CHAPTER 3

OPTIMAL CONTENT ALLOCATION FOR WEB DIGITAL LIBRARIES

Chapter 2 explains the need for an optimal content allocation in the networked system. In this present work a multi-agent based user access pattern oriented optimal content allocation method has been designed to solve this problem of content management.

In this work, Section 3.1 explains user access pattern modeling. Section 3.2 explains the Network system model. Section 3.3 presents the multi-agent system framework design. Section 3.4 presents the distributed algorithm for different node. Section 3.5 gives the empirical modeling for this system 3.6 simulation experiment as well as experiment results. Section 3.6 summaries this work.

3.1 USER ACCESS PATTERN MODELING

The user access pattern furnishes information that explains the essential structure as to what type of contents is accessed by a specific region. For a language region the user access pattern explains how the numbers of the i\textsuperscript{th} language requests are received from the region j. The digital libraries are designed for various specific purposes such as scientific and scholarly content, tourism advertising, virtual universities, geo-referenced information system etc., wherein the contents are accessed across different parts of the world. A
digital library offers different services to users all over the world and also content are from different countries in their regional languages. Usually content of a specific language are accessed frequently by a set of people at a specific region. However owing to the transmigration of people of a specific group from one country to another, they may have to search for the specific contents of their own language (or mother tongue). Besides this, there are people who are very specific in accessing contents of certain specific systems irrespective of their locations. Considering these factors, the static policy for content storage may no longer be applicable and hence storing the content objects of the respective region will not always be optimal, necessitating the need for a dynamic allocation policy.

A video browsing system (Silvia Holffelder 1999) designed to overcome information overload has the facility to pre-select the shots of interest and also to keep it in the buffer for effective browsing. This mechanism supports interactive multimedia browsing applications by exploiting information about the expected browsing behavior of the user, which is estimated on the basis of a rationalistic exploration strategy for the retrieval of results. There is need for a content conceptualization.

In the present design of multi-agent system, an agent tries to fix it in every distributed server and do content conceptualization through content analysis. The content specification gives the details of content language, content style, content region, etc. The agent collects the user profile while accessing the requests. By processing the requests of every user, the agent is able to identify the origin of those requests. The multi-agent system formulates a pattern for that type of access and correlates the frequency of access. This system is able to identify the specific type of content, which is normally accessed by users in these regional people and groups. Such a
system is called the user access pattern learning, which takes a very long period and also changes over time. The new content objects are frequently uploaded into the system, which is able to allocate the content in the optimal location instead of replicating the object to all servers.

The user profile is collected at the first time of user request and the content profile is built at the time of uploading the content. The content access details are collected through the user requests. Language profiles and regional profiles are collected and updated as and when they are required. All these profiles are shown in Table 3.1-3.5.

Table 3.1 Language Profile

<table>
<thead>
<tr>
<th>Language Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>L001</td>
<td>Czech</td>
</tr>
<tr>
<td>L002</td>
<td>Danish</td>
</tr>
<tr>
<td>L003</td>
<td>Dutch</td>
</tr>
<tr>
<td>L004</td>
<td>English</td>
</tr>
<tr>
<td>L005</td>
<td>Estonian</td>
</tr>
<tr>
<td>L006</td>
<td>Finnish</td>
</tr>
<tr>
<td>L007</td>
<td>French</td>
</tr>
<tr>
<td>L008</td>
<td>German</td>
</tr>
<tr>
<td>L009</td>
<td>Greek</td>
</tr>
<tr>
<td>L010</td>
<td>Hungarian</td>
</tr>
</tbody>
</table>
Table 3.2 Object/Content Profile

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Name</td>
<td>Technology Object10</td>
</tr>
<tr>
<td>Content Id.</td>
<td>C0025</td>
</tr>
<tr>
<td>Language</td>
<td>English</td>
</tr>
<tr>
<td>Language Id.</td>
<td>L001</td>
</tr>
<tr>
<td>Size</td>
<td>1 GB</td>
</tr>
</tbody>
</table>

Table 3.3. User Profile

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>James Ronold</td>
</tr>
<tr>
<td>User Id.</td>
<td>U045</td>
</tr>
<tr>
<td>Spoken Language</td>
<td>English</td>
</tr>
<tr>
<td>Region</td>
<td>London</td>
</tr>
<tr>
<td>Region Code</td>
<td>R006</td>
</tr>
<tr>
<td>Address</td>
<td>#8, Downing St., London, UK</td>
</tr>
<tr>
<td>Gender</td>
<td>Male</td>
</tr>
</tbody>
</table>

Table 3.4. Language Region Patterns

<table>
<thead>
<tr>
<th>Language Code</th>
<th>Region Code</th>
<th>No. of Requests / Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>L001</td>
<td>R008</td>
<td>80</td>
</tr>
<tr>
<td>L001</td>
<td>R009</td>
<td>15</td>
</tr>
<tr>
<td>L001</td>
<td>R010</td>
<td>18</td>
</tr>
<tr>
<td>L001</td>
<td>R011</td>
<td>20</td>
</tr>
<tr>
<td>L001</td>
<td>R012</td>
<td>30</td>
</tr>
</tbody>
</table>
Table 3.5. Regional Profile

