5.1 Introduction

The data allocation techniques proposed in the Chapter-4 are based on the static data access patterns of different queries as well as their origin and frequencies. In the static environment, optimum solution can be obtained through static data allocation in which the retrieval and update access frequencies of queries from different sites to fragments never change. But, in a dynamic environment where these access patterns change over time, the static allocation solutions would degrade the performance of distributed database system. The dynamic environment requires the change in data allocation schema in order to reduce the data transmission cost or communication. Therefore, reorganization or reconfiguration of distributed database system is required to maintain the performance of system during its lifetime. The re-configuration process entails the following [83]:

1. *Physical Change*: The changes related to the physical system of the distributed database are known as the physical change. The physical changes are due to the change in network topology, addition of new site in the communication network, removal of a site from the communication network, change in the site capacities etc.

2. *Logical Change*: The logical changes are those changes which are not of physical in nature. The logical changes are due to the migration of one or more application from one site to another, addition of applications to the distributed database, deletion of applications from the distributed database and/or change in execution frequency of different applications in the distributed database.

3. Combination of both (1) and (2).

Wilson and Navathe [103] have classified the reorganization and reconfiguration of distributed database into two different categories: Full/Total Redistribution and Limited Redistribution.
- Full/Total Redistribution: It involves the new fragmentation and allocation of data.

- Limited Redistribution: It involves only the re-allocation of data/fragments.

The present study concentrates only on the limited redistribution or re-allocation of data. The re-allocation process requires the monitoring of dynamic distributed database environment to detect the change in access patterns of different queries in the system. Statistics of these changes are evaluated then re-allocation process is initiated if needed to improve the performance of dynamic distributed database.

An algorithm named Threshold and Time Constraint Algorithm (TTCA) for non-replicated data allocation is being proposed in this chapter for dynamic distributed database environment to decrease the data transfer during the execution of queries. The objective of the proposed algorithm is to reduce the load on the communication network and to increase the overall performance of the distributed database system.

The proposed TTCA algorithm for non-replicated dynamic data allocation is an extension of existing data re-allocation algorithms: Optimal algorithm of Brunstrom et al. [19] and Threshold algorithm of Ulus and Uysal [100,101]. The proposed TTCA algorithm integrates and improves the ideas proposed in Ulus and Uysal [100, 101] and Brunstrom et al. [19].

### 5.2 Optimal Algorithm

Brunstrom et al. [19] have investigated the problem of re-allocating portions of the database in a distributed system with changing workload. They have proposed a heuristic algorithm known as optimal algorithm or simple counter algorithm for non-replicated dynamic data allocation in distributed database. The decision of re-allocation is taken on the basis of variation in access frequencies from a particular site to various fragments of the database over the time period.

A process known as stats process is used to monitor the access pattern of different transactions to the fragments. Initially, all the fragments are distributed to different sites of communication network using any static data allocation method. After the initial allocation, optimal algorithm maintains weighted counters of the number of accesses from each site to each fragment. Every time an access request is made for the stored
fragment then the access counter of the accessing site for the stored fragment is increases by one. The stats process regularly examines the counters for each fragment. The decision of migration of a fragment from one site to another depends on the values of counter. If the counter of a remote site to a fragment is greater than the counter of the current owner of the fragment, then the fragment is migrated to the remote site. So, a site having the highest access count for a particular fragment is the primary candidate for the fragment.

The drawback of this technique is that if the change in frequency of access pattern for each fragment is high, then it will spend more time for migrating fragments from one site to another. For example, there are two site denoted by A and B. Assume that at a time $t$, site A is the current owner of a fragment $F_i$ and access counter value of site A for a fragment $F_i$ is highest and access counter value of site B for fragment $F_i$ is one less than the access counter value of site A. After two consecutive accesses made by site B, the access counter value of a site B for a fragment $F_i$ is highest and access counter value of site A for fragment $F_i$ is one less than the access counter value of site B. This will result in migration of fragment $F_i$ from site A to site B. Shortly site A carry out two consecutive accesses for fragment $F_i$. This will result in site A having highest access count for fragment $F_i$ and access counter value of site B is one less than the access counter value of site A. This will result in migration of fragment $F_i$ from site B to site A. This can be generalized for multiple sites [100].

### 5.3 Threshold Algorithm

Ulus and Uysal [100, 101] proposed another heuristic algorithm known as threshold algorithm for dynamic data allocation in distributed database systems. Threshold algorithm maintains only one counter per fragment as compared to an array of values in the simple counter algorithm. By storing only one counter per fragment, the threshold algorithm decreases the storage cost as compare to simple counter algorithm where separate counter is used for each site.

Initially, all the fragments are distributed over communication network using any static allocation method and the value of the counter for each fragment is set as zero. The counter value of a fragment is increased by one whenever a remote site accesses the fragment. For every local access, the value of the counter is reset to zero. So, the counter value of a fragment is an evidence of the number of successive remote accesses
to the fragment. The ownership of a fragment is transferred from the current owner to another site every time the counter exceeds a predetermined threshold value. The basic structure of the threshold algorithm is given below [100]:

**Step1.** For each (locally) stored fragment, initialize the counter values to zero

**Step2.** Process an access request for the stored fragment

**Step3.** If it is a local access, reset the counter of the corresponding fragment to 0 and

   Go to step 2

**Step4.** If it is a remote access, increase the counter of the corresponding fragment by one

**Step5.** If the counter of the fragment is greater than the threshold value, reset its counter to zero and transfer the fragment to the remote site

**Step6.** Go to step 2.

The threshold algorithm decreases the movement of data over the communication network when change in frequency of access pattern for fragments is quite high. The threshold algorithm promises the stay of the fragment for at least \((T+1)\) accesses at the new site after a migration, where \(T\) is the value of threshold. But threshold algorithm has following drawbacks with its approach:

- Whenever the counter exceeds the threshold value, the ownership of the fragment is transferred to another site. But, it does not specify which site will be fragment’s new owner. Two different approaches have been suggested by Ulus and Uysal [100] for the new owner of the fragment. According to the first approach, the new owner of the fragment is selected randomly. According to the second approach, the last accessing site is the new owner of the fragment. But
both of these approaches are not right choices for selecting the new owner of the fragment. In the first case, the randomly generated site could be the site, which has never accessed the fragment. In the second case, the last accessing site could be the site, which has accessed the fragment only one time i.e. the last access.

