**Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System in Cloud Computing**

**ABSTRACT**

Cloud computing is a virtualized, scalable, ubiquitous, and distributed computing paradigm that provides resources and services dynamically in a subscription-based environment. Cloud computing provides services through Cloud Service Providers (CSPs). Cloud computing is mainly used for delivering solutions to a large number of business and scientific applications. Large-scale scientific applications are evaluated through cloud computing in the form of scientific workflows. Scientific workflows are data-intensive applications, and a single scientific workflow may be comprised of thousands of tasks. Deadline constraints, task failures, budget constraints, improper organization and management of tasks can cause inconvenience in executing scientific workflows. Therefore, we proposed a fault-tolerant and data-oriented scientific workflow management and scheduling system (FD-SWMS) in cloud computing. The proposed strategy applies a multicriteria-based approach to schedule and manage the tasks of scientific workflows. The proposed strategy considers the special characteristics of tasks in scientific workflows, i.e., the scientific workflow tasks are executed simultaneously in parallel, in pipelined, aggregated to form a single task, and distributed to create multiple tasks. The proposed strategy schedules the tasks based on the data-intensiveness, provides a fault tolerant technique through a cluster-based approach, and makes it energy efficient through a load sharing mechanism. In order to find the effectiveness of the proposed strategy, the simulations are carried out on Workflows for Montage and Cyber Shake workflows. The proposed FD-SWMS strategy performs better as compared with the existing state-of-the-art strategies. The proposed strategy on average reduced execution time by 25%, 17%, 22%, and 16%, minimized the execution cost by 24%, 17%, 21%, and 16%, and decreased the energy consumption by 21%, 17%, 20%, and 16%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for Montage scientific workflow. Similarly, the proposed strategy on average reduced execution time by 48%, 17%, 25%, and 42%, minimized the execution cost by 45%, 11%, 16%, and 38%, and decreased the energy consumption by 27%, 25%, 32%, and 20%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for Cyber Shake scientific workflow

**INTRODUCTION**

Fault tolerance in cloud computing means creating a blueprint for ongoing work whenever some parts are down or unavailable. It helps enterprises evaluate their infrastructure needs and requirements and provides services in case the respective device becomes unavailable for some reason.

It does not mean that the alternative system can provide 100% of the entire service. Still, the concept is to keep the system usable and, most importantly, at a reasonable level in operational mode. It is important if enterprises continue growing in a continuous mode and increase their productivity levels.

**Main Concepts behind Fault Tolerance in Cloud Computing System**

* Replication: Fault-tolerant systems work on running multiple replicas for each service. Thus, if one part of the system goes wrong, other instances can be used to keep it running instead. For example, take a database cluster that has 3 servers with the same information on each. All the actions like data entry, update, and deletion are written on each. Redundant servers will remain idle until a fault tolerance system demands their availability.
* Redundancy: When a system part fails or goes downstate, it is important to have a backup type system. The server works with emergency databases that include many redundant services. For example, a website program with MS SQL as its database may fail midway due to some hardware fault. Then the redundancy concept has to take advantage of a new database when the original is in offline mode.

**Techniques for Fault Tolerance in Cloud Computing**

* Priority should be given to all services while designing a fault tolerance system. Special preference should be given to the database as it powers many other entities.
* After setting the priorities, the Enterprise has to work on mock tests. For example, Enterprise has a forums website that enables users to log in and post comments. When authentication services fail due to a problem, users will not be able to log in.

Then, the forum becomes read-only and does not serve the purpose. But with fault-tolerant systems, healing will be ensured, and the user can search for information with minimal impact.

**Objectives of the Proposed system**

1, **EXECUTION TIME:** The results, with regard to the execution time for the FD-SWMS strategy compared with the existing strategies ar. The results reflect that the execution time is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering-based mechanism.

**2, EXECUTION COST**: The results, with regard to the execution cost for the FD-SWMS strategy compared with the existing strategies. The results reflect that the execution cost is minimum for the proposed FD-SWMS strategy. The reason is that the proposed strategy finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering-based mechanism.

**3, ENERGY CONSUMPTION**: The results, with regard to the energy consumption for the FD-SWMS strategy compared with the existing strategies. The results reflect that the energy consumption is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy initially finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism. Finally, it implements the load sharing mechanism by finding the resources with minimum utilization and transfer the data from such nodes to other nodes in order to reduce energy consumption

