Distributed Database Statistics Collection Using Mobile Agents

NUTLADA RATTANAVIJAI, SUPHAMIT CHITTAYASOTHORN
Department of Computer Engineering, Faculty of Engineering
King Mongkut's Institute of Technology, Ladkrabang
Bangkok 10520, THAILAND

Abstract: - Mobile agents are programs that can move and work on computers in computer networks without any program pre-installation. At present, mobile agents are employed by various networked applications. An application domain which is believed to be suitable for agent deployment is distributed database. In this paper, we use mobile agents for distributed database statistic collection. The collected database statistics will be used by the cost-based query optimizer of each site to determine the best physical access plan for a given database query. Statistics collection even though useful is time consuming and could slow down other database operations. Mobile agents are employed to find suitable time for statistic collection in order to reduce the interference.

Key-Words: - Mobile Agents, Distributed Database, Database Statistics, Query Optimization

1 Introduction

Database statistics collection is an important activity which ensures correct decision making by the cost-based optimizer. Statistics such as number of rows of a table, number of distinct values in a column, availability of indexes etc. are either collected manually by database administrators or collected automatically for each preset time interval. The major problem is the statistics collection activity is time and resource consuming and could interfere with normal users' database operations. Proper database statistics collection timing is required. The most common timing is to collect statistics at night where the system is assumed to have lighter loads.

Some attempts have been made by researchers. The Piggyback Method [1] collects database statistics each time a query is issued. The newly collected statistics, even though specific to a particular query, is used to update the main statistics. A major drawback of this method is the added overhead imposed on each database query. Another statistics collection method is the Automated Statistics Collection [2]. This method observes database statistics by checking query results and collects only out-of-date ones. However, this method only refers to centralized databases.

In a distributed database system that comprises several database sites, we introduce agents to perform database collection and accumulate these statistics for global query optimization. The agents use information from the system catalog of each site to determine if new database statistics need to be collected and choose to collect them when the system does not have heavy loads.

2 Mobile Agents

Mobile agents are programs that can move and work on computers in computer networks without any program pre-installation. At present, mobile agents are employed by various networked applications [3,4,5]. Mobile agents may record their status before moving to the next computer and can resume operations upon arrival [6,7].

2.1 Characteristics of Mobile Agents

Mobile agents are software that can work and move in the network. It can be programmed to work on specific tasks and can communicate with other agents in the network.

Each agent may have its own characteristics and independent of other agents. It can make its own decision in order to reach its objectives without human intervention. Human users only give initial values and watch the agent's operations.

Mobile agents can work on any computers in the network regardless of hardware and operating systems and there is no need to install the agent code on the machine. There only need to be a Java Virtual Machine (JVM) installed on it and the agents will work on the JVM platform. Thus, the agent approach to solve problems in a networked environment becomes popular.

2.2 Mobile Agent Lifecycle

A mobile agent can be in one of the following states.
- Creation: This is the state that occurs only once for each agent. An agent id is uniquely created and individually assigned to each agent.

- Starting: This is the starting state that takes place when the agent moves to a new host computer.
- Deactivation: This is the state that the agent stops all the works and saves its current status.
- Disposal: This is the state that the agent finishes all the work and returns it resources.

Fig. 1 shows the state diagram of mobile agent lifecycle.

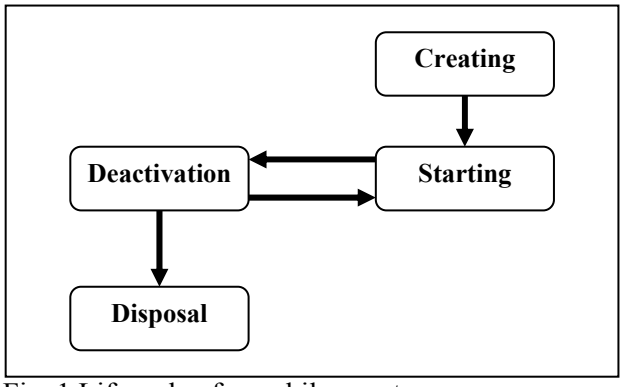

Fig. 1 Lifecycle of a mobile agent

2.3 Communications between Agents

Mobile agents can communicate with one another using one of the three techniques namely procedure call, callback and mailbox. The procedure call is a synchronous communication mode where agent A contacts agent B for a work assignment. While it waits for B to complete the task A cannot perform other tasks and has to wait for the result from B before it can resume operation**.**

The callback is an asynchronous mode where agent A contacts agent B for a work assignment and let B perform the task without waiting for B. After B finishes the task, it will send the result to A. The mailbox mode is similar to the callback except that when agent A wants B to perform a task it will write the request in B's mailbox. When B opens its mailbox and finds the request, it will perform the task and finally put the result into A's mailbox.

