Categorized Cache Prefetching Policy in Mobile Environment

Prof. Sonali I. Ajankar,
Dept of Computer Tech (MCA),
VJTI,
Mumbai, India
ajankar_9@yahoo.co.in

Dr. S. S. Sane
Dept of Computer Tech (MCA),
VJTI,
Mumbai, India
sssane@vjti.org.in

***Abstract* - In mobile environment cache management is a challenging task; it attracts the interest of many researchers to work on this field. Caching and prefetching techniques improves the performances of the use of cache for mobile devices. One of the important problem is the efficient access to data. A proposed solution to this problem is the prefetching technique because of which user is able to access the data with low access latency and in case of disconnection or limited connectivity also.**

This paper deals with the study of cache prefetching policy and it's simulation and analysis. In order to do this, here present a client and server algorithm for prefetching scheme in mobile environment. It includes prefetching techniques based on user interest and a cache invalidation scheme which is used for freeing space from non-validate data. It reduces the latency for accessing data by the user and also reduces the number of data item in the client cache. As percentage of cache updation is reduced, performance of the device improved. We are demonstrating the feasibility of the proposed approach with experiments using simulation.

Keywords – Invalidation; Replacement; Prefetching

I. INTRODUCTION

The rapidly expanding demand for digital mobile communication services, in conjunction with the recent proliferation of portable computers, has led to the development efforts for future mobile systems directed towards Mobile Computing - a new dimension and requirements for future communication and computing networks. The mobile applications are aware of the user context (time, location, weather, temperature, surrounding noise etc). There are many popular context-aware tourist guide applications at present [1].

Whereas some dealing with locations. Mobile systems still suffer from scarce bandwidth, low quality communication and frequent network disconnections. All these factors lead to high delays before satisfying user's queries. But this delay will not occur if the answer is already in the client's cache. However, in location-dependent systems, where the answer of the same query changes if only the user's position is different, and where users rarely return to the same place. But, if useful information is transferred to the client before the user requests it, the problem of latency will be resolved.

Here presents a prefetching technique generally adapted for location-dependent systems for managing an important amount of data. We use the user's location both as a prediction criterion and as a cache invalidation one.

Here data changes with user location, direction, available networks and data access history. The user interest is used for data prefetching, selection of most suitable area and optimal utilization of the device's storage capacity and bandwidth put forth a good prefetching technique.

II. RELATED WORK

There are three important issues involved in client cache management:

1. A cache prefetching policy automatically preloads data items in to the cache for possible future access requests,
2. A cache replacement policy determines which data item(s) should be deleted from the cache when the free space is insufficient for accommodating an item to be cached, and
3. A cache invalidation scheme maintains data consistency between the client cache and the server.

Some of the papers describing these methods are as follows:

A new cache replacement policy called Predicted Region Based Replacement Policy (PRRP) for location dependent data in mobile environment propose in an "A New Cache Replacement Policy for Location Dependent Data in Mobile Environment" by Ajey Kumar, Manoj Misra and A. K. Sarje (2006)[2]. Unlike earlier cache replacement policies that consider only directional/non-directional data distance, PRRP takes into account data distance integrated with predicted region of client's movement that adapts to client's movement nature. It also considers the size of data item in cache. The problem of latency is very important for local queries whereas for non-local queries user movement for a short time does not invalidate the answer.

The system architecture and detailed algorithms describing the tasks executed in the client and the server as suggested in "Prototyping a Prefetching Scheme for Location-Dependent Systems" by Karim Zerioh and Robert Laurini(2006)[6]. They also discuss some additional steps that can be useful for saving energy consumption in the mobile device.

The prediction of the possible and the most relevant way how users will request information is takes into account in "Data Prefetching Algorithm in Mobile Environments" by El Garouani Said, El Beqqali Omar and Laurini Robert(2009) [5],. This approach is considered to deal with these problems and to improve response time and reduce the amount of data to mobile devices users.

Hoarding policy particularly adapted for location-dependent information is propose in "Spatial Hoarding: A hoarding strategy for location-dependent" by K. Zerioh, O. El Beqqali, and R. Laurini (2004) [4]. A system manages a huge amount of multimedia information and where no assumptions can be made about the future user's location. They use the user's position as a criterion for both hoarding and cache invalidation. This hoarding mechanism improves the cache hit ratio, thus reduces the uplink requests, and reduces the query latency.

