

1.Introduction

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict 'rating' or 'preference' that a user would give to an item (such as music, books or movies) or social element (e.g. people or group) they had not yet considered, using a model built from the characteristics of an item (content based approaches) or the user's social environment (collaborative filtering approaches).

A recommender system is a system which provides recommendations to a user. The Recommender System is to generate meaningful recommendations to a collection of users for items or products that might interest them.

Recommendation systems are defined as the techniques used to predict the rating one individual will give to an item or social entity. These items can be books, movies, restaurants and things on which individuals have different preferences.

These preferences are being predicted using two approaches first Content-Based Approach which involves characteristics of an item and second Collaborative Filtering Approaches which takes into account user's past behavior to make choices. In collaborative filtering, partners are chosen who will make recommendations because they share similar ratings history with the target user. One partner who have similar ratings to the target user may not be a reliable predictor for a particular item. So the past record of the partner of making a reliable recommendation also needs to be take into consideration which is dictated by trustworthiness of a partner.

Map Reduce

Map Reduce is a processing technique and a program model for distributed computing based on java. The Map Reduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.

The Map Reduce framework consists of a single master JobTracker and one slave Task Tracker per cluster-node. MapReduce allows for distributed processing of the map and reduction operations.

- **Map step:** Each worker node applies the map() function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.
- **Shuffle step:** Worker nodes redistribute data based on the output keys such that all data belonging to one key is located on the same worker node.
- **Reduce step:** Worker nodes now process each group of output data, per key, in parallel.

Map Reduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel –

though in practice this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process can often appear inefficient compared to algorithms that are more sequential Map Reduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use Map Reduce to sort a Petabyte of data in only a few hours. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available.

Another way to look at Map Reduce is as a 5-step parallel and distributed computation:

- **Prepare the Map() input:** The "Map Reduce system" designates Map processors, assigns the input key value K1 that each processor would work on, and provides that processor with all the input data associated with that key value.
- **Run the user-provided Map() code:** Map() is run exactly once for each K1 key value, generating output organized by key values K2.
- **"Shuffle" the Map output to the Reduce processors:** The MapReduce system designates Reduce processors, assigns the K2 key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.
- **Run the user-provided Reduce() code:** Reduce() is run exactly once for each K2 key value produced by the Map step.
- **Produce the final output:** The MapReduce system collects all the Reduce output, and sorts it by K2 to produce the final outcome.

Collaborative Filtering Algorithm

Collaborative filtering (CF) is a technique used by Recommender Systems.

Collaborative filtering, also referred to as social filtering, filters information by using the recommendations of other people. It is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future.

Collaborative filtering algorithms often require

- (1) users' active participation,
- (2) an easy way to represent users' interests, and
- (3) algorithms that are able to match people with similar interests.

Typically, the workflow of a collaborative filtering system is:

1. A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain.
2. The system matches this user's ratings against other users' and finds the people with most "similar" tastes.

3. With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item)

Methodology

Collaborative filtering systems have many forms, but many common systems can be reduced to two steps:

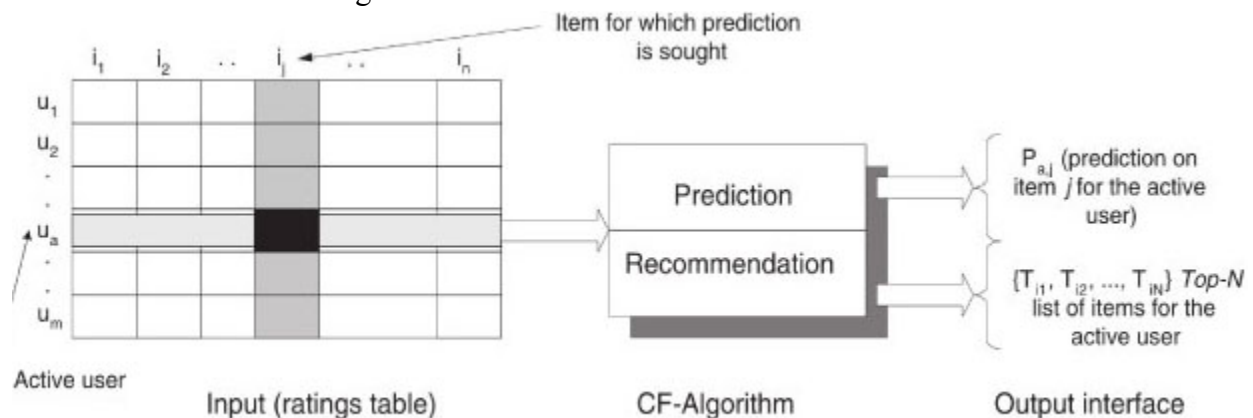
1. Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
2. Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

Alternatively, item-based collaborative filtering (users who bought x also bought y), proceeds in an item-centric manner:

1. Build an item-item matrix determining relationships between pairs of items
2. Infer the tastes of the current user by examining the matrix and matching that user's data

Collaborative Filtering based Recommender system Approach

Collaborative Filtering (CF) systems work by collecting user feedback in the form of ratings for items in a given domain and exploiting similarities in rating behaviour amongst several users in determining how to recommend an item.



Pearson and Tanimoto Coefficient

Pearson Coefficient:

In statistics, the Pearson correlation coefficient is a measure of the linear correlation between two variables X and Y . It has a value between $+1$ and -1 , where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. It is widely used in the sciences and recommender systems.

The ratings of person X and Y of the item k are written as x_k and y_k , while \bar{x} and \bar{y} are the mean values of their ratings. The correlation between X and Y is then given by:

$$r(X,Y) = \frac{\sum_k (X_k - \bar{X})(Y_k - \bar{Y})}{\sqrt{\sum_k (X_k - \bar{X})^2 \sum_k (Y_k - \bar{Y})^2}}$$

In this formula k is an element of all the items that both X and Y have rated. A prediction for the rating of person X of the item i based on the ratings of people who have rated item i is computed as follows:

$$p(X_i) = \frac{\sum_j Y_j \cdot r(X,Y)}{n}$$

Tanimoto Coefficient:

It is the ratio of intersection i.e., both users express the same feelings about the items to the union of the user preferred items

Tanimoto Coefficient uses the ratio of the intersecting set to the union set as the measure of similarity. Represented as a mathematical equation:

$$T(a, b) = \frac{N_c}{N_a + N_b - N_c}$$

In this equation, N represents the number of attributes in each object (a,b). C in this case is the intersection set.

1.1. Project Overview

Volumes of data are exploding in both scientific and commercial domain. With the increasing population and work, people mostly are engaged to online shopping, booking, chatting etc.. People want to take the best and suitable products ever, thus in order to be with the trend there is much use of recommenders.

We in our project had developed a recommender system which gives recommendations of the items based upon the previous users interests, ratings and reviews. We have used Hadoop for parallel processing and Mahout for machine learning techniques.

1.2. Project Deliverables

Data in a Hadoop cluster is broken down into smaller pieces (called blocks) and distributed throughout the cluster. In this way, the map and reduce functions can be executed on smaller subsets of your larger data sets, and this provides the scalability that is needed for Big Data processing. To achieve availability as components fail, HDFS replicates these smaller pieces onto two additional servers by default. (This redundancy can be increased or decreased on a per-file basis or for a whole environment; for example, a development Hadoop cluster typically doesn't need any data redundancy.) This redundancy offers multiple benefits, the most obvious being higher availability. In addition, this redundancy allows the Hadoop cluster to break work up into smaller chunks and run those jobs on all the servers in the cluster for better scalability.

1.3. Project Scope

Collaborative filtering algorithm increases the scope for recommendations and can also be implemented further for future use. The objective of this project is to apply collaborative filtering algorithm with Pearson and Tanimoto coefficients on the data set for generating recommendations. In the real world scenario this algorithm is highly useful in analyzing large amount of data parallelly as well as it helps in reducing processing time and improves performance.

2. Literature Survey

Recommender system has been so extensively used these days that it has become a preferable choice for researchers.

