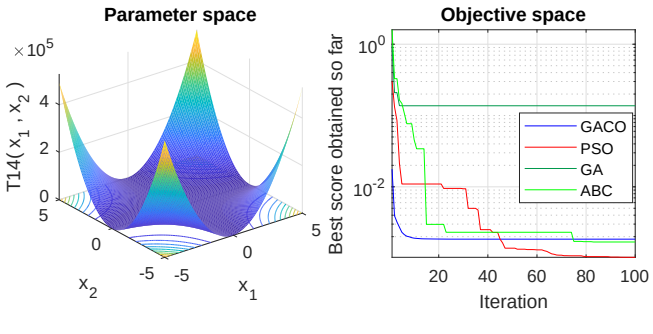
(d) T11



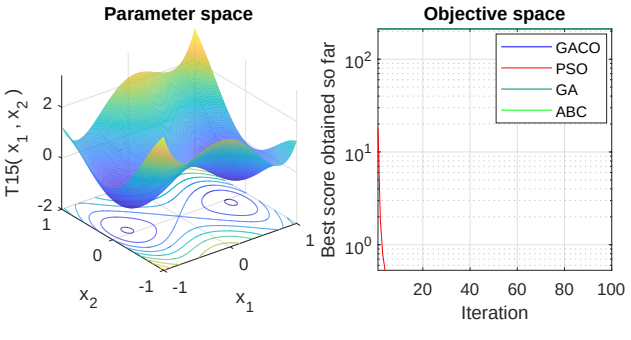
(e) T12



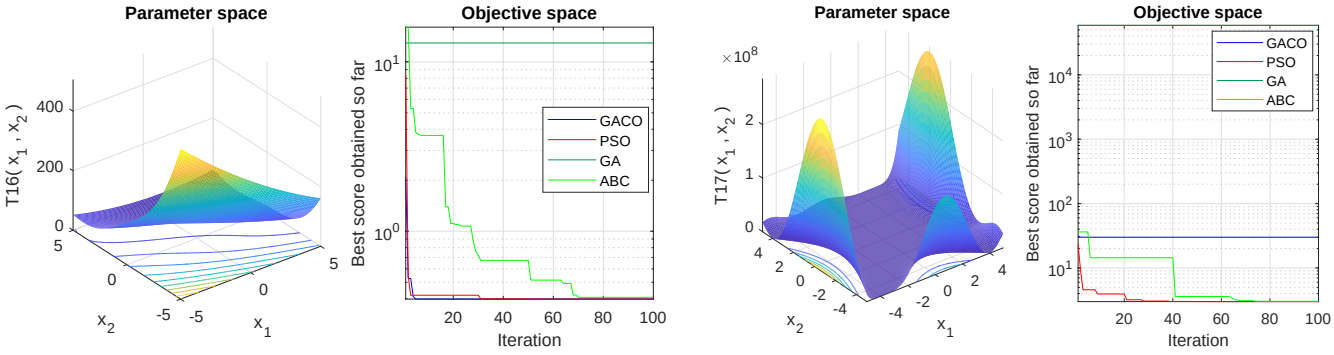
(f) T13



(g) T14



(h) T15



(i) T16 (j) T17

Figure (4.5) Results of multi-model test benchmark functions

Once it is distributed, Map-Reduce makes use of small data-sets to look for matching records from large data-sets and then combines that records and converted into output records. Here, two different data-sets having one common field as shown in Figure 4.6. The goal is to use Map-Reduce to combine two large data-sets. Only a few records are displayed in the Figure 4.6, the actual data-sets contains a million of records. Here, main purpose is that these two data sets have to be integrated through the Map-Reduce model. All concerned codes have been implemented in JAVA 1.8.0 programming language. There are certain steps as given below to implement this model successfully-

Step 1: Make a directory "dataset" inside hdhuser and Copy both data-sets.

Step 2: Uncompress the zip file with following command:

hduser_tietlab-VirtualBox:~$ sudo tar -xvf MapReduceJoin.tar.gz

Step 3: Select MapReduceJoid directory

hduser_tietlab-VirtualBox:~$ cd MapReduceJoin

hduser_tietlab-VirtualBox:~/MapReduceJoin$

Step 4: Now start the Hadoop

hduser_tietlab-VirtualBox:~/MapReduceJoin$HADOOP_HOME/sbin/start-dfs.sh

hduser_tietlab-VirtualBox:~/MapReduceJoin$HADOOP_HOME/sbin/start-yarn.sh

Step 5: Input both files (ServiceRequest.txt and Resource.txt)

$HADOOP_HOME/bin/hdfs dfs -copyFromLocal ServiceRequest.txt Resource.txt/

Step 6: Run the Map-Reduce program

$HADOOP_HOME/bin/hadoop jar MapReduceJoin.jar MapReduceJoin/JoinDriver/ServiceRequest.txt /Resource.txt /output_mapreducejoin

Step 7: After successfully executed output file (part-00000) will stored on HDFS (inside output_mapreducejoin directory)

$HADOOP_HOME/bin/hdfs dfs -cat /output_mapreducejoin/part-00000

4.4 Implementation Details and Interfaces

This section explains overall implementation details including test benchmark, testing parameters, data-sets. Some important implementation steps and print screen have also been highlighted in this section.

4.4.1 Hadoop Test Benchmark and Configuration Parameter Settings

The Hadoop is one of the popular open source framework for implementation of Map-Reduce tasks. It is developed by Apache and Yahoo! foundation. This proposed research work have been implemented in Hadoop 2.7 running on JDK 1.8.0 and performed all concerned experimental evaluations. Some parameter's value have been changed in Hadoop system which is shown in Table 4.6 and rests of are default configuration settings. WAN emulator has also been used to find the impact of dynamicity of DAIS framework. Besides, we configured the system to run two Map instances and a single Reduce instance simultaneously on each COS node. All the input and output data has been stored in Hadoop Distributed File System(HDFS) [156]. It uses a central job tracker and master HDFS demon to coordinate node activities. The central job tracker and master HDFS daemon of Hadoop system coordinate the COS node activities and to ensure that these daemons do not affect the performance of other active COS node. We found the 512 MB buffer size per COS

Table (4.6) Experiment environment and hadoop configuration parameters

Parameter	Description	Remarks
Work Station	DELL T5610	Intel CPU E5-2650, 16 GB RAM
WAN Emulator	TCS WANem (version 3.0 Beta 2)	
Storage nodes	One admin server, 15 COS nodes (each COS having 2TB storage capacity)	
Operating System	Ubuntu 18.04.1	Oracle VM VirtualBox
Databases	Hadoop and GeoMesa HBase	usage for implementation and comparative analysis
Coding language	Java SE 8	
Data block size	256 MB	
Maximum heap size	512 MB	Individual task executor JVM
Maximum heap size	1.5 GB	DataNode JVMs
Enable "Rack awareness"	True	It increases the fault tolerance
Enable block access token	True	For kind secure operation
Enable cross-origin support (CORS)	True	It supports all kind of web services
Methods allowed	GET and POST	
Enable ACL	True	Support for HDFS Access Control List (ACL)
Enable log aggregation	True	
Buffer size per node	512 MB	
Clusters size	250, 500 and 1000 nodes	
Tasks performed	Loading and integrating 10 millions recorded data-sets	
Concurrent request levels	20, 45, 70 and 100	
Data-sets	TPC-DI[139] and TPC-DS[131]	This thesis has used recorded data-sets of both standard specifications

