DATA CACHE CONSISTENCY BY BROADCASTING WITH CATALOG BINDING IN E-COMMERCE SYSTEM

Myo Myo San

M.C.Sc. **June 2022**

DATA CACHE CONSISTENCY BY BROADCASTING WITH CATALOG BINDING IN E-COMMERCE SYSTEM

By

Myo Myo San

B.C.Sc.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Computer Science

(M.C.Sc.)

University of Computer Studies, Yangon

June 2022

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to those who helped me with various aspects of conducting research and writing this thesis. To complete this thesis, many things are needed like my hard work as well as the supporting of many people.

First and foremost, I would like to express my deepest gratitude and my thanks to **Dr. Mie Mie Khin**, Rector of the University of Computer Studies, Yangon, for her kind permission to submit this thesis.

I would like to express my appreciation to **Dr. Si Si Mar Win and Dr. Tin Zar Thaw**, Professors, Faculty of Computer Science**,** University of Computer Studies, Yangon, for their superior suggestions, administrative supports and encouragement during my academic study.

My thanks and regards go to my supervisor, **Dr. Zaw Tun**, Prorector of the University of Computer Studies, Sittway, for his support, guidance, supervision, patience and encouragement during the period of study towards completion of this thesis.

I also wish to express my deepest gratitude to **Daw Hnin Yee Aung**, Lecturer, Department of English, University of Computer Studies, Yangon, for her editing this thesis from the language point of view.

Moreover, I would like to extend my thanks to all my teachers who taught me throughout the master's degree course and my friends for their cooperation.

I especially thank to my parents, all of my colleagues, and friends for their encouragement and help during my thesis.

ABSTRACT

Caching is a simple and effective way to provide faster access to information on the Internet, and it has been used effectively to improve the E- commerce servers. However, there is a growing concern by today's Web service providers to ensure that the data cached at a remote client is up to date with the current value in the server. A unique aspect of cache consistency for E-commerce applications is that most of the applications on the Web can tolerate some degree of inconsistency between the clients and the server. This system presents Broadcasting with Catalog Binding (BWC) Cache Consistency which exploits the tolerance exhibited by E-commerce applications, and which is suitable for E-commerce transactions on the Internet.

. This system is implemented using ASP.Net programming language with Microsoft SQL Server Database 2017 Express.

CONTENTS

		Page
ACKNOWLEDGEMENTS		**i**
ABSTRACT		**ii**
CONTENTS		**iii**
LIST OF FIGURES		**vi**
CHAPTER 1	**INTRODUCTION**	**1**
	1.1 Objective of Thesis	1
	1.2 Overview of the System	2
	1.3 Organization of the Thesis	2
CHAPTER 2	**BACKGROUND THEORY**	**3**
	2.1 Concurrency Control	3
	2.2 Type of Concurrency Control	4
	2.2.1 Pessimistic Concurrency Control	4
	2.2.2 Optimistic Concurrency Control	4
	2.3 Typical Problems between the Concurrency Executions of Transactions	5
	2.3.1 Lost or Buried Updates	5
	2.3.2 Inconsistent Analysis (Non Repeatable Read)	6
	2.3.3 Uncommitted Dependency (Dirty Read)	7
	2.3.4 Phantom Reads	7
	2.4 Distributed Database	8
	2.5 The Structure of Distributed Transactions	9
	2.6 Distributed Transactions	10
	2.7 Modeling A Distributed DBMS	10

2.8 Distributed Concurrency Control Algorithm 11
2.8.1 Distributed Two-Phase Locking (2PL) 11
2.8.2 Wound-Wait (WW) 12
2.8.3 Basic Timestamp Ordering (BTO) 12
2.8.4 Distributed Certification (OPT) 13
2.9 Related Work 14

CHAPTER 3 CACHE CONSISTENCY CONTROLLING

3.1 Web Caching 15
3.2 Caching Scheme 15
3.2.1 Client-Initiated Approach 16
3.2.2 Server-Initiated Approach 16
3.3 Caching in Memory 16
3.4 Mechanism of a Web Request 17
3.5 Caching Web Resources 18
3.5.1 Content Cacheability 21
3.5.2 Cache Latency 22
3.6 Maintaining Consistency of Client-Cache Data 22
3.7 Cache Consistency For the Internet 23
3.7.1 Weak Cache Consistency (Client-Initiated Approach) 24
3.7.2 Strong Cache Consistency (Server-Initiated Approach) 25
3.8 Caching Strategies 26
3.8.1 Static Caching 26
3.8.2 Dynamic Caching 27
3.9 Caching in HTML 28
3.10 Caching Granularity 28
3.11 Caching Effectiveness 30
3.12 Traditional Web Caching Protocols 30
3.13 Simple Caching Rules 31

3.13.1 Rule 1: Cache all Static Content 31
3.13.2 Rule 2: Monitor Popularity And Content Generation Time 32
3.13.3 Rule 3: Document Size Matters 32
3.13.4 Rule 4: Plan Ahead on Invalidation and Expiration 32
3.13.5 Rule 5: Use Partial-page Caching 32
3.13.6 Rule 6: Compression Saves Bandwidth 33
3.13.7 Rule 7: Tune Configuration Parameter 33
3.13.8 Rule 8: Use Web Cache for Secure Socket Layer (SSL) Termination 33
3.14 HTTP Support for Caching and Replication Conditional Requests 34
3.14.1 Request Redirection 35
3.14.2 Range Request 35
3.14.3 The Cache-Control Header 35

CHAPTER 4 DESIGN AND IMPLEMENTATION OF THE SYSTEM 37
4.1 UPDATE PROPAGATION TECHNIQUES 37
4.2 Broadcasting with Catalog Binding Algorithm 39
4.3 The System Overview 41
4.4 Implementation of the System 43

CHAPTER 5 CONCLUSION, LIMITATIONS AND FURTHER EXTENSIONS 49
5.1 Benefit of the System 49
5.3 Limitations and Further Extensions 50

REFERENCES 51

LIST OF FIGURES

FIGURE		PAGE
Figure 2.1	Transaction A Performs an inconsistent analysis	7
Figure 2.2	Distributed Transaction Structure	10
Figure 2.3	Distributed DBMS Model Structure	11
Figure 3.1	A Simple View of the Web	15
Figure 3.2	HTTP Transfer Timing Costs with a New Connection	18
Figure 3.3	Cache in World Wide web	20
Figure 3.4	Caching Strategies	27
Figure 3.5	Fragment Granularity	29
Figure 4.1	The System Flow	41
Figure 4.2	Data Update Propagation Flow	42
Figure 4.3	Main Page	43
Figure 4.4	“SHOP” Menu of the System	43
Figure 4.5	Login/Registration Page	45
Figure 4.6	Catalog Page of the System	46
Figure 4.7	MYCART Page	47

CHAPTER 1

INTRODUCTION

Late development in Web-based business has prompted progressively dynamic examination in the plan of superior execution electronic trade sites. Today there is a developing interest in giving storing backing to E-trade applications on the Web as reserving is a straightforward and powerful method for giving quicker admittance to data on the Internet. Reserving alludes to putting away Web objects got to by clients at places through which the client's solicitation passes. Subsequently, assuming the solicitation is for an item that is put away in the store, the article is provided from the reserve as opposed to the server. A focal issue of storing is reserve consistency, which alludes to the system by which the server guarantees that the information stored at a distant client is in the know regarding the ongoing worth in the server. The main objective is to develop a cache consistency algorithm for Web-based electronic commerce applications, and to evaluate for Web-workloads. Specifically, this system focuses on developing a relaxed cache consistency scheme for E-commerce types of applications, as many of the Web applications are able to tolerate some degree of inconsistency.

1.1 Objectives of the Thesis

The objectives of thesis are as follows:

- To understand how consistency is important in database systems with multiple users.
- To implement a system in which all objects remain in a consistent state when they are accessed by multiple transactions.
- To illustrate a control algorithm, Broadcasting with Catalog Binding that uses catalog (client directory) instead of locking so that it can avoid deadlock.
- To control the network traffic and user latency.

1.2 Overview of the System

In the client-server system, the information duplicate from the server (Global data set) is put away in the client's data set (nearby/store data set) - reserving.

When the client's alteration exchange is effectively dedicated in neighborhood data set (Local Cache) and worldwide data set, information in the data set will be conflicting with the information on the opposite side of clients on the grounds that other client doesn't have the foggiest idea about the adjustment.

To stay away from the circumstance, after the adjustment is effectively dedicated the changed data should be criticism to one more client to keep the information consistency (Cache Consistency).

1.3 Organization of the Thesis

This thesis is organized in five chapters. They are as follows:

Chapter 1 introduction of concurrency control for data shipping, objectives of the thesis and thesis organization are described.

Chapter 2 presents the overview of the concurrency control method, transactions, typical problems between the concurrent execution of transactions, types of concurrency and deadlock.

Chapter 3 discusses the cache consistency problem and resolution the problems, using active data aware cache consistency by catalog binding.

Chapter 4 expresses the design and implementation of the proposed system. Finally,

Chapter 5 presents the conclusions of this thesis, and showing advantage, limitation and further extension.

CHAPTER 2

THEORICAL BACKGROUND

Concurrency control is the movement of organizing simultaneous gets to an information base in a multiuser data set administration framework (DBMS). Simultaneousness control grants clients to get to a data set in a multi-customized design while protecting the deception that every client is executing alone on a devoted framework.

2.1 Concurrency Control

From the past years, Distributed Databases stand out in the data set research local area. Information dissemination and replication offer open doors for further developing execution through equal inquiry execution and burden adjusting as well as expanding the accessibility of information. Truth be told, these open doors play had a significant impact in propelling the plan of the ongoing age of data set machines.

A distributed database system (DDBS) is an assortment of a few coherently related data sets which are genuinely circulated in various PCs (generally called destinations) over a PC organization. In data set administration frameworks, simultaneousness control is a significant issue for the synchronous execution of exchange over a circulated data set that can make a few information respectability and consistency issue. To work exchanges simultaneously on a circulated information base, a simultaneousness control calculation should be embraced to facilitate their exercises. In Optimistic Concurrency Control (OCC), exchanges are permitted to execute unhindered until they come to their commit point, when they are approved. OCC gives independence from gridlock. The execution of an exchange comprises of three stages, read stage, approval stage, and compose stage.

The vast majority of the circulated simultaneousness control calculations come into one of three essential classes: locking calculations, Timestamp calculations, and a hopeful (or accreditation) calculation. Locking calculation is locks which are characterized in the read and compose tasks of exchanges. Stop might happen in the locking calculations. Timestamp requesting partners timestamps with all as of late gotten to information things and expects that clashing information gets to by

exchanges be acted in timestamp request. The disseminated affirmation works by trading certificate data during the commit convention. This framework restarts the contention exchange to deal with and give messages total their work.

2.2 Types of Concurrency Control

Various concurrency control algorithms differ in the time when conflicts are detected, and in the way they are resolved conflict [8].

Concurrency Control methods can be classified into two categories as following:

- Pessimistic Concurrency Control
- Optimistic Concurrency Control

2.2.1 Pessimistic Concurrency Control

The main feature of the pessimistic approach is to forestall potential contentions. Exchange gain admittance to information provided that this won't cause conceivable clash circumstance later. On the off chance that it is beyond the realm of possibilities promptly the exchange ought to hold on until it will become conceivable. A large portion of cynical calculations depend on locks. The old style cynical calculation is the broadly utilized two stage locking (2PL). [5]

Skeptical Concurrency Control, stay away from any simultaneous executions of exchanges when clashes that could bring about future irregularities are distinguished. Pessimistic calculations synchronize simultaneous execution of exchanges right off the bat in their execution life cycle;

Approval (V) Read(R) Computation(C) Write (W)

In skeptical framework, activities can defer superfluously .likewise, critical frameworks can get into halt circumstances, where a gathering of activities can't process due to each activity in the gathering .In a negative framework with locking, it is important to gain the proper lock (by sending a lock request)before getting to an item .Thus, a roundtrip network delay is required in any event, while perusing a reserved article .This postponement is important to guarantee two things: the activity

should peruse (or change) a forward-thinking duplicate of the article and the locking rules should be kept up with. Skeptical Concurrency Control can be delegated follows:

- Two-Phase Locking(2PL)
- Timestamp Ordering

2.2 Optimistic Concurrency Control

Optimistic Concurrency Control permits simultaneous exchanges to process at the gamble of restarting them on the off chance that these thought irregularities appear. Hopeful calculations postpone the synchronization of exchange until their end:

Read(R) Computation(C) Validation (V) Write (W)

In hopeful frameworks, processes that bomb approval cut short and restart, re-trying work that wouldn't be revamped in a critical framework. Also, while hopeful frameworks can't stop. This framework presents the simultaneousness control utilizing OCC approach and the detail cycle is made sense of in section 3.

2.3 Typical Problems between the Concurrent Executions of Transactions

Concurrency means that the various clients approach the data set simultaneously. The errand of a simultaneousness control component is to guarantee the consistency of the data set while permitting a bunch of exchanges to execute simultaneously [1]. Issues happen in the simultaneousness in the accompanying areas.