The rapid development of information technology has produced many changes, especially for enterprises that operate digitally. Both public and private enterprises now accumulate large amounts of data, which has resulted in the era of 'big data'. This amounts to the accumulation of a vast amount of data over time. In the past, such data were considered merely a historical archive. However, with the advent of cloud computing, scholars began to specialize in mining these data-related features

In year 2005 John O'Donovan[1], Barry Smyth have taken trust as the percentage of correct predictions that a profile has made in general (profile-level trust) or with respect to a particular item (item-level trust). Authors have described a number of ways in which these different types of trust values might be incorporated into a standard collaborative filtering algorithm and evaluated each against a tried-and-test benchmark approach and on a standard data-set.

In year 2007 Paul Resnick[2] proposed an idea of "influence limiter algorithm" in recommender system which prevents any attack which shows the irrelevant result for our search and limits the number of content that an attacker can modify.

In year 2007 a trust based recommender system was proposed by Punam Bedi[3], Harmeet Kaur[4], Sudeep Marwaha[5] for semantic net. Description of the design of a recommender system that uses knowledge stored in the form of ontologies is given. The interactions amongst the peer agents for generating recommendations are based on the trust network that exists between them.

In 2008 Kleanthi Lakiotaki, Stelios Tsafarakis[6], and Nikolaos Matsatsini[7] proposed UTA-Rec. UTARec is a Recommender System that incorporates Multiple Criteria Analysis methodologies. The system's performance and capability of addressing certain shortfalls of existing Recommender Systems is demonstrated in the case of movie recommendations.

A. Recio-García[8] presented a prototyping Recommender Systems in jCOLIBRI. The goal of this recommender system is to support system developers in rapid prototyping recommender systems using Case-Based Reasoning (CBR) techniques. It describes how jcolibri can serve to that goal. Jcolibri is an object-oriented framework in Java for building CBR systems that greatly benefits from the reuse of previously developed CBR systems.

In year 2009 Mohsen Jamali, Martin Ester[9] proposed Trust Walker, it is a random walk model for combining trust-based and item-based recommendation. Trust Walker combines the trust-based and the item-based collaborative filtering approach to recommendation. TrustWalker considers not only ratings of the target item, but also those of similar items, with probability increasing with increasing length of the walk. As another contribution, Trust Walker allows us to define and to measure the confidence of a recommendation.

Rong Hu analyse different Acceptance Issues of Personality-based Recommender Systems. Author evaluated an existing PBR using the technology acceptance model (TAM). Author also compare it with a baseline rating-based recommender in a ,within subject user study. In year 2012 Punam Bedi[10], Ravish Sharma proposed Trust based Ant Recommender System (TARS) that produces valuable recommendations by incorporating a notion of dynamic trust between users and selecting a small and best neighborhood based on biological metaphor of ant colonies

The performance of TARS is evaluated using two datasets of different sparsity levels viz. Jester dataset and MovieLens dataset (available online) and compared with traditional Collaborative Filtering based approach for generating recommendations.

In same year Pooja Vashisth, Deepak Chandoliya, Bipin Kumar, Punam Bedi proposed Trust Enabled Argumentation Based Recommender System(TABRS). TABRS recommends item of interest to the user by using a hybrid approach for recommendation. these recommendations are further improved using argumentation to convince users about the product. It is an agent based recommender system.

3.Problem Analysis

3.1.Existing System

3.1.1.Description of Existing System

In Existing System, Collaborative filtering algorithm is used in traditional aspect of hadoop cluster to perform analysis of data in database for recommendations to social Api.As the amount of data increases this system undergo certain difficulties.

3.1.2. DrawBacks of Existing Systems

- Lack of proper data
- Changing and retrieving data is very difficult
- User preferences are unpredictable
- Complex stuff

3.2. Proposed System

We propose an algorithm that is based on mapreduce framework.This allows the algorithm to be executed in a distributed approach, and solves the memory consumption problem.

3.2.1. Problem Statement

In our project we have taken huge amount of amazon data from the amazon service which is having all the user data including user preferences, user ratings and reviews for the purchased items.If the user is new to the site and want to buy some products, our algorithm works with the proposed metrics and generate the recommendations to the user.

3.2.2. Advantages of proposed System

- Our algorithm executes with good performance because we are executing it in a distributed environment.
- Time constraint is less as we execute our algorithm parallely
- Computation Speed is more.
- Highly accurate

4. System Analysis

System analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose. System analysis is the process of studying a procedure in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way.

The development of a computer-based information system includes a system analysis phase which produces or enhances the data-model which itself is a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, system analysis would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- • Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

4.1. Software Requirements Specifications

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. software requirements specifications permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

4.1.1. Functional Requirements

Functional requirements define what a system is supposed to do. Functional requirements are usually in the form of system shall do, an individual action of part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model.

Extracting Data

The high dimensional Amazon data obtained from the Amazon service has the following fields such as,

Username, date, Location, Date, City, Purchased items etc...

Applying Collaborative Filtering Algorithm

In our project by weighing the coefficient value from different metrics coefficients such as Pearson and Tanimoto, we apply Collaborative filtering algorithm to obtain the coefficient value greater than or equal to one.

Output

We get output which shows recommendations for the user of desired items.

4.1.2. Non-Functional Requirements

Non-functional requirements define how a system is supposed to be. Non-functional requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are in the form of an overall property of the system as a whole or of particular aspect and not a specific function.

Reliability: Hadoop is a reliable system, that takes care of data replication (with a suitable replication count) and in case of some failure it ensures that data are not lost.

Scalability: whenever there is a need to handle larger data, Hadoop scales linearly, just by adding another node to the cluster.

Accuracy: Hadoop defines accurate clusters with the defined recommendations.

4.2. Feasibility study

The feasibility study is an evaluation and analysis of the potential of a proposed project. It is based on extensive investigation and research to support the process of decision making. Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained. A well-designed feasibility

study should provide a historical background of a project, a description of a service, details of the operations. Generally, feasibility studies precede technical development and project implementation. A feasibility study evaluates the project's potential for success. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

Scalability:

Ability to process huge amounts of data.

Reliability:

It is an efficient way of processing data without loss.

Technical Feasibility:

Technical feasibility also involves evaluation of the hardware and the software requirements of the proposed system.

Economic Feasibility:

It serves as an independent project assessment, and enhances project credibility. Economic feasibility is an assessment which typically involves cost/ benefits from the analysis of the project.

Operational Feasibility:

It measures how well the proposed system solves problems and takes advantage of the opportunities identified during scope definition. Operational feasibility studies analyze how the project plan satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must inform and guide design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability and others.

Scheduling Feasibility:

It is an important factor for project success. A project will fail if it is not completed on time. In Scheduling feasibility, we estimate how much time the system will take to complete and with our technical skills we need to estimate the period to complete the project using various method of estimation.

Benefits of Conducting a Feasibility Study:

Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of your idea. Below are the key benefits of conducting a feasibility study:

- Gives project teams more focus and provide an alternative outline.
- Narrows the business alternatives.
- Identifies a valid reason to undertake the project
- Enhances the success rate by evaluating multiple parameters.
- Aids decision-making on the project.

4.3. Object Oriented Analysis

Object Oriented Analysis is a popular technical approach for analysing, designing an application, system or business by applying the object oriented paradigm and visual modelling throughout the development lifecycle to faster, better, stakeholder communication and product quality.

In the case of object oriented analysis the process is varies. But these two are identical at use case analysis. Actually the steps involved in the analysis phase are

- Identify the actors.
- Classification-develop a static UML class diagrams.
- Develop use cases.
- Identify classes, relationships, attributes, methods.

System Models

- Scenarios
- Use Case Models
- A Use case is a description of the behaviour of the system. That description is written from the point of a user who just told the system to do something particular.

UML Diagrams

UML is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems. The UML user mostly graphical notations to express the design of software project. It is a very important part of developing object oriented software and software development process.

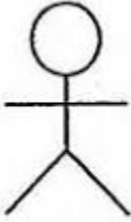


4.3.1. Use Case Diagrams

Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions that some systems should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Graphical Notation

The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.