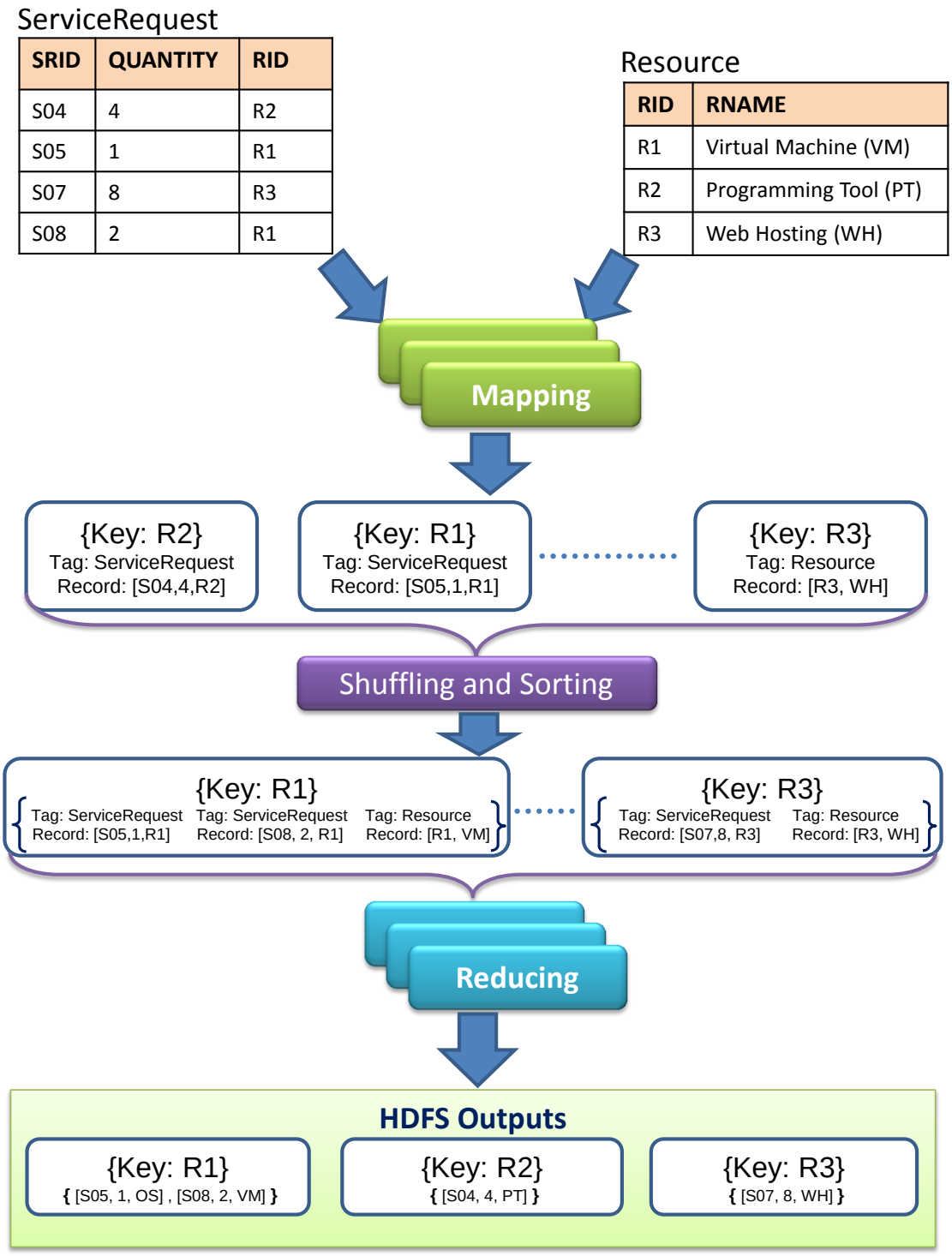


Figure (4.6) Map-Reduce Model for Join Operation

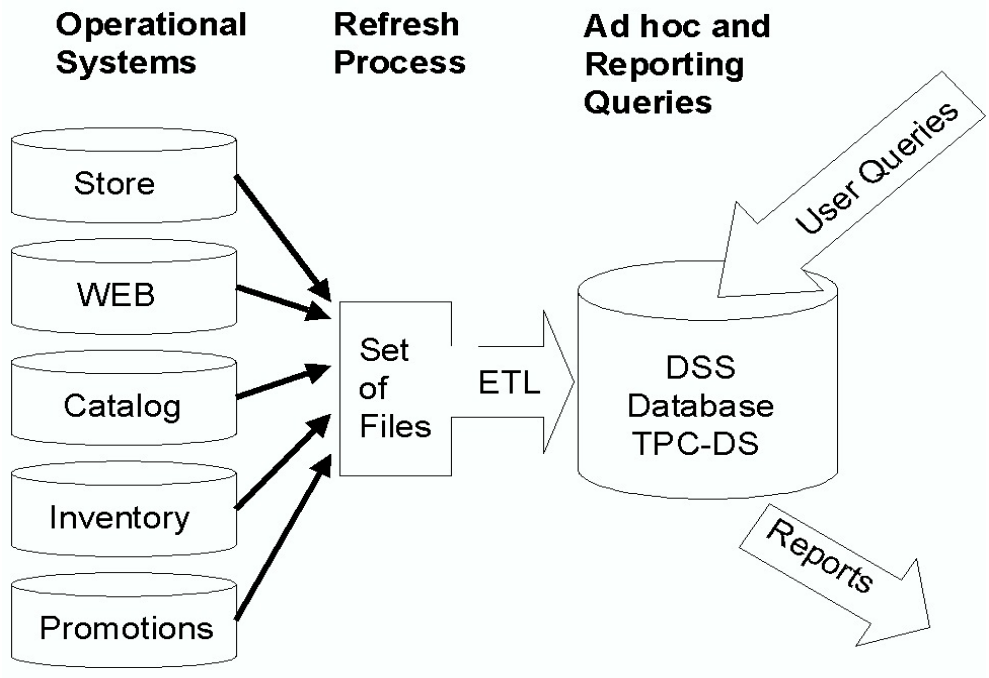
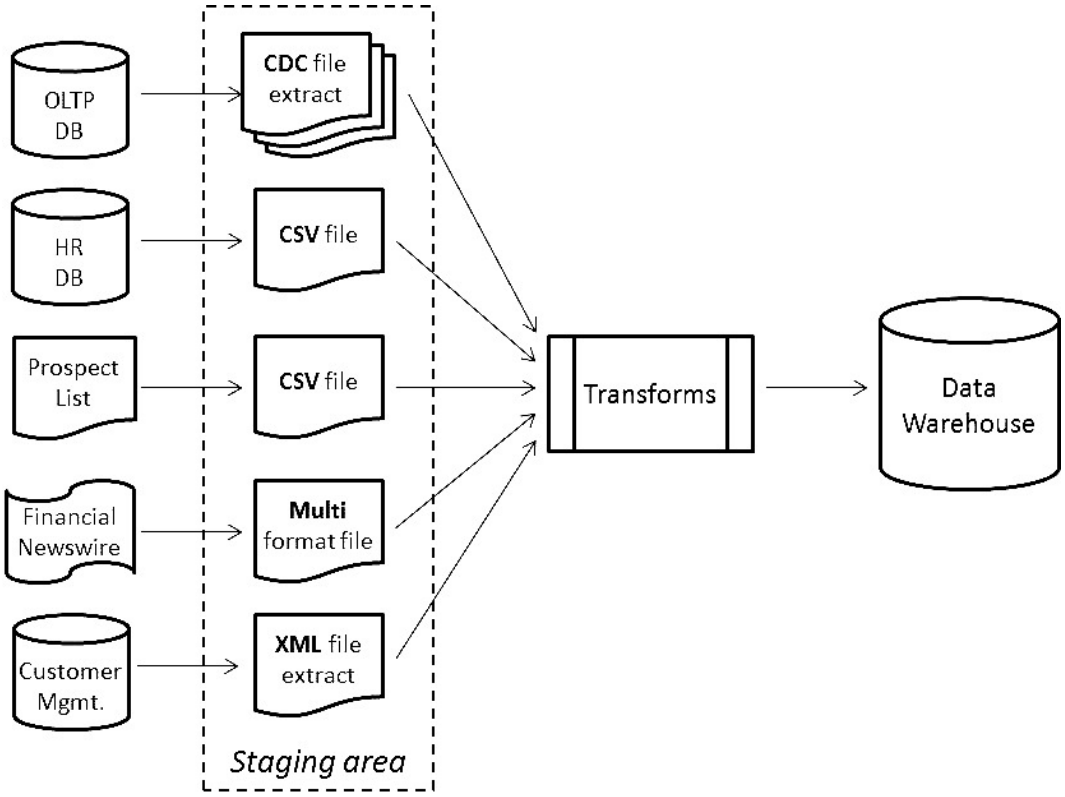


Figure (4.7) TPC-DS benchmark working model

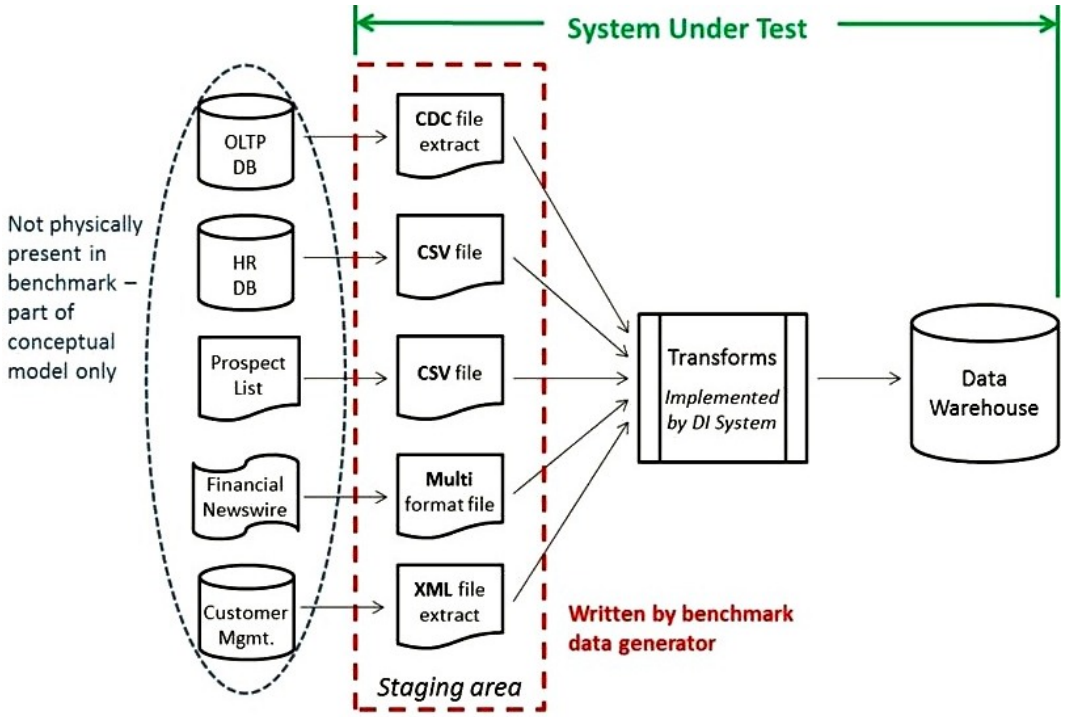
node is more effective for our experiment. Testing benchmarks have been deployed in Hadoop on the cluster of 1000 COS nodes.

4.4.2 TPC Benchmark and Data-sets

In this thesis, the use of Transaction Processing Performance Council (TPC) [41] benchmarks evaluate the performance of large scale data processing. This data processing can be referred as storage access, disk read/write, data transfer, data integration, system calls, etc. Three benchmarks have been used in this thesis- TPC-DS, TPC-DI and TPC-E. Table 4.7 explains brief ideas about these benchmarks. Figure 4.7 [189] and Figure 4.8 [139] illustrate the working layout of TPC-DS and TPC-DI benchmarks respectively. This thesis has been implemented 40 SQL queries which are already exist in these three TPC benchmarks. All these queries are related to four tasks including Loading, Selection, Searching and Joining. The purpose of these queries to evaluate the energy efficiency of the proposed DAIS framework which have been described in next chapter.



(a) TPC-DI overview



(b) TPC-DI system under test

Figure (4.8) TPC data integration layout

Table (4.7) TPC benchmarks

Benchmark	Version	Description	Ref.
TPC-DS	1.1.0	It is a type of decision support system that provides a model for data manipulation and queries. It simulates large amount of data such as Spark and Hadoop. It gives accurate performance report. It supports various operational systems including web storage, catalog, inventory and promotions. There are different ad-hoc and reporting queries have been executed in this benchmark.	[189]
TPC-DI	2.11.0	It is a performance test benchmark that transfer and integrate data from heterogeneous platform. It manipulates and loads large amount of data. Integrate different sources data with more accurate and reliable.	[139]
TPC-E	1.14.0	It provides online transaction processing workloads that simulates all the activities found during complex data processing. It supports multiple transactions simultaneously and moderate application execution time.	[35]

4.5 Conclusion

Overall design and implementation of proposed work has been exhibited in this chapter. Actually, DAIS is a layered architecture - exterior and interior. Four different algorithms have been designed for ACOT topology. All these algorithms have been implemented in JAVA and further their computational analysis has also been performed. the search time on ACOT overlay with m COS node is $O(log_{m-1}n)$. Insertion complexity is $O(log_k N)$, where k indicates the branching factors and N specifies the all active nodes in a complete domain as shown in Figure 4.1. In the worst case, one leaf node is placed to another sub domain and $2log_k N$–1 hopes will traverse. A novel meta-heuristic algorithm namely Generalized Ant Colony Optimizer (GACO) has been designed and implemented for SLA optimization. As per simulation results GACO performs better results in most of the cases.

Chapter 5

Testing and Results Analysis

Previous chapters discussed in detail the design and implementation of the proposed DAIS framework and its related components. Apart from this, DAIS framework has been tested on various aspects with three different networks-1) Grid Computing Lab, 2) Software Engineering Lab and 3) Creative Computing Lab of Thapar Institute. All the computers are connected through wireless access points. The Hadoop 2.7 and Java 1.8.0 have been used for concerned experimental evaluations. Mainly, this chapter highlights various test results of the proposed work in different performance metrics, including bandwidth, dynamic effect, average response time, SLA optimization and energy efficiency evaluation. The statistical analysis (trial and error approach) of experimental results of SLA optimization using GACO algorithm have been conducted in order to assess the 0.05 level of significance.

5.1 Bandwidth Evaluations

The main goal of DAIS framework is to cooperates full range of large-scale data access from different COS nodes which are geographically distributed. Firstly, the performance of DAIS framework has been measured in terms of their bandwidth utilization. In general, bandwidth specifies the amount of data transfer per unit of time. Here, data transfer time identifies the differ time between finish and start time. Time taken in resource discovery are not included for bandwidth evaluation. In this ex-

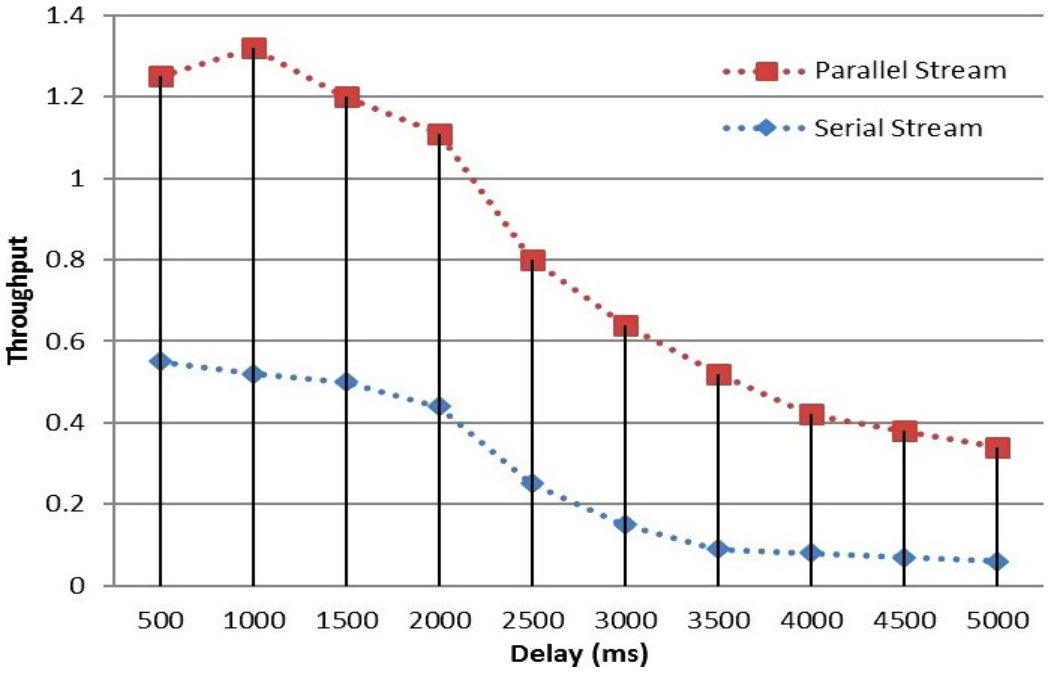


Figure (5.1) Bandwidth evaluation results (Delay vs. Throughput)

periment, all the information have been passed through WAN Emulator and scales correctly as per user demand. Overall, bandwidth evaluation has been conducted in two aspects: Delay vs. Throughput and Workload vs. Throughput. Average of 250 measurement performances have been reported in this experiment. The bandwidth evaluation results of Delay vs. Throughput and Workload vs. Throughput are visualized in Figure 5.1 and Figure 5.2 respectively. The bandwidth exceed the increase of multiple streams, but there is some anomaly. Figure 5.3 illustrates bandwidth evaluation of five different workloads (16MB, 64MB, 256MB, 1024MB and 2048MB) on parallel streams. If we increase the workloads, then bandwidth automatically grows in the same movement. It is interesting to note that the bandwidth for workload of 16 MB, 64 MB and 256 MB reaches the optimum value and then gradually falls. This obstacle can be easily handled by if the number of parallel streams have been increased as aforementioned i Figure 5.3.