2.3.1 Lost or Buried Updates

This problem occurs at the point when at least two exchanges are perused and refreshed on similar information thing at the offer data set. Every exchange knows nothing about different exchanges.

A subsequent exchange read a thing for update after the principal exchange has understood it, yet before the main exchange has committed .Whichever of the exchange commit first, that update will be lost.

2.3.2 Inconsistent Analysis (Non Repeatable Read)

A transaction, which reads the same data item more than once, should always read the same value.

Non repeatable read emerges when a subsequent exchange gets to similar the information thing a few times and peruses various information each time on the grounds that another exchange has been refreshed this thing while the subsequent exchange is perusing. Conflicting investigation includes various read (at least two) of a similar term is non-repeatable perused. Figure 2.1, which show the exchange A plays out a conflicting examination at time t8.

ACC1 **X := 40;**	**ACC2** **Y := 50;**	**ACC3** **Z := 30;**
Transaction A	**Time**	**Transaction B**
- **SUM := 0;** **read_item(x);** **SUM := SUM +x;** -	**t1**	- - - - -
read_item(Y); **SUM := SUM +Y;** -	**t2**	- - -
- -	**t3**	**read_item(z);** **z := z-10;**
- -	**t4**	**Write _item(z);** -
- -	**t5**	**read_item(x);** **x :=x+10;**

- -	**t6**	**Write _item(x);** -
- -	**t7**	**Commit** -
read_item(z); **SUM := SUM +z;**	**t8**	- -

Figure2.1: Transaction A performs an inconsistent analysis

Two exchange an and B procedure on account (ACC) records: exchange An is adding account balance, exchange B is moving a sum 10 from account 3 to account 1.

Exchange A has perused thing X before expansion of (10) by exchange B at time t1, and read thing Z after deduction of (10) by exchange B at time t8. All the outcome delivered by exchange An is clearly wrong; if exchange A were to proceed to compose that outcome back into the data set, it would really leave the data set in a conflicting state on the grounds that the read thing X by exchange An isn't repeatable and exchange B commits its updates before the exchange A has perused thing Z.

2.3.3 Uncommitted Dependency (Dirty Read)

A transaction, assuming it recovers or updates an information thing that has been refreshed by one more exchange however not yet dedicated by that other exchange. Filthy read resembles to conflicting investigation, the thing read by the one exchange was committed by the other exchange that rolled out the improvement.

2.3.4 Phantom Reads

A exchange re-executes a question, tracking down a bunch of information not equivalent to a past one-albeit the pursuit condition is unaltered.

Ghost peruses may cause when inset or erase activity is performed against a column that has a place with the scope of lines being by an exchange. For instance:

Assume exchange A recovers the arrangement of all columns that fulfill some condition (e.g.; all providers the condition that the city in Paris).Suppose that exchange B then, at that point, begin and embeds another line fulfilling that equivalent condition. If exchange A, presently rehashes its recovery demand, it will see a line that didn't beforehand exist.

2.4 Distributed Database

Distributed Databases stand out in the data set research local area. Information conveyance and replication offer open doors for further developing execution through equal question execution and burden adjusting as well as expanding the accessibility of information. As a matter of fact, these potential open doors play had a significant impact in rousing the plan of the ongoing age of data set machines.

A dispersed data set framework (DDBS) is an assortment of a few legitimately related data sets which are genuinely circulated in various PCs (generally called destinations) over a PC network [7]. All destinations taking part in the disseminated data set appreciate neighborhood independence as in the data set at each site has full command over itself as far as dealing with the information. Likewise, the destinations can between work at whatever point required. The client of a dispersed data set has the feeling that the entire data set is nearby with the exception of the conceivable correspondence postpones between the destinations. This is on the grounds that a dispersed data set is an intelligent association of the multitude of destinations and the circulation is stowed away from the client.

A dispersed information base administration framework (DDBMS) includes an assortment of destinations interconnected by an organization. Each site runs at least one of the accompanying programming modules: an exchange chief (TM), an information the executives (DM), and a simultaneousness control scheduler (or basically scheduler). In a client-server model, a site can work as a client, a server, or both. A client runs just the TM module, and a server runs just the DM and scheduler modules. Every server stores a part of the information base. Every information thing might be put away at any server or needlessly at a few servers.

2.5 The Structure of Distributed Transactions

Figure 2.2 shows a general circulated exchange as far as the cycles engaged with its execution. Every exchange has an expert interaction (M) that runs at its site of beginning. The expert cycle thus sets up an assortment of Cohort processes (Ci) to play out the genuine handling engaged with running the exchange. Since virtual all inquiry handling methodologies for dispersed information base frameworks include getting to information at the site(s) where it dwells, instead of getting to it from a distance.

There is no less than one such accomplice for each site where information is gotten to by the exchange. By and large, information might be recreated, in which case every accomplice that update any information things is expected to have at least one update (Uij)processes related with it at different locales. Specifically, a partner will have an update interaction at every remote site that stores a duplicate of the information things that it refreshes. It speaks with its update processes for simultaneousness control purposes, and it likewise sends them duplicates of the important updates during the principal period of the commit convention depicted underneath. The unified two-stage commit convention will be utilized related to every one of the simultaneousness control calculations inspected.

The convention functions as follows [5]: Fig 2.2: Distributed Transaction Structure When a companion wraps up executing its Portion of an inquiry, its sends an "execution complete" message to the expert. At the point when the expert has gotten such a message from every partner, it will start the commit convention by sending "plan to commit" messages to all destinations. Expecting that a partner wishes to commit, it sends a "ready" message back to the expert, and the expert will send "commit" messages to every companion in the wake of getting arranged messages from all companions.

The convention closes with the expert getting "committed" messages from every one of the companions. In the event that any partner can't commit, it will return a "can't commit" message rather than a "ready" message in the principal stage, making the expert send "cut off" rather than "carry out" messages in the second period of the convention. At the point when copy update processes are available, the commit

convention turns into a settled two-stage commit convention. Messages stream between the expert and the accomplices, and the companions thus connect with their updaters. That is, every companion sends "plan to commit" messages to its updaters in the wake of getting such a message from the expert, and it accumulates the reactions from its updaters prior to sending a "ready" message back to the expert; stage two of the convention is comparatively changed.

2.6 Distributed Transactions

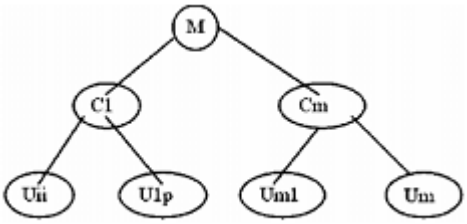

Figure 2.2 Distributed Transaction Structure

Distributed transactions deal with a physical division of transactions caused by the need to access distributed resources. Special distributed algorithms are needed to handle locking of data and committing of transactions.

2.7 MODELING A DISTRIBUTED DBMS

Figure 2.3 shows the general construction of the model. Each site in the model has four parts: a source, which creates exchanges and furthermore keeps up with exchange level execution data for the site, an exchange chief, which models the execution conduct of exchanges, a simultaneousness control director, which carries out the subtleties of a specific simultaneousness control calculation, and an asset supervisor, which models the CPU and I/O assets of the site. Notwithstanding these per-site parts, the model likewise has an organization supervisor, which models the way of behaving of the interchanges organization.

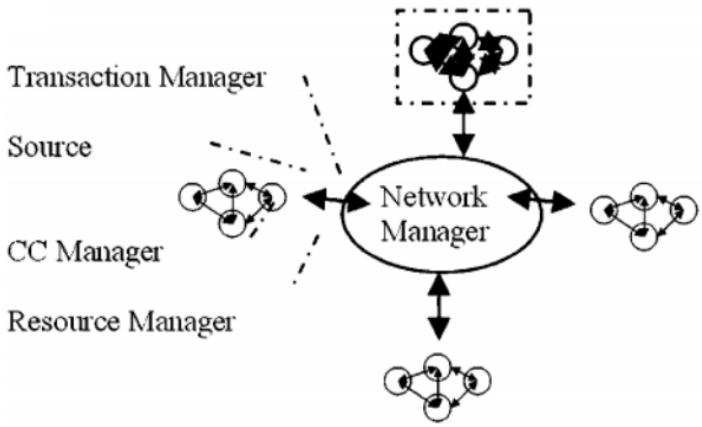

Fig 2.3: Distributed DBMS Model Structure

2.8 Distributed Concurrency Control Algorithm

This is four algorithm and they are

(1) Distributed Two-Phase Locking (2PL)

(2)Wound-Wait (WW)

(3)Basic Timestamp ordering (BTO)

(4)Distributed Certification (OPT)

2.8.1 Distributed Two-Phase Locking (2PL)

The first algorithm is the dispersed "read any, compose each of the" two-stage locking calculation portrayed in [15].Transactions set read locks on things that they read, and they convert their read locks to compose locks on things that should be refreshed. To peruse a thing, it gets the job done to set a read lock on any duplicate of the thing, so the nearby duplicate is locked; to refresh a thing, compose locks are expected on all duplicates. Compose locks are gotten as the exchange executes, with the exchange impeding on a compose demand until the duplicates of the thing to be all refreshed have been effectively locked. All locks are held until the exchange has effectively dedicated or cut off. Halt is plausible, Local stops are checked for any time an exchange impedes, and are settled when fundamental by restarting the exchange with the latest beginning startup time among those associated with the gridlock cycle.

(A companion is restarted by cutting short it locally and sending a "cut off" message to its lord, who thusly tells each of the cycles engaged with the exchange.) Global stop identification is taken care of by a "Sneak" process, which occasionally demands hangs tight for data from all destinations and afterward checks for and settle any worldwide gridlocks (involving similar casualty choice models concerning neighborhood halts). We don't relate the "Sneak" obligation with a specific site. All things being equal, each site takes a turn being the "Sneak" site and afterward hands this errand over to the following site. The "Sneak" obligation consequently pivots among the locales in a cooperative style, guaranteeing that nobody site will turn into a bottleneck because of worldwide gridlock discovery costs.

2.8.2 Wound-Wait (WW)

The second algorithm is the circulated wound-stand by locking calculation, again with the "read any, compose all" rule. It varies from 2PL in its treatment of the stop issue: Rather than keeping up with hangs tight for data and afterward checking for nearby and worldwide halts, gridlocks are forestalled through the utilization of timestamps. Every exchange is numbered by its underlying startup time, and more youthful exchanges are kept from making more seasoned ones pause. In the event that a more seasoned exchange demands a lock, and on the off chance that the solicitation would prompt the more established exchange hanging tight for a more youthful exchange, the An Approach for Concurrency Control in Distributed Database System more youthful exchange is "injured" - it is restarted except if it is now in the second period of its commit convention (in which case the "injury" isn't deadly, and is essentially disregarded). More youthful exchanges can sit tight for more established exchanges so the chance of stops is dispensed with.

2.8.3. Basic Timestamp Ordering (BTO)

The third algorithm is the basic timestamp ordering algorithm of [9, 14]. Like injury pause, it utilizes exchange startup timestamps, yet it utilizes them in an unexpected way. As opposed to utilizing a locking approach, BTO partners timestamps with all as of late gotten to information things and expects that clashing information gets to by exchanges be acted in timestamp request. Exchanges that endeavor to perform messed up gets to are restarted. At the point when a read demand is gotten for a thing, it is allowed if the timestamp of the requester surpasses the

things compose timestamp. When a compose demand is gotten, it is allowed on the off chance that the requester's timestamp surpasses the perused timestamp of the thing; if the timestamp of the requester is not exactly the compose timestamp of the thing, the update is essentially disregarded (by the Thomas compose rule [14]).For duplicated information, the "read any, compose all "approach is utilized, so a read solicitation might be shipped off any duplicate while a compose demand should be shipped off (and supported by) all duplicates. Reconciliation of the calculation with two stage commit is achieved as follows: Writers keep their updates in a hidden work area until commit time.

2.8.4 Distributed Certification (OPT)

The fourth algorithm is the appropriated, timestamp-based, hopeful simultaneousness control calculation from [13], which works by trading affirmation data during the commit convention. For every information thing, a read timestamp and a compose timestamp are kept up with. Exchanges might peruse and refresh information things uninhibitedly, putting away any updates into a nearby work area until commit time. For each read, the exchange should recall the variant identifier (i.e., compose timestamp) related with the thing when it was perused. Then, when the exchange's all's companions have finished their work, and have detailed back to the expert, the exchange is relegated a worldwide remarkable timestamp. All this timestamp is shipped off every partner in the "plan to commit" message, and it is utilized to locally guarantee its peruses and composes as follows: A read demand is ensured in the event that (I) the rendition that was perused is as yet the ongoing form of the thing, and (ii) no compose with a more up to date timestamp has proactively been privately confirmed. A compose demand is guaranteed on the off chance that (I) no later peruses have been ensured and in this way dedicated, and (ii) no later peruses have been privately confirmed as of now updaters. As portrayed before, the expert lives at the site where the exchange was submitted. Every partner makes a succession of perused and composes solicitations to at least one record that are put away at its site; an exchange has one companion at each site where it needs to get to information. Associates speak with their updaters when remote compose access authorization is required for repeated information, and the updaters then make the required compose demands for neighborhood duplicates of the information in the interest of their

partners. An exchange can execute in either a successive or equal design, contingent upon the execution example of the exchange class.