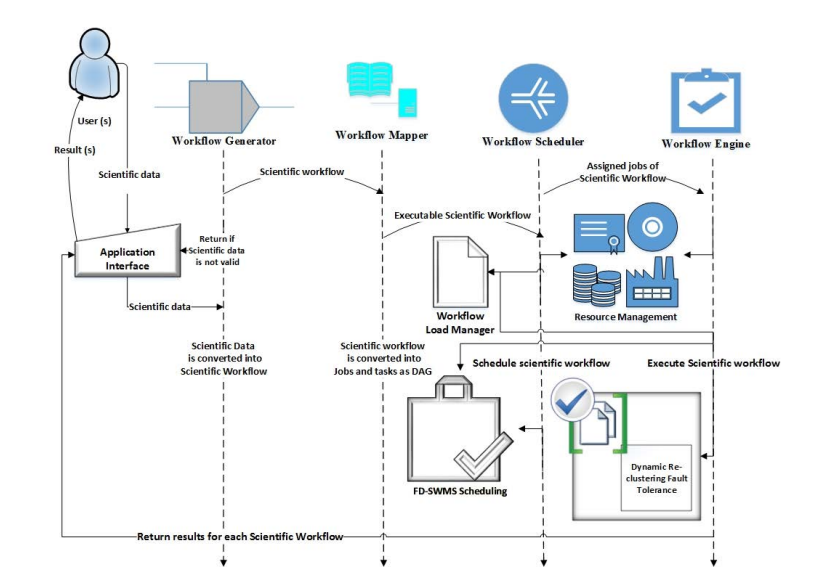
**PROBLEM STATEMENT**

The proposed the FD-SWMS strategy, a re-cluster based, fault-tolerant and data-intensive workflow management and scheduling strategy for scientific workflows in a cloud environment. The FD-SWMS strategy schedules the workflow tasks by the process in which it finds the best suitable resource based on data transfer time. The FD-SWMS ensures fault tolerance by implementing a dynamic re-clustering mechanism. The FD-SWMS minimises the energy consumption by implementing a load sharing mechanism. The simulation was carried out on WorkflowSim for Montage and CyberShake workflows. The proposed FD-SWMS strategy performs better as compared with the existing state-of-the-art strategies.

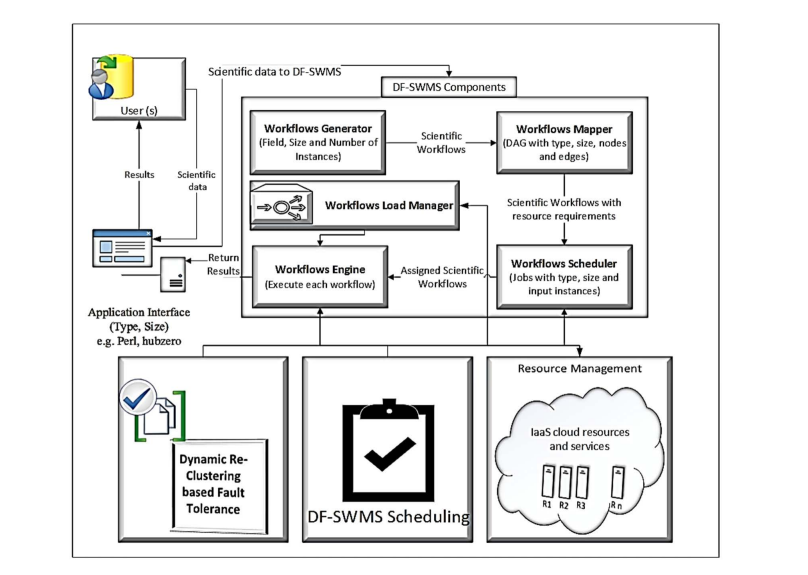
**Existing System**

Existing System is provision of fault-tolerant techniques with data-oriented scheduling is an important approach for execution of scientific workflows in Cloud computing. Accordingly, we have presented enhanced data-oriented scheduling with Dynamic-clustering fault-tolerant technique (EDS-DC) for execution of scientific workflows in cloud computing. We have presented data-oriented scheduling as a proposed scheduling technique. We have also equipped EDS-DC with Dynamic-clustering fault-tolerant technique. To know the effectiveness of EDS-DC, we compared its results with three well-known enhanced heuristic scheduling policies referred to as: (a) MCT-DC, (b) Max-min-DC, and (c) Min-min-DC. We considered scientific workflow of CyberShake as a case study, because it contains most of the characteristics of scientific workflows such as integration, disintegration, parallelism, and pipelining.

**SYSTEM ARCHITECHTURE**



**SYSTEM DESIGN**



**METHODOLOGY ADAPTED**

**CLOUD COMPUTING**

Cloud computing refers to the practice of delivering computing services over the internet, allowing users to access various resources such as servers, storage, databases, applications, and more, on-demand, without having to manage the underlying infrastructure.

In cloud computing, users can use various computing resources without having to invest in the physical hardware, software, or infrastructure required to run them. Instead, these resources are hosted and managed by a cloud service provider, who offers them to users on a subscription basis. Users can access these resources through the internet from anywhere in the world.

Cloud computing offers numerous benefits to organizations, including cost savings, flexibility, scalability, and reliability. With cloud computing, organizations can easily and quickly scale their resources up or down based on their needs, pay only for what they use, and rely on the provider's expertise to manage the infrastructure.

There are three main types of cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides users with access to virtualized computing resources such as servers and storage. PaaS provides users with a platform for developing, testing, and deploying applications, while SaaS provides users with access to pre-built software applications that are hosted and managed by the provider.

FD-SWMS Strategy

Workflow Scheduler gets jobs with multiple tasks from the Workflow Mapper and schedules them by allocating resources. The resources are retrieved from the cloud using Infrastructure as a Service (IaaS) and then scheduled/ managed separately using the FD-SWMS strategy. Workflow Scheduler also turns jobs into tasks and then assigns resources.

The allocation of resources is done in such a way that the tasks consume minimum execution time at the lowest cost. It is achieved by considering the list of tasks and resources.

It finds the resource for each task with minimum data transfer time. For each task, the data transfer time of a task to all the resources is calculated, and then the task is assigned to the resource with the minimum data transfer time. Equation 6 demonstrates the concept of finding a resource with minimum data transfer time.

**HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

PROCESSOR : PENTIUM IV 2.6 GHz, Intel Core 2 Duo.

RAM : 512 MB DD RAM

MONITOR : 15” COLOR

HARD DISK : 40 GB

**SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team’s and tracking the team’s progress throughout the development activity.

