Data Management and Query Processing in Mobile Environments

Whei Yuan Lee Arbee L.P. Chen
Department of computer Science
National Tsing Hua University
Hsinchu, Taiwan, ROC
Email: alpchen@cs.nthu.edu.tw

Abstract

Mobile computing is a new emerging computing paradigm posing many challenging data management problems. It refers to the environment where tens of millions of users will use various palmtop and laptop computers, connected by wireless connections to the fixed network. In this paper, we focus on deriving the condition of data replication. The main goal of data replication is to reduce the network traffic generated by massive queries and to speed up the query processing. We try to make it possible to get the needed data from the local agents to process the queries. Further, we present eight query processing strategies, simulate their behaviors according to five query types and other factors of the wireless environment, and discuss the simulation results.

1 Introduction

Because of the progressing of computer techniques which improve the computation ability and speed, more and more people rely on computers to do their jobs. Due to the reduced size of computers and the rapidly expanding technology of wireless communications, mobile computing has emerged as a new computing paradigm. The resulting computing environment no longer requires a user to maintain a fixed position in the network, and enables almost unrestricted user mobility. Mobility and portability will create an entire new class of applications.

To provide powerful functions for mobile computing, we have to let users(called *mobile hosts*, MHs) to query information without facing the location problem. This involves data acquirement and transition problems [1][4]. In the wireless environment, MHs may want to query other MHs' data, location, or other local information. Typical queries from MHs range from simple requests such as " Where is John", "Find a doctor near me", "How about the weather here", to more complex "Find an obstetrician between John and Helena" or "Find me the best route to the restaurant". To process these queries, it must involve a large number of database accessing [6]. Consider an example query "Find a doctor near me". We may check the MHs' occupations, who stay in the same agent as the user who poses the query. Notice that, there may be MHs who come from the other agents such that we can not check their occupations in the local database. We need to send messages to these agents to get the MHs' occupations, that is, a heavy traffic load due to the massive data transfers will be generated by queries. In order to reduce network traffic load, we do partial data replication, that is, we cache some of the MHs' data in a local database. Under this data replication environment, we propose an efficient way to process various types of queries.

This paper is organized as follows: Section 2 presents an overview of the system environment adopted in this paper. In Section 3, we present the condition of data replication. Section 4 presents eight query processing strategies and also shows the simulation results according to different factors in the wireless environment and five query types. In Section 5, we provide conclusions and the future work.

2 System Environment

We investigate the problems based on the Internet Engineering Task Force(IETF) proposal[7][8][9]. In the following, we present an overview of how IETF supports the wireless network. In the IETF proposal, each MH belongs to a *home network* and is served by a *home agent*. The home agent is responsible for maintaining the current location information of the MH. When a MH moves to another network (called *foreign network* to the MH), that is, moves out of its home network, it will register to the agent of that network, and the agent becomes its *foreign agent*. The *binding* which is used to record the current location of the MHs is stored in the location directory of the form <IP, care-of-address>. When a MH moves to a foreign network, it can either acquire a temporary IP address (changes identity) or register to the foreign agent (maintains identity). The care-of-address is the temporary IP address or the IP address of the foreign agent. If there is a packet to be sent to the MH, it is initially routed to the home agent of the MH and the binding at the home agent is then used to route the packet to the care-of-address of the MH.

The registration and deregistration methods used in this paper is based on [2] which is an extension of the basic IETF method. When a MH moves from a foreign network A to

another foreign network B, agent B must inform the home agent and agent A to let them know where the MH is. The registration/deregistration protocol ensures that a MH's home agent always knows about the binding of the MHs it serves. There are two choices agent A can make. One is simply removing the MH's binding. The other is updating the MH's binding. In this case, agent A then becomes the MH's *cache agent*. We take the second method, that is, we keep the location information about the MH in agent A. This mechanism allows the previous foreign agents to directly forward packets to the MH's new location. We denote the MH to the home agent as a *resident*, to a foreign agent as a *visitor*, and to a cache agent as a *passer*.