2.9 Related Work

The possibility of ADCC started with the memory coherency convention utilized in SGI Origin multiprocessor frameworks [5]. ADCC is recognized from Origin memory coherency convention in a few significant viewpoints. ADCC is programming based and utilizes a two-level catalog, while the convention in Origin frameworks is equipment based and utilizes a solitary level registry in particular. Another significant contrast is that the coherency convention in multiprocessor frameworks keeps a predictable perspective on memory for each processor on every memory activity. ADCC has a coarser granularity of atomicity, which requires a grouping of tasks to be executed overall. In this manner, ADCC should deal with stops and cuts short at the exchange (a succession of tasks) level.

ADCC, with the calculations proposed for information transporting DBMS during the last ten years. The calculations can be grouped into two classifications as per their approach for invalid access counteraction: evasion based and discovery based [4]. The calculations in general, with the exception of ADCC, totally depend on a unified server for simultaneousness control. CBL is broadly acknowledged as the main calculation because of its great presentation and low cut short rate [3]. As a general rule, it has preferable execution over Caching Two-Phase Locking (C2PL) [7], No-Wait Locking (NWL) [7], Cache Locks [8] and Notify Locks [8]. Hopeful Two-Phase Locking (O2PL) [2] and Adaptive Optimistic Concurrency Control (AOCC) [1] have comparable or higher throughput. Be that as it may, the significant downside of these two hopeful methodologies is the conceded consistency check, which prompts high cut short rates. The cut short rate is a basic issue for clients in the profoundly intuitive conditions that are normal for page servers. Offbeat Avoidance-based Cache Consistency (AACC) [6] can bring down the cut short rate while keeping up with high throughput. In any case, both AOCC and AACC were proposed for versatile locking, which switches locking between the page and the article level.

CHAPTER 3

Cache Consistency Controlling

Cache consistency describes the validity of data in the network. With the help of cache consistency, it is ensured that valid data is received in response to query generated by the sink and all stale data is immediately evicted from each cache in the network.

3.1. Web Caching

Web resource caching, one innovation is utilized to make the Web versatile. Web storing can diminish transfer speed use, decline client saw latencies, and decrease Web server stacks straightforwardly. Thus, reserving has turned into a huge piece of the Web's foundation. Reserving has even produced another industry: content conveyance organizations, which are likewise developing at a phenomenal rate. Reserving clarifies how it applies for the Web, and depicts when and why it is valuable.

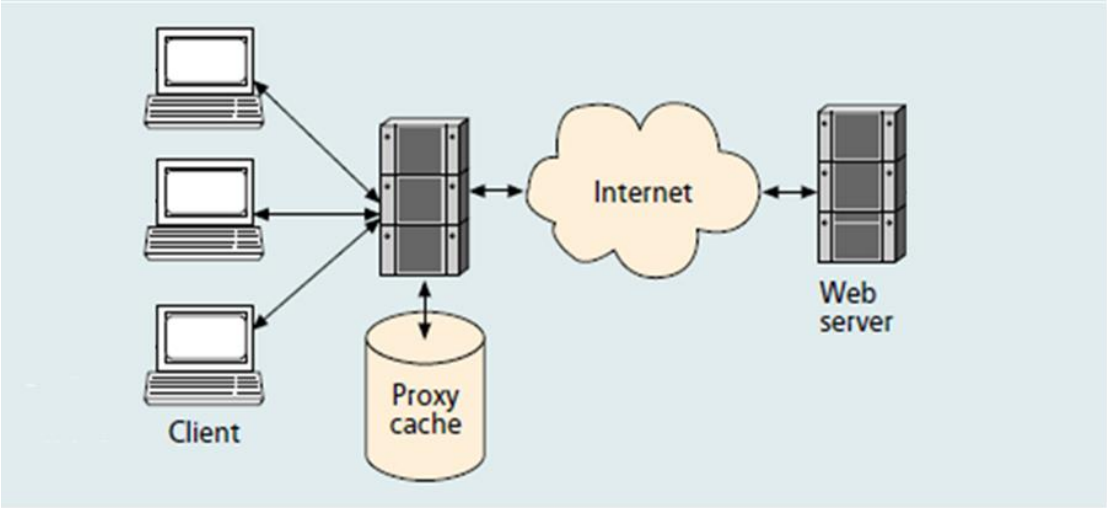

Figure 3.1 A Simple View of the Web

3.2 Caching Scheme

The concept of caching is simple. In the event that the information is expected to fulfill the got to demands are not currently stored, then a duplicate of those information is brought from the server to the client framework. Gets to are performed on the stored duplicate. The thought is to hold as of late gotten to circle blocks in the

store, so that rehashed gets to a similar data can be taken care of locally, without extra organization traffic [6].

A forward-thinking duplicate of the information should be stored. There are two ways to deal with check the legitimacy of stored information.

3.2.1 Client-Initiated Approach

The client initiates a legitimacy check in which it contacts the server and checks whether the neighborhood information are reliable with the expert duplicate. The recurrence of the legitimacy check is the essence of this methodology and decides the subsequent consistency semantics. It can go from a really take a look at before each admittance to a beware of just first admittance to a document. Each entrance combined with a legitimacy check is postponed, contrasted and an entrance served promptly by the store. On the other hand, a check can be started each proper time period. Contingent upon its recurrence, the legitimacy check can stack both the organization and the server.

3.2.2 Server-Initiated Approach

The server records the records for every client into stores. At the point when the server identifies a likely irregularity, it should respond. A potential for irregularity happens when two distinct clients are stored a document while they are in the clashing modes. Assuming UNIX semantics is executed, it tends to be addressed the expected irregularity by having the server assumes a functioning part. The server should be told at whatever point a document is opened, and the planned mode (read or compose mode) should be shown for each open. Expecting such notice, the server can be act when it distinguishes a document that is opened all the while in clashing modes by handicapping reserving for that specific record. Actually, disabling mode by results changing to a remote-administration method of activity [6].

3.3 Caching in Memory

Memory architectures use stores to further develop PC execution. Since focal handling units work at exceptionally high paces while memory frameworks work at a slower rate, CPU planners give at least one degrees of store, a limited quantity of memory that works at, or near, the speed of the CPU. At the point when the CPU

finds the data it needs in the store, a hit doesn't need to dial back. At the point when it neglects to track down the mentioned object in the reserve (a store miss), it should bring the item straightforwardly and cause the related presentation cost.

Ordinarily, when a store miss happens, the CPU puts the got object in the reserve, expecting transient region. An as of late mentioned object is more probable than others to be mentioned from now on. Memory frameworks likewise ordinarily recover numerous successive memory locations and spot them in the store in a solitary activity, expecting spatial territory those close by objects are bound to be mentioned during a specific time frame [7].

Eventually the store will turn out to be full and the framework will utilize a substitution calculation to account for new items, for instance, earliest in, earliest out (FIFO), least as of late utilized (LRU), or least regularly used (LFU). The objective is to upgrade reserve execution, for instance, to expand the probability of a store hit for run of the mill memory structures.

3.4 Mechanism of a Web Request

Web is a set of servers and clients (like Web programs, or some other programming used to make a solicitation of a Web server) in its most straightforward structure. To recover a specific Web asset, the client endeavors to convey over the Internet to the beginning Web server. To associate the server, the client needs the host's mathematical identifier. It inquiries the space name framework (DNS) to interpret the hostname (for instance, www.webcaching.com) to its separate Internet Protocol (IP) address (209.182.1.122), with which can lay out an association with the server and solicitation the substance. When the Web server has gotten and analyzed the client's solicitation, it can create and communicate the reaction. As Figure 2.2 shows, each move toward this cycle requires some investment. The hypertext move convention (HTTP) determines the communication among Web clients, servers, and go-betweens. Solicitations and reactions are encoded as headers that go before discretionary bodies containing content upheld and a reaction code with standard qualities. How much chance to recover an asset when another association is required

can be approximated by two full circle times in addition to an opportunity to communicate the reaction [1].

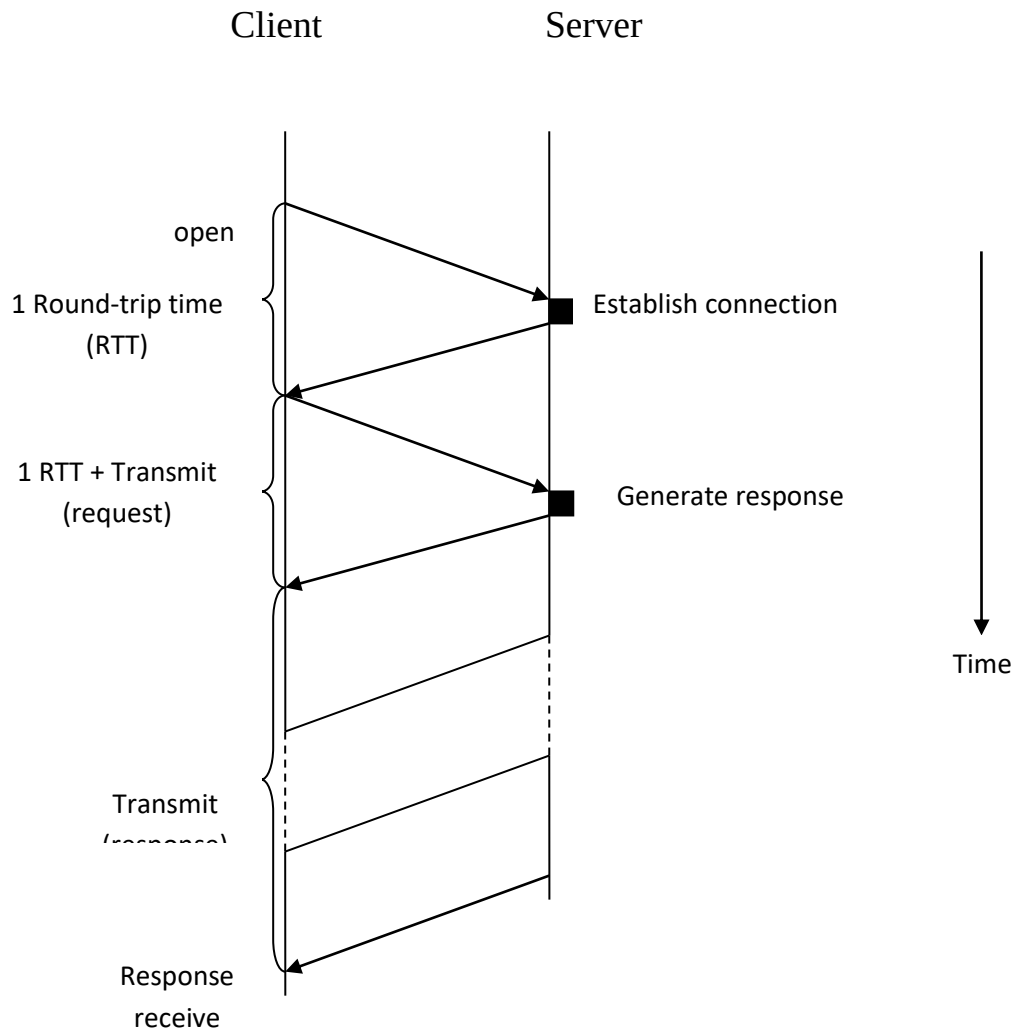

Figure 3.2 HTTP Transfer Timing Costs with a New Connection

3.5 Caching Web Resources

A Web reserve stores Web assets fully expecting future solicitations. Web reserving result relies upon the non-consistency of Web object sizes, recovery expenses, and store capacity. To address object size, store administrators and

fashioners track both the general item hit rate and the general byte hit rate. Customary substitution calculations frequently expect a proper item size, so factor sizes can influence their exhibition. Recovery cost changes with object sizes, distance voyaged, network blockage, and server load. At last, some Web assets can't be stored on the grounds that the asset is customized to a specific client or is continually refreshed.