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Country Name</th>
<th>Region Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vienna</td>
<td>Austria</td>
<td>R001</td>
</tr>
<tr>
<td>Brussels</td>
<td>Belgium</td>
<td>R002</td>
</tr>
<tr>
<td>Nicosia</td>
<td>Cyprus</td>
<td>R003</td>
</tr>
<tr>
<td>Prague</td>
<td>Czech Republic</td>
<td>R004</td>
</tr>
<tr>
<td>Copenhagen</td>
<td>Denmark</td>
<td>R005</td>
</tr>
<tr>
<td>Tallinn</td>
<td>Estonia</td>
<td>R006</td>
</tr>
<tr>
<td>Helsinki</td>
<td>Finland</td>
<td>R007</td>
</tr>
<tr>
<td>Paris</td>
<td>France</td>
<td>R008</td>
</tr>
<tr>
<td>Berlin</td>
<td>Germany</td>
<td>R009</td>
</tr>
<tr>
<td>Athens</td>
<td>Greece</td>
<td>R010</td>
</tr>
<tr>
<td>Dublin</td>
<td>Ireland</td>
<td>R011</td>
</tr>
<tr>
<td>Riga</td>
<td>Latvia</td>
<td>R012</td>
</tr>
<tr>
<td>Vilnius</td>
<td>Lithuania</td>
<td>R013</td>
</tr>
<tr>
<td>Luxembourg</td>
<td>Luxembourg</td>
<td>R010</td>
</tr>
</tbody>
</table>

3.2 THE NETWORK SYSTEM MODEL

As the set of servers are located in a non-hierarchical order, there is a need for a virtual hierarchical tree $T = (V, E)$, ($V$ being the set of nodes and $E$ the set of directed communication links) to represent the communication network. In the case of distributed digital library the contents are distributed and are maintained as the hierarchical tree. The hierarchical tree approach for content management permits high-level of scalability, with each node representing the server’s capability of storing content objects and each server representing the agent program. The system has three types of nodes – head-end, intermediate node and root node. The head-end (switches) connects all
the user nodes through a client node; and stores the content objects, depending on the user access. Each head-end has an agent program, called head-end accessor, which is responsible for storing the content objects in the head-end nodes and also performing the user classification and content access pattern learning by analyzing the user requests. It also collects the information (user profile) about the user who is accessing the object. Such types of information are forwarded to the intermediate nodes. New contents are also uploaded into the system through any one of these nodes, during the process of loading. It collects the information (object profile) about the object and copies of these profiles are then moved to the root node.

The intermediate node stores an agent program and receives a user profile from the head-end and classifies users into different categories. Normally, the system (virtual tree) contains a number of intermediate levels. The very first (line before the head-end) and the intermediate level of nodes do the user classification and also the object profile analysis. After analysing the user content frequency access over the previous period, a pattern is formed to include the probability of user access to particular language objects by combining previous accesses over the particular period of time in various regions. The entire user requests, object information and object copy are moved to the root server. This model is shown in the following Figure 3.1. Here ‘*’ indicates the content/object availability in the particular node.
The following model assumptions are made

1. While satisfying the head-end’s requests for object in the network, any one of the nodes on the path from the root to head-end invariably contains the object.

2. The link capacities are sufficient to provide all types of requests for delivery of the object to all head-ends.

3. The links are bi-directional in terms of message exchange, but necessarily distributed from root node to the leaf node for providing the objects at the head-ends.

4. A labeling process identifies each node. The agent is capable of not only identifying each node connected to all its leaf nodes but also the number of links to which the leaf nodes are linked to the root of the tree.

5. Each head-end represents the regional server located in different regions. (For example above Figure .7, H1 represents region 1 (R1), H2 represents region 2 (R2) and so on.)
6. The root server initially stores the complete set of objects in all languages.
7. Generally, other relevant assumptions are made for network representations, pertaining to node linking and message receipt.

### 3.3 THE MULTI-AGENT SYSTEM FRAMEWORK

In the Multi-Agent Framework, every agent comprising of a single component is able to play various roles according to the place in which the agent lives. The intelligent agent emits specialised behaviour for handling different tasks. Such behaviour can be modelled using role (Georg Gottlob 1996), since, the agent is perceived as an entity with different roles. In the current design, it performs tasks such as 1. Head-end Accessor, 2. User Access Pattern Analyser, 3. Content location Analyser, 4. Agent Replicater, 5. Content Allocator. The basic objective design of this system is its capability to adapt to any changing conditions that may occur in any part of the system in such a manner that every module is independent of each other as is shown in Figure 3.2. The various roles played by the agents are

#### 3.3.1 Head-End Accessor: This task is equipped with a browser style GUI, which is able to collect the user semantic information about the content request with the user profile (if he has not been profiled earlier) and subsequently the user is classified in a particular category. It stores this information in the local server. This is forwarded to the immediate ancestor. The location of the user is also identified by the system by fetching the IP address of the user access location. Also, it provides a provision to load the contents in the system.
3.3.2 **User Access Pattern Analyser**: This is a non-interactive component, which will analyse the different type of users and contents, according to the various attributes and classify them in a certain category. Based on the frequency of access in a particular location, the content access frequency is then computed based on the type of content that is accessed in the particular region. This pattern of information is moved to the root node.