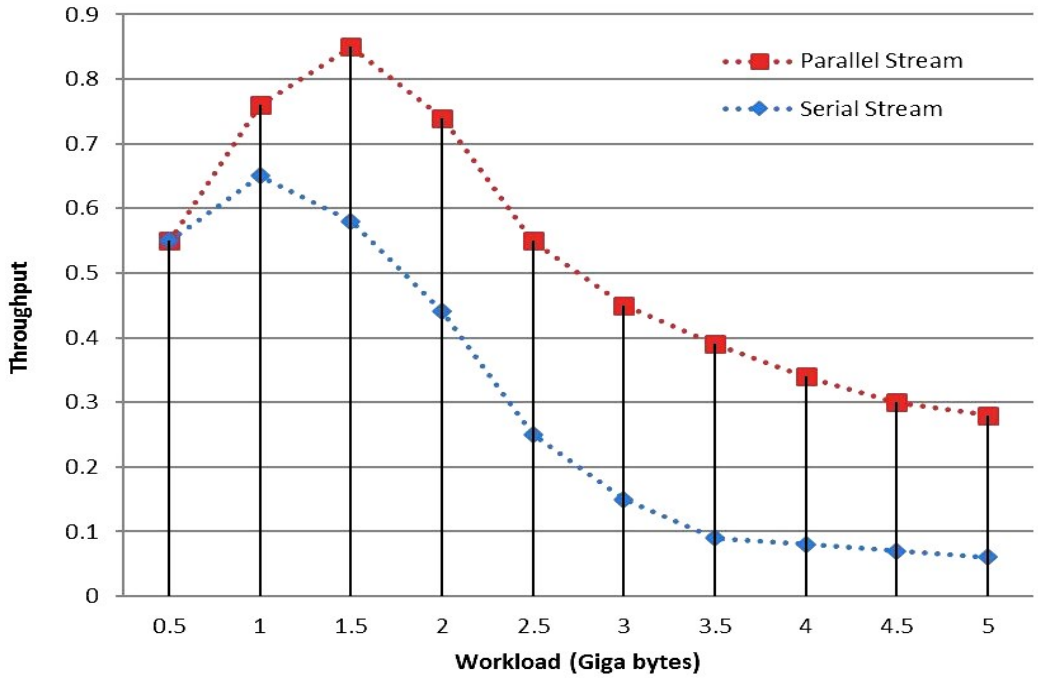


Figure (5.2) Bandwidth evaluation results (Workload vs. Throughput)

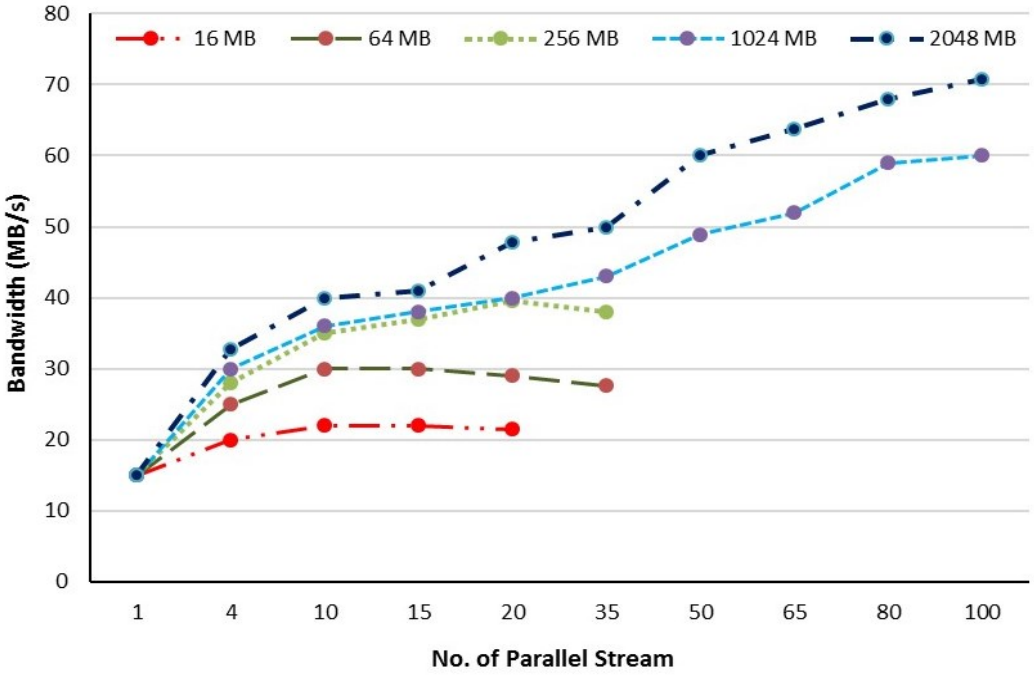


Figure (5.3) Bandwidth evaluation on five different workloads (16MB, 64MB, 256MB, 1024MB and 2048MB)

5.2 Dynamicity and Scalability Evaluation

Each experiment has been executed three times and results are reported the average of all three trials. First, we executed on single node to each experiment and then separately performed on three different cluster sizes (250, 500 and 1000 nodes) for measure the amount of data processed as well as how much time taken to perform the experiment. Results of experiment have been stored in the distributed file system and average result has been highlighted in this thesis. The time of reply is the critical metrics that analyze the performance of storage management in cloud environments. It specifies the time between user request submission and application delivered. Figure 5.4 shows the Average Response Time(ART) in four turnaround times (TT=50ms, TT=65ms, TT=80ms, and TT=100ms). The axis X is linear and axis Y indicates the logarithmic scale in millisecond(ms). Average response time increases linearly when the number of users are increased. Reason behind is that ART has been restricted to processing time for all CR domain. After increasing the number of users, it will automatically generate long queue results in the higher response time. Here, we found the growth of ART was also increase the turnaround time(TT).

The dynamic impact on cloud-oriented storage discovery of proposed DAIS framework has been investigated. Initially, turnaround time of WAN emulator was set to 50ms. We believe that DAIS has a negligible performance impact on the dynamic storage access. Figure 5.5 illustrates the comparative ART between static and dynamic DAIS frameworks. It reflects the performance impact of dynamicity of DAIS framework on storage discovery. All COS node can join or leave the domain any time as per ACOT rules which is already mentioned in Section 3.2. Queries traffic can be easily controlled and shared with multiple COS agents. Figure 5.6 compares the ART between static and dynamic COS storage allocation. Table 5.1 shows the consolidated overall performance report on various input parameters.

Table (5.1) Overall performance report on various input parameters

Input Parameters	Performance Metrics			
	ART	Dynamicity	TT	Speedup
Increase the no. of users	Increased	High	Increased	Normal
Statically increase the no. of COS nodes	Decreased	Low	Increased	Low
Dynamically increased the no. of COS nodes	Increased	High	Decreased	High
Increase the no. of resources shared by COS devices	Normal	High	Increased	Normal
Increase the no. of resources discovered by cloud user	Normal	High	Increased	Normal

5.3 SLA Violation Penalty Evaluation

The objective function mentioned in Equation 3.3 that maximizes the total profit if it is considering the cost of SLA violation along with underutilized resources. Equation 3.7 identifies the total number of allocated VM instances of each vm_j type which never exceed the number of available instances. Equation 3.8 recognizes the successful bids for each pair of resource type. Equation 2.1 inhibits all underutilized resources to be assigned to particular bid.

The SLA optimization problems has been resolved while taking all the constraints (mentioned in Equation 3.4 to 3.8) in this thesis. Here, we have defined set of all the possible bids orderings as search spaces that will be used by all meta-heuristic algorithms. Each bid will be converted to a solution and checked it will feasible or not. As per bid policy there must be at least one ordering that reflects the optimal solution. The proposed GACO algorithm has been implemented in MATLAB and executed 100 times to find the best solution. The comparative statistics of proposed GACO algorithm for SLA violation penalty in unidentified search spaces are reported in Table 5.2. Figure 5.7 shows the comparative best solution with other algorithms in

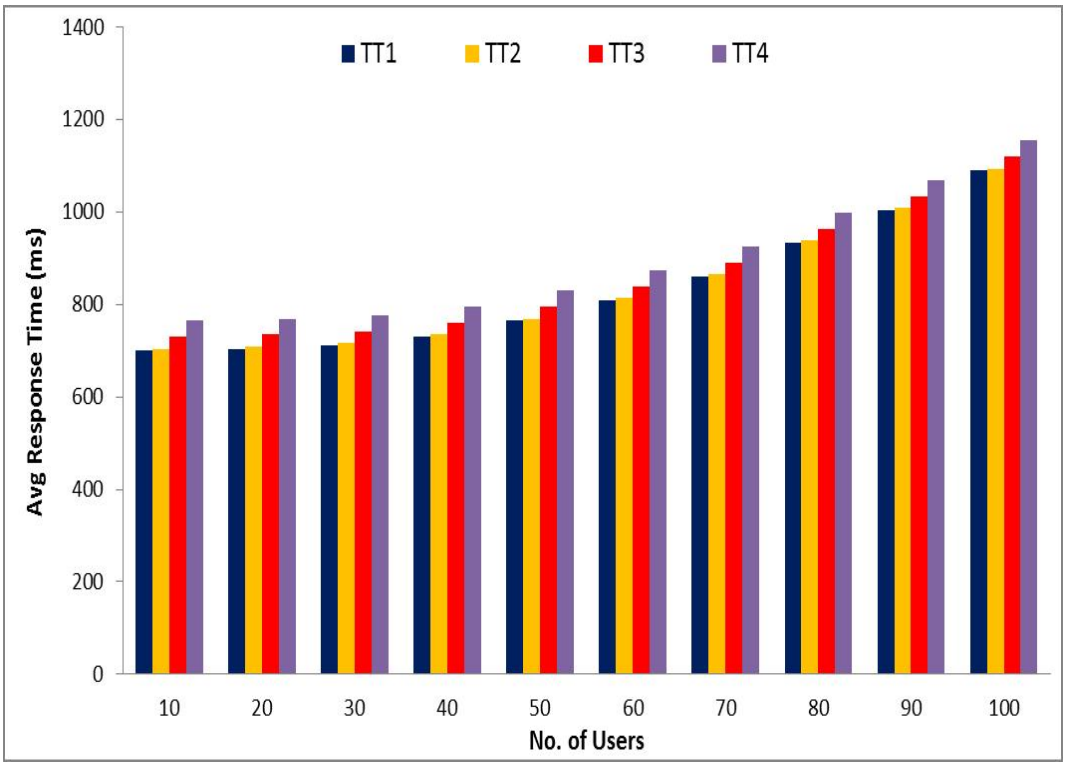


Figure (5.4) Average response time of proposed DAIS framework (in four turnaround time TT1=50ms, TT2=65ms, TT3=80ms, and TT4=100ms)

unidentified search spaces. Moreover, the results of the SLA optimization problems in cloud computing showed the better performance in unidentified search spaces. The results of this problem are improved as compared to existing techniques.

