Savitribai Phule Pune University

Modern Education Societys College of Engineering, Pune

19, Bund Garden, V.K. Joag Path, Pune 411001.

ACCREDITED BY NAAC WITH "A" GRADE (CGPA 3.13)

DEPARTMENT OF COMPUTER ENGINEERING

A

Mini-Project Report

on

"Udankhatola Airline Booking System"

B.E. (COMPUTER)

SUBMITTED BY

Mr. Aditya Ujeniya	**Roll No: Q23**
Mr. Shubham Shingade	**Roll No: Q18**
Mr. Rahul Pawar	**Roll No: Q11**
Ms. Sushila Choudhari	**Roll No: Q21**

UNDER THE GUIDANCE OF

Prof. R. M. WAHUL

(Academic Year: 2018-2019)

Modern Education Societys College of Engineering, Pune

19, Bund Garden, V.K. Joag Path, Pune 411001.

ACCREDITED BY NAAC WITH "A" GRADE (CGPA 3.13)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that seminar entitled

"DS PROJECT"

has been completed by Mr. Aditya Ujeniya, Mr. Rahul Pawar, Mr. Shubham Shingade and Ms. Sushila Choudhari of BE COMP II in the Semester - I of academic year 2018-2019 in partial fulfillment of the Fourth Year of Bachelor degree in "Computer Engineering" as prescribed by the Savitribai Phule Pune University.

Prof R. M. Wahul	**(Dr.(Mrs.) N. F. Shaikh)**
Seminar Guide	**H.O.D**

Place: MESCOE, Pune.

Date: / /2017

ACKNOWLEDGEMENT

It gives me great pleasure and satisfaction in presenting this mini-project on DS.

I would like to express my deep sense of gratitude towards the Principal ***Dr. A.A Keste****.*

I have furthermore to thank Computer Department HOD ***Dr.(Mrs.) N. F. Shaikh*** *and* ***Prof. R. M. Wahul*** *to encourage me to go ahead and for her continuous guidance. I would also like to thanks all the teaching staff for their assistance and guidance for preparing the report.*

I would like to thank all those, who have directly or indirectly helped me for the completion of the work during this mini project.

Mr. Aditya Ujeniya
Mr. Rahul Pawar
Mr. Shubham Shingade
Ms. Sushila Chouadhari

Contents

List of Figures

Chapter 1

Problem Statement

Design, implement, and thoroughly test a distributed system, implementing - An airline reservation system. Each airline would maintain its own collection of servers, with enough state replication to enable automatic fail-over. It would be possible to book travel that involves multiple airlines

Chapter 2

Introduction

2.1 DISTRIBUTED SYSTEM

Leslie Lamport once noted, A distributed system is one within which the failure of a system you didnt even understand existed will render your own system unusable. []

A distributed system typically satisfies the following criteria:

1. Multiple processes
2. Inter-process communication
3. Disjoint address spaces
4. Collective goal

2.2 CHARARCTERISTICS OF D.S

2.2.1 Resource sharing

1. The opportunity to use available hardware, software or data any where in the system
2. Resource managers control access, offer a scheme for naming, and controls concurrency
3. A resource manager is a software module that manages a resource of a particular type

2.2.2 Concurrency

1. Components in DS execute in concurrent processes
2. Components access and update shared resources (e.g. variables, data bases, device drivers)
3. Integrity of the system may be violated if concurrent updates are not coordinated
4. Preservation of integrity requires concurrency control where concurrent access to the same resource is synchronized

2.2.3 Scalability

1. A system is scalable if it remains effective when there is a significant increase in the amount of resources and number of users
2. Scalability denotes the ability of a system to handle an increasing future load

2.2.4 Failure handling

1. DS must maintain availability even in cases where hardware/software/network have low reliability
2. Failures in distributed systems are partial

2.3 APPLICATIONS OF D.S

1. World Wide Web
2. Social Networks
3. Grid and Cloud Computing
4. Massively multiplayer online games
5. Financial Trading

Chapter 3
System Models

Main system models of DS:

1. Physical Models
2. Architectural Models
3. Fundamental Models

3.1 ARCHITECTURAL MODEL

The architecture of a system is its structure in terms of separately specified components and their interrelationships. The overall goal is to ensure that the structure will meet present and likely future demands on it. An architectural model of a distributed system defines the way in which the components of the system interact with each other and the way in which they are mapped onto an underlying network of computers. It mainly comprises of Peer-to-Peer and Client-Server architectural models. The client-server model can be modified by:

1. The partition of data or replication at co-operating servers
2. The caching of data by the proxy servers and clients
3. The use of mobile codes and mobile agents. e.g. JAVA Applets

3.1.1 Proxy Server

A Proxy Server is a server that acts as a middleware for requests from clients requesting resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, or other resources available from a

different server and the proxy server estimates the request as a way to simplify and control its complexity.

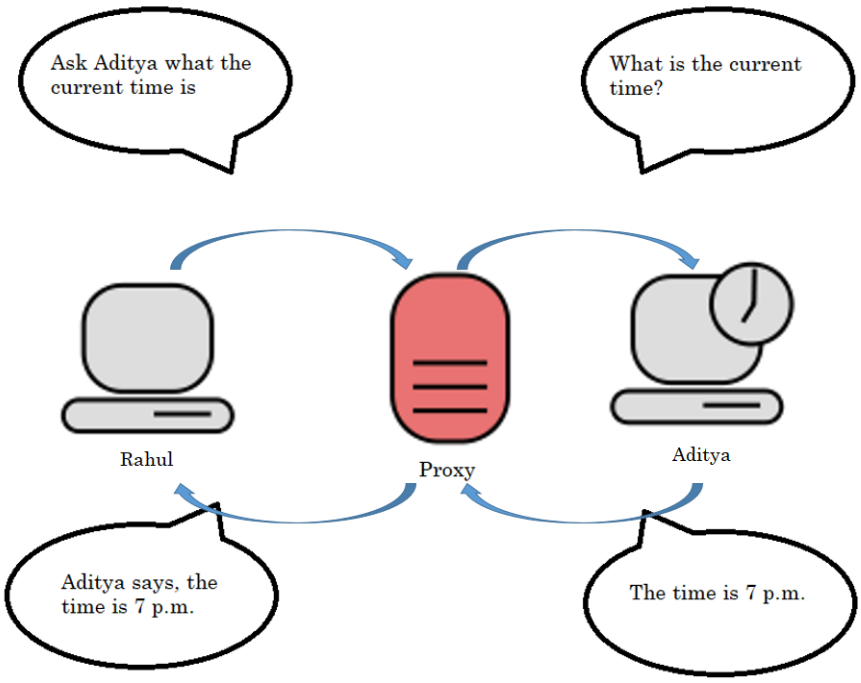

Figure 3.1: Communication through Proxy Server

Bypassing filters In Proxy Server

A caching proxy server accelerates service requests by retrieving content saved from a previous request made by the same client or even other clients. Caching proxies keep local copies of frequently requested resources, allowing large organizations to significantly reduce their upstream bandwidth usage and costs, while significantly increasing performance. Caching proxies were the first kind of proxy server.

A proxy can keep the internal network structure of a company secret by using network address translation, which can help the security of the internal network. This makes requests from machines and users on the local network anonymous. Proxies can also be combined with firewalls.

Chapter 4

Implementation

4.1 PROJECT OBJECTIVE

The objective of this project is to implement and demonstrate a distributed system for An Airline Reservation System using a Proxy Server.

It will highlight the following key-features:

1. Fault-tolerant
2. Consistent database on all servers
3. Concurrency
4. Scalable system with multiple nodes(server/client)
5. Caching mechanism
6. Load balancing

4.2 PROJECT DESCRIPTION

This Mini-Project consists of the following components:

1. Multiple Servers (minimum 2)
2. A Proxy Server
3. Multiple Clients (minimum 4)

Multiple Servers

The main function of the server is to maintain all the records about airline reservations with an updated database copy. The servers are designed in such a way that it can withstand any network or hardware failure, and can still function like a coherent system.