3.3.3 **Content Location Analyser**: This is also a non-interactive component, which analyses regional information and compares the best level of hierarchy location for every individual pattern. It also takes into account the parameters of storage and bandwidth, apart from the access pattern of the user. It uses a distributed algorithm as given in the Section 3.4 developed by the authors. It also identifies the intra-correlation between languages and regions. If it finds any intra-correlation between two language and regions then it will automatically allocate the content of one language to the other related language regions. This correlation pattern identification method is explained in Section 3.5.1.
3.3.4 Agent Replicater: This component performs the agent cloning when any one node is added, and instructs the necessary nodes to do the adjustments if any one of the components is removed.

3.3.5 Content Allocator: For every pattern, this component performs the content allocation according to the information given by content location Analyser.

3.4 DISTRIBUTED ALGORITHM

In this system an agent fixes itself in every server and uses three different algorithms according to the location. This constitutes a new design of distributed algorithm approach justifying the three different roles. The first algorithm is used by the head-end, second by the intermediate nodes and the root server uses the third one. The agent is able to pass the computed information to the parents. In turn each of the parent nodes also passes the required information to its entire children. This agent also passes the depth of information while cloning the new server. These algorithms compute the average request service cost for storing the particular language pattern objects in the concerned nodes.

3.4.1 Algorithm for Head-End Node

1. \( C_{ci} \) - Cost of moving particular language (\( L_j \)) pattern objects from the root to the Head-end node \( H_i \).

2. \( S_{ci} \) - Storage Cost for the objects at node \( i \), \( T_i \) - total number of requests at Head-end \( H_i \) for various objects of particular language (\( L_j \)) pattern, Set LOCATION = “UP”.

3. \( C_i = S_{ci} + C_{ci} \).
4. Compute the average request cost at Head-end Hi (for L_j) using the following equation \( C_{\text{avg}} = C_i / T_i \).

5. Communicate the average cost, depth, and number of requests to the parent node.

### 3.4.2 Algorithm for Intermediate Node

1. \( C_{ri} \) - Cost of moving particular language (L_j) pattern objects from the root to the current node.

2. \( C_{si} \) - Cost of serving the particular language (L_j) objects to the Head-ends.

3. \( C_{ci} = C_{ri} + C_{si} \).

4. \( S_{ci} \) - Storage Cost for the particular language (L_j) objects at the i\(^{th}\) node, \( T_i \) - the total number of requests at Node \( I_{ij} \) from its Childs for the particular language (L_j) pattern objects. Set LOCATION = “UP”.

5. \( C_i = S_{ci} + C_{ci} \).

6. Compute the average request cost at Node \( I_{ij} \) using the following equation.
\[
C_{\text{Avg}} = C_i / T_i
\]

7. If any of the child’s average is less than the Caver and not equal to zero then
   i. Pass the LOCATION = “HERE” to those nodes.
   ii. Drop the requests from that node, recompute the value of \( T_i, C_{si}, C_{\text{Avg}} \).

8. Communicate the average cost, depth, and number of requests to the Parent nodes.
3.4.3 Algorithm for Root Node

1. $C_{si}$ - Cost of serving the particular language ($L_j$) pattern objects to the Head-ends.

2. $C_{ci} = C_{si}$.

3. $S_{ci}$ - Storage Cost for the particular language ($L_j$) pattern objects, $T_i$ - the total number of request at root from its child for the particular language ($L_j$) pattern objects.

4. $C_i = S_{ci} + C_{ci}$.

5. Compute the average request cost at root using the following equation
   \[ C_{Avg} = \frac{C_i}{T_i}. \]

6. If any of the child’s average is less than the $C_{Avg}$ and not equal to zero then Pass the LOCATION = “HERE” to those nodes.

Notations

$C_{ci}$ - Communication Cost for particular patterned objects at node i, $C_{ri}$ - Cost of moving particular patterned objects from the root to the current node i, $C_{si}$ - Cost of serving particular patterned objects to the Head-ends from the current node i, $S_{ci}$ - Storage Cost for particular patterned objects at node i, $C_i$ - Total Cost for particular patterned at node i, $C_{Avg}$ - Average Total Cost/Request for particular patterned objects, $T_i$ - Total number of requests served from node i for particular patterned objects, $L_j$ - $j^{th}$ Language.

The content allocator is moving all the objects of the specific language pattern to all the nodes having LOCATION information “HERE”.
3.5 COMPLEXITY

Based upon the algorithms in Section 3.4.1, regarding the Nodes, the algorithms analysis pertaining to the computational, message, time complexity are established here.