Fig:Graphical representation of Use Case diagrams

Actor	An actor in the Unified Modelling Language specifies a role played by a user or any other system that interacts with the subject.	 Actor
Use case	A Use Case is the functionality provided by the system. Use Cases are depicted with an ellipse. The name of the Use Case is written in ellipse.	 Use Case
Association	Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation.	 Association

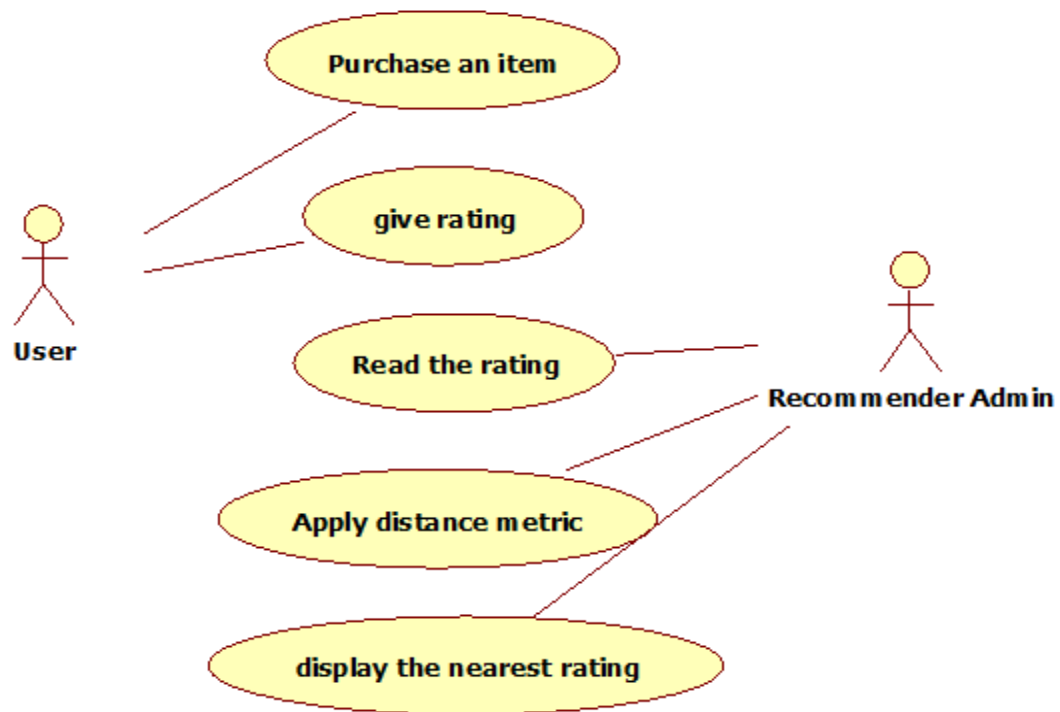
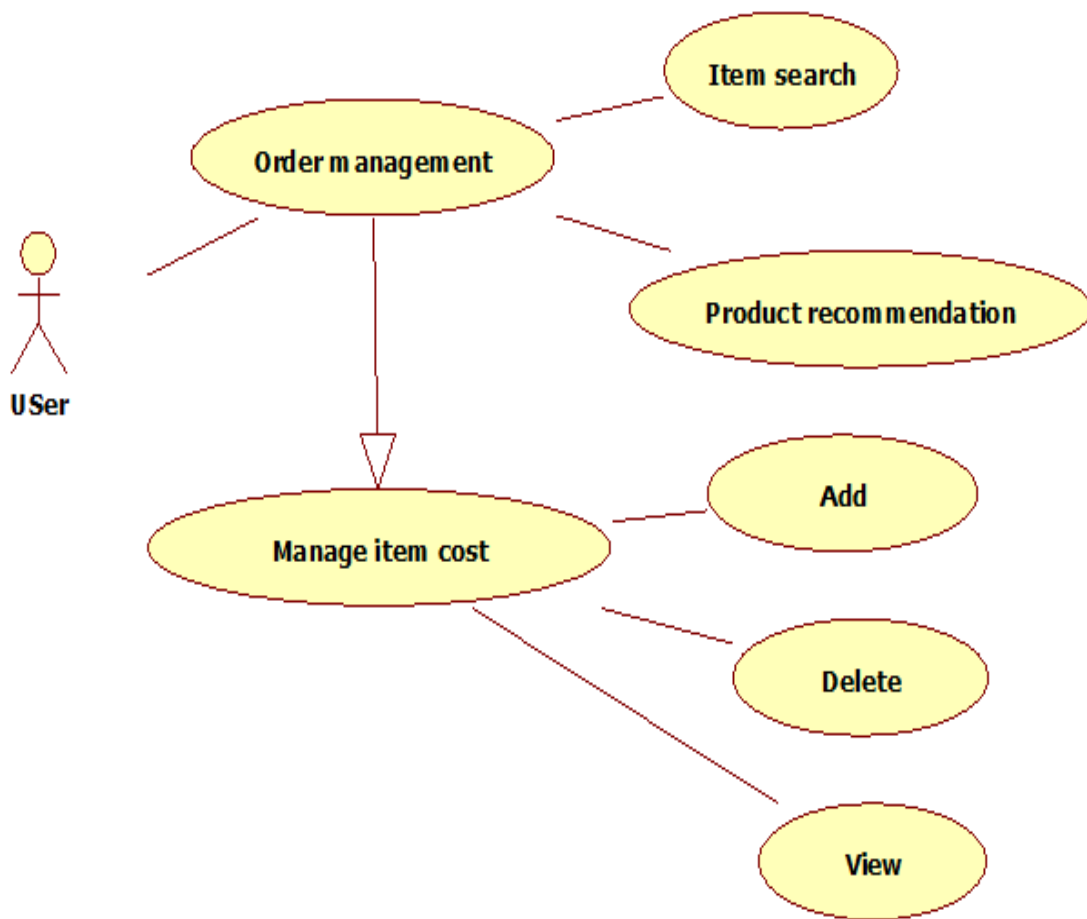


Fig:Use Case diagram

4.3.2. Use Case Scenarios

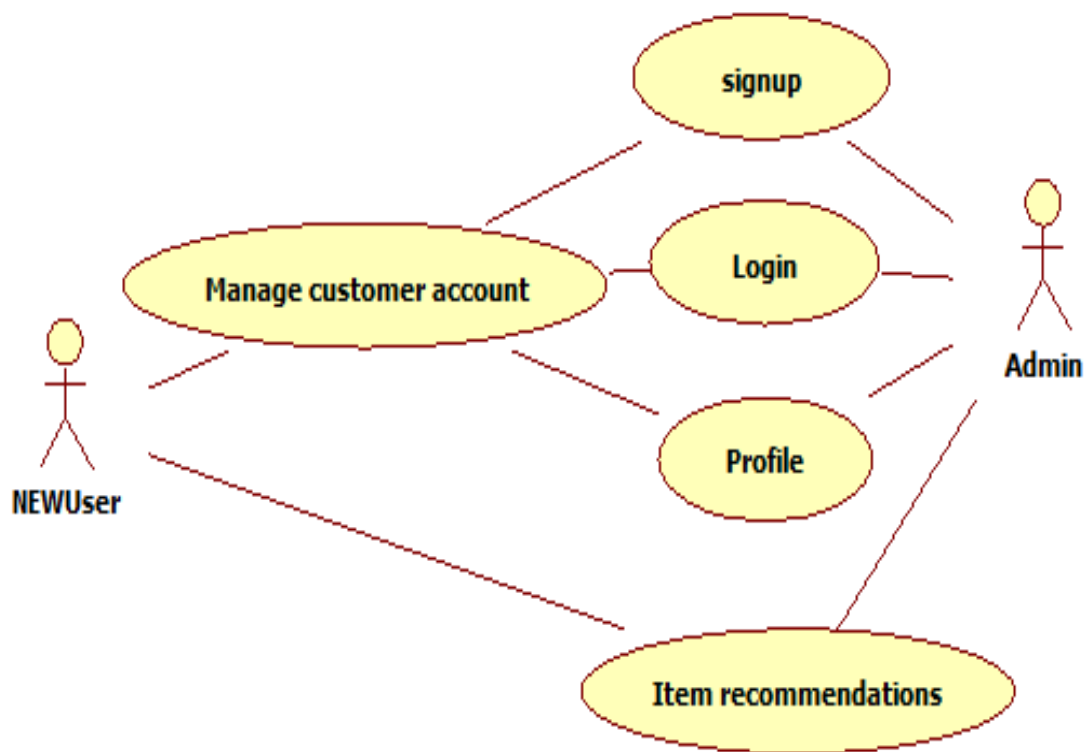
i)Use case for the existing user

This use case diagram is defined to manage his account and purchased items



ii) Use case for New User

This use case diagram defines new user to manage his account with correspondence with the admin.



iii) Use case for New User and Recommender Admin

This use case defines with recommendations of the products to the new user based on his/her interest.