In order to provide a more powerful query processing environment, for each agent we add two additional databases, one is Home Database (HD) which records the data of the residents. The other is Visitor Database (VD) which records a part of the data of the visitors or passers. Moreover, we divide the location directory into two parts. One is Home Location Directory(HLD) which records the binding of the residents. The other is Visitor Location Directory(VLD) which records the bindings of the visitors or passers. Therefore, there are four databases in each agent, two are MHs' databases, and the others are location databases. In HD, we store the complete data about the residents, such as name, occupation, age, address and blood type. However, in VD, we only replicate the data about the visitors or passers, which are often queried. The existence of the VD helps to process queries more efficiently. On the contrary, the excessive replication will cause an unexpected traffic load of the wireless network.

3 Data Replication

To improve the system performance and service quality, we need to replicate the often queried data of a MH in the VD of the agent to which the MH is a visitor or passer. In the following, we discuss the data replication condition under the following assumptions :

1.The size of a data packet is 1, and the size of a control packet is ω. We assume the size of a control packet is always smaller or equal to the size of a data packet, that is, $0<\omega<=1$.

2.The MHs' databases are not updated.

3.The packet will be reliably transmitted.

4.The query is restricted to finding the data of a particular MH..

Also, we define the parameters which will be used in our discussion:

β : the number that a MH moves out of an agent in a time unit.

γ : the number that a data item is queried by a user in a time unit.

Let C_n be the registration cost for a MH plus the query cost for accessing the data item of the MH in a time unit, where the data are not replicated when the MH moves out of the home agent; and C_r be the same cost, where the data are replicated when the MH moves out of the agent.

C_n = registration cost + query cost

registration cost = $4\beta\omega$ (see Figure 1) for four control packets, one informing the home agent, one informing the previous foreign agent and the other two ack messages.

query cost = $2\gamma\omega + \gamma*1$ (see Figure 2 and Figure 3) for two control packets, one query packet to the home agent and the other ack message from the querying agent to the home agent, and a data packet from the home agent to the querying agent.

We get $C_n = 4\beta\omega + (2\omega + 1)*\gamma$.

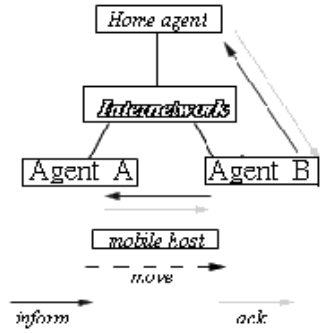


Figure 1 : A MH moves to another network without a replicate.

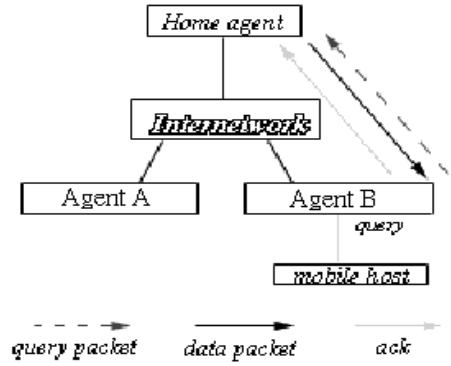


Figure 2 : A data item is queried by a user in an agent with a replicate.

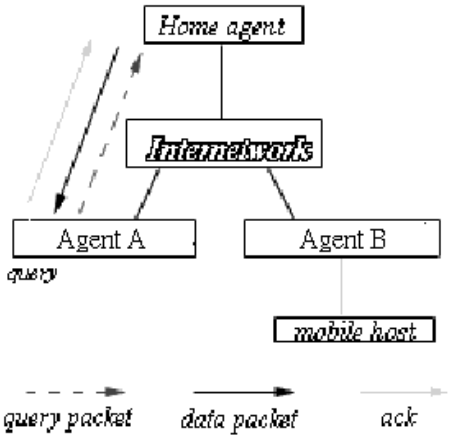


Figure 3 : A data item is queried by a user in an agent without a replicate.