Using additional updated invalidation reports (UIR) to improve query delay is suggests in, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," by G. Cao (2003) [3]. In IR + UIR the server broadcasts a number of UIRs between successive IR. Each UIR only contains information about the most recently updated data since the last IR.

III. A CATEGORIZED CACHE PREFETCHING POLICY:

In this research, the focus is on developing more efficient caching and prefetching techniques in mobile environment. In particular, these mechanisms have been investigated in connection with management of cache in many embedded devices, and even more general ad hoc networks.

We formulate the requirements of a good prefetching technique for location-dependent systems and mobile environments as follows:

- User interested Information related to the user's current area must be prefetched first,
- Prefetch only information related to user interest from its neighborhoods area for not wasting bandwidth and the mobile device resources,
- Closely related data must be grouped together and from a different category (and this categorized data sent together to the client, because once the user accesses a data item he will need to access also the data closely related to it),
- User direction and speed is also taken into consider (if user moving on fast vehicle then only high access probability data will be prefetched).
- A cache invalidation scheme used for freeing space from non-validate data where there is need for more relevant data,

Here divide geographical area dynamically into squares of unit length and makes the prefetching decision after user will travel ¾ of length of square by considering the user's direction and restricts the prefetched geographical area. This scheme the cache invalidation criterion is executed when there is need to store prefetch data. Here an access probability table also maintain where each data item is associated with the average probability that it will be requested. Accordingly it divides into the hot data and cold data (first 40% data consider as hot data and remaining as a cold data). Only a hot data items are prefetch for the purpose of not wasting bandwidth and the user's device resources when user's moving speed is fast. The prefetching decision is made far before the user is outside the prefetched region. In this policy, we assume that we have no knowledge about future user's movement. Even the user moves randomly, this prefetching policy can perfectly adapt itself to its movement.

IV. ARCHITECTURE FOR PROPOSED SYSTEM

The central server issued to model a service site for centralized data. In addition to a central information database, prefetching manager is responsible for proactively sending prefetch data items toward the clients in response to the notification of prefetching request. Since the downlink bandwidth is usually much larger and cheaper than uplink connection, pushing techniques result into efficient and valuable tools.

The local server is the data manager and wireless information server for a single cell. Each cell is assumed to have a unique local server which provides wireless access for all the clients in its cell and acts as a bridge between the central server and the client devices at the same time. This is an important factor since neighboring local servers must work closely together to provide efficient location-based services. The local data managers responsible for maintaining the information received from the central server as well as other local servers. Fig. 1 depicted general system architectural design consisting of the central server, local server, and the client device.

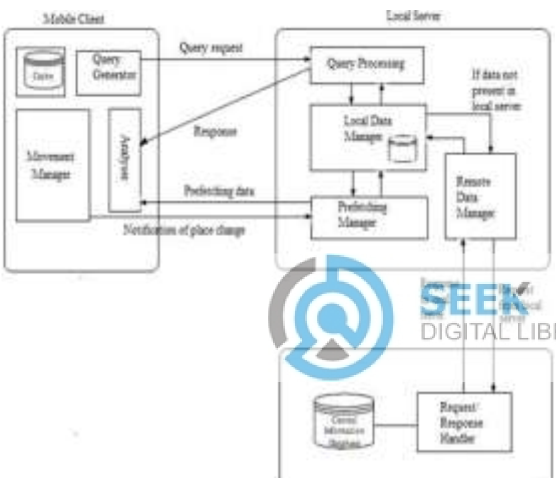

Figure 1. Architecture for Proposed System.

The mobile client is any end user device that is capable of wireless communication as well as user interface services. A client can change its position at will, in and out of a cell, from one cell to another, without notifying any server in advance. Since a client can issue a query at any time anywhere, this is especially challenging for information service providers. In our architectural design, a client always sends requests to the local server of the cell where the client resides. The local data manager checks whether the data is present at the local server or not if present then the local server can answer the query without consulting other servers. If the target information resides in other cells or in a central server, the local server must request the desired information from other servers and pass them to the client. Since the client can move at will, a target that can be accessed from within the same cell may have to be accessed from other cell for the next request. Static query processing strategies have little use in such environments. Dynamic data management strategies that can effectively locate the desired data as well as quickly respond to the location changes are in order.

V. ALGORITHMS

The algorithm given below describes the steps executed by the server.