4.4. System Requirements

System requirements specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the systems is to do, without making any commitment as to how the system is to do it.

A system requirements specification is normally produced in response to a user requirements specifications or other expression of requirements, and is then used as the basis for system design. The system requirements specification typically differs from expression of requirements in both scope and precision the latter may cover both the envisaged system and the environment in which it will operate, but may leave many broad concepts unrefined.

4.4.1. Software Requirements

- Hadoop FrameWork runs well on linux
- Mahout: Library for Machine Learning Algorithms
- Ubuntu: Server edition of Ubuntu is good fit
- Python Environment

4.4.2. Hardware Requirements

- Memory : 4GB
- Processor : Dual Core
- Network : 1GB Ethernet

5.System Design

5.1. Introduction

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data

that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

System design mainly concentrates on defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development.

System design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. System design then overlaps with systems analysis, systems engineering and systems architecture.

The system design approach first appeared right before World War II, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science in general.

5.1.1. Class Diagram

The Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing , describing and documenting different aspects of a system but also for constructing executable code of the software application. It describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

Purpose

The purpose of the class diagram is to model the static view of an application .The class diagrams are only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community .So the purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe the responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

Visibility

Use visibility markers to signify who can access the information which is in a class. There are four visibilities. They are:

- Private visibility hides information from anything outside the class partition.
- Public visibility allows all other classes to view the marked information .
- Protected visibility allows child classes to access information which is inherited from a parent class.

Active Class

Active classes initiate and control the flow of activity , while passive classes store data and serve other classes. Illustrate active classes with a thicker border.

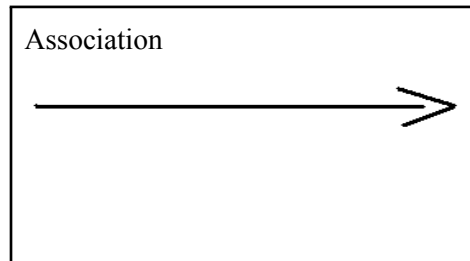
Multiplicity (Cardinality)

Place multiplicity notations at the ends of an association. These symbols indicate the number of instances of one class linked to the instances of other class.

Associations

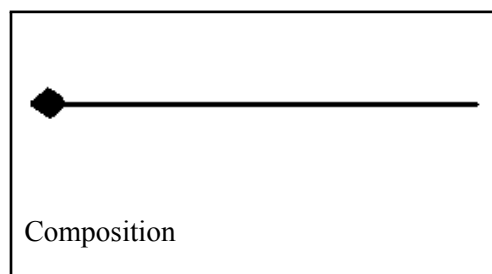
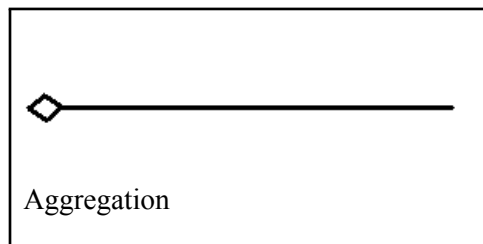
Associations represent static relationship between the classes. Place the association names above, on or below the association line. Use a filled arrow to indicate the direction of the

relationship. Place roles at the end of an association. Roles represent how the two classes see each other.



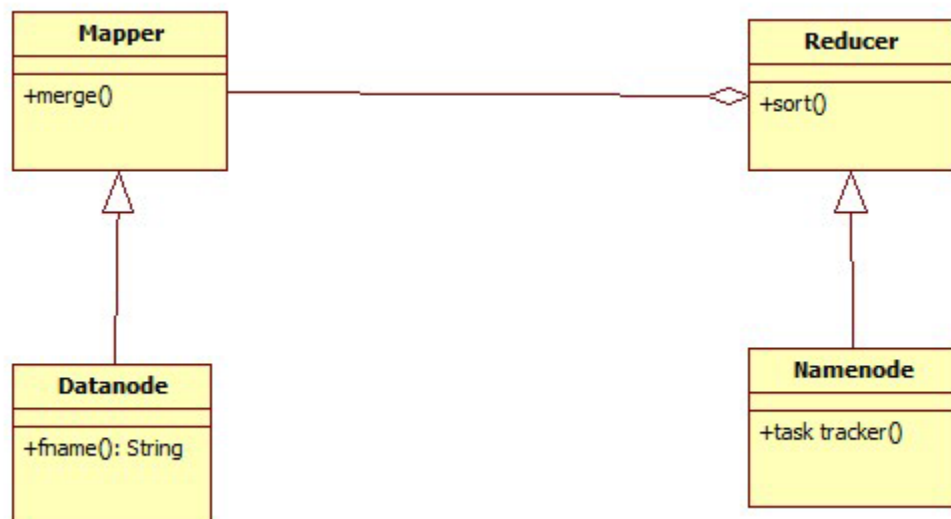
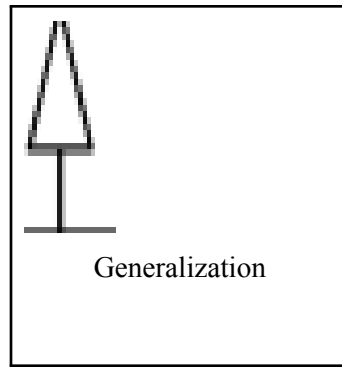
Composition and Aggregation

Composition and Aggregation links a semantic association between two classes in UML diagram. They are used in class diagram. They both differ in their symbols.



Generalization

It is a specification relationship in which objects of the specialized element (the child) are suitable for objects of the generalization element (the parent). It is used in class diagram.



Class Diagram for Mapreduce Scenario

5.1.2. Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and




organisational processes (i.e. workflows). Activity diagrams show the overall flow of control.


Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

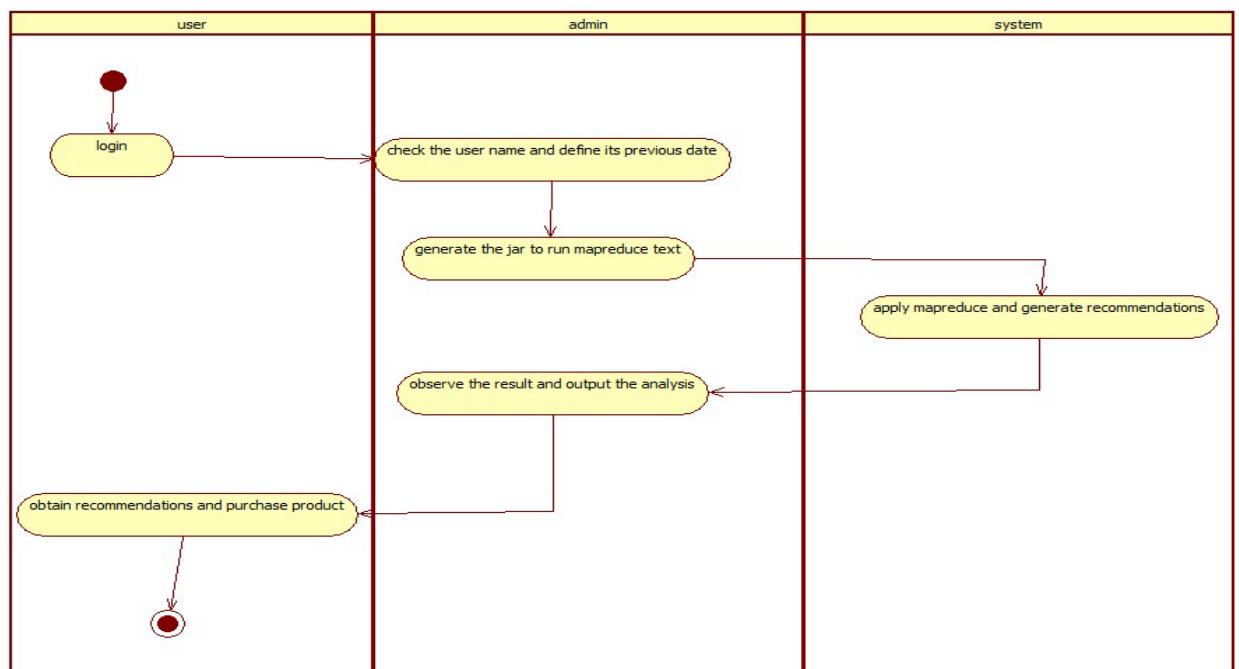
- Rounded rectangles represent actions
- Diamonds represent decisions
- A black circle represents the start (initial state) of the workflow
- An encircled black circle represents the end (final state).

Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

Graphical Representation of Activity Diagrams

Action State	It represents the execution of an atomic action. An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing the execution of the entry action.	
Initial state	.An initial node is a control node at which flow starts when the activity is invoked. An activity may have more than one initial states.	
Transition	Transition is used to show the flow of action from one state	

	to another.	
Final state	An final node is a control node at which flow starts when the activity is invoked.	



Activity Diagram

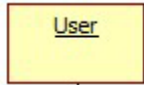
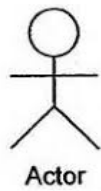


5.1.3.Sequence Diagram

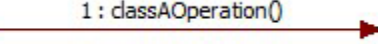
A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is like a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the

scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

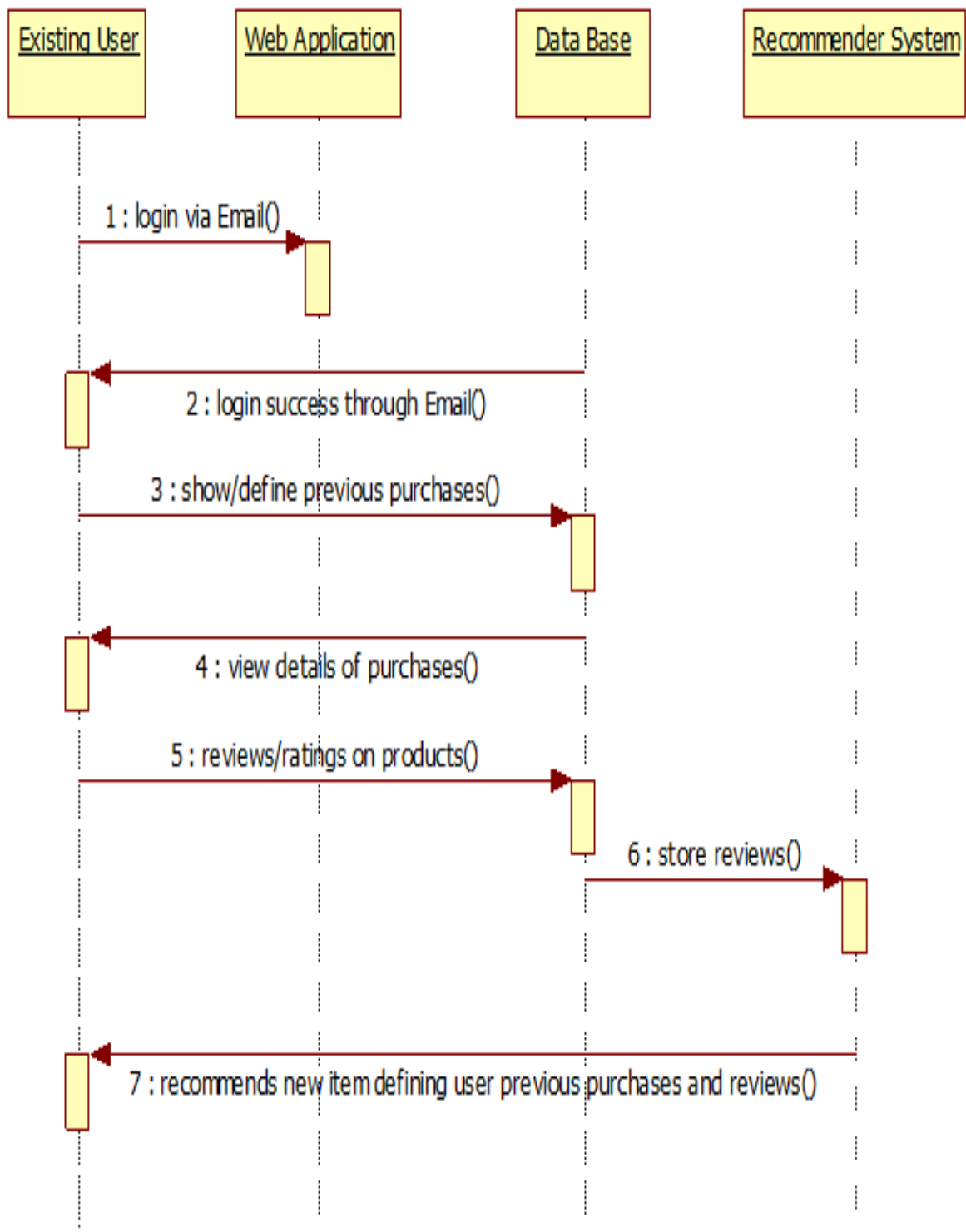
A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom : the interaction starts near the top of the diagram and ends at the bottom.

Graphical Representation of Sequence diagram

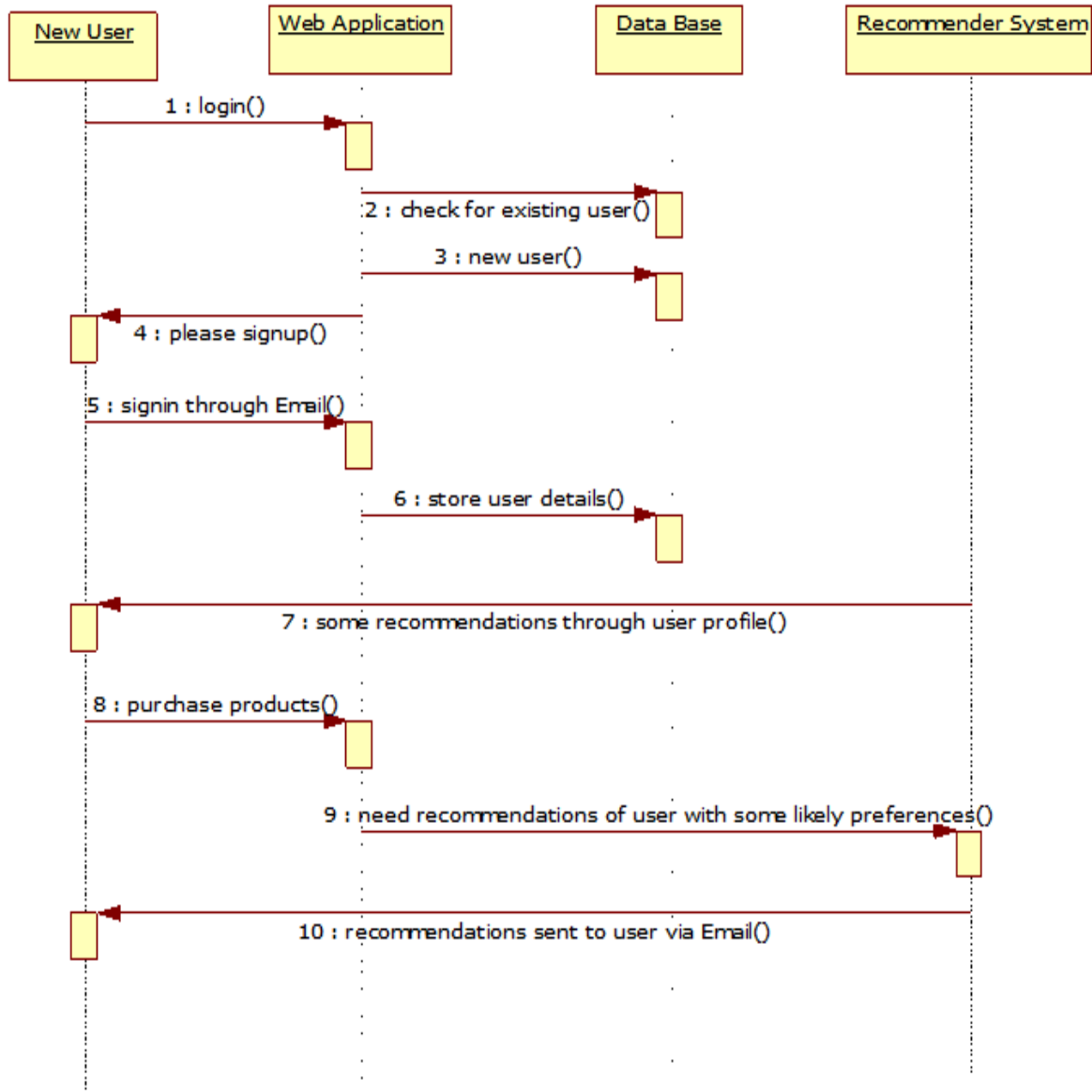
Object	Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name (optional).	
Actor	Actor can also communicate with objects so they too can be listed as a column. An Actor is modeled using the stick figure.	
Lifeline	The Lifeline identifies the existence of the object over time. The notation for a lifetime is a vertical dotted line extending from an object.	
Activation	Activation modeled as rectangular boxes on the lifeline indicate when	

	the object is performing an action	
Message	Messages modeled as horizontal arrows between Activations indicate the communication between the objects.	