Front End : HTML5, CSS3, JAVA SCRIPT

Back End : CoreJAva,J2EE(SERVLETS,JSP)

Data Base : MY SQL 5.5

Operating System : Windows 07, 10

IDE : Eclipse

EXCEPCTED RESULTS

**package** com.browser.Bean;

**public** **class** RegisterBean {

**private** String name;

**private** String username;

**private** String password;

**private** String email;

**private** String address;

**private** String contact;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getContact() {

**return** contact;

}

**public** **void** setContact(String contact) {

**this**.contact = contact;

}

}

package com.browser.Database;

import java.sql.Connection;

import java.sql.DriverManager;

public class DatabaseConnection {

static Connection con;

public static Connection createConnection(){

try{

Class.forName("com.mysql.jdbc.Driver");

con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sicher","root","root");

}catch(Exception e){

e.printStackTrace();

}

return con;

}

}

**package** com.browser.Implementation;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** java.util.ArrayList;

**import** com.browser.Bean.RegisterBean;

**import** com.browser.Bean.UploadBean;

**import** com.browser.Database.DatabaseConnection;

**import** com.browser.Interface.BrowserInterface;

**public** **class** BrowserImplementation **implements** BrowserInterface {

Connection con;

@Override

**public** **int** register(RegisterBean rb) {

**int** ans = 0;

con = DatabaseConnection.*createConnection*();

**try**{

PreparedStatement ps = con.prepareStatement("insert into register values (?,?,?,?,?,?)");

ps.setString(1, rb.getName());

ps.setString(2, rb.getUsername());

ps.setString(3, rb.getPassword());

ps.setString(4, rb.getEmail());

ps.setString(5, rb.getAddress());

ps.setString(6, rb.getContact());

ans = ps.executeUpdate();

}**catch**(Exception e){

e.printStackTrace();

}

**return** ans;

}

@Override

**public** **int** login(String email, String password) {

**int** ans = 0;

**try**{

con = DatabaseConnection.*createConnection*();

Statement st = con.createStatement();

ResultSet rs = st.executeQuery("select \* from register where email='"+email+"' and password='"+password+"'");

**while**(rs.next()){

String emailch = rs.getString("email");

String passwordch = rs.getString("password");

**if**(emailch.equals(email)&&passwordch.equals(password)){

ans = 1;

}

}

}**catch**(Exception e){

e.printStackTrace();

}

**return** ans;

}

}

package com.browser.Interface;

import com.browser.Bean.RegisterBean;

import com.browser.Bean.UploadBean;

public interface BrowserInterface {

public int register(RegisterBean rb);

public int login(String email, String password);

}

package com.browser.Servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.browser.Implementation.BrowserImplementation;

import com.browser.Interface.BrowserInterface;

/\*\*

\* Servlet implementation class BrowserLoginServlet

\*/

@WebServlet("/BrowserLoginServlet")

public class BrowserLoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public BrowserLoginServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

String email = request.getParameter("email");

String password = request.getParameter("password");

System.out.println(email+" "+password);

BrowserInterface bi = new BrowserImplementation();

int i = bi.login(email, password);

System.out.println("The value of i is: "+i);

if(i == 1){

HttpSession session = request.getSession();

session.setAttribute("email", email);

response.sendRedirect("BrowserHome.jsp");

}else{

response.sendRedirect("Error.jsp");

}

}

}

package com.browser.Servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import com.browser.Bean.RegisterBean;

import com.browser.Implementation.BrowserImplementation;

import com.browser.Interface.BrowserInterface;

/\*\*

\* Servlet implementation class BrowserRegisterServlet

\*/

@WebServlet("/BrowserRegisterServlet")

public class BrowserRegisterServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public BrowserRegisterServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

String name = request.getParameter("name");

String username = request.getParameter("username");

String password = request.getParameter("password");

String cpassword = request.getParameter("cpassword");

String email = request.getParameter("email");

String address = request.getParameter("address");

String contact = request.getParameter("contact");

System.out.println(name+" "+username+" "+password+" "+cpassword+" "+email+" "+address+" "+contact);

if(password.equals(cpassword)){

RegisterBean rb = new RegisterBean();

rb.setName(name);

rb.setUsername(username);

rb.setPassword(password);

rb.setEmail(email);

rb.setAddress(address);

rb.setContact(contact);

BrowserInterface bi = new BrowserImplementation();

int i = bi.register(rb);

System.out.println("The value of i is: "+i);

if(i == 1){

HttpSession session = request.getSession();

session.setAttribute("email", email);

session.setAttribute("password", password);

response.sendRedirect("BrowserLogin.jsp");

}else{

response.sendRedirect("Error.jsp");

}

}

}

}

package com.browser.Servlet;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class ServerLoginServlet

\*/

@WebServlet("/ServerLoginServlet")

public class ServerLoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public ServerLoginServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

String username = request.getParameter("username");

String password = request.getParameter("password");

System.out.println(username+" "+password);

if(username.equals("admin")&&password.equals("admin")){

response.sendRedirect("ServerHome.jsp");