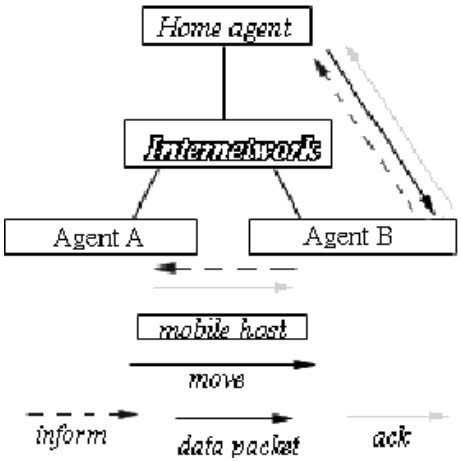


Figure 4 : A MH moves to another network with data replication.

C_r = registration cost + query cost

registration cost = $4\beta\omega + \beta * 1$ (see Figure 4) for four control packets, one informing the home agent, one informing the previous agent and the other two ack messages, and a data packet from the home agent to the foreign agent.

For γ (the number that a data item is queried by a user in a time unit), we can divide it into two parts:

δ : the number that a data item is queried by a user in the agent with a replicate in a time unit.

θ : the number that a data item is queried by a user in the agent without a replicate in a time unit.

query cost = $2\theta\omega + \theta * 1$(see Figure 3) Notice that we need to send a data request only when the data item is queried by a user in the different region, for two control packets one query packet to the home agent and the other ack message from the querying agent to the home agent and a data packet from the home agent to the querying agent.

We get $C_r = \beta(4\omega + 1) + \theta(2\omega + 1)$.

When a data item satisfies the condition "$C_n > C_r$", we replicate it. Simplify this condition, we get

$$\delta > \beta/(2\omega+1).$$

Therefore, when the number that a data item is queried by a user in the same agent in a time unit is greater than $\beta/(2\omega + 1)$, this data item will be replicated when the corresponding MH moves out of an agent.

We can gain from data replication not only the reduction of the execution time, but also the traffic load needed for transmitting the packets when processing a query.

4 Query Processing

If the users pose a query " Find a doctor near me", how can we process this query? Which query processing strategies are more efficient? In this section, we present eight query processing strategies and use a simulation to evaluate their performance according to five query types. In the following, we discuss the query processing strategies, the simulation environment and the simulation results in detail.

4.1 Query Processing Strategies

Before discussing the query processing strategies, there are several concepts which should be noticed. For an entry in VLD, it can be the visitor's binding or passer's binding. Similarly, an entry in VD can be a visitor's data or passer's data. That is, if we find some MH in VD, it does not mean this MH exists in this network, and we should check VLD to determine where the MH is. The query predicate can be divided into two parts. One is *location predicate* which specifies location conditions; the other is *non-location* predicate which specifies other conditions. For the attributes which appear in the *non-location predicate*, we call them *non-location attributes*. Because we only replicate part of the data, some non-location attributes may not be found in VD. Moreover, we denote the HD, HLD, VD and VLD at the query site (the network where a user poses a query) as local HD, local HLD, local VD and local VLD, and the others as remote HD, remote HLD, remote VD and remote VLD. The *nearby agents* are the agents which are not the querying agents but satisfy the location predicate.

The main difference of the query processing strategies lies in the starting local database(VLD, VD, HLD, or HD) to search. Consider the case that starts from searching local VD. We can search VD to find the MHs who satisfy the non-location predicate and then check VLD to see whether these MHs are visitors. If there is any attributes not replicated in VD, we must send the query to the home agents of the MHs selected from the previous step and get an answer from these home agents. If there is no result produced, we can continue the query processing by searching HD or HLD. These two strategies are named strategy1 and strategy2, respectively. We can also start from searching HD to find the MHs who satisfy the non-location predicate and then check the HLD to

determine whether these selected MHs satisfy the location predicate. If no result produced, we continue our query processing by searching VD and checking VLD, or searching VLD and checking VD. These two strategies are named strategy3 and strategy4, respectively.

Further, we consider the strategies that search from the location directory (either VLD or HLD) and then check the database (either VD or HD). We can start by searching VLD to eliminate the MHs who have moved out of this network and can not satisfy the location predicate, then evaluate the non-location predicate of the MHs who satisfy the location predicate at VD if the needed data are replicated. If there is no result produced, we continue to search HD or HLD. These two strategies are named strategy5 and strategy6, respectively. Finally, we can start to search from HLD then HD, and then VD or VLD. These two strategies are named strategy7 and strategy8, respectively. Due to space limit, we only show strategy1 in detail in Flowchart 1. Table 1 summarizes these eight query processing strategies.