Sequence Diagram for Existing User

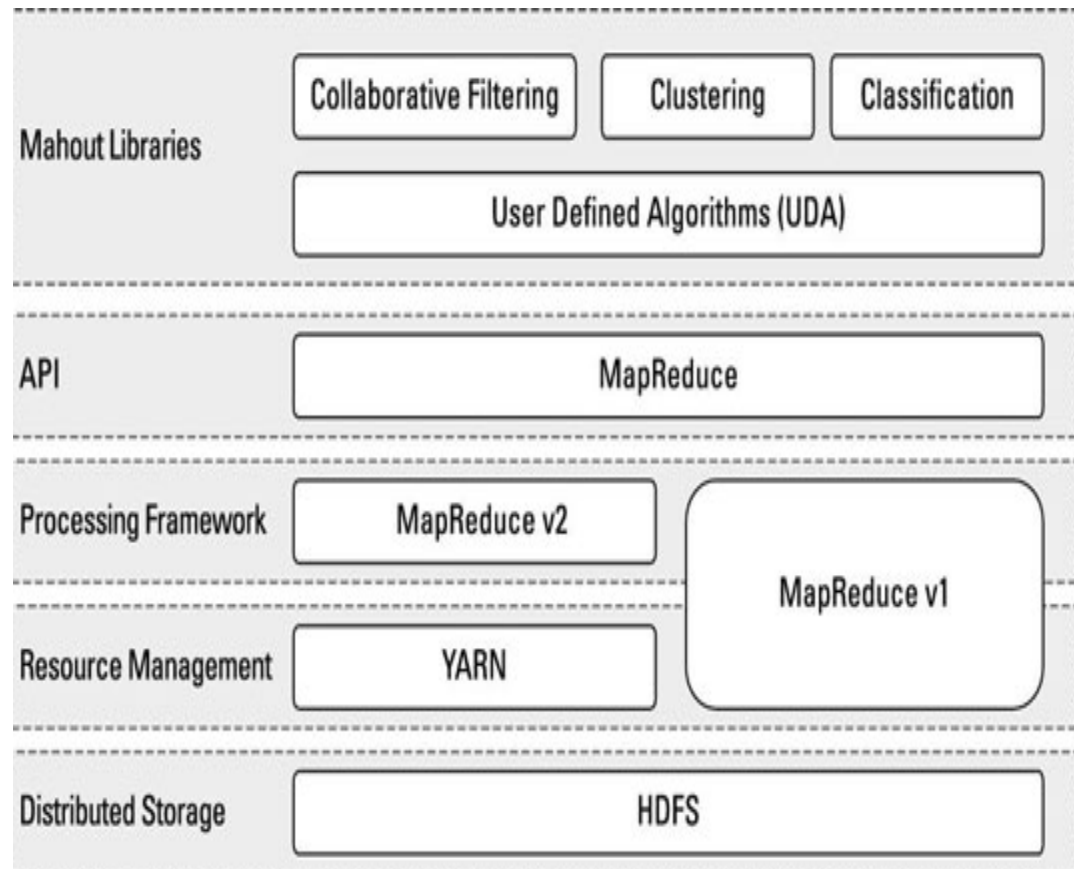


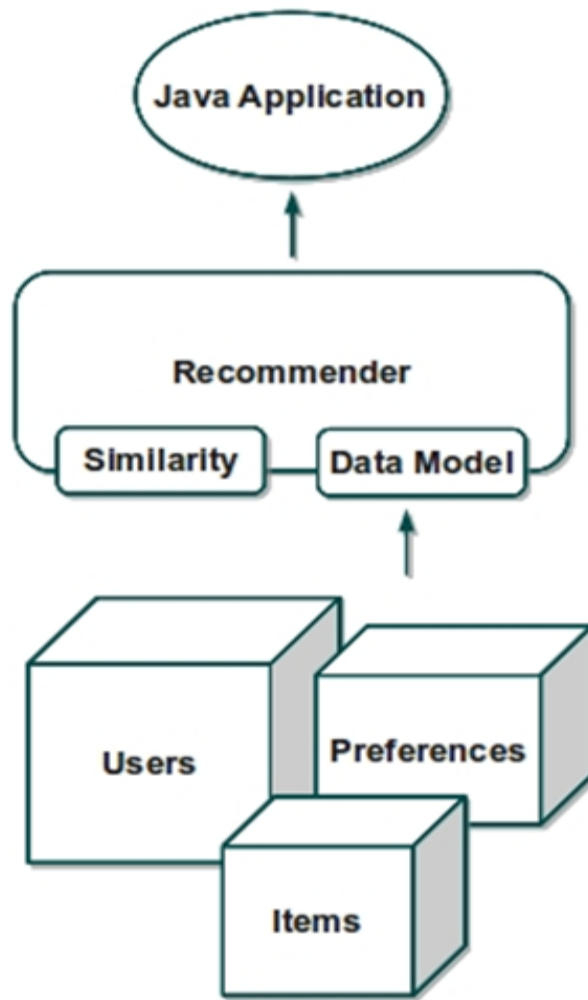
Sequence Diagram for New user Login



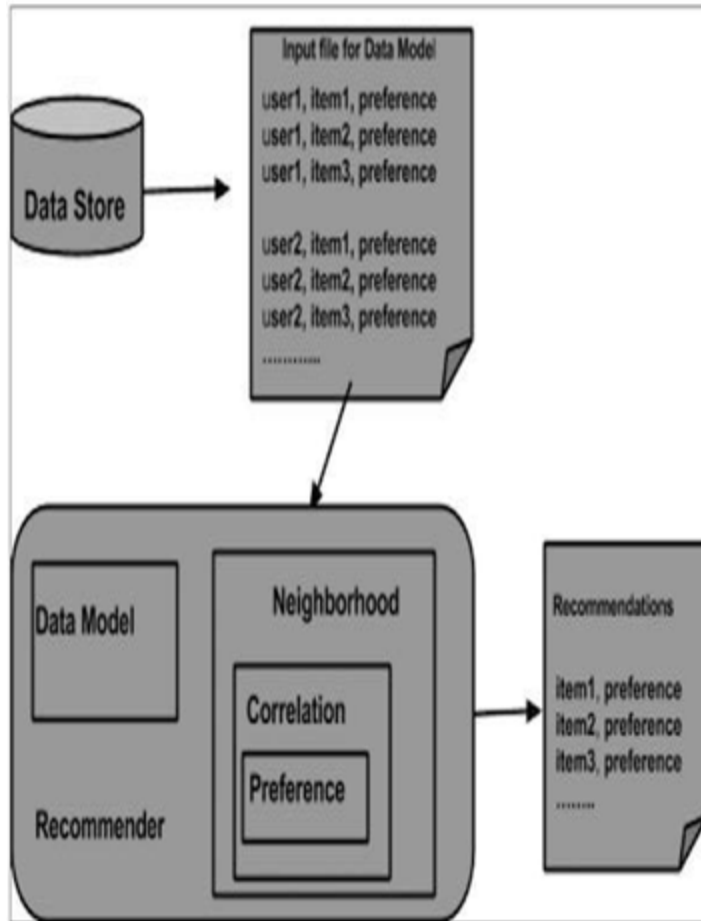
5.2. Architecture Design

HDFS Architecture with Mahout



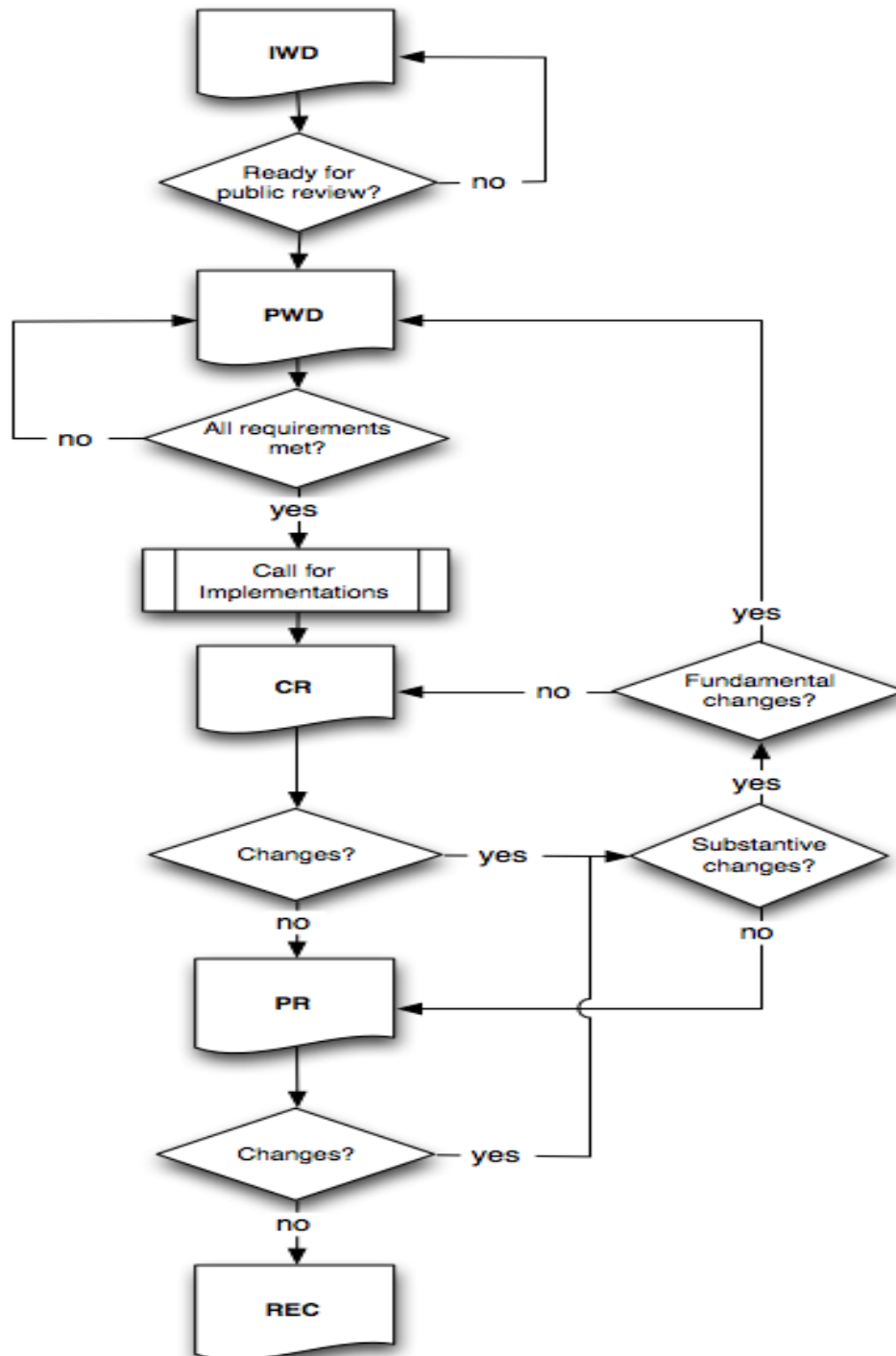


System Architecture



System Design

5.2.1. Algorithm Specification



Algorithm of Map-Reduce

Map Task: // one for each split

Input: S1 // Split i, line - transaction

Output: pairs, where key is an element of candidate itemsets.

1. For each transaction t in Si.
2. Map (line offset,t) // Map function.
3. For each itemset I in t/ *I-all possible subsets of t*/.
4. Out(1,1).
5. End for each.
6. End map.
7. End foreach.
8. End..

Reducer Task

Input: pairs, minimum_support_count, where key2 is an element of the candidate itemsets and value2 is its occurrence in each split.

Output: pairs, key3 is an element of frequent itemsets and value3 is its occurrence in the whole dataset.

1. Reduce (key2,value2)// Reduce function
2. Sum=0;
3. While (value2.hasNext());
4. Sum+= value2.getNext();
5. End while
6. If (sum>= min_sup_count)
7. Out (key2,sum);
8. End if
9. End reduce
10. End

Prerequisites to run the Algorithm

- Divide the database evenly into partitions among all processes
- Each process scans its local database partition to collect the local count of each item
- All processes exchange and sum up the local counts to get the global counts of all items and find frequent 1-itemsets
- On these itemsets we run our technique which is the calculation of support to get the desired output.

Technique that we used

Support

Support is an indication of how frequently the items are recommended

Formula

We use similarity weighing formulas for accuracy which include pearson and tanimoto coefficients

Pearson Coefficient:

$$r(X,Y) = \frac{\sum_i (x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_i (x_i - \bar{X})^2 \sum_i (y_i - \bar{Y})^2}}$$

Tanimoto Coefficient

$$T(a, b) = \frac{N_c}{N_a + N_b - N_c}$$

5.3. User Interface Design

A user interface, also called a "UI" or simply an "interface," is the means in which a person controls a software application or hardware device. A good user interface provides a "user-friendly" experience, allowing the user to interact with the software or hardware in a natural and intuitive way.

The user interface in the industrial design field of human–computer interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

Generally, the goal of user interface design is to produce a user interface which makes it easy (self-explanatory), efficient, and enjoyable (user-friendly) to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human. With the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface is generally assumed to mean the graphical user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