3 Distributed Database System

A typical distributed database system comprises several autonomous computers with an autonomous DBMS and databases connected together in a computer network. Such a system allows local applications that use only resources from only one site and global applications that may use resources from several sites. Hence, global query optimization becomes an interesting issue. Global query optimization that utilizes cost-based technology requires global database statistics from all related sites. A typical distributed database system is shown in Fig. 2.

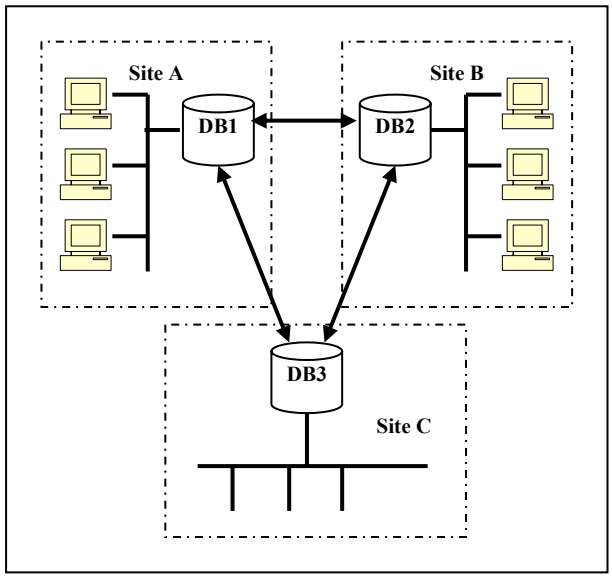

Fig.2 A Typical Distributed Database System

Global query processing in a distributed database system is performed by the query optimizer of the site where the request is originated (the coordinator). In principle, the global statistics should be available at all participating sites. However, this is not the case in practice and the coordinator normally has access only to its own database statistics. Global queries are decomposed in to several smaller ones and sent to the sites that have the data. Local query optimization is done at each site and results are sent back to the coordinator site for the final assembly.

4 Query Optimization

Query optimization is the process of transforming a user's logical query to the most efficient physical access path to the result. Nowadays most DBMSs employ one of the two popular query optimization techniques namely rule based (or heuristic based) and cost based optimizers.

The rule based optimizer is the older technology. Programmers are required to know the preference choice of the optimizer and write SQL statements accordingly to encourage the optimizer. The cost based optimizer requires almost no human choice of the access plan but requires database statistics for decision making.

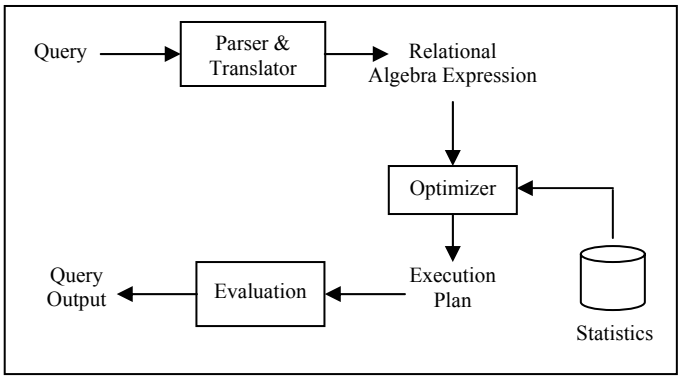

Fig. 3 Steps of cost based optimization

Fig. 3 shows steps of cost based query optimization. A user's logical query enters the system, has the syntax checked by the parser and then translated into relational algebra statements. The cost based optimizer then consults pre-collected database statistics to find the access path with minimum calculated cost. The final execution plan is executed by the evaluation engine. Some DBMSs keep the plan for later usage.