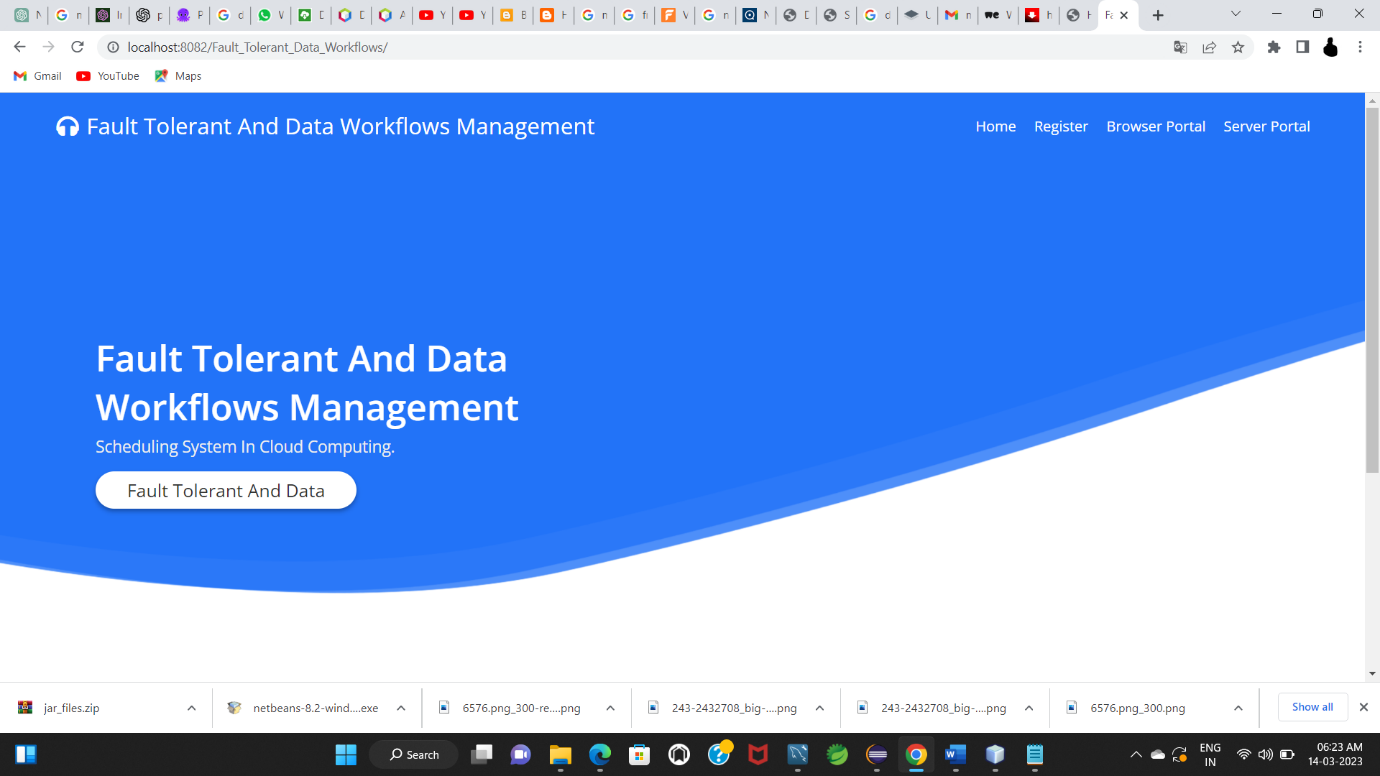
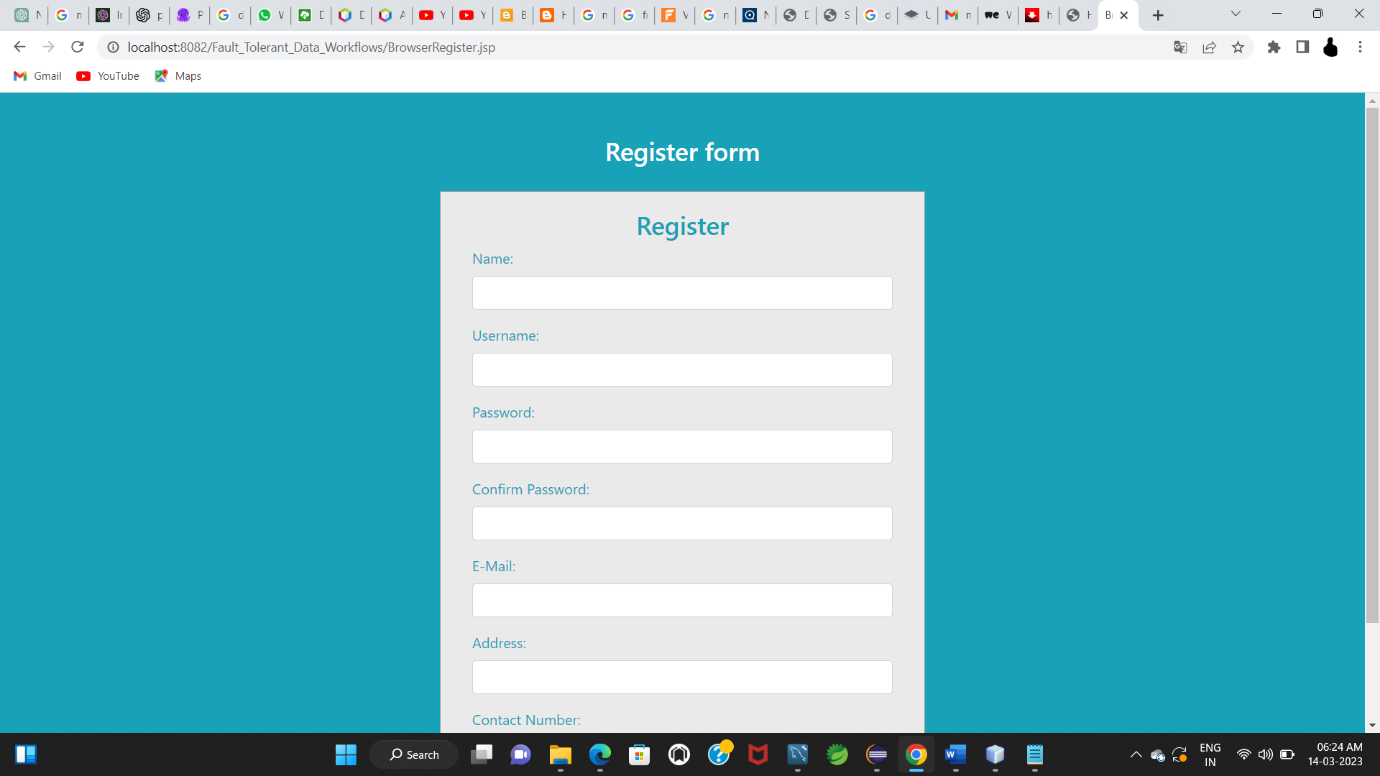