5.3.1 Wilcoxon Statistical Test Analysis

The comparison of algorithms does not guarantee the effectiveness and superiority of particular algorithm.The possibility of getting good results, by trial and error method, can't be ignored. In this regards, statistical analysis has also been performed in this thesis using seven CEC2015 [34] special section test benchmark functions. Table 5.3 shows the details of CEC2015 special section test benchmark functions. Test work is taken from the CEC2015 special session, which is the most challenging test benchmark functions in the literature. These functions are also known as black-box optimization, because it solves real parameter based single objective optimization without knowing

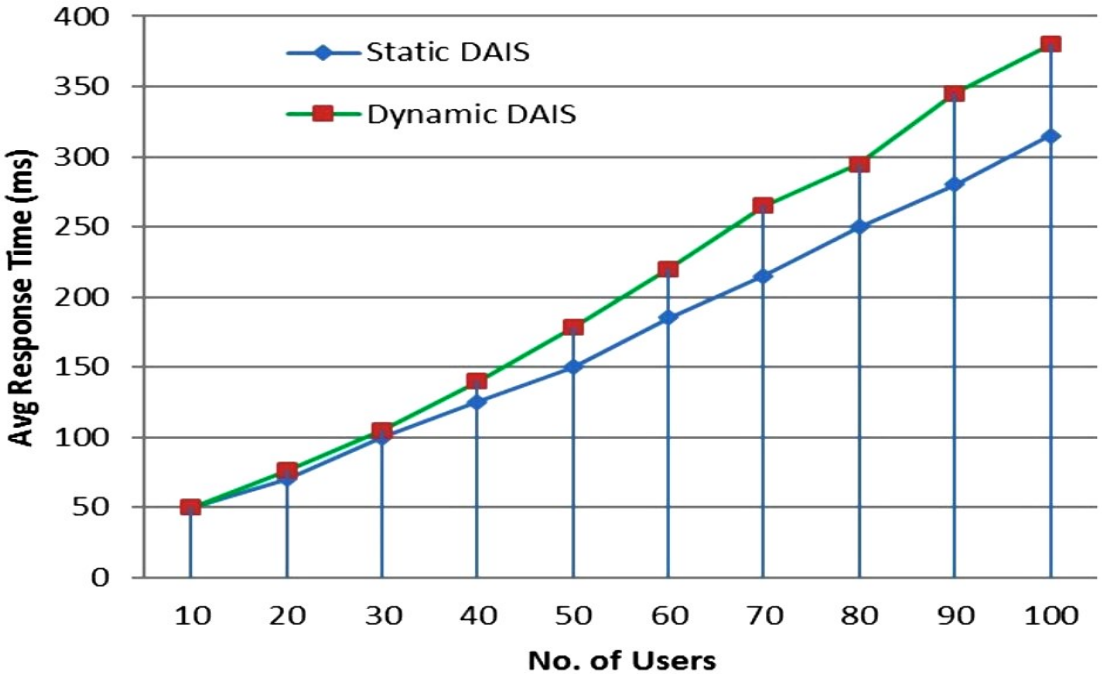


Figure (5.5) Average response time(Static vs. Dynamic DAIS Framework)

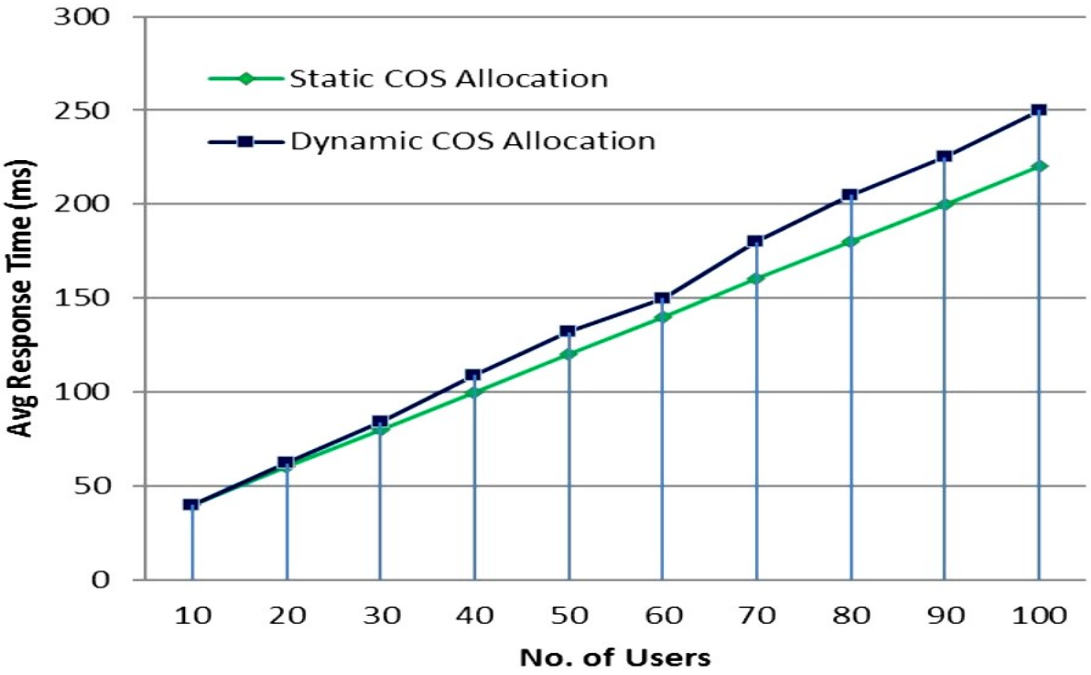


Figure (5.6) Average response time(static vs. dynamic COS allocation)

Table (5.2) Comparative statistics of GACO for SLA violation penalty optimization

Algorithms	Worst case	Best case	Average case	STDEV
GACO	0.013792	**43701.79**	2796.085	8053.906
PSO	0.342633	42021.53	3271.718	**8175.305**
GA	146.2593	23769.62	1347.263	3141.235
ABC	919.4537	39802.37	7302.713	7546.448

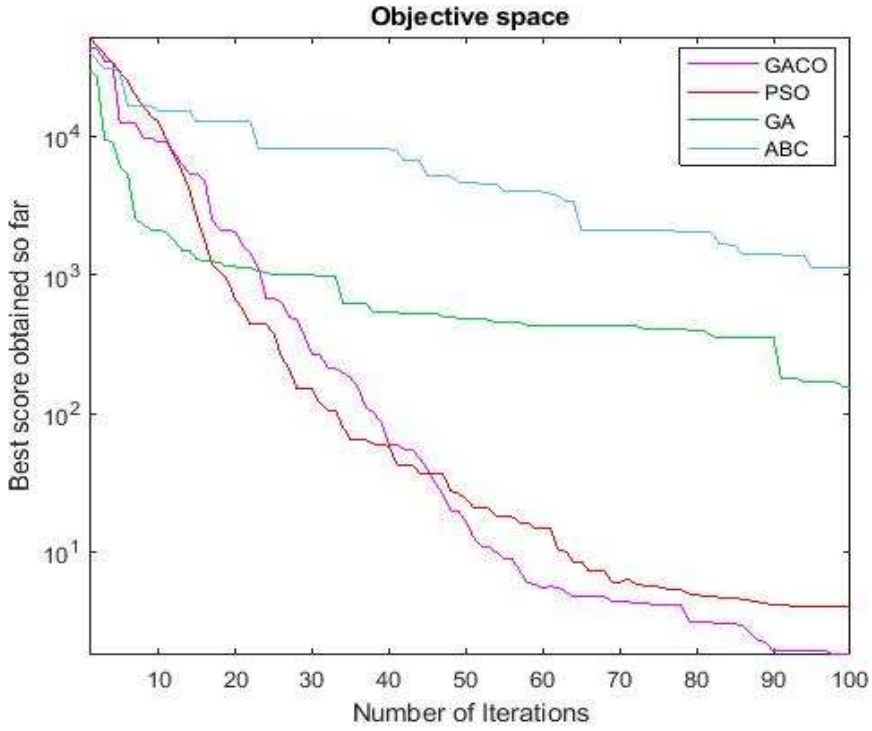


Figure (5.7) Comparative best solutions of GACO in unidentified search space

the equation of benchmark functions.

Table (5.3) CEC2015 test benchmark functions

Function No.	Function Name	Related basic function	Dim	f_{min}
CEC1	Rotated Bent Cigar	Bent Cigar function	20	100
CEC2	Rotated Discus	Discus	20	200
CEC3	Shift and Rotated Ackley	Ackley's function	20	300
CEC4	Shift and Rotated Schwefels	Schwefel's function	20	400
CEC5	Shift and Rotated Katsuura	Katsuura function	20	500
CEC6	Shift and Rotated HappyCat	HappyCat function	20	600
CEC7	Shift and Rotated HGBat	HGBat function	20	700

The Wilcoxon statistical test [172] has been performed at 0.05 level of significance. The obtained p-values from Wilcoxon statistical test are computed in Table 5.4. At the time of statistical analysis, we have compared all the four algorithms (GACO, PSO, GA and ABC) to each other. All those algorithms are demonstrated statically superiority whose p-value is less than 0.05. In addition, MannWhitney-U rank test has also been conducted on the average values of CEC-2015 special session benchmark functions. MannWhitney-U rank sum test results of statically superiority at 0.05 level of significance have been reported in Table 5.5. All the significant differences are denoted as- ns indicates for $p > 0.05$, '*' indicates for $p <= 0.05$, '**' indicates for $p <= 0.01$ and '***' indicates for $p <= 0.001$. A significant difference between two data sets has been observed by the MannWhitney-U ranks sum test.

5.4 Energy Efficiency Evaluation

In general, energy efficiency having many factors to consume energy such as storage server, network equipment, infrastructure related etc. This thesis consider only storage server related energy consumption as mentioned in Figure 2.2 . Efficiency analysis of proposed framework has been addressed in three units: workload, energy

Table (5.4) Obtained p-value from Wilcoxon statistical test

Function Name	GACO	PSO	GA	ABC
CEC1	0.01871	0.00061	0.00021	0.00201
CEC2	0.00113	0.01792	0.00661	0.00411
CEC3	0.00021	0.00193	0.00011	0.08403
CEC4	0.00070	0.03041	0.00262	0.01100
CEC5	0.00360	0.00031	0.03102	0.06301
CEC6	0.00380	0.00030	0.06571	0.00071
CEC7	0.00061	0.00091	0.00412	0.00560

Table (5.5) Statistical superiority of MannWhitney-U rank sum test (where ns denotes for $p > 0.05$, '*' denotes for $p <= 0.05$, '**' denotes for $p <= 0.01$ and '***' denotes for $p <= 0.001$)

Function Name	GACO	PSO	GA	ABC
CEC1	*	***	***	**
CEC2	***	*	**	**
CEC3	***	**	***	ns
CEC4	***	*	**	*
CEC5	**	***	*	ns
CEC6	**	***	ns	***
CEC7	***	***	**	**

efficiency and measurement [163].