5 Database Statistics

Database statistics are collected and kept in the system catalog of the DBMS. It is used by the cost-based optimizer and at least has the following information:

- Table statistics
  - Number of rows
  - Number of blocks
  - Average row length
- Column statistics
  - Number of distinct values in column
  - Number of null in column
  - Data distribution (histogram)
- Index statistics
  - Number of leaf blocks
  - Levels
  - Clustering factor
- System statistic
  - I/O performance and utilization
  - CPU performance and utilization

Each cost based DBMS is supplied with a database statistics collector or database analyzer which can be manually activated by the database system administrator. However, there are problems that database administrators must be aware of. The problems are listed as follow:

- Heavy system load: During statistics collection, tables are scanned to find information about the databases thus disturb and slowdown other activities.
- Out-of-date Statistics: In the case that the database has many insert, delete and update activities, the database statistics may not reflect the correct status of the database. In this case, database statistics should be collected more often or the query optimizer may make incorrect decisions regarding the best physical access plan.
- Incomplete Statistics: Most DBMSs allow only parts of the database to have statistics collected to reduce the statistics collection overheads. However, when the database table that does not have statistics collected are use in the same query with those with statistics collected, most DBMSs will invoke the cost based optimizer and activate the statistics collection process for the table. This process will significantly slow down the query processing.
- High Cost for Analyzing Large Database: Since statistics collection is an expensive task, in the case that the majority of the database is unchanged, statistics should not be collected for those parts.
- Inconvenience for Users: All in all, the database statistics collection process should be automated for user's conveniences and to achieve up-to-date statistics with minimum interference with other users' activities.

6 Mobile Agents and Distributed Database Statistics

This research project presents the use of mobile agents for distributed database statistics collection. The mobility of the agents and the ability to work on any computers without program installation make them suitable to be employed in the networked environment. Three kinds of agents are adopted namely, the UDI agent, the QF agent and the scheduler agent.

6.1 The UDI Agent

This kind of agent performs the task of watching Update/Insert/Delete (UDI) operations on the database. It performs the following steps:

- The agent observes a system table that keeps the number of update, insert and delete operations on tables. This system table is maintained by the DBMS [9]. It also counts number of rows of each table.
- It calculates the percentage of changes made to each table by comparing the number of

activities and the number of rows. In the case that the percentage of change is grater than a margin value, the data which are related to the table are sent to another agent that performs statistics collection scheduling.

Table 1 shows the sample data which is collected and calculated by UDI agents.

6.2 The QF Agent

This kind of agent performs the task of watching the query optimizer cost estimation performance in order to suggest new statistics collection. For each query that enters the system, the query optimizer estimates the cost and keeps the estimated value in a system table [9]. The QF agent performs the task of submitting the queries to the DBMS at suitable time and compares the actual cost with the estimated one. In the case that the estimated value is different from the actual value by a wide margin, it is possible that the cost estimation process uses an out-of-date statistics and new statistics need to be collected. The QF agent then informs another agent that performs statistics collection scheduling.

6.3 The Scheduling Agent

This kind of agent performs statistics collection scheduling. It accepts input data from the UDI agents and QF agents and arranges scheduling priorities. It also checks database utilization and collects database statistics of tables based on information in the priority queues. There are five priority queues. They are as follow:

a) The useful queue: Tables in this queue have been modified by less than 50% according to the UDI agent report.
b) The need queue: Tables in this queue are those reported by the QF agents.
c) The pressing queue: Tables in this queue have been modified by more than 50% according to the UDI agent report.
d) The urgent queue: Tables in this queue have been modified by more than 50% according to the UDI agent report and also reported by the QF agents.
e) The critical queue: Tables in this queue have been modified but have never had statistics collected.

The critical queue is the highest priority queue.

7 Some Experimental Results

Two identical distributed database systems are set up and identical insert/delete/update operations are performed. The difference is one system employs mobile agents for statistics collection but the other one collects statistics based on preset time intervals. Twenty different queries are applied to both databases and the response time obtained from the agent-based one is shown to be smaller than the interval one. This is due to the fact that the optimizer has more up-to-date statistics than the interval one. Fig 4 shows the comparison results.

The second experiment demonstrates the case when queries are issued during the statistics collection time against the case of queries on databases that use mobile agents for statistics collection. From Fig. 5 the interval statistic collection starts when query number 4 starts. The difference between the response times is shown in wide margin.