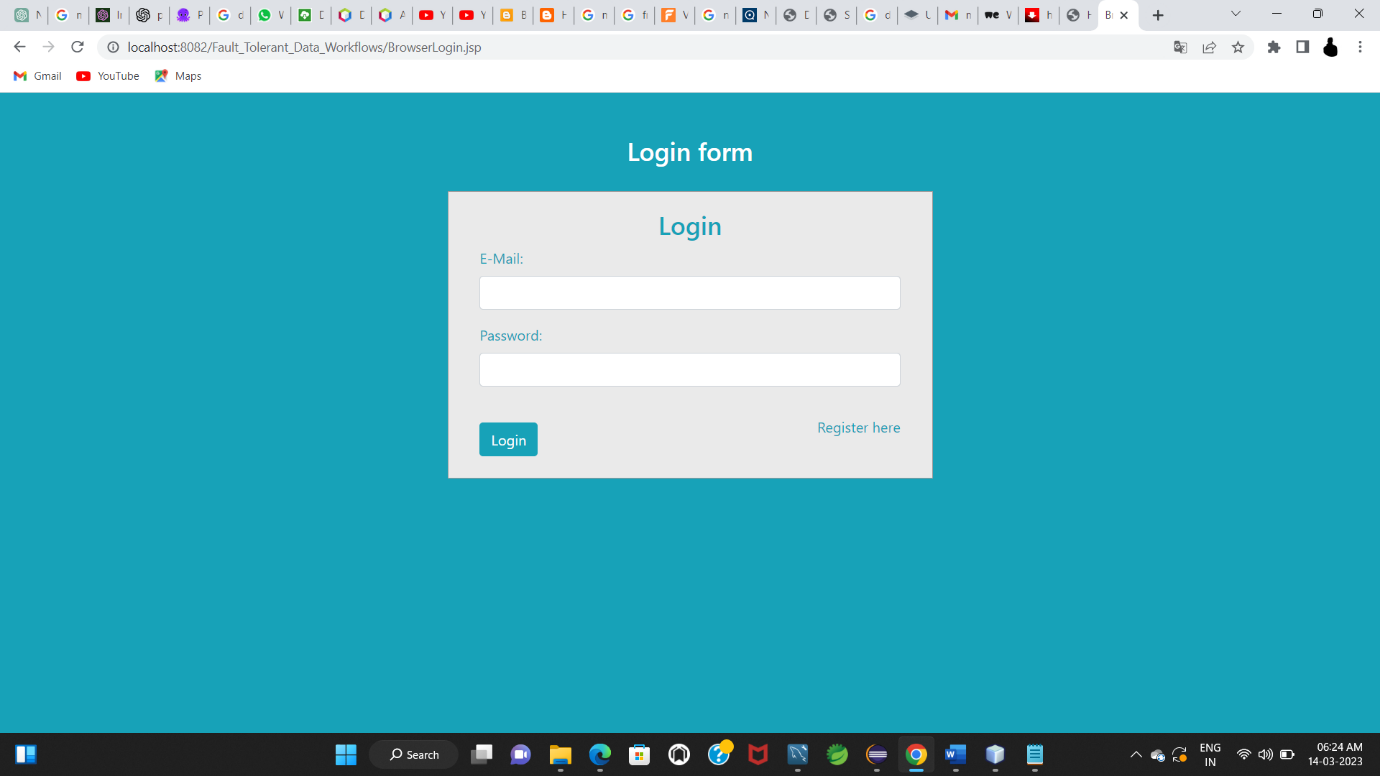
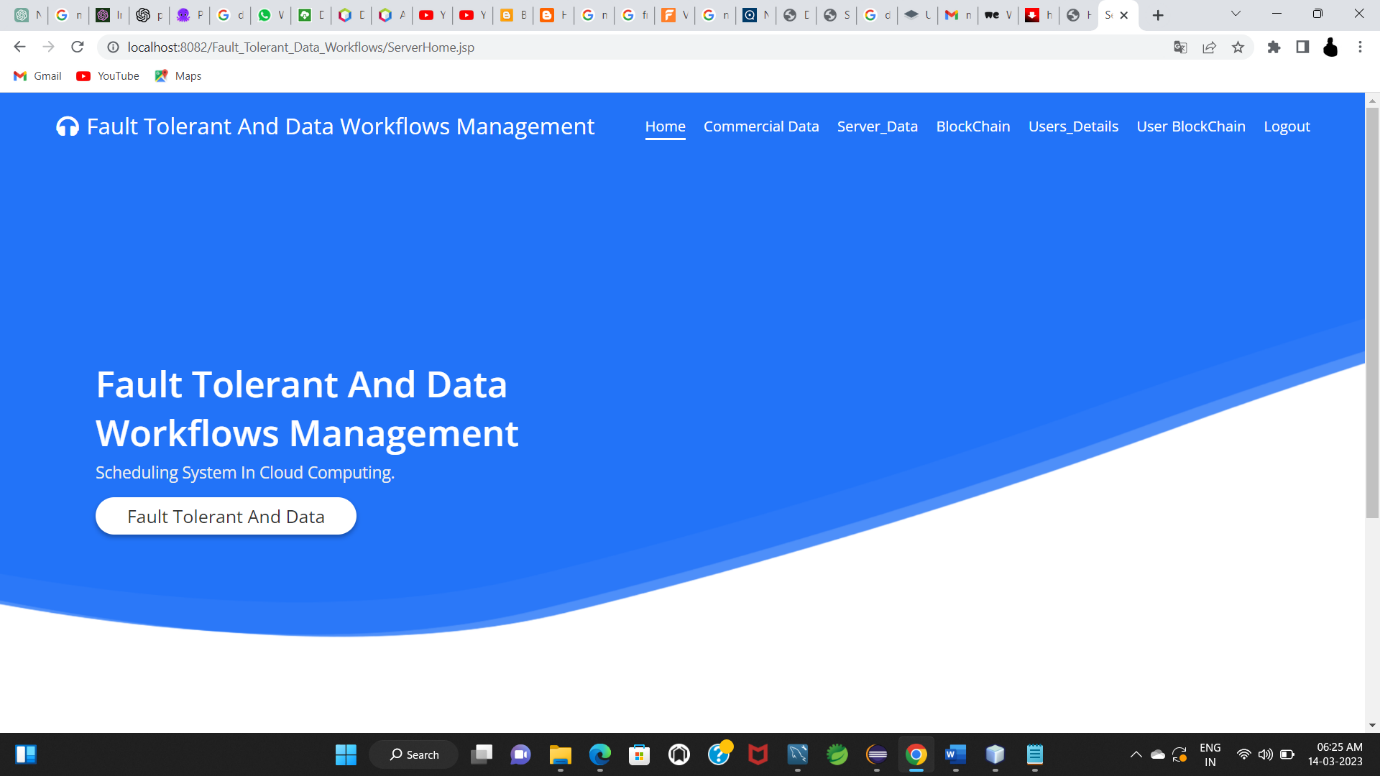