In our query processing strategies, we stop processing when there is an answer returned.

4.2 Simulation Environment

4.2.1 User databases and query types

We assume that there are two tables in HD as shown in Figure 5. The tables in HD store the information about the residents and their cars. Two attributes, Name and Occupation in table "*Residents*" and two attributes Owner and Factory in table "*Cars*" are replicated in VD. Therefore, if any of these attributes is a non-location attribute, we can check its value in VD.

In the following, we classify five types of queries according to the schema described above. Moreover, we assume that in the query predicate, the word "near" means in the querying agent and there is always a satisfied MH for easier simulation.

Query type 1 : Simple query, containing a single non-location attribute which is replicated in VD.
Ex : Find a doctor near me.

Query type 2 : Simple query, containing a single non-location attribute which is not replicated in VD.
Ex : Find a person near me whose hobby is tennis.

Query type 3 : Complex query, tables are needed to be joined to process the query and the data which need to be joined are replicated in VD.
Ex : Find a doctor near me, who owns a car whose factory is VOLVO.

Query type 4 : Complex query, tables are needed to be joined to process the query, only part of the data which need to be joined are replicated in VD.
Ex : Find a doctor near me, who purchased a car in 1997.

Query type 5: Complex query, tables are needed to be joined to process the query and the data are not replicated in VD.
Ex : Find a person near me, whose age is below 30 and who purchased a car in 1997.

4.2.2 Cost Model

We assume the time of scanning the entire Residents table in HD is 1. Therefore, the time needed to scan other tables is the ratio between the tuples in these tables and that in the Residents table in HD. Also, the cost of getting one tuple in the table is equal to the cost of scanning the entire table divided by the number of tuples. The cost of joining two tables is the multiplication of the scanning cost of one table and the number of the tuples of the other table. The connection cost between two agents is modeled as the ratio of the cost for scanning the Residents table in HD. Moreover, we assume that the query can be executed in parallel at remote agents, which reduces the processing time.

4.3 Simulation Results

The values of the parameters defined above are assumed as follows :

- The number of tuples of the replicated Residents table in VD : 150.
- The number of tuples of the Residents table in HD : 100.
- Move-to-Query Ratio(the ratio between the number of MH moves to the number of queries posed) : 3.
- Motion probability (the probability for each MH to move out of an agent): 0.4.

A one-to-one correspondence between VD and VLD, and between HD and HLD is also assumed.

4.3.1 Different query types

- Goal : Simulate how the different query types affect the eight query processing strategies.
- Result : Shown in Figure 6.
- Discussion :

1. For query type1, start query processing by searching HD or HLD is better than VD and VLD. This is because the number of tuples of the Residents table in HD is smaller than that in VD and so does the number of entries in HLD and HD. The cost for scanning the HD (HLD) is smaller than VD(VLD).
2. For query type2, starting query processing by searching HD or HLD is better than VLD(the non-location attribute is not replicated in VD, so starting query processing by searching VD is the same as searching VLD). Moreover, searching VLD involves setting connections to remote agents. The performance is worse compared to searching HD or HLD.
3. For query type3, starting query processing by searching HLD or VLD is a good policy. This is because we can eliminate some unsatisfied MHs before processing

the join to reduce the time of table scanning. Moreover, the non-location attributes are all replicated in VD. No connections need to be set when processing this query type.

4. For query type4, starting query processing by searching HLD is a good policy. Searching VLD may also reduce the time of the table scanning. However, the non-location attributes are only partially replicated in VD. If we start query processing by searching VLD, we have to set connections to the remote agents to evaluate the non-location attributes not replicated in VD. The detailed discussion of the effect of connection costs will be presented in the following.

5. For query type5, starting query processing by searching HLD is the best policy. The reason is the same as that for query type4.