TABLE_NAME	INSERTS	UPDATES	DELETES	SUM	NUM_ROWS	CHANGE_RATE
PART	5	2	0	7	20	35%
SUPPLIER	10	0	0	10	10	100%

Table 1 UDI-driven data

SQL	TABLE_NAME	COL_NAME	ESTIMATED	REAL
SELECT * FROM LINEITEM WHERE L_SHIPDATE<=’1998-05-27’	LINEITEM	L_SHIPDATE	17626	20899
SELECT * FROM CUSTOMER WHERE C_MKTSEGMENT=’BUILDING’	CUSTOMER	C_MKTSEGMENT	2941	2941

Table 2 shows the sample data which is collected by QF agents.

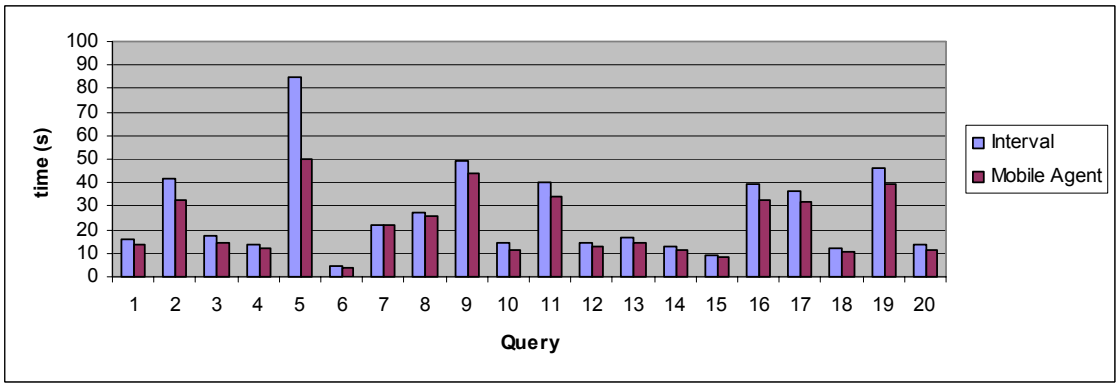

Fig. 4 The comparison between systems without and with mobile agents

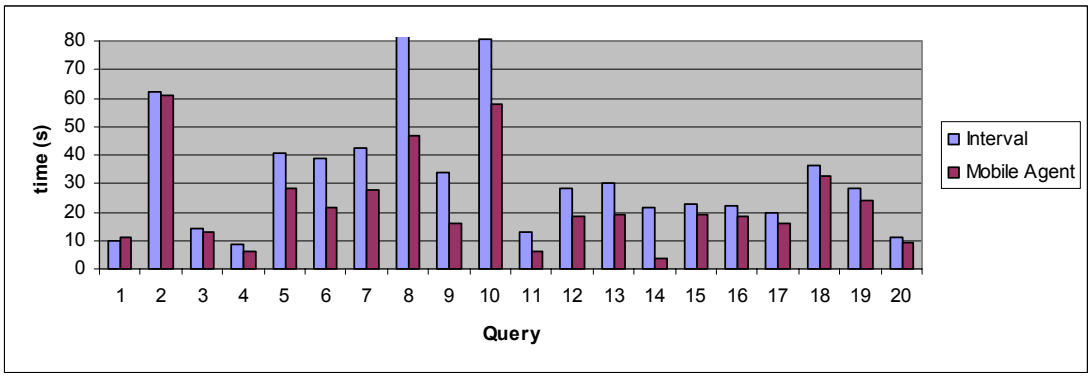

Fig. 5 The comparison between queries issued during statistics collection and the ones which have agents

8 Conclusion

This paper presents the application of mobile agents for statistics collection in distributed databases. Agents choose statistics collection timing according to the characteristics of individual tables and try to avoid interference with other users' activities. Agents observe changes made to tables and collect statistics accordingly. Experiments show convincing results.

Please, follow our instructions faithfully, otherwise you have to resubmit your full paper. This will enable us to maintain uniformity in the conference proceedings as well as in the post-conference luxurious books by WSES Press. The better you look, the better we all look. Thank you for your cooperation and contribution. We are looking forward to seeing you at the Conference.

References:

[1] Q. Zhu, B. Dunkel, N. Soparkar, S. Chen, B. Schiefer, T. Lai, "A piggyback method to collect statistics for query optimization in database management systems," *Proc. 1998 Conf. Center for Advanced Studies on Collaborative Research (CASCON '98)*, 1998.