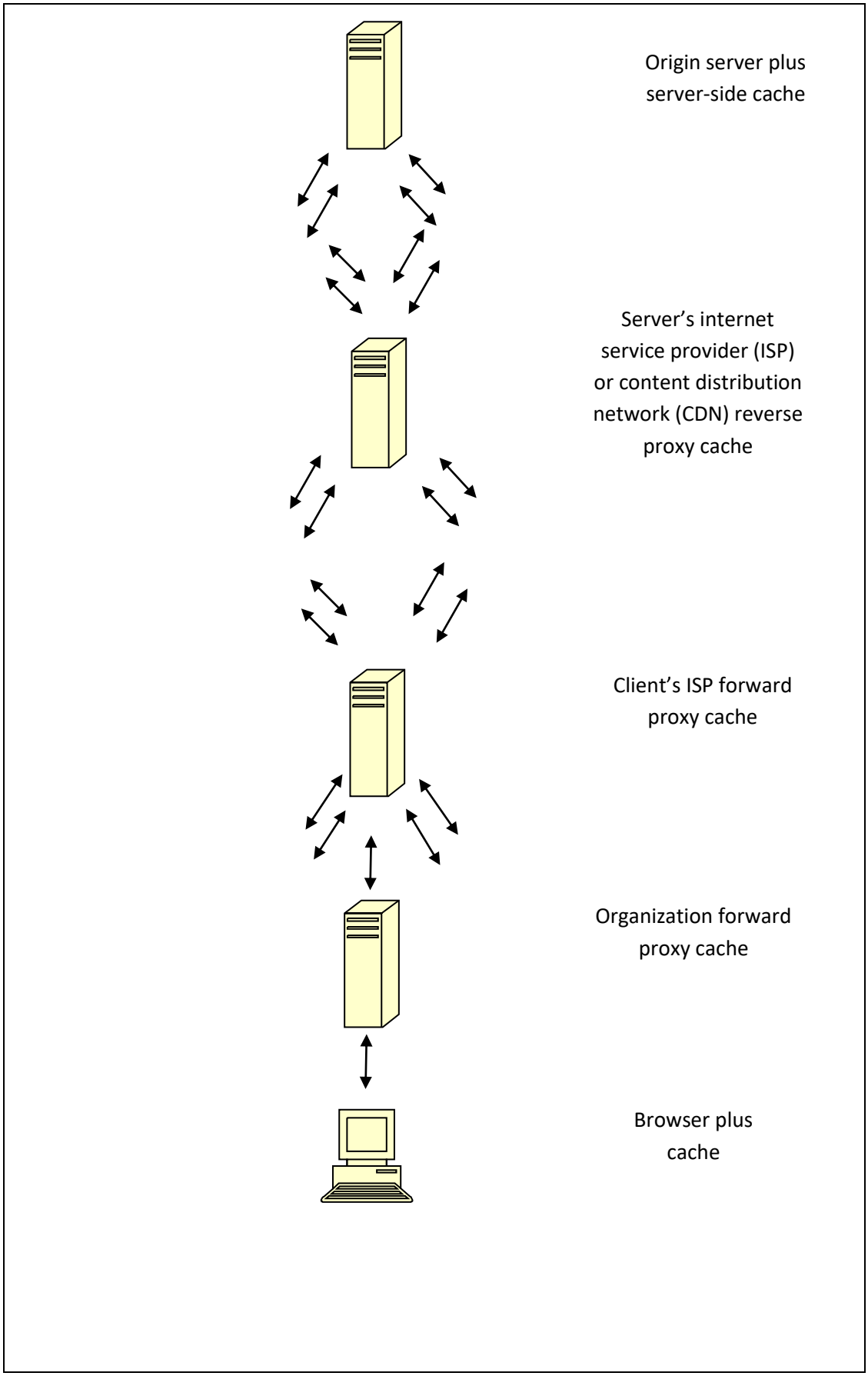
Origin server plus
server-side cache
Server's internet
service provider (ISP)
or content distribution
network (CDN) reverse
proxy cache
Client's ISP forward
proxy cache
Organization forward
proxy cache
Browser plus
cache

Figure 3.3 Caches in World Wide Web

The request might be passed to the association wide intermediary store on the off chance that a reaction caught in a program reserve doesn't fulfill a client. In the event that a substantial reaction is absent there, an intermediary store will be worked by the client's network access supplier (ISP) could get the solicitation. Assuming the web access supplier (ISP) reserve doesn't contain the mentioned reaction, it will probably endeavor to contact the beginning server. Nonetheless, the opposite intermediary stores are worked by the substance supplier's ISP or content circulation organization (CDN) may answer the solicitation. In the event that they don't have the mentioned data, the solicitation could at last show up at the beginning server. Indeed, even at the beginning server, items or bits of content can be put away in a server-side reserve to decrease the server load. The reaction moves through the converse way back to the client. Each move toward Figure 2.3 has numerous bolts, meaning associations with different elements at each level. For instance, the converse intermediary can act as an intermediary store for the substance from different beginning servers and can get demands from various downstream clients.

In a progressive reserving structure, each store serves numerous clients, which can be clients or different stores. At the point when a nearby reserve can't serve a solicitation, it passes the solicitation to a more elevated level in the ordered progression. On the off chance that the solicitation misses at a root store (which has no parent), the reserve demands the item from the beginning server.

3.5.1 Content Cacheability

Every Web resource is not cacheable. Of those, some can be reserved for extensive stretches by any store, while others have limitations, for example, reserving for brief periods or to specific sorts of stores (for example, non-intermediary reserves). This adaptability augments the chance for reserving individual asset. The cacheability of a site influences the two its client seen execution and the versatility of a specific facilitating arrangement. Rather than requiring seconds or minutes to stack, a stored item can show up promptly. Notwithstanding how much the facilitating cost, a store well-disposed plan will permit a server to serve more pages before it requirements to move up to a more costly arrangement. Luckily, the substance

supplier decides its assets' cacheability. The Web server programming sets and sends the HTTP headers to decide cacheability, as indicated by the server's reserving strategy for that information. To boost a site's cacheability, all static substance (buttons, designs, sound and video records, and pages that seldom change) are regularly given termination dates far in the future so they can be reserved for weeks or months information time.

3.5.2 Cache Latency

Caching can provide only a restricted advantage (object hit rates commonly arrive at 40% to 50 percent with adequate traffic), as a reserve can give protests that have been recently mentioned. In the event that future solicitations can be expected, articles can be gotten ahead of time. When accessible in a neighborhood store, those items can be recovered with negligible deferral, upgrading the client experience. In spite of the fact that web storing shows guarantee, it is hard to assess and has not been generally carried out in business frameworks. A few programs and workgroup intermediary reserves will store the connections of the ongoing page, or occasionally prefetch the pages in a client's bookmarks. One critical trouble is precisely foreseeing which assets will be required the close to limit botches (i.e, the outcome in squandered data transmission and expanded server loads [3].

3.6 Maintaining Consistency of Client-Cached Data

In a client-server database environment, the server gives shared information base admittance to numerous client workstations and that client workstations might reserve a part of the data set. This framework expects to keep up with the consistency of the client store. The calculation is a straightforward expansion two-stage locking and comprises of three extra lock modes that should be upheld by the server lock.

The application program runs as a client interaction and speaks with the information base server through messages. This builds the expense of every information demand. One arrangement is to decrease the quantity of solicitations by storing a part of the information base on the client. At the point when a client reserve

is utilized, there should be a convention between the client and server to guarantee that the client store stays steady with the common data set [5].

In this sense, the client store might be seen as a functioning information since it's updates ought to set off a reserve revive activity. The dynamic data set permits applications to be educated regarding changes to some piece of a common information base by different exchanges. Practically speaking, this has implied that all updates to the data set should be observed by the information base administration framework to decide whether the updates influence the dynamic information. At the point when the dynamic information is refreshed, the data set framework will be educated the impacted clients that a change has happened.

Hence, all exchanges cause extra above to help a help that they might in all likelihood never use i.e., identification and notice of updates. The test is to make this above as low as conceivable since this makes an interpretation of straightforwardly into lower reaction time and higher throughput. This is particularly significant when the data set is on a focal server.

To keep up with store consistency, the reserve consistency calculation with the lock director of the information base administration framework can be incorporated.

3.7 Cache Consistency for the Internet

The value of the cache is significantly decreased in the event that reserved duplicates are not refreshed when the first information change. Reserve consistency instruments guarantee that stored duplicates of the information are at last refreshed to keep consistency with the first information. An ideal reserve consistency arrangement will authorize the consistency to the greatest degree, while lessening the organization data transmission utilization and server load.

There are fundamentally two classes of store consistency draws near: frail reserve consistency approach major areas of strength for and consistency approach gave a definition based on these two conditions. Powerless reserve consistency is the model where client reaction time is more stressed, however a lifeless report may be

gotten back to the client, solid store consistency, then again, implements the newness of record constantly, and yet has the cost of additional server asset utilization.

3.7.1 Weak Cache Consistency (client-initiated approach)

Existing Web Caches generally give powerless consistency, and that implies it is feasible for the client to get an old record from the reserve, in light of the fact that right now, the items on the web server have changed yet the store has not done the synchronization yet. Two systems fall into this classification: Time-To-Live (TTL) and Client Polling. Their component in like manner is that the client store, which starts the consistency cycle, sends approval messages to server.

Time-To-Live (TTL)

Under this approach, each article (report, picture, record and so forth) is doled out to TTL values, like two hours or at some point. This worth is a gauge of the article's lifetime, after which it should change. At the point when the TTL terminates, the information is viewed as invalid, and the following solicitation for the article will make the item be mentioned from the first server. A slight improvement to this essential instrument is that when a solicitation for a lapsed item is shipped off the reserve, rather than mentioning record move from the server, the store initially sends an "if-changed since" control message to the server to check whether a document or information move is fundamental.

Client Polling

Client polling is one more methodology of powerless reserve consistency. It implies the client (Cache) occasionally seeks out the server to decide whether reserved objects are as yet substantial. It is to some degree like the versatile TTL, in light of the fact that the client conveys approval message to check assuming the archive is as yet legitimate under the two cases, when the client begins to figure the report may be stalled. Alex FTP store utilizes an update limit to decide how successive to survey the server. The update edge is communicated as a level of the item's age.

An article is nullified when the time since last approval surpasses the update edge times the item's age. For instance, consider a stored document whose age is 30 days and whose legitimacy was checked one day prior. In the event that the update limit is set to 10%, the article ought to be set apart as invalid following 3 days (10% *30 days). i.e., 3 days is the modest amount of a month.

Since the article was checked yesterday, demands that happen during the following two days will be fulfilled locally, and there will be no correspondence with the server including control message. After the two days have passed, the record will be checked invalid, and the following solicitation for the document will make the reserve recover another duplicate of the record from the server. Like TTL, the stunt this is the way to choose the update limit.

3.7.2 Strong Cache Consistency (Server-Initiated approach)

Weak cache consistency techniques save network traffic and client dormancy to the detriment of returning lifeless records to the clients. Under circumstances where archive adjustment doesn't occur much of the time, or client doesn't have severe necessity on the newness of the record, feeble reserve consistency is a financial methodology. Notwithstanding, in the event that the legitimacy of the information is significant, for example, stock statement, a solid store consistency must be implemented. The two generally acknowledged techniques for solid store consistency are negation and surveying without fail [5].

Invalidation

Many circulated document frameworks depend on this strategy to guarantee that reserved duplicates never become old. In the web climate, this basic strategy likewise applies in light of the fact that clashing updates never occur (assuming there are clashing updates, a more muddled value-based consistency calculation is required). Under this component, the web server assumes a pivotal part. It is liable for monitoring the stored information.

The server carries out this by keeping all reserve tends to that have the duplicate of the information. When the information is altered on the server, the server

conveys message to advise the reserves on the rundown that their information are presently not legitimate. This warning system is viewed as complete solely after every one of the clients on the rundown has gotten the message. Negation ensures that when the client demands a report, the returned record is cutting-edge. The tradeoff is the above at server side.

Polling-every-time

This method is an outrageous instance of client surveying that is utilized in frail store consistency. Under this methodology, at whatever point the client reserve gets a report demand from the end client and there is a duplicate in the store, it will initially contact the web server to approve the reserved duplicate. In the event that it is new, the duplicate will be gotten back to the client, generally another duplicate will be shipped off the store and supplant the former one. This approach likewise includes a ton of message move, conceivably a huge piece of which is pointless. Yet, given a short lifetime of the items and successive solicitations from the client, this technique is supposed to be proficient [5].

3.8 Caching Strategies

A proper caching strategy includes effective use of both write-through and lazy loading of your data and setting an appropriate expiration for the data to keep it relevant and lean.

3.8.1 Static Caching

Content changes rarely in static storing. (e.g, abnormal landing page). And afterward it is helpful to republish the site at whatever point its substance changes. It is generally protected to empower program and intermediary server storing. Static reserving permits programs and intermediary servers storing for static items in HTMLs. The presentation can't be fundamentally improved by utilizing static reserving, on the off chance that the substance is steady over an enormous number of solicitations.