A Proxy Server

Proxy server acts as a middleware which filters all the client requests' and directs it to the servers. The main function of a proxy server is that it maintains a cache. Whenever a request is made for any transaction, proxy server will look into the cache for similar transactions made previously. If any result to the query is matched the proxy server sends a suggestion to the user. Based on the criteria proxy server saves all frequently made requests' by user in its cache for further use. This will improve the user response time.

Multiple Clients

Clients here refer to thin clients, who have minimum access to resources and heavily rely on a server for its computational role. They are centrally managed by the proxy server. They can be scalable.

4.2.1 Minimum Requirements

Software Requirements

1. Ubuntu - Any version
2. Apache server - 7.0.5+
3. PHP - Any version
4. MySQL - 3.5+
5. Bootstrap - 3+

Hardware Requirements

1. 512 MB RAM
2. 500 MB Disk Space
3. 64-bit Architecture

4.2.2 Screenshots

MainPage

This is the homepage of flight booking system where the user who wants to book the flight will provide the details into the **To** and **From** DropDown menu. The menu will show only the flights which are available at that moment. The user need to input the number of seats too. If the desired number of seats are not available, then use may not get any details of available flights with desired seats.

The database here is linked with phpmyadmin where the available flights are fetched from database to present infront of user. PHP is used as a hybrid language where GUI and backend can be connected with ease.