Algorithm is said to be efficient if there exists a polynomial \( p(n) \) such that any instance of size \( n \) in a time will be of \( O(p(n)) \) or by \( \Omega (p(n)) \). Such an algorithm is said to be in polynomial time.

3.5.1 Computational

Since Node \( i \)'s computation consists of just \( (d_i+1) \) sums of \( |CH_i| \) elements, clearly the complexity for Node \( i \) is \( (d_i+1) |CH_i| \)

where \( d_i \) is Node i’s distance from the root and \( |CH_i| \) is the number of Node i’s children. The overall complexity of all nodes is

\[
\sum_{i \in V} (d_i + 1)|CH_i|, \text{ for } |V| = N \text{ and } d_i \leq N - 1 \text{ for } \forall i. \tag{3.1}
\]

Therefore worst case computational complexity:

\[
\sum_{i \in V} (d_i + 1)|CH_i| \leq N \cdot \sum_{i \in V} |CH_i| = N (N - 1) = O(N^2) \tag{3.2}
\]

The last equality holds because the total number of children in the tree equals \( N-1 \). Hence, the algorithm complexity is

\[
O(dN) = O(N^2). \tag{3.3}
\]
Best case computational complexity:

Number of nodes the content consist is 1, therefore

\[(d_i+1) |CH_i| \text{ where } N=1, \ O(d) = O(N). \tag{3.4}\]

Average case computational complexity:

Number of nodes for the worst case \(N-1\) and Number of nodes for the best case is 1.

The average number of nodes \((N-1+1)/2 = N/2\) \tag{3.5}
Therefore the average complexity \(O(d(N/2)) = O(N^2). \tag{3.6}\)

3.5.2 Message

In this algorithm there are two types of messages. The first is sent by each node to its parent and consists of the cost vector calculated by the node. The second type, content, is sent by each node to its children. There are \(N\) links and each link passes two messages, one of each type.

The message best message complexity therefore, \(O(N)\).

The longest message is the one with the longest cost vector. The length of the cost vector is determined by the distance (number of hops) from the node to the root. Let \(d = \max \{ d_i \}\) and \(\text{COST} = \max \{ \text{Cost} [i] \}\). The longest message is \(d\) long and worst complexity is \(O(\ d.N \ \log\ \text{COST}). = O(\ d.N.c), c\ is\ \log\ \text{COST}\ is\ a\ constant. = O(\ d.N) = O(N^2).\)

The average message complexity \(O(N) + O(N^2)/2 = O(N^2). \tag{3.7}\)
3.5.3 Time

Assuming that the time to send a message over a link is 1 time unit, the running time of the algorithm is 2d time units.

The worst case time complexity: $O(d)$.

The best case time complexity: $O(0) = 1$, where $d = 0$, that is content is available in head-end nodes.

The average case time complexity: $(1+O(d))/2 = O(d)$. \hfill (3.8)

<table>
<thead>
<tr>
<th></th>
<th>Computational</th>
<th>Message</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>1</td>
</tr>
<tr>
<td>Average</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$</td>
<td>$O(d)$</td>
</tr>
<tr>
<td>Worst</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$</td>
<td>$O(d)$</td>
</tr>
</tbody>
</table>

3.6 CORRECTNESS

A Proof for the generalized problem (with the tree attached to a string) that is based on induction and the assumption that optimal content location has terminated. The proof that content allocation algorithm terminated is trivial: the termination of phase I can easily we shown by induction, starting with the leaves and ending with the root. The termination of phase 2 is trivial by induction, staring with the root and ending with the Leaves. Lemma 1 is used as the induction base.
Proofs of Lemma 1 and Lemma 2

Proof of Lemma 1: Either one of two possibilities holds.

1. There is no string connected. CAA (Content Allocation Algorithm) allocates the object at Node $i$, which is the only possibility, and thus the optimum.

2. A string with length $j$ is connected to Tree $T_i$. If the communication cost is less than the storage cost ($C_i < S_{ci}$), it is clearly better to use (as CAA does allocate) the copy from the string ($cost$ is $C_i$ and $location$ is up). Otherwise, CAA allocates the object at Node $i$ ($cost$ is $S_{ci}$ and $location$ is here), which is obviously optimal.

Proof of Lemma 2: We prove by contradiction. Assume that the lemma does not hold; thus, if $COST^{ALG}$ is the cost of the solution $SOLUTION^{ALG}$ calculated by the algorithm, then, for the optimal cost $COST^{OPT}$ of an optimal solution, the assumption is that $COST^{OPT} < COST^{ALG}$. For each case, in the following case analysis, we use $SOLUTION^{OPT}$ to calculate solutions for the children of Node $i$. We show that at least one of these solutions has a lower cost than the solution found for the child by CAA. This contradicts the assumption in the lemma that the solutions calculated for the children are optimal.

Either Lemma 1 there is no string connected to Node $i$, or Lemma 2 there is a string connected.
1. If there is no string connected, then CAA's allocation (of the value cost \( i[0] \)) is the minimum between:

1 (a) not allocating an object copy at Node \( i \), and 1b) allocating an object copy at Node \( i \). In each of these cases, we consider separately the case that \( SOLUTION^{OPT} \) allocates a server at Node \( i \), and the case that \( SOLUTION^{OPT} \) does not.