A. Algorithm in the Server Side

```
if (prefetching_request) then
  stop previous prefetching thread;
  select_region ();
  prefetching ();
else then //Message is a question
  search_prefetch data list;
// which is the list of Id's of sending data
    if (found) then
    exit;
    else then //Search in local database
      if (found) then
        send response to the client;
      else then
        send request towards central server;
          central server send requested data towards local
server which further sends towards client;
      end if
    end if
end if
```

If prefetching request is received at local server, the local server stops the previous prefetching thread for sending the new requested data immediately to the client. Otherwise if a query request is received, then before answering a query, the list is checked. A list of already prefetched information (Id's of sending data) is maintained by the Prefetching Manager. If data found in this list then server does nothing otherwise it first searches in local database and then in central database. The cache invalidation algorithm is also executed with prefetching request for freeing space of the cache. Then, data invalidated by the cache invalidation criterion are discarded from the list of prefetched elements. Since the client is responsible for its cache invalidation policy there is no need to execute the cache invalidation criterion in the server. Another important advantage of this list is not to resend already prefetched information, related to squares not totally prefetched due to a new prefetching request, which occurred while the previous prefetching process was still active. In the case where the user request for new prefetching data and the server is still prefetching data, a new set is arranged based on the list of prefetched elements, the already sent elements are discarded from the set of elements to prefetch. At this stage the server can begin sending prefetched data.

B. Algorithm in the Client Side

The algorithm given below describes the steps executed by the client

```
if (message received from server) then
  if (prefetch data) then
    if (it answer one of the pending query) then
      display and cache data;
    else then
      cache data;
    end if
  else then //response to previous query
    display and cache data;
  end if
else if (query occurs) then
  search in local cache;
  if (found) then
    display requested data;
  else then
    send request towards local server;
  end if
else if (traveled distance reach threshold)
```

```
    // request for prefetching relevant data
notification sends towards local server
    // call for select_region() and prefetching();
invalidation ();
end if
```

The client informs the local server of its profile. When message is received from server, then client checks whether it is a prefetching data or response to one of its previous pending query. If it is an answer to one of the pending query then display information and cached the data.

The invalidation criteria of the cache were carried out to drop the data of the old regions. The user location and the direction of the device are the most important criteria for the cache invalidation.

For answering queries, the client looks for the answer in the cache before sending it to the local server. When a message is received from the server, it is first determined if either the message is a response or a previously sent query or a prefetched data from the server. In the later case, the client checks if the data can answer one of its pending queries before caching this data. In this way, the client will not have to wait longer time for the answer from the server if this prefetched element answers a pending query. At the same time the server will not have to re-send a previously sent element as noted before.

The client device traces the travelled distance since the last prefetching request, when it reaches to the threshold it sends the notification for requesting prefetching data to the local server and then invalidation criteria of the cache were carried out.

V. SIMULATIONS AND ANALYSIS

At first, the development of an approach consists of storing information in database. We simulated this model by implementing it in java, to compare the performance of our approach with previous prefetching algorithms. In this prototype, we are showing a simulation that allows potential clients (e. g. tourist) to receive the data they need in the future movement, by the server according to their positions, device storage space, speed and direction. In this prototype, the potential clients are equipped with GPS enabled mobile devices with network connectivity. The shared data is stored on the server in the form of tables in database with its location (latitude and longitude).

In this system geographical area is dynamically divided into the small squares of equal length considering user location as a center of the side of a square. After traveling ¾th of the length of the side of square (consider as threshold) the device will send prefetching request and division of geographical area will be calculated again and it will fetch the data of its current square and three next square in the direction of user. Fig. 2 is showing the area considered for prefetching data. If user is in 0th area then data for the area having number 1, 2 and 3 are also fetched.

Figure 2. Data in the cache of the client following his direction.

With prefetching request, the user informs the server of its profile. The server sends relevant data to the client of the squares where he could be in its next displacement.

These simulations are performed for testing the efficiency of running location based application.

At the start of application user must have given his/her own information (like user interest) by completing the user profile form. User can ask about any information they want but the criteria used for prefetching is different for different user, based on their profile.

VI. RESULT of COMPARISON

The proposed method compared to the other two methods Directed Prefetching (DP) and Spatial Prefetching (SP); gives competitive results for the rate of success, compared to volume of data stored in user cache. It decreases the mass of data in cache. The difference between the three techniques is in the evolution of the number of items pre-fetched into the cache of mobile device of a client. So the uplink and downlink bandwidths are saved for fetching fewer amounts of data.