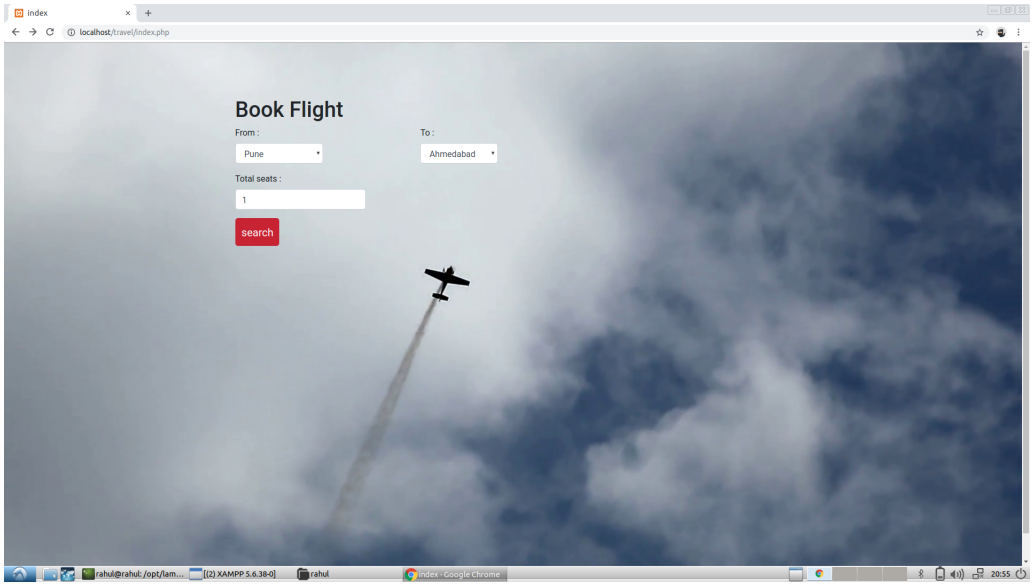

Figure 4.1: Main Page

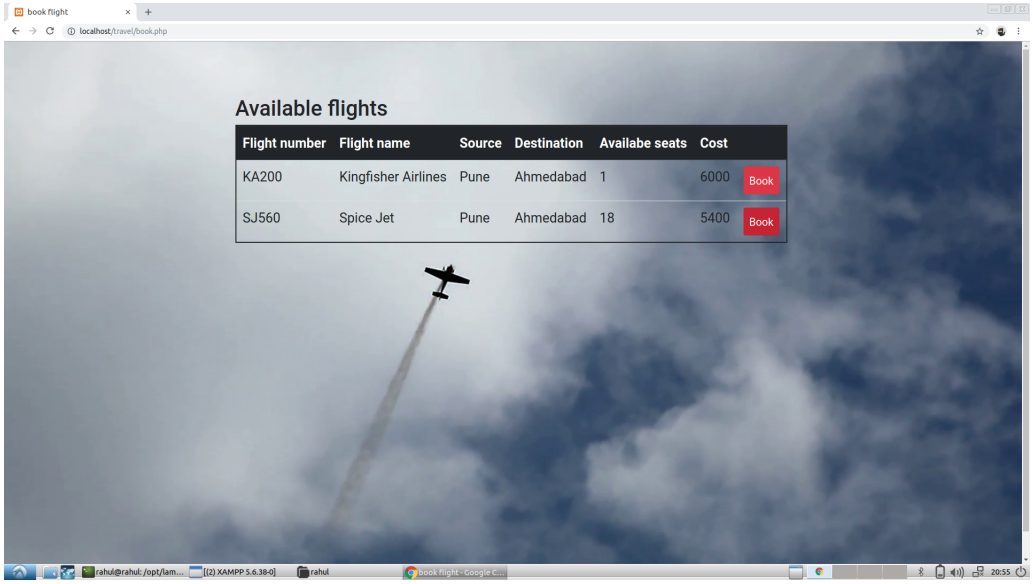

Figure 4.2: Checking availability of tickets

Checking availability

The data is fetched from the thee database, where the Flight details entered by user are compared with Fight table and the number of seats are checked whether the desired seats are available or not. Various flights with available seats are then displayed onto the screen. The cost of each flight is multiplied by the desired seats entered and the total cost is displayed to user.

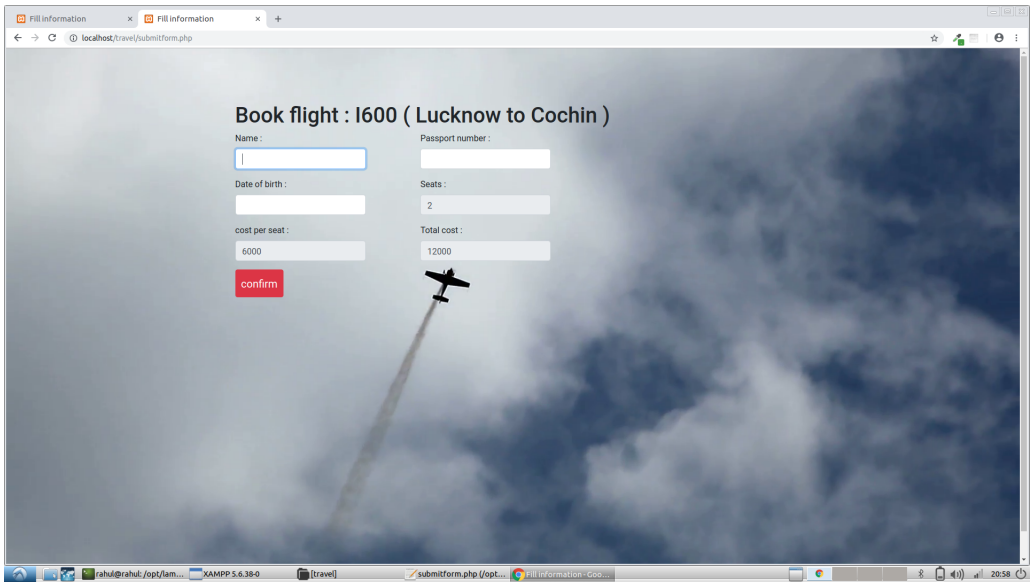

Figure 4.3: Confirmation Page

Ticket Confirmation

Atlast, when the user enters the details like **Name**, **Passport Number** and **Date**, the details are fetched by php, processed in backend by the database and stored in Booking table. Simultaneously, the data fetched is also sent to the main servers where the servers update their existing Booking table. The proxy server maintains the cache for ease of management and load distribution. Even if any of the main server goes down, the existing proxy server and the connected server will continue to update their Booking and Flight tables.