(a) Consider the case that the minimum is obtained by CAA by not allocating an object copy at Node \( i \). The cost calculated by CAA is, thus, the sum of the costs of the trees that are rooted at the children of Node \( i \) (\( T_1, T_2, \ldots, T_{|S|} \)) with no string connected to them (\( \sum_{i \in \text{CHI}} cost_i \)). If \( SOLUTION^{OPT} \) does not allocate a copy at Node \( i \), then, by definition, \( SOLUTION^{OPT} \) too is the sum of the costs calculated (say, by some optimal algorithm) for the children. Thus, for \( COST^{OPT} \) to be smaller than \( COST^{ALG} \), the cost of the solutions for at least one of the children in \( SOLUTION^{OPT} \) must be smaller than in \( SOLUTION^{ALG} \). This is a contradiction.

If \( SOLUTION^{OPT} \) does allocate a copy at Node \( i \), then consider the solutions for the children of Node \( i \) in \( SOLUTION^{OPT} \). Each such child solution is a correct solution for the problem of allocating an object copy to the child's tree, with a string of length 1 attached to the child (since each such solution for a child of Node \( i \) assumes that there is a copy of the object at the parent Node \( i \)). The sum \( CC(OPT) \) of the costs of these child solutions is smaller than the sum \( CC(ALG) \) of the costs of the child solutions calculated by Algorithm CAA for the same children with the same string of length 1 attached. (These follows from the facts that \( COST^{OPT} = CC(OPT) + S_{C_i} \) and \( COST^{ALG} < CC(ALG) + S_{C_i} \) (since CAA found that the minimum is obtained without assigning an object.
copy at Node $i$) and the assumption that $COST^{OPT} < COST^{ALG}$.) Thus, at least one of the child solutions computed by CAA was not optimal. This is a contradiction.

(b) Consider the case that the minimum is obtained by CAA by allocating an object copy at Node $i$. - The cost calculated by CAA is, thus, the sum of the costs of the trees that are rooted at the children of Node $i$ ($T_1, T_2, \ldots, T_{|S_i|}$) with a string of length one ($S_i$) connected to them, and the storage cost at Node $i$ ($\sum_{i \in CHI} cost_t[1] + S_{Ci}$). The rest of the proof is very similar to the proof in case 1(a).

2. If a string of length $j > 0$ is connected, CAA's allocation (of the value $cost[i][j]$) is the minimum between:

   (a) Using the object copy from the string - The cost is the sum of the costs of the trees which are rooted at the children of Node $i$ with a string of length $j + 1$ ($<S_{j+i}$) connected to them ($\sum_{i \in CHI} cost_t[j+1]$).

   (b) Allocating an object copy at Node $i$ (see 1(a)), ($\sum_{i \in CHI} cost_t[1] + S_{Ci}$)

   (c) Not allocating an object copy at Node $i$, and not using the object copy from the string (see 1(b)), ($\sum_{i \in CHI} cost_t[0]$).

These are the only three possibilities when the string of length $j$ is connected to Node $i$. Thus, there are 3 cases for the minimum calculated by the algorithm, and for each such case, there are 3 cases for the decisions of the optimal algorithm assumed. For each of the 9 cases, assuming that $COST^{OPT} < COST^{ALG}$ implies that the solution of CAA for the children was not optimal, a contradiction to the assumption of the lemma. The proof in each of 9 cases is very similar to the proof of given example. Hence, the allocation of the object copies is optimal for the tree $T'$ rooted at Node $i$, for every length of a string connected to it.
3.7 EMPIRICAL MODELING

Consider the \(i^{th}\) object pattern, it has \(C_i\) as the cost element for optimal storage and communication given by.

\[ C_i = S_{ci} + C_{ci} \]  

(3.9)

The \(i^{th}\) object may pertain to language \(L_i\), \((i=1, 2, \ldots, n)\) and may spread over different regions \(R_j\) \((j=1, 2, \ldots, n)\). The first region \(R_1\) takes \(x_1\) number of requests for object 1, \(x_2\) number of requests for object 2, \(x_3\) number of requests for object 3 and so on under this category.

For example, it is known \(x_1 + x_2 + x_3 + x_4 = 163\) pertaining to the current (starting) month; \(x_4 + x_5 + x_6 = 736\) and \(x_7 + x_8 + x_9 + x_{10} = 101\) totaling 1000 requests for the required regions for different objects pertains to language \(L_1\). The set of requests \((x_1, x_2, x_3, \ldots, x_n)\) is random in nature and distributed over specific types of regions needing \(L_1\) (Language 1).