Nearly all software programs have a graphical user interface, or GUI. This means the program includes graphical controls, which the user can select using a mouse or keyboard. A typical GUI of a software program includes a menu bar, toolbar, windows, buttons, and other controls. The Macintosh and Windows operating systems have different user interfaces, but they share many of the same elements, such as a desktop, windows, icons, etc. These common elements make it possible for people to use either operating system without having to completely relearn the interface. Similarly, programs like word processors and Web browsers all have rather similar interfaces, providing a consistent user experience across multiple programs.

Most hardware devices also include a user interface, though it is typically not as complex as a software interface. A common example of a hardware device with a user interface is a

remote control. A typical TV remote has a numeric keypad, volume and channel buttons, mute and power buttons, an input selector, and other buttons that perform various functions. This set of buttons and the way they are laid out on the controller makes up the user interface. Other devices, such as digital cameras, audio mixing consoles, and stereo systems also have a user interface.

While user interfaces can be designed for either hardware or software, most are a combination of both. For example, to control a software program, you typically need to use a keyboard and mouse, which each have their own user interface. Likewise, to control a digital camera, you may need to navigate through the on-screen menus, which is a software interface. Regardless of the application, the goal of a good user interface is to be user-friendly.

6. Implementation

6.1. Technology Description

6.1.1. Apache Hadoop

Apache Hadoop is an open source framework for distributed storage. It process large amounts of datasets on a commodity hardware. Hadoop enables is to take instant decisions in business from massive amounts of structured and unstructured data.

Apache Hadoop is a set of algorithms (a software framework written in Java) for distributed storage and distributed processing of very large data sets (Big Data) on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework.

The base Apache Hadoop framework is composed of the following modules:

- **Hadoop Common** – contains libraries and utilities needed by other Hadoop modules
- **Hadoop Distributed File System (HDFS)** – a distributed file-system that stores data on commodity machines, providing very high aggregate storage across the cluster
- **Hadoop MapReduce** – a programming model for large scale data processing.

Hadoop EcoSystem

Hadoop platform consists of two key services: a reliable, distributed file system called Hadoop Distributed File System (HDFS) and the high-performance parallel data processing engine called Hadoop MapReduce, described in MapReduce below. Hadoop was created by Doug Cutting and named after his son's toy elephant. Vendors that provide Hadoop-based platforms include Cloudera , MapR, IBM, and Amazon.