4.3.2 Different Connection Costs

- Goal : Simulate different query processing strategies according to different connection costs.
- Result : We only need to set connections when processing query type2, type4 and type5. Since they all have the same effect, we only present the results for query type4 in Figure 7.
- Discussion : For query type4, the higher the connection cost is, the more the time is needed to process a query. There is also one thing which should be noticed. Consider the case that the connection cost is low(say, 0.5), and compare the results of query type3 shown in Figure 6. If we have replicated all the non-location attributes in VD and process the query locally, it may not be as good as the partial replication situation. This is because for partial replication, we can first search VLD and VD to eliminate unsatisfied MHs, setting connections to remote agents and process the query parallel. However, for a high connection cost, it is not worthy processing the query remotely.

4.3.3 Summary

To summarize the above observations, we get the following conclusions. If there is a join operation in a query, it is better to search the HLD to prune unsatisfied MHs to decrease the join cost. Moreover, searching from HD or HLD often performs better than searching from VD or VLD. Also, data replication can speed up the query processing. However, for a low connection cost, if there is a join operation in the query, processing the query at remote agents may perform better than processing query locally. This is because we can process queries in parallel and the needed connection cost is low.

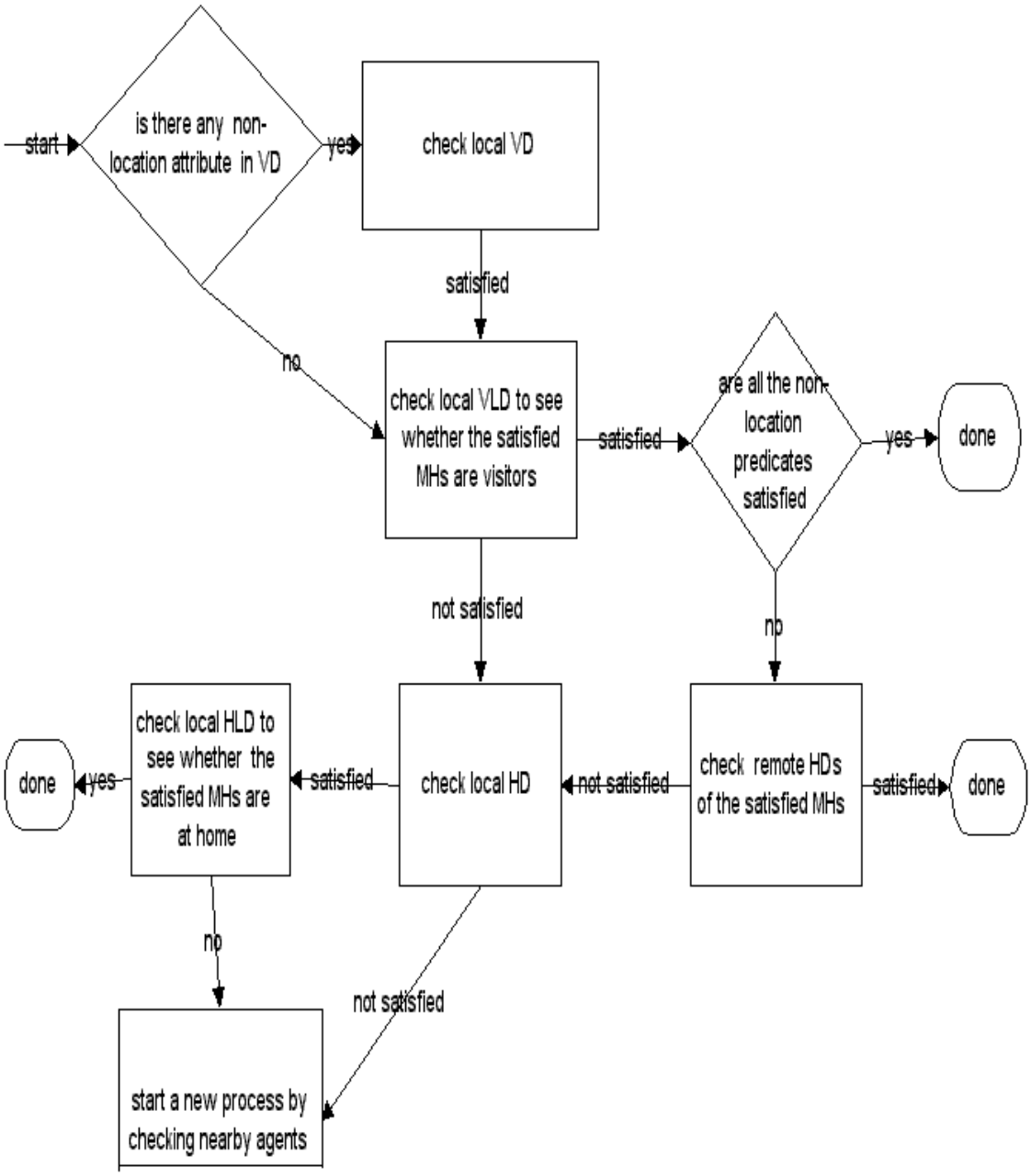
5 Conclusion and Future work

In this paper, we consider the condition of data replication which can reduce the traffic load and speed up the query processing. Moreover, we present query processing strategies, simulate their performance according to different query types and different connection costs, discuss simulation results, and conclude with efficient strategies according to different conditions.