The distribution of objects pertaining to Language \(L_2\) to various regions, is assumed to take into account, the semantic aspect revealed by the requirements indicated by \((x_1, x_2, \ldots, x_n)\) for the current month. It may spread over different types of regions irrespective of the objects in \(L_1\) in various regions the previous month. If the successive stage of transition from the current period to the immediate successor period follows Markov property, then the corresponding one step transition probability matrix with the transition probability is as follows.

\[ p_{jk} = P(x_n = k \mid x_{n-1} = j), \quad n \geq 1 \]  

(3.10)

is given by \( P = [p_{ij}] \), \(i, j = 1, 2, \ldots, n\), such that

\[ \sum_{j=1}^{n} p_{ij} = 1 \quad \text{for } i=1, 2, \ldots, n \]  

(3.11)
In terms of the conditional probability notation, the probability distribution of the language $L_i$ to the various regions $R_j$ is given by

$$\Pr (L_i) = \Pr (L_iR_1) + \Pr (L_iR_2) + \ldots + \Pr (L_iR_j) + \ldots + \Pr (L_iR_n) \tag{3.12}$$

Where $\Pr (L_iR_j) = \Pr (R_j/L_i)\Pr (L_i)$ for $i,j=1,2,\ldots,n$. In other words, the conditional probabilities $\Pr (R_j/L_i)$ are given by $\Pr (L_iR_j)/\Pr (L_i)$ for $i,j=1,2,\ldots,n$. Similarly the conditional probabilities of $L_i$ with respect to $R_j$ are given by $\Pr (L_i/R_j) = \Pr (L_iR_j)/\Pr (R_j)$, for $i,j=1,2,\ldots,n$. It is necessary to assume that the conditional probabilities are already known for the current month. This prior information of knowing the language $L_i$, access frequency at the region $R_j$ is explained as a pattern for content allocation of new objects pertaining to language $L_i$. From the given requests; the content semantics of a particular specific region for every given language is identified. Instead of storing all the objects in the root node for unknown future requests, we are moving only the particular language objects needed to the required regions. This will enable us to achieve optimal content allocation.

### 3.7.1 Intra-Correlation Pattern

A mathematical technique is adopted in finding the intra-correlation between different languages and regions. Assume that there are $n$ languages $L_i$ ($i=1,2,3,\ldots,n$) and there are $n$ regions $R_j$ ($j=1,2,3,\ldots,n$). Let $x_{ij}$ denote the number of object content requests accessed by the customers (users) for the language $L_i$ in the region $R_j$ at any point of time. (The number of requests for $L_i$ accessed by the region $R_j$ need not be the same). This number $x_{ij}$ takes into account the unit bandwidth cost $B_{ij}$, unit storage cost $S_{ij}$ and delay time cost $D_{ij}$ for additional bandwidth. The movie object is accessed from local/remote server, reckoned to compensate the latency that may be inherent in certain environments. Among these individual costs some of the costs may not be
reckoned at all, consistent with the location, anywhere in the hierarchical graph tree representation. All these costs are added to constitute $C_{ij}$ as unit cost representation for the concerned $(i,j)^{th}$ entry.

$$C_{ij} = B_{ij} + S_{ij} + D_{ij} \quad (3.13)$$

where $B_{ij}$ and $D_{ij}$ become zero, if and only if $S_{ij} \geq 0$, indicating the availability of the object at the local server (Head-end). The set of measurement $x_{ij}$ can be represented in a bi-variate table in a matrix form $[X_{ij}]$, $i,j=1,2,\ldots,n$ indicating $(L_i,R_j)^{th}$ measurement. From this matrix, we shall have $n(n-1)$ pairs in the $R_j$ region, like $(x_{ij}, x_{il})$, $j \neq l$. There will be $n.n(n-1) = N$ (say) entries for all the $n$ regions. Such a table is called the intra-class correlation table (Table 3.8) and the correlation is called intra-class correlation.

As there is nothing to distinguish $x_{ij}$ (x) from $x_{il}$ (y), we have

$$\overline{x}_j = \overline{y}_j = \frac{1}{n(n-1)} \sum_{i=1}^{n(n-1)} x_{ij} \quad (3.14)$$

and

$$\sigma_x^2 = \sigma_y^2 = \frac{1}{n(n-1)} \sum_{i=1}^{n(n-1)} (x_{ij} - \overline{x}_j)^2 \quad (3.15)$$

$$cov(x_j, x_l) = \frac{1}{n(n-1)} \sum_{j,l} (x_{ij} - \overline{x}_j)(x_{il} - \overline{x}_l) \quad (3.16)$$

For instance for $j=1$,

$$\overline{x}_1 = \overline{y}_1 = \frac{1}{n(n-1)} \sum_{i=1}^{n(n-1)} x_{i1} \quad \text{etc.} \quad (3.17)$$

For the entire measurements as $L_i$ and $R_j$. We have

$$\overline{x} = \overline{y} = \frac{1}{n} \sum_{j=1}^{n} x_j \quad (3.18)$$
\[
\sigma^2_x = \sigma^2_y = \frac{1}{n} \sum_{j=1}^{n} (x_j - \bar{x})^2
\]

(3.19)

\[
\begin{align*}
 r &= \frac{\text{cov}(X,Y)}{\sqrt{V(X)V(Y)}} = \\
 &= \frac{n^2 \sum_{i=1}^{n} (x_i - \bar{x}) \sum_{j=1}^{n} (x_j - \bar{x})}{(n-1) \sum_{i=1}^{n} (x_i - \bar{x})^2} \\
 &= \frac{1}{n-1} \left[ \frac{n \sigma^2_m}{\sigma^2} - 1 \right] \\
\end{align*}
\]

(3.20)

(3.21)

It can be shown that

\[
-1 \leq r \leq 1
\]

(3.22)

An example from the observed data is worked out.