3.8.2 Dynamic Caching

No content is completely static because everything changes eventually (e.g., products in e-commerce, white pages). Figure 2.4 illustrates how content in an HTML page should be treated differently depending on how often it changes:

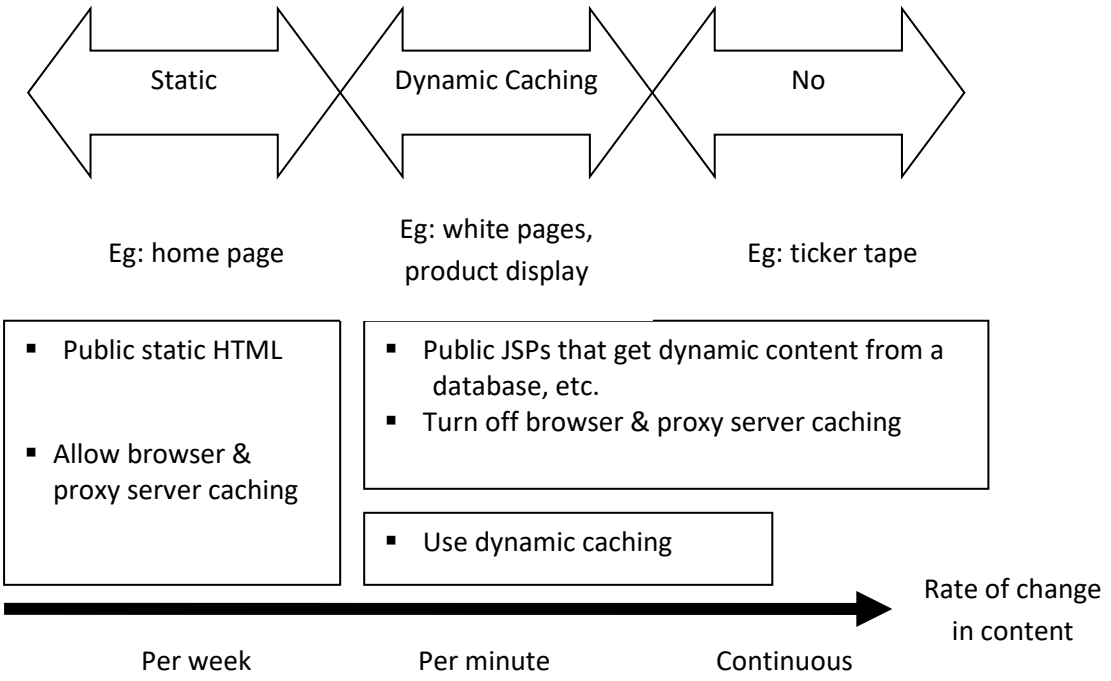

Figure 3.4 Caching Strategies

Content frequently changes that it is irrational to republish the site each time it changes, then JSPs ought to be utilized that powerfully get the substance from a record or information base, and afterward render (i.e., design) it into HTML.

With dynamic storing, either time limits or an occasion driven in approval system can be utilized to keep the substance in the reserve forward-thinking. In the event that the substance changes constantly (e.g., a paper feed), any type of storing is a poorly conceived notion. JSPs ought to be utilized with practically no storing. One method for review storing is that it robotizes the distributing system. For refreshes that are excessively continuous for manual republishing each time a few basic unique

substance changes, JSP/servlets with dynamic substance can be utilized alongside reserving to republish steadily and naturally [3].

3.9 Caching in HTML

Caching rendered HTML offers the following performance improvements when the underlying dynamic content has not changed.

Avoiding access to backend servers (e.g.,database, transaction monitor, internal application, news service) and to get the dynamic content.

Avoiding the delivering the unique substance into HTML. Reserving the fundamental unique substance (i.e., information) rather than HTML requires delivering the information into HTML during the quick way. In any case, the quick way happens more regularly. At the point when similar information is delivered in more ways than one, reserving delivered HTML requires getting to the backend server once for each delivering. The tradeoff working framework will be hard to make for delivering on the grounds that it is between continuously staying away from a regularly more affordable movement (i.e.,rerendering) and rarely keeping away from a normally more costly action (i.e., backend access) [3].

3.10 Caching Granularity

Caching rendered HTML with dynamic substance requires adaptability in the granularity of the reserve. A piece is a section or a delivered HTML page which can be all stored. A piece can contain at least 0 kid sections and can be contained by at least 0 parent parts.

-The href for a picture that shows what the item resembles. The basic data set record for the item contains this URL.

-A designed table that incorporates the definite portrayal of the item (e.g., item request number, name, choices, and cost).

-A customized welcoming (e.g., "Hi, John! Welcome to AcmeCorp.").

-A designed shopping basket, including the request number, name, amount and cost of the items that have been decided for conceivable buy.

-A href for a picture that shows a notice. The notice href is different each time a page is shipped off a customer. This makes the general page too unpredictable to even think about storing. Nonetheless, section granularity actually permits the remainder of the page to be reserved. The href to the item picture and the point by point item depiction are fantastic possibility for sections to be reserved, on the grounds that the basic information portraying a specific item changes inconsistently. Nonetheless, the hidden information depicting some item changes extremely oftentimes for static distributing. The customized welcoming has the lifetime of a client meeting yet just for a specific customer. It very well might be utilized a few times inside a genuinely brief time frame span, so it is a decent contender for dynamic storing. The shopping basket changes on different occasions inside a client meeting (each time something is added or the amount changes), so it isn't as great a contender for dynamic storing as the customized welcoming. Nonetheless, in the event that it is remembered for each page got back to the customer, it is normally returned a few times between changes, so there is a sensible case for reserving it. The notice href is an exceptionally unfortunate possibility for reserving in light of the fact that the hit proportion would be zero and reserving has its own above (i.e., putting away it in the store and discrediting it).

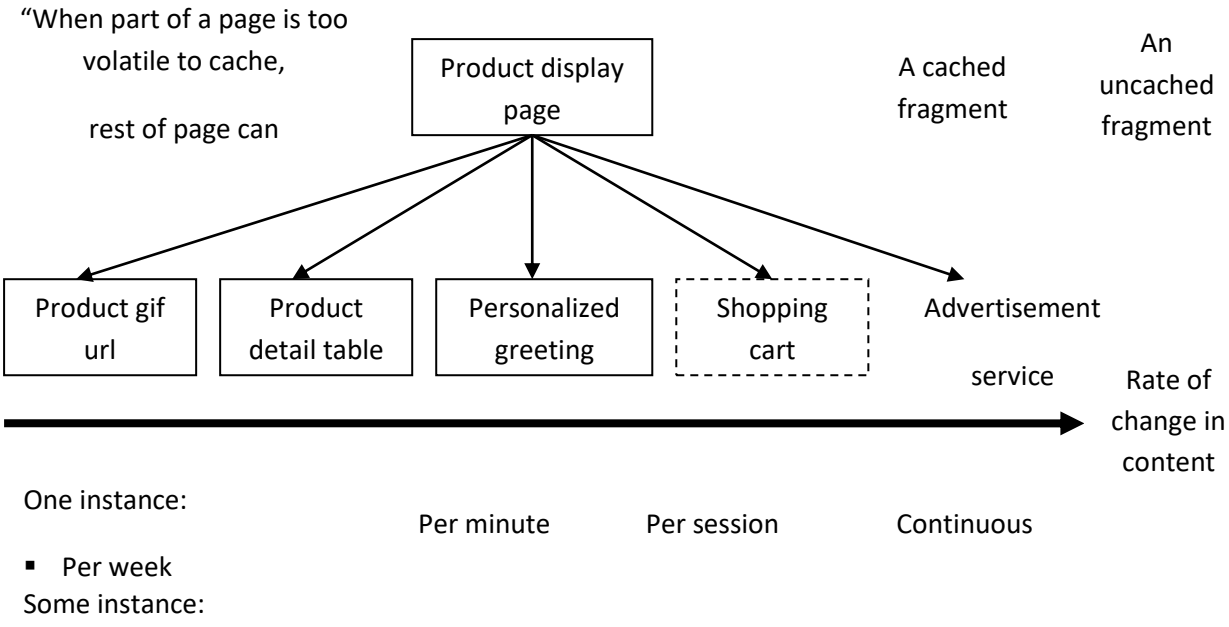

Figure 2.5 Fragment Granularity

3.11 Caching Effectiveness

Caching effectiveness is generally estimated by two amounts: the (record) hit proportion is the level of the complete solicitations that are fulfilled straight by archives put away in the reserve; and the byte hit proportion is the level of the absolute mentioned Web content bytes that are fulfilled straight by reports put away in the store. The two measurements are expected since Web objects fluctuate fundamentally in size. Different measurements, for example, client saw reaction time are reliant upon the hit proportion and the byte hit proportion, as well as organization data transmission, full circle deferral, and server load. To improve the presentation of Web reserving, staggered Web storing pecking orders have as of late gotten expanding research consideration. In a progressive design, intermediaries at or close to the end-client comprise the most reduced level of the order, frequently with kin associations with each other. The most reduced level intermediaries might have a kid parent relationship to a more significant level intermediary, generally a (geologically) provincial intermediary. A local intermediary can interface with a more significant level intermediary, for example, a public intermediary [1]. A solicitation that can't be fulfilled by one intermediary reserve can be shipped off a close by kin or to the parent

utilizing an Inter-Cache Protocol. Reaching the beginning server to acquire the archive servers if all else fails.

3.12 Traditional Web Caching Protocols

Polling-every-time gives solid store consistency, similar to the wide range of various conventions. The intermediary generally sends an "If-Modified-Since" solicitation to the server prior to returning any stored duplicate to clients. The server answers with either a changed duplicate or "Not Modified". The last option case is known as a sluggish hit on the grounds that the reserved duplicate is gotten back to the client after a full circle to the server. On the other hand, in a negation based convention, all hits are quick hits in light of the fact that the reserved duplicate is promptly gotten back to the client.

Versatile TTL [3] gives powerless reserve consistency and depends on the perception that "more seasoned" records are less inclined to be changed. The intermediary sets the TTL of a reserved duplicate to L times the "age" of the item (i.e., from its last change to now). Before the TTL lapses, client demands are served straightforwardly from the store. They are quick hits yet might be lifeless. Upon the primary client demand after the TTL terminates, the intermediary sends an "If-Modified-Since" solicitation to the server. The outcome might be a changed duplicate or a sluggish hit. Then, at that point, the TTL of the stored duplicate is changed as needs be.

3.13 Simple Caching Rules

Rule 1: Cache All Static Content

Rule 2: Monitor Popularity and Content Generation Time

Rule 3: Document Size Matters

Rule 4: Plan Ahead on Invalidation and Expiration

Rule 5: Use Partial-Page Caching

Rule 6: Compression Saves Bandwidth

Rule 7: Tune Configuration Parameters

Rule 8: Use Web Cache for Secure Socket Layer (SSL)

Termination

3.13.1 Rule 1: Cache all Static Content

Static content that changes rarely is the most appropriate for reserving. Web Cache requires no application changes to store content in arrangements, for example, jpeg, gif, pdf, css, wav or JavaScript. This is helpful storing diverts. Web Cache supplies a bunch of default cacheability rules to reserve the fundamental static substance. A cacheability rule of a specific URL articulation might be applied on objects finishing with a record expansion, matching a way prefix, or matching standard articulation punctuation [2].

3.13.2 Rule 2: Monitor Popularity And Content Generation Time

Popularity and Content Generation Time are vital elements in deciding if an article ought to be stored. The Popular Requests page shows both stored and non-reserved demands in the request for their fame. A mentioned item's notoriety depends on the quantity of late demands for an item. The Performance page incorporates a breakdown of client demands, classified by store hit (objects served straightforwardly from the reserve) or reserve miss (protests that needs to demand from the beginning server). The higher the reserve hits rate, the better the framework advances. For store misses, a high level of non-cacheable miss shows that there might be other conceivable cacheable items.

3.13.3 Rule 3: Document Size Matters

While large objects may be costly to create, the vital models for the decision about whether to reserve them is the articles' prevalence. At the same time, when the reserve moves toward its most extreme store size, trash assortment eliminates less well known or old items. Putting away enormous items requests more memory and

subsequently builds the likelihood of trash assortment during which any articles can't be placed into the reserve.

3.13.4 Rule 4: Plan Ahead on Invalidation and Expiration

Invalidation and expiration are two vital components for checking stored objects as not useable. Utilizing them cautiously and admirably upgrades the proficiency and eventually the general execution of the Web application. Termination is the greatest time that Cache is permitted to serve a given item. Lapse is valuable for objects whose changes are unsurprising or for objects that change often. The nullification message might be produced automatically by an application or physically through the managerial control center.

3.13.5 Rule 5: Use Partial-page Caching

Web pages with sections of the page that get refreshed at an alternate rate from different segments, while not alluring for full page reserving, are great possibility for incomplete page storing. Since the page sections are expected for reuse, the most effective way to move toward fractional page storing is through a standard improvement technique. The piece can be set to terminate or be negated automatically.

3.13.6 Rule 6: Compression Saves Bandwidth

Compression is especially helpful in conditions though network data transmission is restricted. Most programs today support gzip encoding. Reserve stores the archive in compacted structure to save memory. Store serves a compacted report streamed (meaning it begins to convey the record as packs the archive) from the end client's point of view, the substance shows up on the program prior as opposed to sitting tight for the appearance of the whole report.

3.13.7 Rule 7: Tune Configuration Parameter

The effectiveness of Web caching is reliant upon the equipment on which it runs the organization and the working framework. Network engineering likewise assumes a part in the store's exhibition and accessibility [2].