Hadoop Ecosystem Core Components:

- **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop
- **Hadoop YARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

The Sub Components

- **Hbase:** HBase is a column-oriented database that uses HDFS for underlying storage of data. HBase supports random reads and also batch computations using MapReduce. With HBase NoSQL database enterprise can create large tables with millions of rows and columns on hardware machine. The best practice to use HBase is when there is a requirement for random 'read or write' access to big datasets.
- **Hive:** Hive developed by Facebook is a data warehouse built on top of Hadoop and provides a simple language known as HiveQL similar to SQL for querying, data summarization and analysis. Hive makes querying faster through indexing.
- **Pig:** Apache pig is a convenient tools developed by Yahoo for analysing huge data sets efficiently and easily. It provides a high level data flow language Pig Latin that is optimized, extensible and easy to use. The most outstanding feature of Pig programs is that their structure is open to considerable parallelization making it easy for handling large data sets.
- **Mahout:** Mahout is an important Hadoop component for machine learning, this provides implementation of various machine learning algorithms. This Hadoop component helps

with considering user behavior in providing suggestions, categorizing the items to its respective group, classifying items based on the categorization and supporting in implementation group mining or itemset mining, to determine items which appear in group. While Mahout's core algorithms for clustering, classification and batch based collaborative filtering are implemented on top of Apache Hadoop using the map/reduce paradigm, it does not restrict contributions to Hadoop-based implementations. Contributions that run on a single node or on a non-Hadoop cluster are also welcomed. Mahout also provides Java libraries for common maths operations (focused on linear algebra and statistics) and primitive Java collections.

6.1.2. Python

Python is an interpreted high-level programming language for general-purpose programming. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms including object-oriented, functional and procedural, imperative and has a large and comprehensive standard library.

6.2. System Modules

HDFS

HDFS means Hadoop Distributed File System. We are using hadoop single node cluster in our proposed system. In Hadoop single node, both the master node and slave node resides in the same node. Hadoop processing normally contains 5 main components. They are

- **Name Node** : The name node is the center piece for HDFS file system. It keeps the directory tree of all the files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. client applications talk to the name node whenever they wish to locate a file. The namenode responds the successful requests by returning a list of relevant data node servers where the data lives.
- **Data Node** : The data node stores data in the hadoop file system. a Functional file system has more than one data node, with data replicated across them. Client applications can talk directly to the data node, once the namenode has provided the locations of the data.
- **Secondary Name Node** : Secondary NameNode is specially dedicated node in HDFS cluster whose main function is to take checkpoints of the file system metadata present on the namenode. The secondary name node is a helper to the primary namenode.
- **Node Manager** : It is a Java utility that runs as separate process from WebLogic Server and allows you to perform common operations tasks for a Managed Server, regardless of its location with respect to its Administration Server.
- **Resource Manager** : Resource Manager (RM) is responsible for tracking the resources in a cluster, and scheduling applications (e.g., MapReduce jobs). Prior to Hadoop 2.4, RM is the single point of failure in a YARN cluster.

6.3. Sample Source Code

Mapper
Reducer

7. Testing

7.1. Introduction

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test have the following

- Meets the requirements that guided its design and development
- Responds correctly to all kinds of inputs
- Performs its functions within an acceptable time
- Is sufficiently usable
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholders desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

Testing Types

A software engineering product can be tested in one of two ways:

- Black Box Testing
- White Box Testing

Black box testing

Knowing the specified function that a product has been designed to perform, determine whether each function is fully operational.

White box testing

Knowing the internal workings of a software product determine whether the internal operation implementing the functions perform according to the specification, and all the internal components have been adequately exercised.

Testing Strategies

Four Testing Strategies that are often adopted by the software development team include: Testing Strategies Four Testing Strategies that are often adopted by the software development team include:

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

Unit Testing

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase[9]. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the data flow diagram and ensure that they perform the purpose intended for them.

All the names of the variables are scrutinized to make sure that they are truly reflected of the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Beside these, we trace through the code manually to capture syntax errors and logical errors.

Integration Testing

After finishing the Unit Testing process, next is the integration testing process. In this testing process we put our focus on identifying the interfaces between components and their functionality as dictated by the DFD diagram. The Bottom up incremental approach was adopted during this testing. Low level modules are integrated and combined as a cluster before testing.

The Black box testing technique was employed here. The interfaces between the components were tested first. This allowed identifying any wrong linkages or parameters passing early in the development process as it just can be passed in a set of data and checked if the result returned is an accepted one.

Validation Testing

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

System Testing

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and perform allocated functions. System testing also ensures that the project works well in the environment. It traps the errors and allows convenient processing of errors without coming out of the program abruptly.

Recovery testing is done in such a way that failure is forced to a software system and checked whether the recovery is proper and accurate. The performance of the system is highly effective.

Software testing is critical element of software quality assurance and represents ultimate review of specification, design and coding. Test case design focuses on a set of technique for the creation of test cases that meet overall testing objectives. Planning and testing of a programming system involve formulating a set of test cases, which are similar to the real data that the system is intended to manipulate. Test cases consist of input specifications, a description of the system functions exercised by the input and a statement of the expected output.

In principle, testing of a program must be extensive. Every statement in the program should be exercised and every possible path combination through the program should be executed at least once. Thus, it is necessary to select a subset of the possible test cases and conjecture that this subset will adequately test the program.

Guidelines for developing test cases

- Describe which feature or service your test attempts to cover
- If the test case is based on a use case it is a good idea to refer to the use case name.
- Remember that the use cases are the source of test cases. In theory the software is supposed to match the use cases not the reverse. As soon as you have enough use cases , go ahead and write the test plan for that piece
- Specify what you are testing and which particular feature. Then specify what you are going to do to test the feature and what you expect to happen.
- Test the normal use of the object's methods. Test the abnormal but reasonable use of the object's methods.
- Test the abnormal but unreasonable use of the object's methods.
- Test the boundary conditions. Also specify when you expect error dialog boxes, when you expect some default event, and when functionality till is being defined.
- Test object's interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process
- when the revisions have been made, document the cases so they become the starting bases for the follow- up test

Attempting to reach agreement on answers generally will raise other what-if questions. Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions.

7.2. Test Cases

TestcaseNo.	User Input Specification	Test Condition	Output
1	Invalid input exception	Exception	Input file was not found
2	Not a valid jar	Error	Empty file
3	Empty Output	Error	Empty file

7.3.Test Results

//screenshots of test cases

8.ScreenShots

//of project execution and results

9.Conclusion

The main aim of this project is to run an algorithm on MapReduce parallelly on all the clusters and obtain the data analysis results. We run our algorithm on distributed environment so that time consumption is less and also our algorithm is easy to implement. By applying our algorithm we generate the recommendations for the users.

We have identified various datasets which are high dimensional in nature. Ability to grasp advanced programming techniques to solve contemporary issues. We formulated the threshold called the support upon analysis of the data. We designed mapreduce program to define the user preference reviews and ratings.

With the algorithm using metrics users get recommendations of certain products based on user and item collaborative filtering. We define mapper and reducer for fast execution. Our recommender system improves the scalability of the users and provide them with accurate recommendations

10. Bibliography

1. C. Sammut and G. I. Webb, Encyclopedia of Machine Learning. Springer, 2011.

2. (2014, November) IBM what is big data? Bringing big data to the enterprise. [Online]. Available: <http://www.ibm.com/big-data/us/en/>
3. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th international conference on World Wide Web. ACM, 2001, pp. 285–295.
4. Z.-D. Zhao and M.-S. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in Knowledge Discovery and Data Mining, 2010. WKDD 10. Third International Conference on. IEEE, 2010, pp. 478–481.
5. J. Jiang, J. Lu, G. Zhang, and G. Long, "Scaling-up item-based collaborative filtering recommendation algorithm based on hadoop," in Services (SERVICES), 2011 IEEE World Congress on. IEEE, 2011, pp. 490–497.
6. Apache Mahout, <http://mahout.apache.org>.
7. J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. Commun. ACM, 51:107–113, 2008.
8. T. Dunning. Accurate methods for the statistics of surprise and coincidence. Comput. Linguist., 19:61–74, 1993.
9. S. Funk. Netflix Update: Try This at Home, <http://sifter.org/simon/journal/20061211.html>. 2006.
10. Y. Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. ICDM, pages 263–272, 2008.