[2] A. Aboulnaga, P. Haas, M. Kandil, S.Lightstone IBM Almaden Research Center, G. Lohman, V.Markl, I. Popivanov, V. Raman IBM Toronto Development Labe, "Automated Statistics Collection in DB2 UDB," *In Proceeding of 30th VLDB Conference, Toronto, Canada*, 2004.

[3] Brewington, B., Gray, R., Moizumi. K., Kotz, D., Cybenko, C., & Rus, D., "Mobile agents for distributed information retrieval," *Intelligent information agents, Springer-Verlag*, 1999, pp. 355- 395.

[4] Sahuguet, A., B. Pierce and V. Tannen, "Distributed Query Optimization: Can Mobile Agents Help?," Unpublished draft.

[5] Thandia Win, Khin Mar Lar Tun, "Mobile Agent Cooperation Methods in Hybrid Query Optimization," *In APSITT 2005 Proceedings. 6th Asia-Pacific Symposium on Information and Telecommunication Technologies*, 2005, pp 71-76.

[6] Alf Inge Wang and Carl-Fredrik Sørensen, "A Comparison of Two Different Java Technologies to Implement a Mobile Agent System," *In proceedings of the IASTED International*

Conference on Applied Informatics 2003 (AI'2003), 2003.

[7] Damir Horvat, Dragana Cvetkovic, Veljko Milutinovic, Petar Kocovic and Vlada Kovacevic, "Mobile Agents and Java Mobile Agents Toolkits," *In Proc. of the 33rd Hawaii International Conference on System Sciences* ,2000.

[8] C.J. Date, *An Introduction to Database Systems*, Addison-Wesley 1995.

[9] *Performance Tuning Guide*, http://download-west.oracle.com/docs/cd/B14117_01/server.101/b10752 /title.htm

Appendix A: Sample SQL queries that are used in the experiments

Query 1

```
select   l_returnflag,l_linestatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price,
   sum(l_extendedprice * (1 - l_discount)) as
   sum_disc_price,
   sum(l_extendedprice * (1 - l_discount) * (1 +
   l_tax)) as sum_charge,
   avg(l_quantity) as avg_qty,avg(l_extendedprice)
   as avg_price,
   avg(l_discount) as avg_disc,count(*) as
   count_order
from
   lineitem
where
   l_shipdate <= date '1998-12-01' - interval '96'
   day (3)
group by
   l_returnflag,l_linestatus
order by
   l_returnflag,l_linestatus;
```

Query 2

```
select   s_acctbal,s_name,n_name,p_partkey,
         p_mfgr,s_address,s_phone,s_comment
from     part,supplier,partsupp,nation,region
where
   p_partkey = ps_partkey
   and s_suppkey = ps_suppkey
   and p_size = 28
   and p_type like '%STEEL'
   and s_nationkey = n_nationkey
   and n_regionkey = r_regionkey
   and r_name = 'MIDDLE EAST'
   and ps_supplycost = (
         select   min(ps_supplycost)
         from     partsupp,supplier,nation,region
         where
                  p_partkey = ps_partkey
                  and s_suppkey = ps_suppkey
                  and s_nationkey = n_nationkey
                  and n_regionkey = r_regionkey
                  and r_name = 'MIDDLE EAST')
   and rownum <= 100
order by
         s_acctbal desc,  n_name,s_name,p_partkey;
```

Query 3

```
select   l_orderkey,sum(l_extendedprice * (1 -
   l_discount)) as revenue,
   o_orderdate,o_shippriority
from     customer,orders,lineitem
where
   c_mktsegment = ' BUILDING'
   and c_custkey = o_custkey
   and l_orderkey = o_orderkey
   and o_orderdate < date ' 1995-03-31'
   and l_shipdate > date ' 1995-03-31'
   and rownum <= 10
group by
   l_orderkey,o_orderdate,o_shippriority
order by
   revenue desc,o_orderdate;
```

Query 4

```
select   o_orderpriority,count(*) as order_count
from     orders
where
   o_orderdate >= date ' 1997-10-01'
   and o_orderdate < date ' 1997-10-01' + interval
   '3' month
   and exists (   select * from lineitem where
   l_orderkey = o_orderkey
                  and l_commitdate < l_receiptdate)
group by
   o_orderpriority
order by
   o_orderpriority;
```