Our future work include considering data updates when we derive the condition of data replication. For the simulation, we need to allow the MHs to move freely with different probabilities. The move-to-query ratio may affect the cost of different query processing strategies. Moreover, in order to reduce the registration and deregistration times, some proper grouping of agents is needed. That is, agents should be grouped into regions to reduce the times of the MHs moving out of a region. We will analyze the MHs' motion patterns to decide a proper grouping for a better performance.

Reference

1. Tomasz Imielinski and Henry F. Korth, "Location Management for Networks With Mobile Users." Mobile Computing p.129--p.150.
2. Charles E. Perkins and Andrew Myles, "The Internet MH Protocol." Proceedings of INET '94.
3. Andrew Myles, "Mobile IP Extensions." Internet Draft.
4. Rafael Alonso and Henery F. Korth, "Database system Issues in Nomadic Computing." SIGMOD '93.
5. Yixiu Huang, Prasad Sistla and Ouri Wolfson, "Data Replication for Mobile Computers." SIGMOD '94.
6. Tomasz Imielinski and B. R. Badrinath, "Querying in Highly Mobile Distributed Environments." Proc. Of VLDB '92.
7. Fumio Teraoka, "Mobile IP : The VIP approach." IETF Presentation, Boston, July 1992.
8. C. Perkins, P. Bhagwat, "A mobile networking system based on internet protocol." IETF Personal Communications, Vol. 1, No. 1, pp32—41, January 1994.
9. S. Rajagopalan and B. R. Badrinath, "An adaptive location management strategy for Mobile IP." First Intl. Conf. on Mobile Computing and Networking, Nov. 1995.



Flowchart 1 : Query strategy 1.

	Step 1	Step 2	Step 3	Step 4
Strategy 1	VD	VLD	HD	HLD
Strategy 2	VD	VLD	HLD	HD
Strategy 3	HD	HLD	VD	VLD
Strategy 4	HD	HLD	VLD	VD
Strategy 5	VLD	VD	HD	HLD
Strategy 6	VLD	VD	HLD	HD
Strategy 7	HLD	HD	VD	VLD
Strategy 8	HLD	HD	VLD	VD

Table1 : The eight query processing strategies.

Residents

Name	Occupation	Hobby	Age
John	doctor	baseball	35
Mary	teacher	tennis	40
Hellen	doctor	tennis	33
Sue	nurse	football	24
Bob	engineer	basketball	27
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Cars

Owner	Car_id	Factory	Purchasing Date
John	1111	VOLVO	90/3/25
John	1222	BENZ	91/5/15
Mary	1333	BMW	89/2/3
Sue	1444	BMW	85/6/17
Bob	1555	CIVIC	96/1/8
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Figure 5 : Two tables in HD.