- Each time a site is going for a local access to a fragment, it resets the counter of the fragment to zero. For example, after every T-1 remote accesses, if a site X goes for a local access to a fragment F_i and resets the counter of the fragment F_i to zero. This can result in increase in data transmission cost of executing queries because the site having maximum accesses for the fragment F_i may not be the owner of fragment F_i.
- It does not give the information about past accesses of the fragments because for every local access, the access counter is set as zero.

### 5.4 Proposed TTCA Algorithm

In this section, a new heuristic algorithm is being proposed to improve the performance of simple counter algorithm and threshold algorithm. The proposed algorithm re-allocates fragments with respect to the changing data access patterns. This algorithm is named as Threshold and Time Constraint Algorithm (TTCA). TTCA algorithm includes the time constraint to the existing Threshold algorithm and uses an access counter to keep track of access history of fragments. TTCA algorithm takes into consideration both the threshold value as well as the time at which accesses are made to a particular fragment before migrating the fragment from one site to another site.

Initially all the fragments are distributed over different sites using any static allocation method in a non-redundant manner. TTCA maintain a access counter matrix M of size m x n, where m denotes the total number of fragments and n denotes the total number of sites. M_{ij} is the number of accesses to fragment i by site j.

**Step1:** For each fragment, initialize the counter values equal to zero (i.e. set M_{ij} = 0, where i = 1,2,----,m and j = 1,2,----,n)
Step 2: Process an access request for the stored fragment

Step 3: Increase the corresponding access counter of the accessing site by one for the stored fragment and also store the time of corresponding access

Step 4: If the accessing site is the current owner, Go to Step 2. (i.e. Local access, otherwise it is remote access)

Step 5: If the counter of a remote site for a fragment is greater than the counter of current owner by a threshold value T and all last “T+1” accesses are made in a specified time (t) then migrate the fragment to the remote site and reset corresponding fragment’s counter equal to zero for all the site

Step 6: Else Go to step 2

Figure 5.1 shows the data allocation framework for dynamic distributed database system using TTCA algorithm, where CRS is the counter of remote site and CCO is the counter of the current owner site of the fragment.

TTCA algorithm will further decrease the migration of fragments over different sites as compare to Threshold algorithm. It will migrate a fragment only if that fragment is most recently accessed for T+1 times, where T is the threshold value. So if we increase the value of time variable then the migration of fragment will be more. But, if the value of time variable decreases then there will be less migration of fragments.

5.5 Comparison

Comparison of TTCA with the optimal and threshold has been made on the following three different parameters:
Allocate Fragments using any Static Allocation Method in Non-redundant Manner

For each Fragment
Initialize Counter Value = 0 in Access Counter Matrix

Process an access request for the stored Fragment

Increase the corresponding Access Counter of the accessing Site for the stored Fragment and also store the Time of access

Is the Access Local?

Yes

Is CRS > (T+CCO)? AND Are all the last T+1 Accesses of the Remote Site in Time (t)?

No

No

Yes

Reset corresponding Fragment’s counter equal to zero for all the sites and migrates the Fragment to the Remote Site

Figure 5.1 Data Allocation Framework for Dynamic Environment
• **Migration Condition:** Simple counter algorithm of Brunstrom et al. [19] migrate a fragment to a remote site when the counter of remote site for a fragment is greater than the counter of the current owning site. Threshold algorithm of Ulus and Uysal [100] migrates the fragment when the counter with particular fragment is greater than the threshold value. TTCA algorithm migrates a fragment when the counter of the remote site for the fragment is greater than the counter of current owner by a threshold value T and all last “T+1” accesses are made in a specified time (t).

• **Storage Cost:** Optimal algorithm uses extra storage cost for access counter matrix. Threshold algorithm required less storage cost as compare to Optimal algorithm, because it stores only one counter for each fragment. But TTCA algorithm requires more storage as compare to both Optimal and Threshold algorithms, as it stores access counter matrix and respective time of particular access.

• **Network overhead:** Optimal algorithm increases the traffic on network when changing frequency of access pattern for each fragment is high. Threshold algorithm decreases the overhead on the network by limiting the migration of fragments as compare to Optimal algorithm when changing frequency of access pattern for each fragment is high. TTCA algorithm further decreases the network overhead as compare to both optimal and Threshold algorithms by including both the number of access and most recent access conditions for migration of fragments.

### 5.6 Summary

In this chapter, a new algorithm, named Threshold and Time Constraint Algorithm (TTCA), is proposed for non-replicated allocation of data in dynamic distributed database systems. The proposed algorithm re-allocates fragments with respect to the changing data access patterns. TTCA algorithm migrates a fragment when the counter of the remote site for the fragment is greater than the counter of current owner by a threshold value T and all last “T+1” accesses are made in a specified time (t). The objective the proposed TTCA algorithm is to reduce the load on the communication network and to increase the overall performance of the distributed database system.