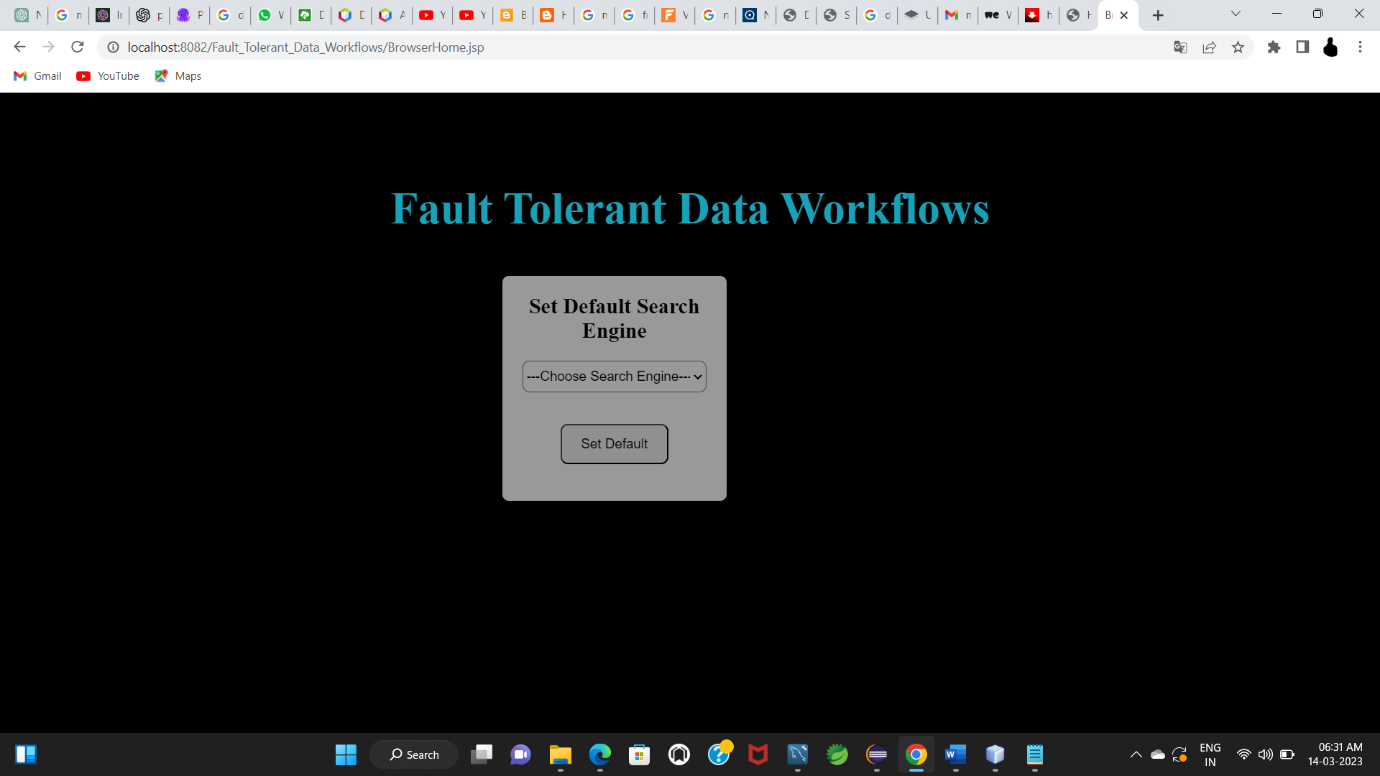
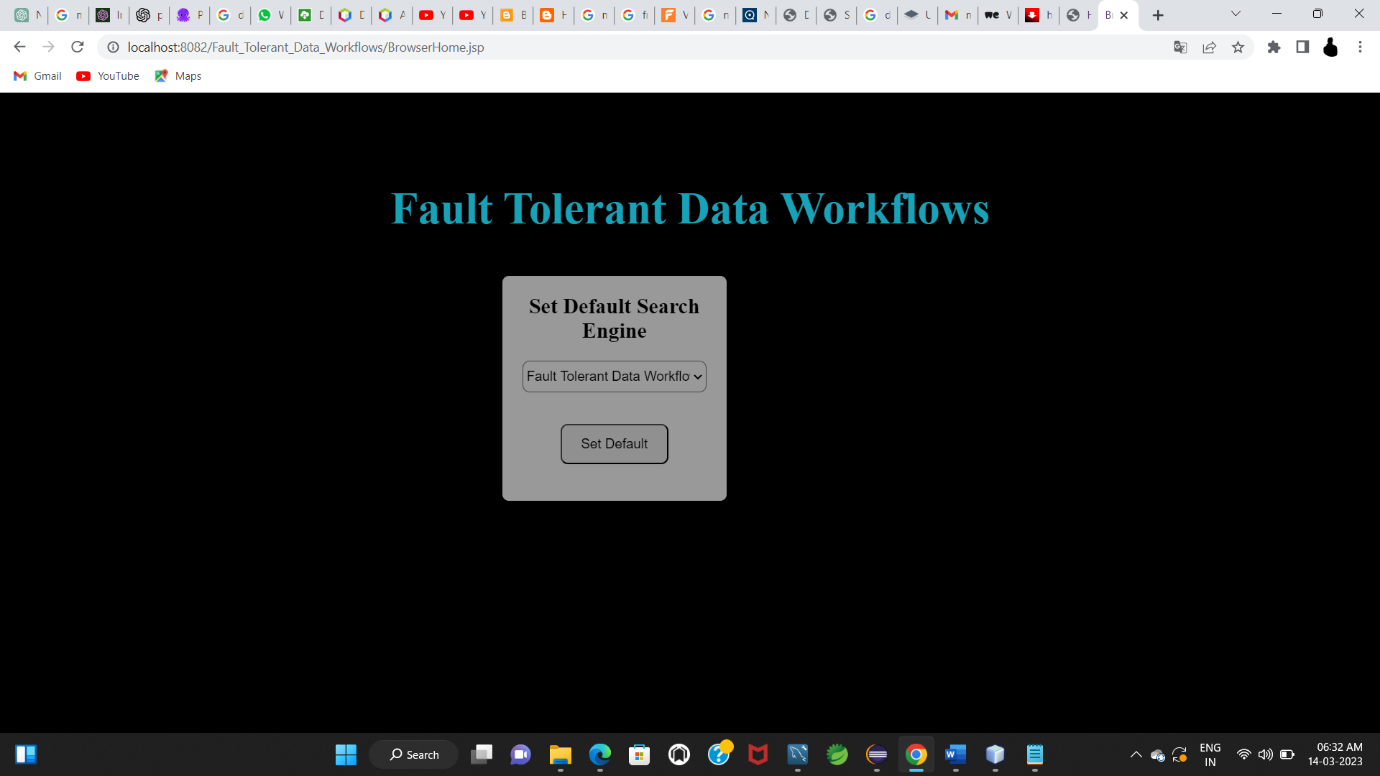
}else{