3.13.8 Rule 8: Use Web Cache for Secure Socket Layer (SSL) Termination

SSL acceleration is offloaded by the SSL encryption/decryption processing from the application server tier to the Web caching tier. Cache supports the following SSL related capabilities:

-Multi-site hosting

-SSL client certificate support

-SSL hardware accelerator support

-SSL encryption to each tier [2]

3.14 HTTP Support for Caching and Replication

Hyper Text Caching Protocol is a trial convention which is utilized to find HTTP stores and reserved information, overseeing sets of HTTP reserves and observing reserve movement. It very well may be utilized to screen remote stores increments and cancellations and sending hints about web articles like the outsider areas of cacheable articles or the deliberate un-cacheability or inaccessibility of an item. HTCP incorporates HTTP headers while ICP doesn't. The headers have indispensable data which can be utilized by the intermediary reserves.

Conditional Requests

An HTTP feature vital for storing and replication upholds for restrictive solicitations. These solicitations determine specific circumstances in their headers. A server executes a contingent solicitation and answers with the message body provided that the condition is valid. In any case, the server basically answers with a unique

status code, by the same token "304 Not Modified" or "412 Precondition Failed" which demonstrate that the condition doesn't hold. The headers that indicate the circumstances are called contingent headers. Clients plan restrictive solicitations utilizing keep up with arrangement about the articles.

Conditional Headers Used for Caching

The most important conditional headers utilized with regards to reserving. The if-changed since header contains the last-altered date of the stored object. At the point when a server gets a solicitation with this header (regularly a GET demand), the server possibly returns the item in the event that the last-changed date of the item at the server is not quite the same as the date contained in the if-adjusted since demand header. In any case, the server returns a "304 Not Modified" status code. Clients utilize these headers to check whether their stored items' duplicates are as yet legitimate and if not, to download the ongoing articles in a single server access.

Conditional Headers Used for Replication

Conditional requests can also be utilized in replication assuming that servers utilize HTTP to spread new item forms to one another. A server can utilize POST to send object updates to one more server with an item imitation, PUT to send the new rendition of the item completely and DELETE to erase object reproductions when the actual item has been eliminated.

13.14.1 Request Redirection

Other HTTP features which are significant for reserving and replication are its components for diverting a solicitation starting with one server then onto the next. A server might divert a solicitation to an intermediary reserve or a mirror server, or it might tell the client of the arrangement of intermediaries and mirrors that can support the solicitation. HTTP servers execute demand redirection by returning a reaction with exceptional redirection status code.

13.14.2 Range Request

When an object downloaded has been interfered with, the client may as of now have gotten an enormous piece of the item. The client could then utilize a reach demand header to download just the lacking pieces. The reach header indicates the byte scope of element being mentioned. On a fundamental level, the solicitation might contain a few reach headers that determine numerous disjoint segments of the element. Demands with range headers are called range demands.

3.14.3 The Cache-control Header

The cache-control header can contain numerous orders that control the utilization of the relative multitude of reserves that are arranged between the client that initially given the solicitation and the beginning server. The worth of the store control header is a rundown of mandates, and every order comprises of the catchphrase distinguishing the order and alternatively, the order esteem.

The system utilized for expulsion choices is frequently alluded to as a substitution strategy. Generally, research in the space of reserving has been practically inseparable from concentrating on store substitution arrangements. Much work on intermediary Web storing has additionally been committed to shrewd substitution arrangements and their impacts on hit rates.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF THE SYSTEM

Throughout the long term Web storing has turned into a subject of expanding significance. The utilization of Internet reserves has turned into a modest and powerful method for further developing execution for Internet clients. Storing endeavors to further develop execution in three ways:

In the first place, storing endeavors to lessen the client saw idleness related with getting Web objects. Dormancy can be diminished in light of the fact that the reserves are ordinarily a lot nearer to the client than the server, and subsequently store hits bring about quicker conveyance of information when contrasted with getting it from the server.

Second, reserving framework endeavors to bring down the organization traffic from the Web servers. Network burden can be brought down since certain articles are served from the store instead of from the server, consequently forestalling rehashed excursions to the server for information.

At last, storing can lessen the assistance requests on the server since reserve hits forestall rehashed demands for reserved information, consequently decreasing how much handling that should be finished at the server. Storing is presently one of the fundamental ways of decreasing dormancy, server burden, and transmission capacity for conveying Web contents. Caching helps E-commerce servers to achieve better performance, which is considered to be vital for the success of an E-commerce company.

4.1. Update Propagation Techniques

There are five possible strategies for data propagation. The main distinction among the different algorithms is based on whether the server keeps track of the data cached by clients, which is known as the *data binding information*.

- ***On Demand Strategy (ODM)*:** This approach alludes to the clients mentioning the information from the server on an on request premise. The server needs to does no accounting to monitor the client store status. Whenever there is a solicitation, the client gives the server the limiting data. The server utilizes the limiting data to channel the bits of the server logs that should be shipped off the clients.

- ***Broadcasting with No Catalog Binding (BNC)*:** This methodology utilizes broadcasting strategies. The server pushes updates or information changes to all clients upon the commit of an update exchange. This technique keeps up with no limiting data in regards to the client's store status and subsequently doesn't monitor which pages are reserved at which client; it pushes the updates to all clients in the framework, whether or not a client has reserved the page or not. This framework isn't versatile considering an enormous number of clients. The benefit of this strategy is that the server dodges a portion of the above, for example, look-into logs tasks and the calculation expected to decide the objective of updates. Endless supply of an update from the server, the client verifies whether the update influences its nearby activity; assuming this is the case, it cuts off.

- ***Broadcasting with Catalog Binding (BWC)*:** In this scheme, the server keeps track of the status of client caches. Upon an update, the server decides on the clients that must be notified, and propagates updates based on the binding information. This technique reduces the number of updates to be propagated at the cost of maintaining binding information for all the clients.

- ***Periodic broadcasting with catalog binding (PWC)*** and ***Periodic broadcasting with no catalog binding (PNC)*** consolidate the possibility of intermittent update broadcasting. In these methods, the server gathers the progressions which have not seen by the client at some normal time period and starts the engendering of the updates to the client. On the off chance that the server keeps up with accounting data about the client's store status, the server sends just a part of the updates to the client (PWC). Then again, on the off

chance that the server keeps up with no limiting data, then, at that point, every one of the updates are engendered to every one of the clients (PNC).

4.2. Broadcasting with Catalog Binding Algorithm

```
BEGIN
Let CD = Client Data Store, SD = Server Data;
At the start of the System, CD ← SD;
SD ← Number of Clients and Clients' ID;
If (type of transaction = = "Read")
{
        Check_notification( );
        If (notification-status ! = alert)
        {
                The data is in the latest and consistence data;
                  Read Commit( );
        }
        Else
        {
                received  notification ( ) ;
                get_Update_from_server( );
                The data is in latest and consistence data;
                Read Commit();
```

```
        }
}
Else If (type of transaction = = "Write")
{
        Check_Concurrency( )
        If (status ! = concurrent access)
        {
                CD ← Update data in client's local database;
                Send_Update_to_Server( )
                Server_checks_catalog ()
                Server_send_notification( )
        }
        Else If (status = = the concurrent writing on the same data item)
        {
                The system checks the clients' timestamp from the catalog and then the
                earliest timestamp will be granted the write operation.
                CD ← Update data in client's local database;
                Send_Update_to_Server( )
                Server_checks_catalog ()
                Server_send_notification( )
        }
}END
```

4.3. The System Overview

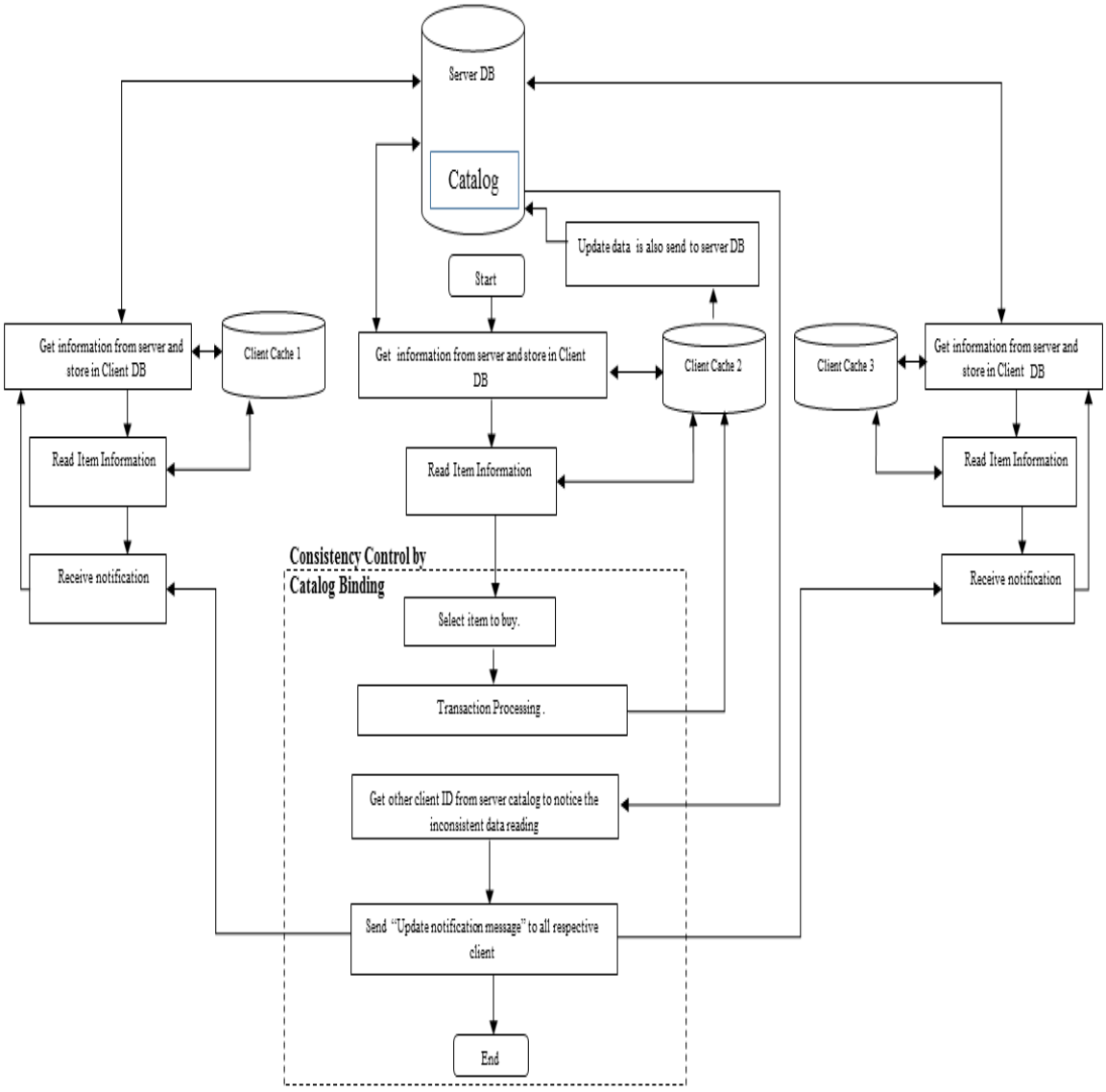

Figure 4.1: The System Flow

This system proposed to develop a data consistent online shopping system. In this system, the online customer receives a catalog of all the items he/she may need and adds the items he/she is interested into a shopping cart. When he/she wants to check out, the client submits an order for all the items placed in the shopping cart. However, not all orders are accepted by the server; there are cases in which a requested item has been sold out by the time the customer checks out. Moreover, the customer usually

spends a considerable amount of time making their decision to purchase an item and, hence, most of the transactions are lengthy.