5.4.1 Workload Unit

Let, $L(T)$ and $E(T)$ denotes the workload and energy efficiency of active COS node during time T. N and c_i represents the number of total COS and number of total active COS respectively, whereas $1 \leq i \leq N$. The frequency, power and usage of c_i node at time t are denoted by $f_i(t)$, $p_i(t)$ and $u_i(t)$ respectively. Hence, $L_i(t) = f_i(t) * u_i(t)$

and $E_i(t) = p_i(t)$. Similarly, $L(T)$ and $E(T)$ can be defined as follows:

$$L(T) = \sum_{i=1}^{N} \int_0^T f_i(t) * u_i(t)dt \tag{5.1}$$

$$E(T) = \sum_{i=1}^{N} \int_0^T p_i(t)dt \tag{5.2}$$

We assume $\eta(T)$ represents energy efficiency during time T and it can be calculated as:

$$\eta(T) = \frac{L(T)}{E(T)} \qquad where, E(T) \neq 0 \tag{5.3}$$

Hence, by equation 5.1, 5.2 and 5.3, we can calculate energy efficiency as:

$$\eta_i(t) = \frac{f_i(t) * u_i(t)}{p_i(t)} \tag{5.4}$$

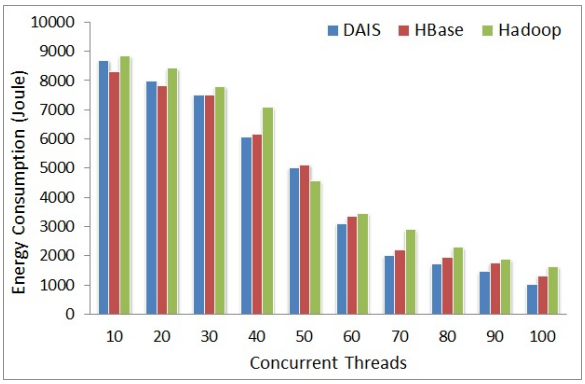
and

$$\eta(T) = \frac{\sum_{i=1}^{N} \int_0^T f_i(t) * u_i(t)dt}{\sum_{i=1}^{N} \int_0^T p_i(t)dt} \tag{5.5}$$

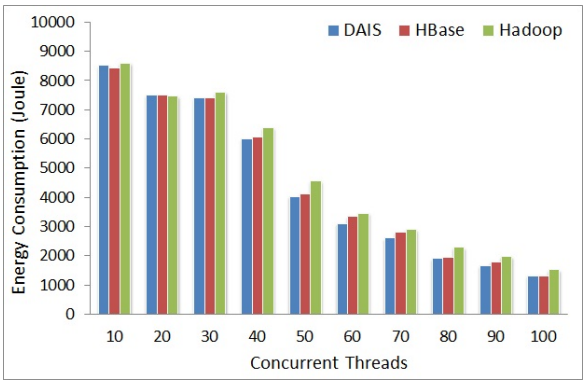
In general, Million Instruction Per Second (MIPS) or Floating Point Operations Per Second (FLOPS) is often used to evaluates energy effectiveness of computer systems, but watt or power is used to evaluates consumed energy by respective cloud storage system or any peripheral nodes. There is no specific algorithm to evaluate energy efficiency of a storage system. At the present time, most algorithms have focused on either time complexity or space complexity. Now focusing on energy efficiency has become very important.

5.4.2 Energy Efficiency and its Measurement

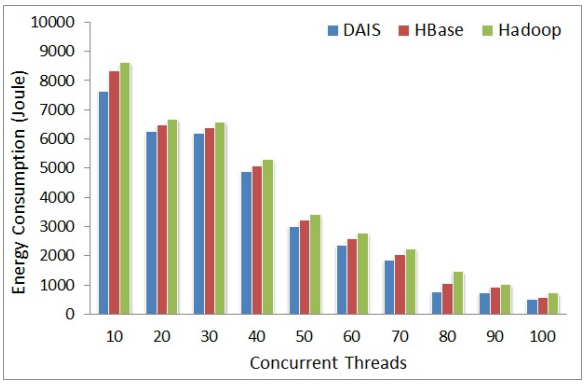
This thesis evaluates the energy consumption and efficiency of three most popular storage approaches like Hadoop, HBase and DAIS. Experiment has been conducted on loading of 10 millions recorded data-set (TPC-DI[139] and TPC-DS[131]). Data is averagely distributed on each COS nodes. Four different tasks (Loading, Selection, Searching and Joining) have been executed in this experiment. Energy efficiency of



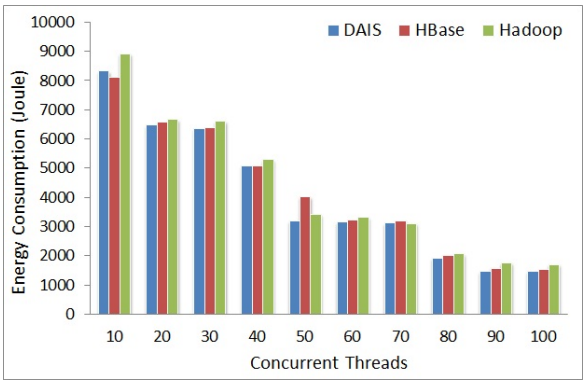
(a) Loading



(b) Selection



(c) Searching



(d) Joining

Figure (5.8) Energy consumption on various tasks

DAIS framework has been compared with two other popular approaches including Hadoop and HBase. Experiment results of each task has been reported separately as shown in Figure 5.8. It also shows that the energy consumption of DAIS is less than HBase and Hadoop in most of the cases. Such superiority prevails even when the number of concurrent requests increases.

Apart from this Figure 5.9 illustrates the average energy efficiency per COS node of all four tasks. As per experiment results proposed work performs better on higher concurrent threads. For better understanding some critical results (maximum and minimum) are reported in Table 5.6. It shows that the average energy efficiency of all four tasks between $6m\eta$ to $10m\eta$. This result also proves that the proposed DAIS framework is less complex than HBase and Hadoop.

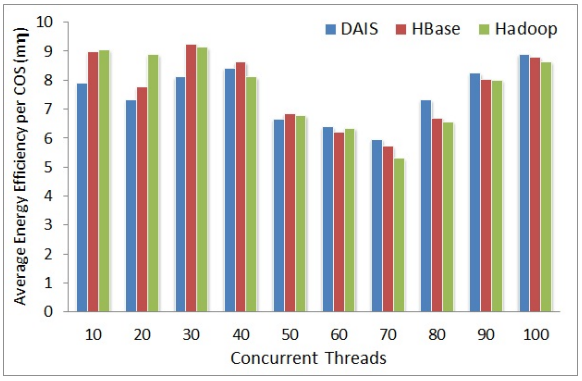
Table (5.6) Average Energy Efficiency (maximum and minimum values in $m\eta$) of four tasks-Loading, Selection, Searching and Joining

Approaches		Tasks			
		Loading	Selection	Searching	Joining
DAIS	Min	5.975001	6.200134	5.200032	5.710005
	Max	8.902012	8.555556	7.722222	7.931428
HBase	Min	5.722503	6.210005	5.210034	5.268945
	Max	9.258751	8.777778	8.803001	7.917695
Hadoop	Min	5.312404	5.576667	4.710107	5.067400
	Max	9.145045	8.900000	8.303000	8.865650

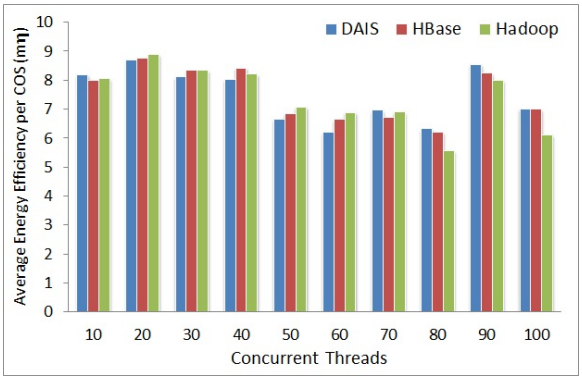
5.5 Conclusion

In this chapter, DAIS framework has been tested in Hadoop 2.7 with Java 1.8.0 environment. Mainly, this chapter has been performed various test along with results analysis. Results are computed in different performance metrics including bandwidth evaluation, impact of dynamicity, average response time, SLA optimization and energy efficiency evaluation.

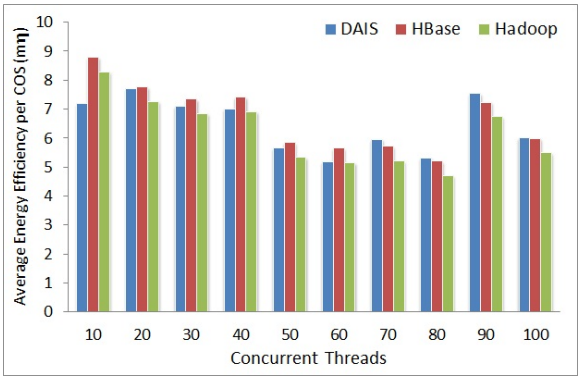
Apart from above, the MannWhitney-U rank sum statistical test (trial and error) of SLA optimization have been conducted in order to assess the 0.05 level of significance. Results are reported on seven special session benchmark (CEC-2015) along with competitors. As per reported results, 95% values are accessed under significance level.



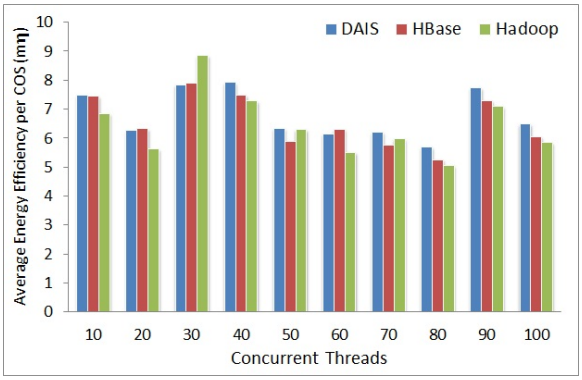
(a) Loading



(b) Selection



(c) Searching



(d) Joining

Figure (5.9) Average energy efficiency of various tasks

Chapter 6

Conclusions and Future Scope

This chapter provides the concluding remarks on the work carried out in this thesis. It highlights overall contributions and outcomes of this research work. It also addresses remains major challenges for cloud storage management. To demonstrate the validity, applicability and complexity of the proposed framework and algorithms. This chapter start with describing the major contributions then discuss outcomes in this research work. At the end, future scope of this research work has been discussed.

6.1 Conclusions

The thesis titled "Dynamic and Scalable Data Access and Integration Services in Cloud Computing Environment" focuses the cloud-oriented storage management issues including dynamicity, scalability, self-organizing, Quality of Service (QoS) and integration services. Following are major contributions of this thesis:

Chapter 2 addresses the comprehensive literature reviews based on prevailing models and applications related to cloud-oriented storage systems. Comparative analysis of 20 well-known existing cloud storage systems and 6 most popular service providers have been explained in chapter 2. The meta-heuristic-based optimization approaches and self-organizing models for cloud storage management has been well explained.

In chapter 3 a unique framework named DAIS has been proposed which provides dynamic data access and integration services in cloud computing environment.

DAIS framework has four major components- Adjacency COS Overlay Topology (ACOT), Cloud Storage Resource Discovery (CSRD), COS Integration Service (CIS) and Cloud Usage Monitor (CUM). Each of these four components have been explained thoroughly in this chapter. The various architectural diagrams of DAIS framework and its components has also been visualized in well manner. A highly flexible and self-organizing topology named ACOT has also been proposed in this thesis. DAIS framework removes the centralized and static approaches for cloud-oriented storage systems. The Cloud Storage Resource Discovery (CSRD) module is one of the major component of DAIS framework that provides integrated storage discovery in cloud environment.

Overall design and implementation of proposed work have been explained in chapter 4 . Various algorithms have been analyzed and designed in this chapter. Each algorithm has been thoroughly computationally analyzed and evaluated its searching cost and time. A novel swarm-based meta-heuristic algorithm named Generalized Ant Colony Optimizer (GACO) has been proposed and well designed in this chapter. GACO has tested on 17 well-known benchmarks test functions and compared with three most popular algorithms.