**Table 3.7. Experiment Data for Five Different Regions and Five Different Languages**

<table>
<thead>
<tr>
<th>REGION</th>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>L5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8 (Paris)</td>
<td>80</td>
<td>62</td>
<td>70</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td>R9 (Berlin)</td>
<td>15</td>
<td>65</td>
<td>30</td>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>R10 (Athens)</td>
<td>18</td>
<td>62</td>
<td>15</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>R11 (Dublin)</td>
<td>20</td>
<td>66</td>
<td>20</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>R12 (Riga)</td>
<td>30</td>
<td>69</td>
<td>10</td>
<td>9</td>
<td>40</td>
</tr>
</tbody>
</table>
Table 3.8: Intra-Correlation between Languages and Regions

<table>
<thead>
<tr>
<th></th>
<th>R8</th>
<th>R9</th>
<th>R10</th>
<th>R11</th>
<th>R12</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>62</td>
<td>15</td>
<td>65</td>
<td>18</td>
<td>62</td>
</tr>
<tr>
<td>80</td>
<td>70</td>
<td>15</td>
<td>30</td>
<td>18</td>
<td>15</td>
</tr>
<tr>
<td>80</td>
<td>5</td>
<td>15</td>
<td>3</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>80</td>
<td>70</td>
<td>15</td>
<td>80</td>
<td>18</td>
<td>90</td>
</tr>
<tr>
<td>62</td>
<td>70</td>
<td>65</td>
<td>30</td>
<td>62</td>
<td>15</td>
</tr>
<tr>
<td>62</td>
<td>5</td>
<td>65</td>
<td>3</td>
<td>62</td>
<td>2</td>
</tr>
<tr>
<td>62</td>
<td>70</td>
<td>65</td>
<td>80</td>
<td>62</td>
<td>90</td>
</tr>
<tr>
<td>62</td>
<td>80</td>
<td>65</td>
<td>15</td>
<td>62</td>
<td>18</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
<td>30</td>
<td>3</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>30</td>
<td>80</td>
<td>15</td>
<td>90</td>
</tr>
<tr>
<td>70</td>
<td>80</td>
<td>30</td>
<td>15</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>70</td>
<td>62</td>
<td>30</td>
<td>65</td>
<td>15</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>3</td>
<td>80</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>3</td>
<td>15</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>62</td>
<td>3</td>
<td>65</td>
<td>2</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>70</td>
<td>3</td>
<td>30</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>70</td>
<td>80</td>
<td>80</td>
<td>15</td>
<td>90</td>
<td>18</td>
</tr>
<tr>
<td>70</td>
<td>62</td>
<td>80</td>
<td>65</td>
<td>90</td>
<td>62</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>80</td>
<td>30</td>
<td>90</td>
<td>15</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
<td>80</td>
<td>3</td>
<td>90</td>
<td>2</td>
</tr>
<tr>
<td>1148</td>
<td>1148</td>
<td>772</td>
<td>772</td>
<td>748</td>
<td>748</td>
</tr>
<tr>
<td>624</td>
<td>624</td>
<td>624</td>
<td>624</td>
<td>624</td>
<td>624</td>
</tr>
<tr>
<td>632</td>
<td>632</td>
<td>632</td>
<td>632</td>
<td>632</td>
<td>632</td>
</tr>
</tbody>
</table>

\[
\bar{x}_1 = \bar{y}_1 = 57.4 \\
\bar{x}_2 = \bar{y}_2 = 38.6 \\
\bar{x}_3 = \bar{y}_3 = 37.4 \\
\bar{x}_4 = \bar{y}_4 = 31.2 \\
\bar{x}_5 = \bar{y}_5 = 31.6
\]
3.7.1.1 Calculation of Intra Correlation among Languages and Regions.

\[
\bar{x} = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{5} = 35.24 = \bar{y} \quad (3.24)
\]

\[
\sum_{i=1}^{5} (x_i - \bar{x})^2 = 456.6257 \quad (3.25)
\]

\[
\sum_{i=1}^{5} \sum_{j=1}^{5} (x_{ij} - \bar{x})^2 = 29414.51 \quad (3.26)
\]

\[
r = \frac{x^T \sum (x_i - \bar{x}) - \sum \sum (x_{ij} - \bar{x})}{(n-1) \sum \sum (x_{ij} - \bar{x})}
\]

\[
r = \frac{25 \times 456.6257 - 29414.51}{4 \times 29414.51} \quad (3.27)
\]

\[
r = -0.153 \quad (3.28)
\]

This indicates that the inter correlation among the languages and the region coefficient is negative. Also the regression lines of Y and X is

\[Y = -0.153 \quad \text{where} \quad Y = y - \bar{y} \quad X = x - \bar{x}\]

\[Y = y - 57.4 \quad X = x - 57.4\]

And this regression line lies in the second and fourth quadrants which indicates a negative correlation.