Fig. 3 shows the comparison between Directed Prefetching, Spatial Prefetching and our Categorized Prefetching. Results show that the Categorized Prefetching client cache contains less number of data items than Directed Prefetching and Spatial Prefetching. Also the cache update time is less in Categorized Prefetching.

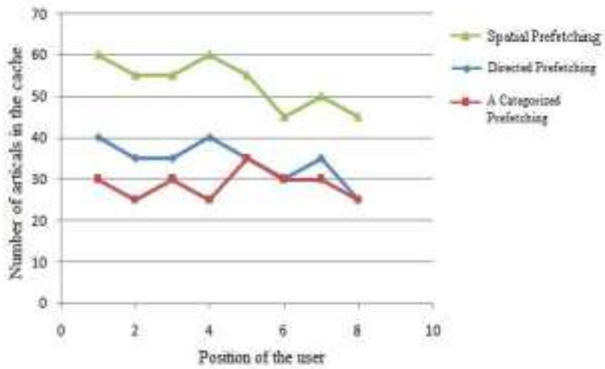

Figure 3. The number of items in the cache of client over its location.

Fig. 4 shows that Categorized Prefetching has less cache update time for user movement compare to the Directed Prefetching. From the above results and comparisons we also conclude that with the increase in performance of the device the battery life is also increases.

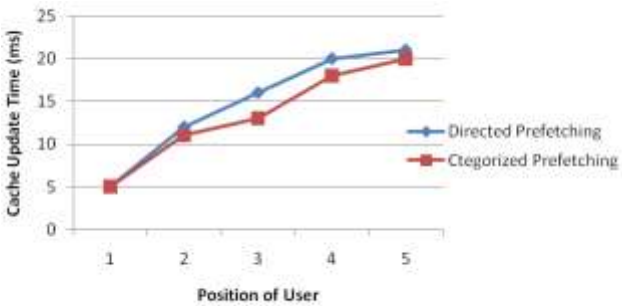

Figure 4. The cache Update Over the location of user

VII. CONCLUSION

For reducing the data in the cache and not wasting bandwidth and the mobile device resources, only user interested data must be prefetch and area selected for prefetching must be calculated dynamically. A cache invalidation scheme must be used for freeing space by removing non-relevant data and prefetching more bandwidth are saved for fetching fewer amounts of data. As the user interested data already in the user cache reduces the latency for accessing data.

REFERENCES

[1] “A Generic Framework for Developing Map-Based Mobile Application” by Haslinda Harun, Norleyza Jailani, Marini Abu Bakar, Mohamad Shanudin Zakaria, Salha Abdullah at International Conference on Electrical Engineering and Informatics 5-7 August 2009, Selangor, Malaysia.

[2] “A New Cache Replacement Policy for Location Dependent Data in Mobile Environment” by Ajey Kumar, Manoj Misra and A. K. Sarje 2006.

[3] G. Cao, “A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments,” IEEE Trans. Knowledge and Data Eng., vol. 15, no. 5, Sep 03.

[4] K. Zerioh, O. El Beqqali, and R. Laurini, “Spatial Hoarding: A hoarding strategy for location-dependent Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC’06) systems”, proceedings of the 11th international symposium on spatial data handling, Leicester, UK, 2004, pp. 217-230.

[5] “Data Prefetching Algorithm in Mobile Environments” by El Garouani Said, El Beqqali Omar and Laurini Robert publish in European Journal of Scientific Research Vol.28 No.3 having ISSN 1450-216X (2009), pp.478-491.

[6] “Prototyping a Prefetching Scheme for Location-Dependent Systems” by Karim Zerioh and Robert Laurini publish in proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (2006).

[7] Kristian Kvilekval and Ambuj Singh, “SPREE: Object Prefetching for Mobile computers” (2004) Distributed Objects and Applications (DOA).

[8] U. Kubach, and K. Rothermel, “Exploiting location information for infostation-based hoarding”, Proceedings of the 7th international conference on mobile computing and networking, 2001, pp. 15-27.

[9] V.P. De Nitto, V. Grasi, and A. Morlupi, “Modelling and evaluation of prefetching policies for context-aware information services”, Proceedings of the 4th international conference on mobile computing and networking, Dallas, USA, 1998, pp. 55-64.