The proposed DAIS framework has been implemented in JAVA 1.8.0 and thoroughly tested in Hadoop 2.7 environment. Performance, bandwidth and energy consumption are some of the major metrics that have been evaluated in this thesis. To test energy efficient, measurement unit and mathematical model have been used. Energy consumption results have been compared with two well-known approaches HBase and Hadoop-MR.

Overall the Quality of Service has also been enhanced in order to data accessibility level, on-demand services, data integration services and Service Level Agreement.

6.2 Future Scope

In this thesis, the cloud storage problem has been resolved to some extent, even there are some challenges and open issues in it, which will be extended in the future work.

a. **IoTs-based applications:** Rapid growth of IoTs devices that will attract to cloud computing not only because they must be highly available. All these devices generate huge amount of data every seconds which are very difficult to store and manage such data-sets.

b. **The growth of analytical application:** Today's most of the organizations usage web analytics services to enhance their business. A growing of such web analytic data is directed to large-scale data-sets. Such data is more useful for business purpose but analysis of that data is very difficult task.

c. **Parallel database management system** for product line business applications. It will be reduce the number of servers and storage resources.

d. **Backend support for intensive computational services** that keep large amount of data in the cloud and rely on sufficient bandwidth for service execution.

e. **High availability ultra-large data** storage in the cloud environment.

References

[1] Daniel J Abadi, Peter A Boncz, and Stavros Harizopoulos. Column-oriented database systems. *Proceedings of the VLDB Endowment*, 2(2):1664–1665, 2009.

[2] Monir Abdullah and Mohamed Othman. Simulated annealing approach to cost-based multi-quality of service job scheduling in cloud computing enviroment. *American Journal of Applied Sciences*, 11(6):872, 2014.

[3] Laith Mohammad Abualigah, Ahamad Tajudin Khader, and Essam Said Hanandeh. Hybrid clustering analysis using improved krill herd algorithm. *Applied Intelligence*, 48(11):4047–4071, 2018.

[4] Laith Mohammad Abualigah, Ahamad Tajudin Khader, and Essam Said Hanandeh. A hybrid strategy for krill herd algorithm with harmony search algorithm to improve the data clustering?? *Intelligent Decision Technologies*, (Preprint):1–12, 2018.

[5] Laith Mohammad Qasim Abualigah. *Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering*, volume 816. Springer, 2018.

[6] Laith Mohammad Qasim Abualigah and Essam S Hanandeh. Applying genetic algorithms to information retrieval using vector space model. *International Journal of Computer Science, Engineering and Applications*, 5(1):19, 2015.

[7] Mushtaq Ahmed, Kunwar Pal, and MC Govil. Comparative analysis of new hybrid approach for overlay construction in p2p live streaming. In *International Conference on Emerging Research in Computing, Information, Communication and Applications*, pages 239–250. Springer, 2016.

[8] Mushtaq Ahmed, Kunwar Pal, and MC Govil. Utilization-based hybrid overlay for live video streaming in p2p network. In *Recent findings in intelligent computing techniques*, pages 331–338. Springer, 2019.

[9] Norman Ahmed and Bharat K Bhargava. Towards dynamic qos monitoring in service oriented architectures. In *CLOSER*, pages 163–171, 2015.

[10] Ashish Ahuja, Sanjoy Das, and Anil Pahwa. An ais-aco hybrid approach for multi-objective distribution system reconfiguration. *IEEE transactions on power systems*, 22(3):1101–1111, 2007.

[11] Talal Alsedairy, Yinan Qi, Ali Imran, Muhammad Ali Imran, and Barry Evans. Self organising cloud cells: a resource efficient network densification strategy. *Transactions on Emerging Telecommunications Technologies*, 26(8):1096–1107, 2015.

[12] EC Amazon. Amazon web services. *Available in: http://aws. amazon. com/es/ec2/(November 2012)*, 2015.

[13] EC Amazon. Xeround cloud database for mysql applications. *Available in: https://aws.amazon.com/marketplace/pp/B0085GVT7O(November 2012)*, 2015.

[14] Artur Andrzejak, Sven Graupner, Vadim Kotov, and Holger Trinks. Algorithms for self-organization and adaptive service placement in dynamic distributed systems. 2002.

[15] Arcitura. Cloud usage monitor. *Available in: https : //patterns.arcitura.com/cloud − computing − patterns/mechanisms/cloud$_u$sage$_m$onitor (Accessed on: July 10, 2019)*, 2019.

[16] Tapas Badal, Neeta Nain, and Mushtaq Ahmed. Online multi-object tracking: multiple instance based target appearance model. *Multimedia Tools and Applications*, 77(19):25199–25221, 2018.

[17] Luyi Bai, Li Yan, and ZM Ma. Determining topological relationship of fuzzy spatiotemporal data integrated with xml twig pattern. *Applied intelligence*, 39(1):75–100, 2013.

[18] Qinghai Bai. Analysis of particle swarm optimization algorithm. *Computer and information science*, 3(1):180, 2010.

[19] Thar Baker, Omer F Rana, Radu Calinescu, Rafael Tolosana-Calasanz, and José Ángel Bañares. Towards autonomic cloud services engineering via intention workflow model. In *International Conference on Grid Economics and Business Models*, pages 212–227. Springer, 2013.

[20] Veronica Barassi and Emiliano Treré. Does web 3.0 come after web 2.0? deconstructing theoretical assumptions through practice. *New media & society*, 14(8):1269–1285, 2012.

[21] Christopher D Bienko, Marina Greenstein, Stephen E Holt, Richard T Phillips, et al. *IBM Cloudant: Database as a Service Advanced Topics*. IBM Redbooks, 2015.

[22] Ken Birman, Gregory Chockler, and Robbert van Renesse. Toward a cloud computing research agenda. *ACM SIGACt News*, 40(2):68–80, 2009.

[23] Spyros Blanas, Jignesh M Patel, Vuk Ercegovac, Jun Rao, Eugene J Shekita, and Yuanyuan Tian. A comparison of join algorithms for log processing in mapreduce. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 975–986. ACM, 2010.

[24] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[25] Dejene Boru, Dzmitry Kliazovich, Fabrizio Granelli, Pascal Bouvry, and Albert Y Zomaya. Energy-efficient data replication in cloud computing datacenters. *Cluster computing*, 18(1):385–402, 2015.

[26] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. On the integration of cloud computing and internet of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 23–30. IEEE, 2014.

[27] Stuart A Boyer. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[28] Aron Brand. Storage device and method thereof for integrating network attached storage with cloud storage services, April 4 2017. US Patent 9,614,924.

[29] Marcos M Campos and Gail A Carpenter. S-tree: self-organizing trees for data clustering and online vector quantization. *Neural Networks*, 14(4-5):505–525, 2001.

[30] Bogdan Alexandru Caprarescu, Nicolo Maria Calcavecchia, Elisabetta Di Nitto, and Daniel J Dubois. Sos cloud: Self-organizing services in the cloud. In *International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, pages 48–55. Springer, 2010.

[31] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[32] Artem Chebotko, Andrey Kashlev, and Shiyong Lu. A big data modeling methodology for apache cassandra. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 238–245. IEEE, 2015.

[33] I-Pao Chen. Flash storage device, data storage system, and data writing method, February 19 2013. US Patent 8,380,919.

[34] Q Chen, B Liu, Q Zhang, J Liang, P Suganthan, and B Qu. Problem definitions and evaluation criteria for cec 2015 special session on bound constrained single-objective computationally expensive numerical optimization. *Technical Re-*

port, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, 2014.

[35] Shimin Chen, Anastasia Ailamaki, Manos Athanassoulis, Phillip B Gibbons, Ryan Johnson, Ippokratis Pandis, and Radu Stoica. Tpc-e vs. tpc-c: characterizing the new tpc-e benchmark via an i/o comparison study. *ACM SIGMOD Record*, 39(3):5–10, 2011.

[36] Bo Cheng. Hierarchical cloud service workflow scheduling optimization schema using heuristic generic algorithm. *Przeglad Elektrotechniczny*, 88(2012):92–95, 2012.

[37] Kristina Chodorow. *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. " O'Reilly Media, Inc.", 2013.

[38] Yeongho Choi and Yujin Lim. Optimization approach for resource allocation on cloud computing for iot. *International Journal of Distributed Sensor Networks*, 2016:23, 2016.

[39] CloudOYE. Cloud usage monitor. *Available in: https : //www.cloudoye.com/wiki/c/cloud − usage − monitor (Accessed on: July 15, 2019)*, 2019.

[40] Brian F Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, 1(2):1277–1288, 2008.

[41] Transaction Processing Performance Council. Transaction processing performance council. *Web Site, http://www. tpc. org*, 2005.

[42] Yong Cui, Ningwei Dai, Zeqi Lai, Minming Li, Zhenhua Li, Yuming Hu, Kui Ren, and Yuchi Chen. Tailcutter: Wisely cutting tail latency in cloud cdns under cost constraints. *IEEE/ACM Transactions on Networking*, 27(4):1612–1628, 2019.

[43] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[44] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS operating systems review*, volume 41, pages 205–220. ACM, 2007.

[45] Ganesh Chandra Deka. A survey of cloud database systems. *IT Professional*, 16(2):50–57, 2014.

[46] Yuhui Deng and Frank Wang. A heterogeneous storage grid enabled by grid service. *ACM SIGOPS Operating Systems Review*, 41(1):7–13, 2007.

[47] Gaurav Dhiman and Vijay Kumar. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 114:48–70, 2017.

[48] Gianni Di Caro and Marco Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In *International Conference on Parallel Problem Solving from Nature*, pages 673–682. Springer, 1998.

[49] Marco Dorigo and Mauro Birattari. Ant colony optimization. In *Encyclopedia of machine learning*, pages 36–39. Springer, 2011.

[50] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[51] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[52] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 congress on*, volume 2, pages 1470–1477. IEEE, 1999.

[53] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new metaheuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 1470–1477. IEEE, 1999.

[54] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[55] Marco Dorigo and Thomas Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of metaheuristics*, pages 250–285. Springer, 2003.

[56] Georgios Drakopoulos, Aikaterini Baroutiadi, and Vasileios Megalooikonomou. Higher order graph centrality measures for neo4j. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6. IEEE, 2015.

[57] Shu Du, Ahamed Khan, Santashil PalChaudhuri, Ansley Post, Amit Kumar Saha, Peter Druschel, David B Johnson, and Rudolf Riedi. Safari: A self-organizing, hierarchical architecture for scalable ad hoc networking. *Ad Hoc Networks*, 6(4):485–507, 2008.

[58] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[59] Alex Feinberg. Project voldemort: Reliable distributed storage. In *Proceedings of the 10th IEEE International Conference on Data Engineering*, 2011.

[60] Iztok Fister Jr, Xin-She Yang, Iztok Fister, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

[61] Jun-Song Fu, Yun Liu, Han-Chieh Chao, Bharat K Bhargava, and Zhen-Jiang Zhang. Secure data storage and searching for industrial iot by integrating fog

computing and cloud computing. *IEEE Transactions on Industrial Informatics*, 14(10):4519–4528, 2018.

[62] Amir Hossein Gandomi, Xin-She Yang, and Amir Hossein Alavi. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, 29(1):17–35, 2013.

[63] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Yang Hou, and Liang Liu. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8):1230–1242, 2013.

[64] Zong Woo Geem, Joong Hoon Kim, and Gobichettipalayam Vasudevan Loganathan. A new heuristic optimization algorithm: harmony search. *simulation*, 76(2):60–68, 2001.

[65] Garth A Gibson and Rodney Van Meter. Network attached storage architecture. *Communications of the ACM*, 43(11):37–45, 2000.

[66] Hector Gonzalez, Alon Halevy, Christian S Jensen, Anno Langen, Jayant Madhavan, Rebecca Shapley, and Warren Shen. Google fusion tables: data management, integration and collaboration in the cloud. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 175–180. ACM, 2010.

[67] Suji Gopinath and Elizabeth Sherly. A dynamic replica factor calculator for weighted dynamic replication management in cloud storage systems. *Procedia computer science*, 132:1771–1780, 2018.