3.7.1.2 Calculation of Correlation among Languages in Region R8

\[
\bar{x}_i = \bar{y}_1 = 57.4 \quad (3.30)
\]

\[
\sigma_{x_1}^2 = \sigma_{y_1}^2 = 1219.04, f_{x,y} = 320.24 \quad (3.31)
\]

\[
r = 0.2626. \quad (3.32)
\]
There is a low pointer correlation among languages within the region R8. Similar calculations are made for the other languages.

3.8 SIMULATION EXPERIMENT AND RESULTS

The system under consideration is simulated using the Java RMIServelets (James Goodwill 1999). It is able to pass all the user requests to the parent servers. Also, two RMI clients are developed to get the user request for object and the content uploading. These RMI clients pass their requests to the RMIServelet server. This RMI server agent of the content content server will automatically perform the aforesaid roles as explained in Section 3.3.

In the current design the RMI server agent process listens on a particular port. When the server receives an RMI request message, it parses the request and finds the content availability. If it is available then it will supply it from the local server. Otherwise, it will forward the request to the parent server. The system is designed to accept the RMI messages in order to elicit the effective functionality. This agent is also able to replicate itself, as and when a new server is installed and will automatically replicate to that server and is playing various roles as explained in above design; these roles are played in according to the location (as explained in Section 3.3.).

3.8.1 Data Sets

In this experiment, during the first phase, data sets were collected from different Internet portals. A set of 20 e-books are collected from different computer science portals ranging from 20 kb to 25 kb in different forms by submitting different queries. The well known meta-data is constructed for all the documents in the server. This meta-data also specifies the various content
details as given in Table 3.2. This meta-data helps the agent to do optimal content allocation.

### 3.8.2 Experiment Results Metrics

The system is tested with above said data sets collected from Internet. Simulation of such a system allocates various types of movies of different languages at different regions. Especially the English language pattern is identified in five regions. There are also 4 English language content objects whose data are loaded into the system. It is assumed that the storage cost is 25 Units/object and Communication cost is 1 Unit/link for all objects irrespective of size.

Initially all the head-end’s average access cost is computed using the head end algorithm. This is shown in Figure 3.3. Later, all the intermediate average cost is computed and it is compared with the head-end’s average cost. This is shown in Figure 3.4. Here the average cost of H1 is less than the node I1 cost, so this pattern object copy is moved to the node H1. Again the intermediate node I1 recomputed the average cost and it is forwarded to the root. The root’s average cost is computed and is compared with intermediate average cost if it is less then a copy will be moved to that node. But, it is testing whether the average value is non-zero. Here it allocates copies of objects in nodes I1 and I2, because the computed average cost of I1, I2 is less than that of R1. The resulting request cost average is shown in Figure 3.5, Figure 3.6 and the resulting content allocation is shown in Figure. 3.1.

It is found from the experiment that language L001 (English) requests are occurring from the regions R008 (Paris), R009 (Berlin), R010 (Athens), R011 (Dublin) and R012 (Riga). That is L004 is receiving 80 requests from region R008, 15 requests from region R009, 18 requests from region R010, 20 requests from R011 and 30 requests from R012. Such a pattern is called
language-region pattern. The identification of correlation inherent among all the languages in all the regions reflects over all relationship among them. The correlation of all languages distributed in one specific region revels the inherent relationship between the languages pertaining to that region. Future work can be attempted by making use of bi-variate frequency distribution, based upon region/language will estimate the relationship between any two languages in particular regions and the relationship between regions for any specific language can also be distributed. This will enable the system to allocate related language objects for additional allocation to the regions. New objects (uploaded in to the network) belong to this language and need to be allocated in these regions, because requests for these objects usually occur only from these regions. But, at the same time, it is necessary to compute the best level of hierarchy for these objects for optimal delivery. Likewise it is necessary to identify the occurrence of sub-patterns within the particular language pattern. Thus a particular set of movies may have similar set of attributes, which need to be identified, and its best location can be computed for content allocation.

Figure 3.3 Experiment Results at Head End Nodes
Figure 3.4 Experiment Results at Intermediate Nodes

Figure 3.5 Experiment Results at Root Node and Recomputed Cost at Intermediate Nodes
3.9 CHAPTER SUMMARY

An attempt is made in this work to design and develop a multi-agent framework for optimal content allocation on federated digital libraries. This work has revealed that the user access pattern learning contributed to improve the dynamic optimal content allocation which is established by means of simulation, even though the simulations does not take into account the transmission delay time. Such handling of dynamic requests will be the focus on future research for improvement in optimal content allocation by means of sub pattern identification.