Chapter 5
Main Highlited Characteristics

Fault Tolerance

Fault tolerance refers to the ability of a system to continue functioning even in case of partial failure. Since Distributed systems are made up of large number of components developing a system which is hundred percent fault tolerance is practically impossible.

We designed a system in such a way that even if one node/sever crashes down, the users are still able to access the resources of the system via other replicated copies of servers. This doesn't lead to total failure of the system. When the server fails to give resources to the user, that is the node/server has crashed or can no longer be accessible, the entire network traffic is redirected to the rest of the server. These internal issues are kept hidden from the user and they freely operate in a consistent state.

We detect for failures by constantly monitoring the performance and comparing it with the expected outcome.

Load Balancing

Load balancing is the way of scattering load units/components across a set of processors which are connected to a distributed network. The surplus load or the remaining unexecuted load from a processor is migrated to other processors which have load below the threshold load.(citation).

Threshold load is an amount of load to a processor that which it can withstand in future.

In our distributed system, there is high chance of some nodes being overloaded, while others being idle. To tackle this situation the proxy server identifies the processors in a system according to their present load as Heavily Loaded Processors, Lightly Loaded Processors and Idle processors. Through load balancing technique it is possible to make each processor equally busy and to approximately finish the tasks at the same time.

Timestamp

Timestamp is a datatype derived from SQL where consistency is maintained by comparing two or more time-stamps. The timestamp is derived by taking the local time of the PC and timestamp attached with the message from the server. Timestamp is in the format of **DD/MM/YYYY HH:MM:SS.ss** where precision is obtained even in milliseconds.

We are using timestamp in a specific time constraint manner where, the bookings are based on the timestamp of the user. Suppose there are two users who want to book the same flight and limited number of tickets are available, then based on the timestamp of the user whoever is having lower timestamp will get the tickets booked. Thus, concurrency is maintained.

Chapter 6

CONCLUSION

Hence we have successfully implemented an airline booking system using Distributed System. The characteristics that we have implemented are Load Balancing, Concurrency, Fault Tolerance and Transaction Control.

Bibliography

[1] Hybrid prefetching for WWW proxy servers, Yui-Wen Horng ; Wen-Jou Lin ; Hsing Mei, ISBN: 0-8186-8603-0, DOI: 10.1109/ICPADS.1998.741130

[2] A scalable and distributed WWW proxy system, K.L.E. Law ; B. Nandy ; A. Chapman, ISBN: 0-8186-7819-4,DOI: 10.1109/MMCS.1997.609770

[3] A distributed object-oriented system with multi-threads of services, F.-J. Wang ; J.-L. Chen ; C.-H. Hu, ISBN: 0-8186-4430-3, DOI: 10.1109/FTDCS.1993.344149

[4] A proxy server management scheme for continuous media objects based on object partitioning, Seh Chul Park ; Yong Woon Park ; Yoo Ek Son, ISBN: 0-7695-1153-8, DOI: 10.1109/ICPADS.2001.934894

[5] Self-organizing cooperative WWW caching, S. Inohara ; Y. Masuoka ; J. Min ; F. Noda, ISBN: 0-8186-8292-2, DOI: 10.1109/ICDCS.1998.679489

[6] Consideration about the cache server optimization and stable network access, T. Miyoshi, ISBN: 0-7803-4316-6, DOI: 10.1109/KES.1998.725888

[7] Distributed load balancing model for grid computing environment, Jagdish Chandra Patni ; Mahendra Singh Aswal ,ISBN: 978-1-4673-6809-4, DOI: 10.1109/NGCT.2015.7375096

[8] Efficient load balancing using improved central load balancing technique, Simranjit Kaur ; Tejinder Sharma, ISBN: 978-1-5386-0807-4, DOI: 10.1109/ICISC.2018.8398857

[9] A comparative study of static and dynamic load balancing algorithms in distributed systems, T. Deepa ; Dhanaraj Cheelu, ISBN: 978-1-5386-1887-5, DOI: 10.1109/ICECDS.2017.8390086