response.sendRedirect("Error.jsp");

}

}

}



**LITERATURE SURVEY**

**TITLE: Attribute-based data access control in mobile cloud computing: Taxonomy and open issues**

**AUTHER: Mehdi Sookhak**

**YEAR: 2016**

**ABSTRACT:**

the thriving growth of the cloud computing, the security and privacy concerns of outsourcing data have been increasing dramatically. However, because of delegating the management of data to an untrusted cloud server in data outsourcing process, the data access control has been recognized as a challenging issue in cloud storage systems. One of the preeminent technologies to control data access in cloud computing is Attribute-based Encryption (ABE) as a cryptographic primitive, which establishes the decryption ability on the basis of a user’s attributes. This paper provides a comprehensive survey on attribute-based access control schemes and compares each scheme’s functionality and characteristic. We also present a thematic taxonomy of attribute-based approaches based on significant parameters, such as access control mode, architecture, revocation mode, revocation method, revocation issue, and revocation controller. The paper reviews the state-of-the-art ABE methods and categorizes them into three main classes, such as centralized, decentralized, and hierarchal, based on their architectures. We also analyzed the different ABE techniques to ascertain the advantages and disadvantages, the significance and requirements, and identifies the research gaps. Finally, the paper presents open issues and challenges for further investigations.