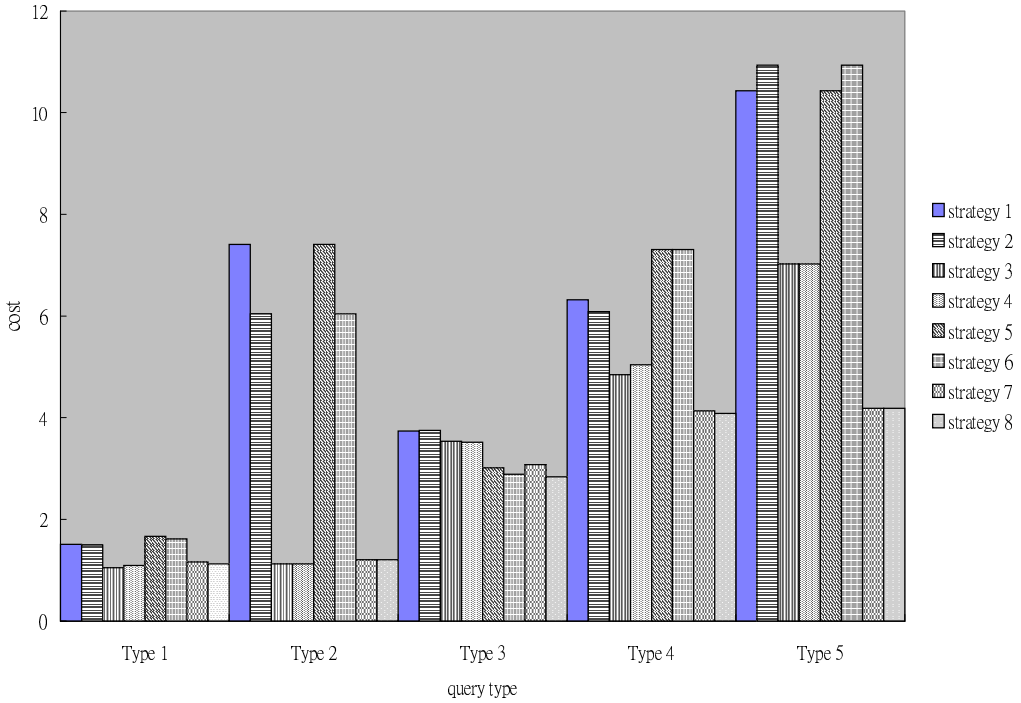


Figure 6 : Results of processing different query types by strategy1-strategy8.

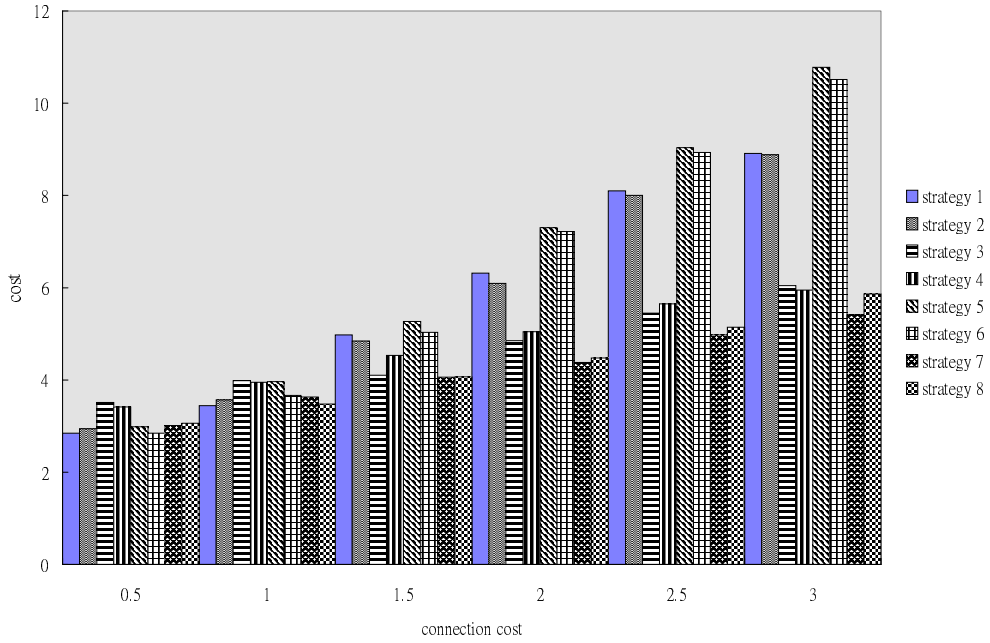


Figure 7 : Results of different connection costs, for processing query type4 by strategy1-strategy8. for query type4