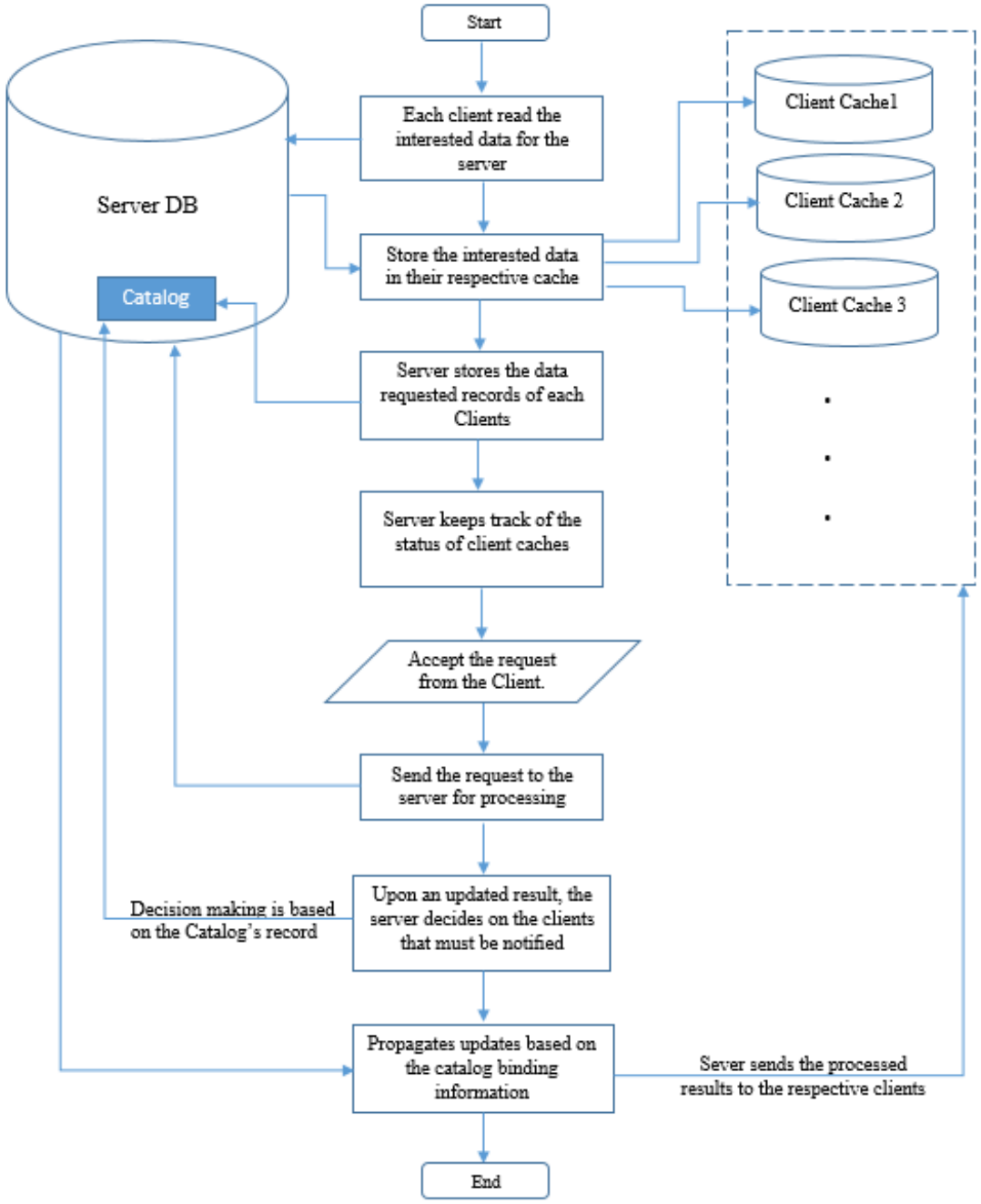

Figure 4.2: Data Update Propagation Flow

Since it is of some concern to the customer transaction, there is a need for the server to send notifications regarding the availability of the item, and any price changes, to the client before he/she checks out. This proposed system decides on the clients that must be notified, and propagates updates based on the binding information

by the “Broadcasting with Catalog Binding (BWC) approach”. This minimizes the number of aborted transactions and helps better satisfy the demands of customers. The data broadcasting steps are explained in details in figure 4.2. The server can also provide hints about the depletion rate of an item from its inventory so that customers can make an immediate decision to reserve or purchase an item.

4.4. Implementation of the System

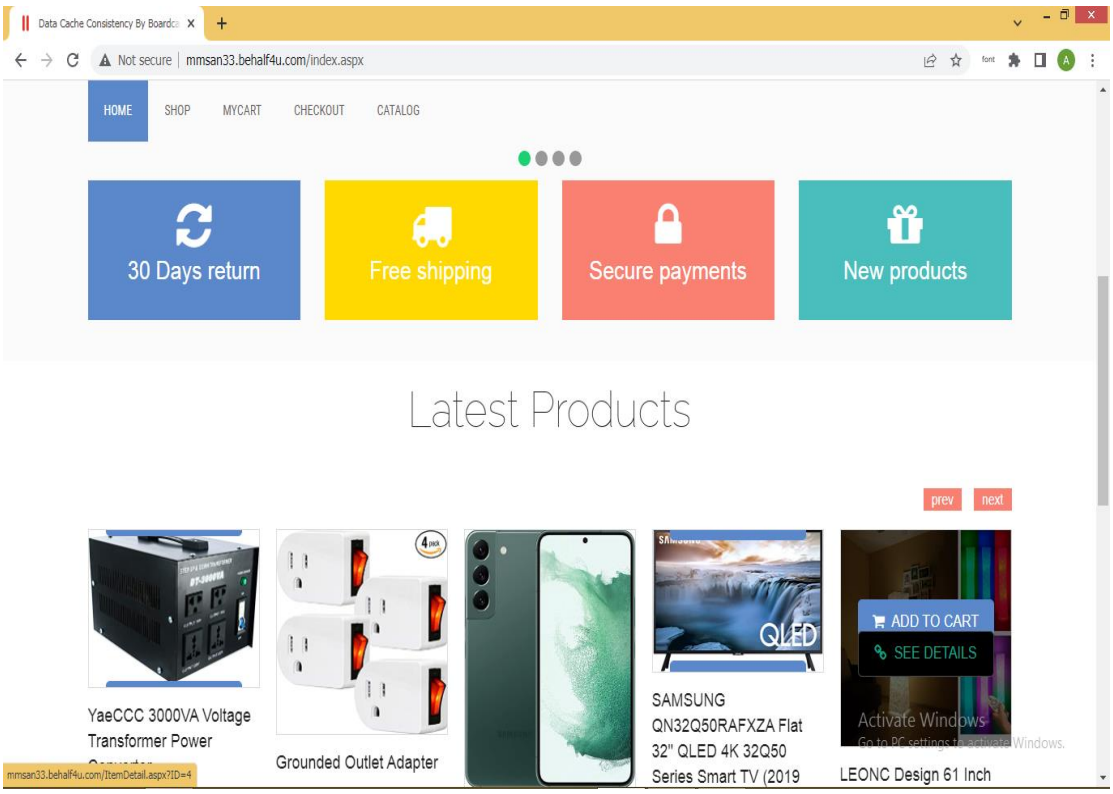

Figure 4.3: Main Page

Main page of the system is shown in figure 4.3. There are five main menus: “HOME” page, “SHOP” page, “MYCART” page, “CHECKOUT” page and “Catalog” page. “SHOP” menu will show the dealing item of the proposed website. Each item in “SHOP” page is described the item name, price of each item, available balance of item and “Add to cart” button as shown in figure 4.4. “Add to cart” button is supplied to be temporarily stored the customer desire item before buying it. But this button is only effective for the registered user. So, the customer must be registered first and then can use this button for temp storage of desire items.

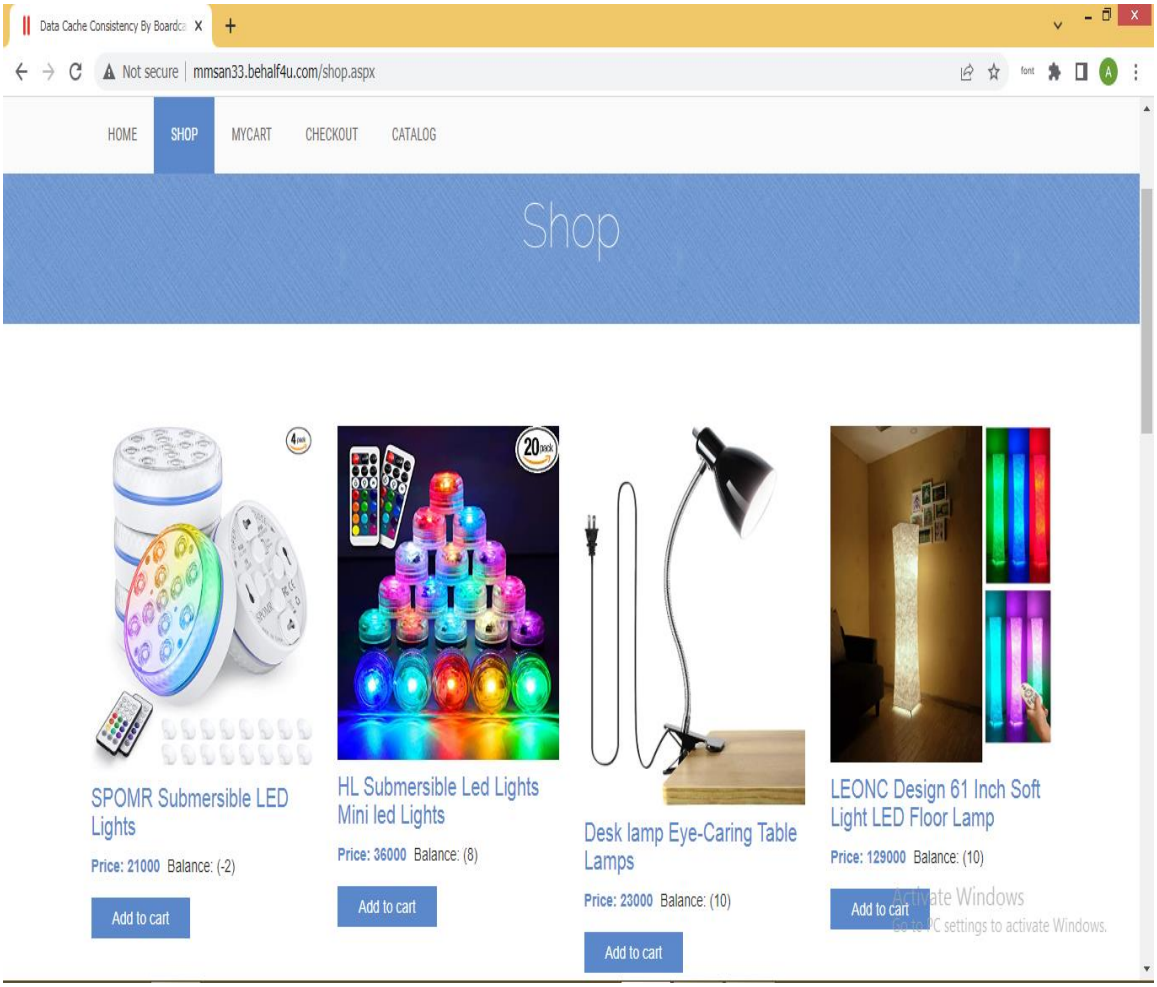

Figure 4.4: "SHOP" Menu of System

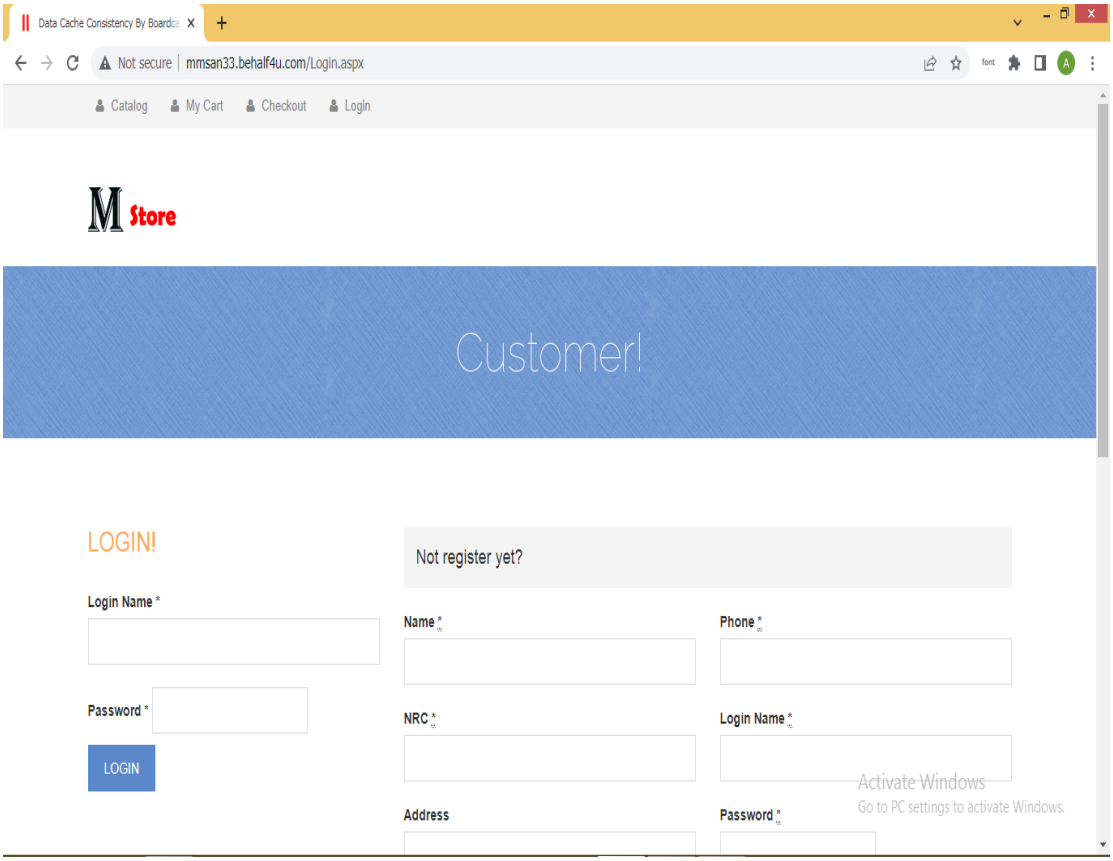

Figure 4.5: Login / Registration Page

Before adding the user desire item or buying item, the user must be login in first. Figure 4.5 shows the Login / Registration page of the system. This page contains two portions: Login section (support for registered user) and Register section (support for new user). This system only supports the registered user for buying product because of this system control the users' cache data consistency by using broadcasting with catalog binding algorithm. So, the system needs the customer ID, and "add to cart" item information. In case of data update situations, the information are needed to broadcast the data update to the specifically related user (i.e. the system does not send all users in the system and only send the related user with the updated data). To control the data consistency and broadcast to the related user, this system also maintains the "Catalog" page. This page only shows for the proposed system theory concept (no need to show in real world of web site).