[68] Julie Greensmith and Uwe Aickelin. The deterministic dendritic cell algorithm. In *International Conference on Artificial Immune Systems*, pages 291–302. Springer, 2008.

[69] Jyotsana Grover and Madasu Hanmandlu. New evolutionary optimization method based on information sets. *Applied Intelligence*, pages 1–17, 2018.

[70] Min Gu, Xiangping Li, and Yaoyu Cao. Optical storage arrays: a perspective for future big data storage. *Light: Science & Applications*, 3(5):e177, 2014.

[71] Yunhong Gu and Robert L Grossman. Lessons learned from a year's worth of benchmarks of large data clouds. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, page 3. ACM, 2009.

[72] J Octavio Gutierrez-Garcia and Kwang Mong Sim. Agent-based cloud service composition. *Applied intelligence*, 38(3):436–464, 2013.

[73] Henry Hai and Seitaro Sakoda. Saas and integration best practices. *Fujitsu Scientific and Technical Journal*, 45(3):257–264, 2009.

[74] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on nosql database. In *2011 6th international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011.

[75] Abdolreza Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information sciences*, 222:175–184, 2013.

[76] Mike Hogan. Cloud computing & databases. *How databases can meet the demands of cloud computing. ScaleDB Inc*, 2008.

[77] Florian Holzschuher and René Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[78] Yin Huai, Ashutosh Chauhan, Alan Gates, Gunther Hagleitner, Eric N Hanson, Owen O'Malley, Jitendra Pandey, Yuan Yuan, Rubao Lee, and Xiaodong Zhang. Major technical advancements in apache hive. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1235–1246. ACM, 2014.

[79] Mihai-Dorin Istin, Andreea Visan, Florin Pop, and Valentin Cristea. Sopsys: Self-organizing decentralized peer-to-peer system based on well balanced multi-

way trees. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pages 369–374. IEEE, 2010.

[80] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[81] Shantenu Jha, Andre Merzky, and Geoffrey Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and computation: Practice and Experience*, 21(8):1087–1108, 2009.

[82] Xuxian Jiang and Dongyan Xu. Violin: Virtual internetworking on overlay infrastructure. In *International Symposium on Parallel and Distributed Processing and Applications*, pages 937–946. Springer, 2004.

[83] Anirudh Kadadi. *Challenges of Data Integration in Big Data*. PhD thesis, North Carolina Agricultural and Technical State University, 2015.

[84] Dervis Karaboga, Beyza Gorkemli, Celal Ozturk, and Nurhan Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

[85] Dervis Karaboga and Celal Ozturk. A novel clustering approach: Artificial bee colony (abc) algorithm. *Applied soft computing*, 11(1):652–657, 2011.

[86] Shaya Karimkashi and Ahmed A Kishk. Invasive weed optimization and its features in electromagnetics. *IEEE transactions on antennas and propagation*, 58(4):1269–1278, 2010.

[87] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[88] Jeffrey O Kephart et al. A biologically inspired immune system for computers. In *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*, pages 130–139, 1994.

[89] Hyun-Woo Kim, Jong Hyuk Park, and Young-Sik Jeong. Efficient resource management scheme for storage processing in cloud infrastructure with internet of things. *Wireless Personal Communications*, 91(4):1635–1651, 2016.

[90] Hyun-Woo Kim, Jong Hyuk Park, and Young-Sik Jeong. Human-centric storage resource mechanism for big data on cloud service architecture. *The Journal of Supercomputing*, 72(7):2437–2452, 2016.

[91] Young-Hoon Kim, Sang Hoon Chu, Seong-Woo Kang, Dong-Ho Oh, Yun-Sik Han, and Tae Yeon Hwang. Servo controller method and apparatus for high tracks per inch hard disk drives using a delay accomodating state estimator, April 18 2006. US Patent 7,031,095.

[92] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

[93] P Venkata Krishna. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing*, 13(5):2292–2303, 2013.

[94] Jayaram Krishnaswamy. *Microsoft SQL Azure Enterprise Application Development*. Packt Publishing Ltd, 2010.

[95] Ajay Kumar and Seema Bawa. Virtualization of large-scale data storage system to achieve dynamicity and scalability in grid computing. In *Advances in Computer Science, Engineering & Applications*, pages 323–331. Springer, 2012.

[96] Ajay Kumar and Seema Bawa. Generalized ant colony optimizer: swarm-based meta-heuristic algorithm for cloud services execution. *Computing*, pages 1–24, 2018.

[97] Ajay Kumar and Seema Bawa. Adjacency cloud-oriented storage overlay topology using self-organizing m-way tree. In *2019 International Conference on Innovative Computing and Communication (ICICC-2019)*, page 76. Springer, 2019.

[98] Ajay Kumar and Seema Bawa. A comparative review of meta-heuristic approaches to optimize the sla violation costs for dynamic execution of cloud services. *Soft Computing*, Jun 2019.

[99] Chunbo Lai, Song Jiang, Liqiong Yang, Shiding Lin, Guangyu Sun, Zhenyu Hou, Can Cui, and Jason Cong. Atlas: Baidu's key-value storage system for cloud data. In *2015 31st Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–14. IEEE, 2015.

[100] Walter S Lasecki, Christopher D Miller, Iftekhar Naim, Raja Kushalnagar, Adam Sadilek, Daniel Gildea, and Jeffrey P Bigham. Scribe: deep integration of human and machine intelligence to caption speech in real time. *Communications of the ACM*, 60(9):93–100, 2017.

[101] Philipp Leitner, Johannes Ferner, Waldemar Hummer, and Schahram Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. *Distributed and Parallel Databases*, 31(3):447–470, 2013.

[102] Philipp Leitner, Waldemar Hummer, and Schahram Dustdar. Cost-based optimization of service compositions. *IEEE Transactions on Services Computing*, 6(2):239–251, 2013.

[103] Weidong Li, Xi Liu, Xiaolu Zhang, and Xuejie Zhang. Dynamic fair allocation of multiple resources with bounded number of tasks in cloud computing systems. *Multiagent and Grid Systems*, 11(4):245–257, 2015.

[104] Weiwei Lin and Deyu Qi. Research on resource self-organizing model for cloud computing. In *Internet technology and applications, 2010 international conference on*, pages 1–5. IEEE, 2010.

[105] Desheng Liu, Hong Zhu, Chengzhi Xu, Ian Bayley, David Lightfoot, Mark Green, and Peter Marshall. Cide: An integrated development environment for microservices. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 808–812. IEEE, 2016.

[106] Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, and Dawn Leaf. Nist cloud computing reference architecture: Recommendations of the national institute of standards and technology (special publication 500-292). 2012.

[107] Esteban Lopez-Falcon, Andrei Tchernykh, Nikolay Chervyakov, Mikhail Babenko, Elena Nepretimova, Vanessa Miranda-López, Alexander Yu Drozdov, Gleb Radchenko, and Arutyun Avetisyan. Adaptive encrypted cloud storage model. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 329–334. IEEE, 2018.

[108] Siriluck Lorpunmanee, Mohd Noor Sap, Abdul Hanan Abdullah, and Chai Chompoo-inwai. An ant colony optimization for dynamic job scheduling in grid environment. *International Journal of Computer and Information Science and Engineering*, 1(4):207–214, 2007.

[109] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

[110] Guan-Chun Luh and Chun-Yi Lin. Optimal design of truss structures using ant algorithm. *Structural and Multidisciplinary Optimization*, 36(4):365–379, 2008.

[111] Jie Ma, Tao Chen, Songfeng Wu, Chunyuan Yang, Mingze Bai, Kunxian Shu, Kenli Li, Guoqing Zhang, Zhong Jin, Fuchu He, et al. iprox: an integrated proteome resource. *Nucleic acids research*, 47(D1):D1211–D1217, 2018.

[112] Carlo Mastroianni, Michela Meo, and Giuseppe Papuzzo. Probabilistic consolidation of virtual machines in self-organizing cloud data centers. *IEEE Transactions on Cloud Computing*, 1(2):215–228, 2013.

[113] Giacomo V Mc Evoy and Bruno Schulze. Using clouds to address grid limitations. In *Proceedings of the 6th international workshop on Middleware for grid computing*, page 11. ACM, 2008.

[114] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A database management system for semistructured data. *SIGMOD record*, 26(3):54–66, 1997.

[115] Hemant Mehta, Priyesh Kanungo, and Manohar Chandwani. Generic data access and integration service for distributed computing environment. *International Journal of Grid Computing & Applications (IJGCA)*, pages 14–21.

[116] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[117] Rohit Menon. *Cloudera administration handbook*. Packt Publishing Ltd, 2014.

[118] Daniel Merkle and Martin Middendorf. Ant colony optimization with global pheromone evaluation for scheduling a single machine. *Applied Intelligence*, 18(1):105–111, 2003.

[119] Mike Mesnier, Gregory R Ganger, and Erik Riedel. Object-based storage. *IEEE Communications Magazine*, 41(8):84–90, 2003.

[120] Mohand Mezmaz, Nouredine Melab, Yacine Kessaci, Young Choon Lee, E-G Talbi, Albert Y Zomaya, and Daniel Tuyttens. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel and Distributed Computing*, 71(11):1497–1508, 2011.

[121] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[122] Marcin Molga and Czesław Smutnicki. Test functions for optimization needs. *Test functions for optimization needs*, 101, 2005.

[123] Muhammad Baqer Mollah, Kazi Reazul Islam, and Sikder Sunbeam Islam. Next generation of computing through cloud computing technology. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE, 2012.

[124] Brototi Mondal, Kousik Dasgupta, and Paramartha Dutta. Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. *Procedia Technology*, 4:783–789, 2012.

[125] Justin Moore, David Irwin, Laura Grit, Sara Sprenkle, and Jeff Chase. Managing mixed-use clusters with cluster-on-demand. *Duke University Department of Computer Science Technical Report*, 2002.

[126] Basma Mostafa, Abderrahim Benslimane, Mohamed Saleh, Sally Kassem, and Miklos Molnar. An energy-efficient multiobjective scheduling model for monitoring in internet of things. *IEEE internet of things journal*, 5(3):1727–1738, 2018.

[127] Seyedmajid Mousavi, Amir Mosavi, Annamria R Varkonyi-Koczy, and Gabor Fazekas. Dynamic resource allocation in cloud computing. *Acta Polytechnica Hungarica*, 14(4), 2017.

[128] Emmy Mugisha, Gongxuan Zhang, Maouadj Zine El Abidine, and Mutangana Eugene. A tpm-based secure multi-cloud storage architecture grounded on erasure codes. In *Cloud Security: Concepts, Methodologies, Tools, and Applications*, pages 295–307. IGI Global, 2019.

[129] Kashif Muhammad, Shang Gao, Sohail Qaisar, MannanMasood Abdul, Atif Muhammad, Ashraf Usman, Akhtar Aleena, and Ali Shahid. Comparative analysis of meta-heuristic algorithms for solving optimization problems. In *2018 8th International Conference on Management, Education and Information (MEICI 2018)*. Atlantis Press, 2018.

[130] Amir Nakib, Boussaad Ismail, Salma Ouchraa, Laurent Schmitt, et al. *Meta-heuristics for Intelligent Electrical Networks*, volume 10. John Wiley & Sons, 2017.

[131] Raghunath Othayoth Nambiar and Meikel Poess. The making of tpc-ds. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1049–1058. VLDB Endowment, 2006.

[132] Mehdi Neshat, Ghodrat Sepidnam, Mehdi Sargolzaei, and Adel Najaran Toosi. Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial intelligence review*, 42(4):965–997, 2014.

[133] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems. In *UCSB Technical Report*. Citeseer, 2008.

[134] Ali Haydar Özer and Can Özturan. An auction based mathematical model and heuristics for resource co-allocation problem in grids and clouds. In *Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009. ICSCCW 2009. Fifth International Conference on*, pages 1–4. IEEE, 2009.