**TITLE:** HIGH PERFORMANCE CLOUD COMPUTING

**AUTHER:** **VIKTOR MAUCH**

**YEAR:2012**

**ABSTRACT:**

Today’s high performance computing systems are typically managed and operated by individual organizations in private. Computing demand is fluctuating, however, resulting in periods where dedicated resources are either underutilized or overloaded. A cloud-based Infrastructure-as-a-Service (IaaS) approach for high performance computing applications promises cost savings and more flexibility. In this model virtualized and elastic resources are utilized on-demand from large cloud computing service providers to construct virtual clusters exactly matching a customer’s specific requirements. This paper gives an overview on the current state of high performance cloud computing technology and we describe the underlying virtualization techniques and management methods. Furthermore, we present a novel approach to use high speed cluster interconnects like InfiniBand in a high performance cloud computing environment.

**TITLE:** **CLOUD COMPUTING AND EMERGING IT PLATFORMS: VISION, HYPE, AND REALITY FOR DELIVERING COMPUTING AS THE 5TH UTILITY**

**AUTHER:** **Rajkumar Buyya**

**YEAR:2008**

**ABSTRACT:**

With the significant advances in Information and Communications Technology (ICT) over the last half century, there is an increasingly perceived vision that computing will one day be the 5th utility (after water, electricity, gas, and telephony). This computing utility, like all other four existing utilities, will provide the basic level of computing service that is considered essential to meet the everyday needs of the general community. To deliver this vision, a number of computing paradigms have been proposed, of which the latest one is known as Cloud computing. Hence, in this paper, we define Cloud computing and provide the architecture for creating Clouds with market-oriented resource allocation by leveraging technologies such as Virtual Machines (VMs). We also provide insights on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain Service Level Agreement (SLA)-oriented resource allocation. In addition, we reveal our early thoughts on interconnecting Clouds for dynamically creating global Cloud exchanges and markets. Then, we present some representative Cloud platforms, especially those developed in industries, along with our current work towards realizing market-oriented resource allocation of Clouds as realized in Aneka enterprise Cloud technology. Furthermore, we highlight the difference between High Performance Computing (HPC) workload and Internet-based services workload. We also describe a metanegotiation infrastructure to establish global Cloud exchanges and markets, and illustrate a case study of harnessing ‘Storage Clouds’ for high performance content delivery. Finally, we conclude with the need for convergence of competing IT paradigms to deliver our 21st century vision

**TITLE:** **Elastic Scheduling of Scientific Workflows under Deadline Constraints in Cloud Computing Environments**

**AUTHER:** **Nazia Anwar**

**YEAR:2018**

**ABSTRACT:**

Scientific workflow applications are collections of several structured activities and fine-grained computational tasks. Scientific workflow scheduling in cloud computing is a challenging research topic due to its distinctive features. In cloud environments, it has become critical to perform efficient task scheduling resulting in reduced scheduling overhead, minimized cost and maximized resource utilization while still meeting the user-specified overall deadline. This paper proposes a strategy, Dynamic Scheduling of Bag of Tasks based workflows (DSB), for scheduling scientific workflows with the aim to minimize financial cost of leasing Virtual Machines (VMs) under a user-defined deadline constraint. The proposed model groups the workflow into Bag of Tasks (BoTs) based on data dependency and priority constraints and thereafter optimizes the allocation and scheduling of BoTs on elastic, heterogeneous and dynamically provisioned cloud resources called VMs in order to attain the proposed method’s objectives. The proposed approach considers pay-as-you-go Infrastructure as a Service (IaaS) clouds having inherent features such as elasticity, abundance, heterogeneity and VM provisioning delays. A trace-based simulation using benchmark scientific workflows representing real world applications, demonstrates a significant reduction in workflow computation cost while the workflow deadline is met. The results validate that the proposed model produces better success rates to meet deadlines and cost efficiencies in comparison to adapted state-of-the-art algorithms for similar problems

**TITLE:** **ANALYZING, MODELING AND EVALUATING DYNAMIC ADAPTIVE FAULT TOLERANCE STRATEGIES IN CLOUD COMPUTING ENVIRONMENTS**

**AUTHER:** **DAWEI SUN**

**YEAR:2013**

**ABSTRACT:**

In cloud environments, it has become critical to perform efficient task scheduling resulting in reduced scheduling overhead, minimized cost and maximized resource utilization while still meeting the user-specified overall deadline. This paper proposes a strategy, Dynamic Scheduling of Bag of Tasks based workflows (DSB), for scheduling scientific workflows with the aim to minimize financial cost of leasing Virtual Machines (VMs) under a user-defined deadline constraint. The proposed model groups the workflow into Bag of Tasks (BoTs) based on data dependency and priority constraints and thereafter optimizes the allocation and scheduling of BoTs on elastic, heterogeneous and dynamically provisioned cloud resources called VMs in order to attain the proposed method’s objectives. The proposed approach considers pay-as-you-go Infrastructure as a Service (IaaS) clouds having inherent features such as elasticity, abundance, heterogeneity and VM provisioning delays. A trace-based simulation using benchmark scientific workflows representing real world applications, demonstrates a significant reduction in workflow computation cost while the workflow deadline is met. The results validate that the proposed model produces better success rates to meet deadlines and cost efficiencies in comparison to adapted state-of-the-art algorithms for similar problems