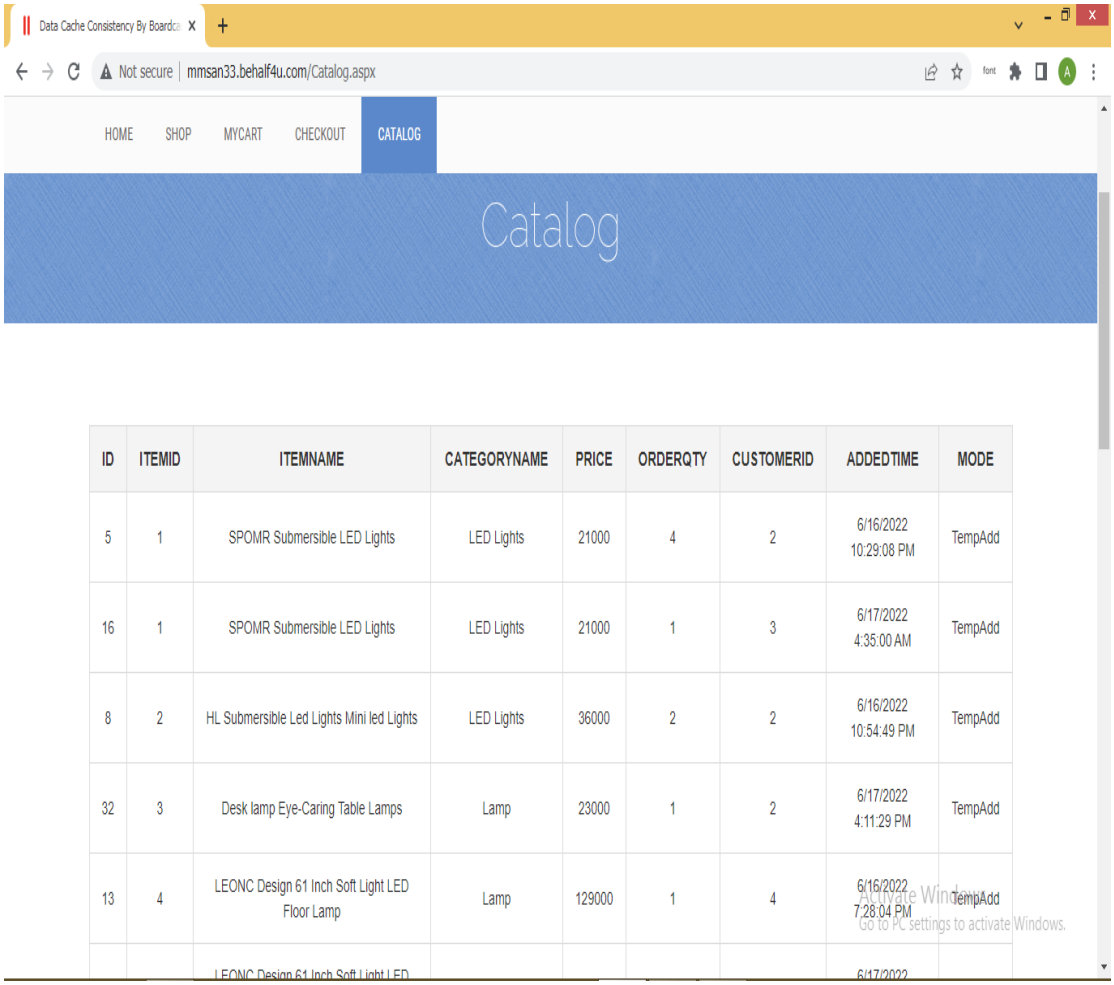

Figure 4.6: Catalog Page of the System

In the catalog page, the user selected item ID, item name, category name, price, order quantity, item added customer ID, added time (timestamp is used to control the concurrency in case of bottle neck time to perfectly make the concurrency control decision of the system) and mode ("TempAdd" mode aims to show the added item is before buy status) as shown in following figure 4.6.

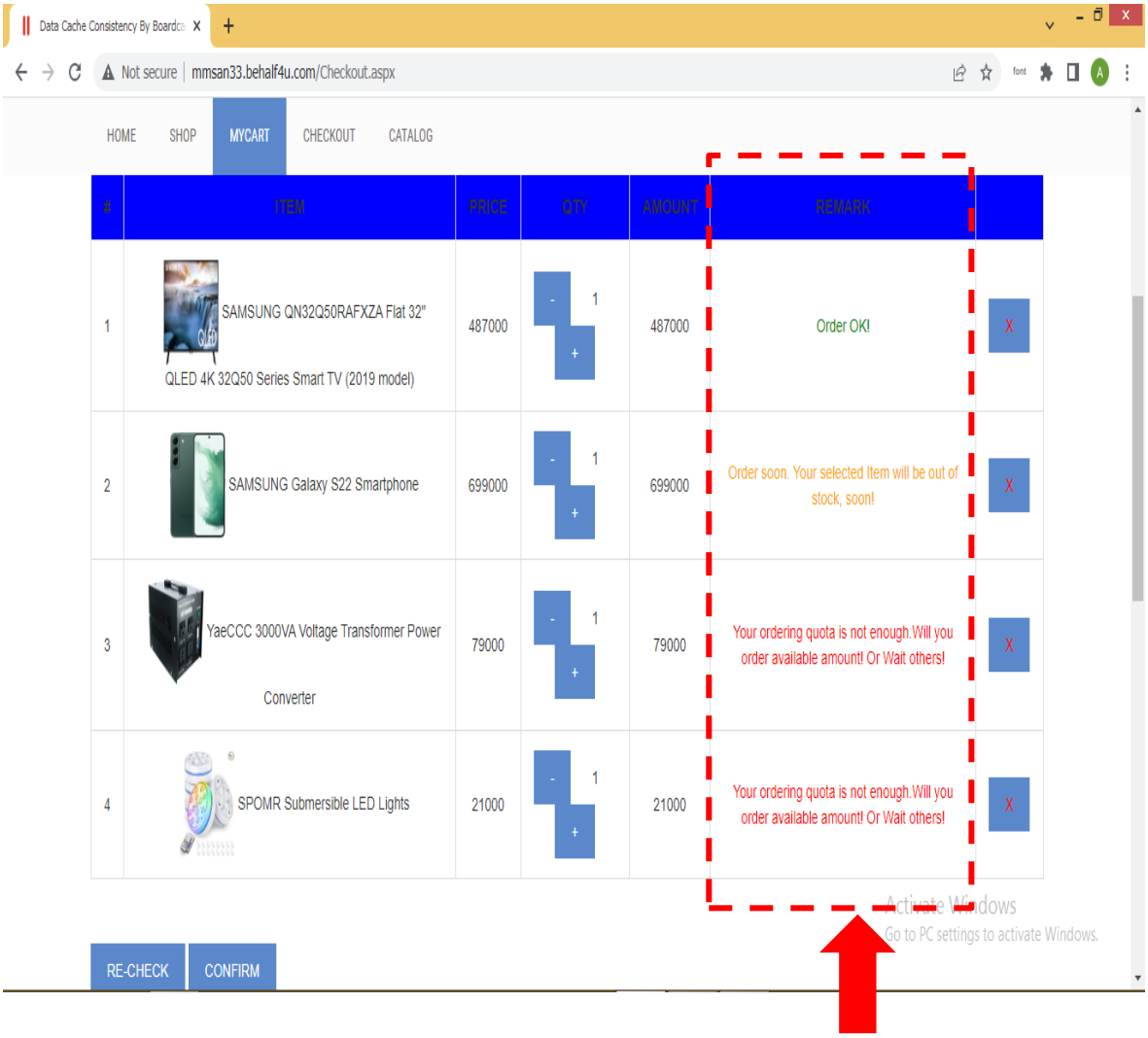

Figure 4.7: MYCART Page

"MYCART" page is supported to temporarily add the user desire item (serve as customer cache) before confirming the item to buy. In this page, registered user's added item information is shown as figure 4.7. Although the user added the item amount is shown in "QTY" field, the user can increase or decrease the desire amount of item. The "REMARK" field is the main notification section of the consistency control and concurrency control. In this field, there are one of three possible notification messages. "Order OK!" message is used to show the notification that is the user selected/added item is ready to busy and there is no concurrency case. "Order Soon. The selected item will be out of stock soon" message is to warn the user who is early selected user and the user selected item amount will not be enough soon. Because of the later user also added this item. In this situation, the early user adds amount of the item and later user add amount of the same item are not totally enough

to fulfill their requirements. So, the later users will also be received the notification message as “The order quota is not enough. Will the order be available amount. Or Wait Others!” But all of the messages are only sent to the respective user. Not to send no overlap item added user.

CHAPTER 5

CONCLUSION

Cache consistency control framework can be controlled the concurrent admittance to the data set and information consistency issue in multi-client climate. In this way, the impedance among applications doesn't cause a deficiency of consistency. This proposed framework expects to give the consistency control of disseminated simultaneous exchange for online business framework by utilizing information broadcasting with index restricting. This control works by trading advise data during the commit convention. In the event that the notice of information update is gotten, rehash the important altered information on data set by the warning and update the information on the client side. By utilizing this methodology, a few advantages are staying away from the stops; keeping up with information consistency and safeguarding the showing up of simultaneousness issues enjoyed lost update issue and conflicting issue. Also, correspondence cost is lower and the extra expense for the framework can be decreased. The proposed technique which is differentiated based largely on how they implement updates. The proposed E-commerce system can impact a large number of clients, client cache size, and epsilon value for Web workloads.

5.1 Benefits of the System

The main advantage of this system is deadlock free, thus saving the expense that deadlock detection usually required in locking approach. Processes can be run concurrently without affecting other processes and without failing. Fetching objects at the client side and working there locally, thus reduced the processing time and network latency. “Broadcasting with catalog binding”, inform each client sides about the task according to the user ID, Item ID, data processed time (timestamp) and current updated data. So, all objects remain in a consistent state when they are accessed by multiple transactions.

5.2 Limitations and Further Extensions

This system does not contain any payment method with bank. It implements the online sale system for approving consistency and concurrency control at the distributed database by catalog binding. Since it is the distributed database system, it depends on the server database and client database. If the server database is crashed, this system cannot effort to get the original data. So, this system can be extended the data recovery service to be perfect data reliability.

REFERENCES

[1] Adya, A., Gruber, R., Liskov, B. and Maheshwari, U. “Efficient optimistic concurrency control using loosely synchronized clocks,” in Proceedings of the ACM SIGMOD Conference on Management of Data. San Jose, CA, pp. 23– 34.

[2] Carey, M.J., Franklin, M.J., Livny M. and Shekita, E. J. “Data Caching Tradeoffs in Client-Server DBMS Architectures,” in Proceedings of the ACM SIGMOD, pp. 357-366.

[3] Franklin, M.J. “Client Data Caching: A Foundation for High Performance Object Database Systems,” Kluwer Academic Publishers, Boston, MA.

[4] Franklin, M.J., Carey, M.J. and Livny, M. “Transactional client-server cache consistency: alternatives and performance,” ACM Transactions on Database Systems, vol. 22(3), pp. 315-363.

[5] Laudon, J. and Lenoski, D. “The SGI Origin: A ccNUMA highly scalable server,” in Proceedings of the 24th Annual International Symposium on Computer Architecture, vol. 25(2), pp. 241-251.

[6] Ozsu, M.T., Voruganti, K. and Unrau, R. “An asynchronous avoidance-based Cache Consistency Algorithm for Client Caching DBMSs,” in Proceedings of the Conference on Very Large Data Bases (VLDB). New York, NY, pp. 440-451.

[7] Wang, Y. and Rowe, L.A. “Cache consistency and concurrency control in a client/server DBMS architecture,” in Proceedings of the ACM SIGMOD Conference on Management of Data. Denver, CO, pp. 367–377.

[8] Wilkinson, K. and Neiman, M.-A. “Maintaining consistency of client-cached data,” in Proceedings of the Conference on Very Large Data Bases (VLDB). Brisbane, Australia, pp. 122-133.