[135] Rainer Palm and Abdelbaki Bouguerra. Particle swarm against market-based optimisation for obstacle avoidance. *Electronics Letters*, 49(22):1378–1379, 2013.

[136] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, pages 400–407. IEEE, 2010.

[137] Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J Abadi, David J DeWitt, Samuel Madden, and Michael Stonebraker. A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 165–178. ACM, 2009.

[138] Teykhrib Anton Pavlovich and Teykhrib Genrikh Ivanovich. Analysis of cloud services integration with enterprise information systems. *Journal of Theoretical & Applied Information Technology*, 82(2), 2015.

[139] Meikel Poess, Tilmann Rabl, Hans-Arno Jacobsen, and Brian Caufield. Tpc-di: the first industry benchmark for data integration. *Proceedings of the VLDB Endowment*, 7(13):1367–1378, 2014.

[140] Evangelos Pournaras, Martijn Warnier, and Frances MT Brazier. Adaptive self-organization in distributed tree topologies. *International Journal of Distributed Systems and Technologies (IJDST)*, 5(3):24–57, 2014.

[141] Meikang Qiu, Keke Gai, Bhavani Thuraisingham, Lixin Tao, and Hui Zhao. Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry. *Future Generation Computer Systems*, 80:421–429, 2018.

[142] Pham Tran Anh Quang, Kamal Deep Singh, Abbas Bradai, and Abderrahim Benslimane. Qaav: Quality of service-aware adaptive allocation of virtual network functions in wireless network. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[143] P Rabl, D DeMille, John M Doyle, Mikhail D Lukin, RJ Schoelkopf, and P Zoller. Hybrid quantum processors: molecular ensembles as quantum memory for solid state circuits. *Physical review letters*, 97(3):033003, 2006.

[144] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems*. McGraw Hill, 2000.

[145] Rohit Ranchal, Bharat Bhargava, Ruchith Fernando, Hui Lei, and Zhongjun Jin. Privacy preserving access control in service-oriented architecture. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 412–419. IEEE, 2016.

[146] Simón J Rascovsky, Jorge A Delgado, Alexander Sanz, Víctor D Calvo, and Gabriel Castrillón. Informatics in radiology: use of couchdb for document-based storage of dicom objects. *Radiographics*, 32(3):913–927, 2012.

[147] Thomas Rings, Geoff Caryer, Julian Gallop, Jens Grabowski, Tatiana Kovacikova, Stephan Schulz, and Ian Stokes-Rees. Grid and cloud computing: opportunities for integration with the next generation network. *Journal of Grid Computing*, 7(3):375, 2009.

[148] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4–1, 2009.

[149] Thomas A Runkler. Ant colony optimization of clustering models. *International Journal of Intelligent Systems*, 20(12):1233–1251, 2005.

[150] Eric E Schadt, Michael D Linderman, Jon Sorenson, Lawrence Lee, and Garry P Nolan. Computational solutions to large-scale data management and analysis. *Nature reviews genetics*, 11(9):647, 2010.

[151] Neha Sharma, Rashi Agarwal, and Narendra Kohli. Review of features and machine learning techniques for web searching. In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 312–317. IEEE, 2016.

[152] Wenting Shen, Jia Yu, Rong Hao, and Xuan Wang. A public cloud storage auditing scheme for resource-constrained clients. *International Journal of High Performance Systems Architecture*, 6(3):121–130, 2016.

[153] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In *Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on*, volume 3, pages 1945–1950. IEEE, 1999.

[154] Serhiy D Shtovba. Ant algorithms: theory and applications. *Programming and Computer Software*, 31(4):167–178, 2005.

[155] Amy Shuen. *Web 2.0: A Strategy Guide: Business thinking and strategies behind successful Web 2.0 implementations*. O'Reilly Media, 2018.

[156] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, et al. The hadoop distributed file system. In *MSST*, volume 10, pages 1–10, 2010.

[157] Carlos A Silva, JMC Sousa, Thomas A Runkler, and JMG Sá Da Costa. Distributed supply chain management using ant colony optimization. *European Journal of Operational Research*, 199(2):349–358, 2009.

[158] Bhupinder Singh and Seema Bawa. Aco based optimized scheduling algorithm for computational grids. In *Proceedings of the third conference on IASTED International Conference*, pages 283–286, 2007.

[159] Bhupinder Singh and Seema Bawa. Aco based optimized scheduling algorithm for computational grids. In *Proceedings of the third conference on IASTED International Conference*, pages 283–286, 2007.

[160] Saurabh Singh, Pradip Sharma, Seo Moon, and Jong Park. Eh-gc: an efficient and secure architecture of energy harvesting green cloud infrastructure. *Sustainability*, 9(4):673, 2017.

[161] Krzysztof Socha. Aco for continuous and mixed-variable optimization. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 25–36. Springer, 2004.

[162] Krzysztof Socha and Marco Dorigo. Ant colony optimization for continuous domains. *European journal of operational research*, 185(3):1155–1173, 2008.

[163] Jie Song, Tiantian Li, Xuebing Liu, and Zhiliang Zhu. Comparing and analyzing the energy efficiency of cloud database and parallel database. In *Advances in Computer Science, Engineering & Applications*, pages 989–997. Springer, 2012.

[164] Katarina Stanoevska, Thomas Wozniak, and Santi Ristol. *Grid and cloud computing: a business perspective on technology and applications*. Springer Science & Business Media, 2009.

[165] Andreas Thor and Erhard Rahm. Cloudfuice: A flexible cloud-based data integration system. In *International Conference on Web Engineering*, pages 304–318. Springer, 2011.

[166] Elton Torres, Gustavo Callou, and Ermeson Andrade. A hierarchical approach for availability and performance analysis of private cloud storage services. *Computing*, 100(6):621–644, 2018.

[167] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.

[168] Jeffrey Voas and Jia Zhang. Cloud computing: New wine or just a new bottle? *IT professional*, 11(2):15–17, 2009.

[169] Mehul Nalin Vora. Hadoop-hbase for large-scale data. In *Computer science and network technology (ICCSNT), 2011 international conference on*, volume 1, pages 601–605. IEEE, 2011.

[170] Mladen A Vouk. Cloud computing—issues, research and implementations. In *ITI 2008-30th International Conference on Information Technology Interfaces*, pages 31–40. Ieee, 2008.

[171] Daniel S Weile and Eric Michielssen. Genetic algorithm optimization applied to electromagnetics: A review. *IEEE Transactions on Antennas and Propagation*, 45(3):343–353, 1997.

[172] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[173] Chenggang Wu, Vikram Sreekanti, and Joseph M Hellerstein. Autoscaling tiered cloud storage in anna. *Proceedings of the VLDB Endowment*, 12(6):624–638, 2019.

[174] Jiyi Wu, Lingdi Ping, Xiaoping Ge, Ya Wang, and Jianqing Fu. Cloud storage as the infrastructure of cloud computing. In *2010 International Conference on Intelligent Computing and Cognitive Informatics*, pages 380–383. IEEE, 2010.

[175] Shengjun Xue, Fei Liu, and Xiaolong Xu. An improved algorithm based on nsga-ii for cloud pdts scheduling. *JSW*, 9(2):443–450, 2014.

[176] Gao-Wei Yan and Zhan-Ju Hao. A novel optimization algorithm based on atmosphere clouds model. *International Journal of Computational Intelligence and Applications*, 12(01):1350002, 2013.

[177] Jing-Yu Yan, Qing Ling, and De-min Sun. A differential evolution with simulated annealing updating method. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 2103–2106. IEEE, 2006.

[178] Yunxue Yan, Lei Wu, Ge Gao, Hao Wang, and Wenyu Xu. A dynamic integrity verification scheme of cloud storage data based on lattice and bloom filter. *Journal of information security and applications*, 39:10–18, 2018.

[179] Lin-lin Yang, Wei-yi Qian, and Qi Zhang. Central force optimization. *Journal of Bohai University (Natural Science Edition)*, 3:001, 2011.

[180] Xin-She Yang. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation*, 3(5):267–274, 2011.

[181] Xin-She Yang and Mehmet Karamanoglu. Swarm intelligence and bio-inspired computation: an overview. In *Swarm Intelligence and Bio-Inspired Computation*, pages 3–23. Elsevier, 2013.

[182] Xin-She Yang, Mehmet Karamanoglu, and Xingshi He. Flower pollination algorithm: a novel approach for multiobjective optimization. *Engineering Optimization*, 46(9):1222–1237, 2014.

[183] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary computation*, 3(2):82–102, 1999.

[184] Tin Tin Yee and Thinn Thu Naing. Pc-cluster based storage system architecture for cloud storage. *International Journal on Cloud Computing: Services and Architecture(IJCCSA)*, 1(3):117–128, 2011.

[185] Sangho Yi, Derrick Kondo, and Artur Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 236–243. IEEE, 2010.

[186] Sisi Yuan, Gaoshan Deng, Quanxi Feng, Pan Zheng, and Tao Song. Multi-objective evolutionary algorithm based on decomposition for energy-aware scheduling in heterogeneous computing systems. *Journal of Universal Computer Science*, 23(7):636–651, 2017.

[187] Gexiang Zhang, Fen Zhou, Xiaoli Huang, Jixiang Cheng, Marian Gheorghe, Florentin Ipate, and Raluca Lefticaru. A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems. *J. UCS*, 18(13):1821–1841, 2012.

[188] Zhaojun Zhang, Funian Hu, and Na Zhang. Ant colony algorithm for satellite control resource scheduling problem. *Applied Intelligence*, pages 1–11, 2018.

[189] Hongwei Zhao and Xiaojun Ye. A practice of tpc-ds multidimensional implementation on nosql database systems. In *Technology Conference on Performance Evaluation and Benchmarking*, pages 93–108. Springer, 2013.

[190] Zhen Zhu, Long Chen, Chaochun Yuan, and Changgao Xia. Global replacement-based differential evolution with neighbor-based memory for dynamic optimization. *Applied Intelligence*, pages 1–15, 2018.

[191] Ajay Kumar and Seema Bawa. DAIS: dynamic access and integration services framework for cloud-oriented storage systems. *Cluster Computing*, pages 1–20, 2020.

List of Publications

1. Ajay Kumar and Seema Bawa. "DAIS: Dynamic Access and Integration Services Framework for Cloud-Oriented Storage Systems", Cluster Computing (2020), Springer. DOI: 10.1007/s10586-020-03088-0 **(SCI Indexed, Impact Factor: 2.040)**

2. Ajay Kumar and Seema Bawa. "A comparative review of meta-heuristic approaches to optimize the SLA violation costs for dynamic execution of cloud services." Soft Computing (2019): 1-14. DOI: https://doi.org/10.1007/s00500-019-04155-4 **(SCI Indexed, Impact Factor: 2.784)**

3. Ajay Kumar and Seema Bawa. Adjacency cloud-oriented storage overlay topology using self-organizing m-way tree. In 2019 International Conference on Innovative Computing and Communication (ICICC-2019), Advances in Intelligent Systems and Computing (**Scopus Indexed**), vol 1059, Springer, Singapore. **(Paper presented in the conference at Ostrava Technical University, Ostrava, UK)**

4. Ajay Kumar and Seema Bawa. "Generalized ant colony optimizer: swarm-based meta-heuristic algorithm for cloud services execution." Computing (2018): 1-24. DOI: https://doi.org/10.1007/s00607-018-0674-x **(SCI Indexed, Impact Factor: 2.063)**

5. Ajay Kumar and Seema Bawa. "Virtualization of large-scale data storage system to achieve dynamicity and scalability in grid computing." Advances in Intelligent Systems and Computing, Vol-167, pp. 323-331. Springer, Berlin, Heidelberg, 2012. DOI:10.1007/978-3-642-30111-7_31 **(Scopus Indexed)**

6. Ajay Kumar and Seema Bawa. "Performance Modeling and Run Time Estimation for Large-Scale Data and Computational Grid." International Conference on Advance in Modeling, Optimization and Computing, IIT Roorkee (UK), India. 2011. **(Paper presented in the conference)**