This is the published version (version of record) of:


Available from Deakin Research Online:
http://hdl.handle.net/10536/DRO/DU:30004946

Reproduced with the kind permission of the copyright owner.

Copyright : 2003, World Scientific
Traditionally, accessing databases on the Internet is based on client/server paradigm (including web-based paradigm). However, there are some drawbacks when using the techniques. Mobile agent technology provides an excellent solution that can easily eliminate those disadvantages. By conducting a series of experiments and analyzing the experimental results, this paper presents in which circumstance mobile agents can outperform traditional client/server paradigm, and how they can overcome the shortcomings in client/server paradigm in database access on the Internet.

1. Introduction

Mobile agent (MA) technology becomes more and more popular these days. Mobile agents have been developed to assist people and act on their behalf. They provide an interesting approach and a new way of thinking for solving different kinds of problems in particular client/server computing. Because of the features of mobile agents, they are used for accessing databases on the Internet that can eliminate different disadvantages in the traditional paradigms such as client/server paradigm.

The use of Internet is growing rapidly every day. In addition, accessing databases on the Internet is one of the main tasks in many Internet applications. Currently the most popular way to access databases on the Internet is the client/server paradigm (including its special case--web-based paradigm). However, there are some drawbacks when using these methods.

In client/server paradigm, the transaction often fails if the network connection is unstable. Once the network connection is disconnected, the whole transaction will be failed and has to be restarted from the beginning. The second disadvantage is excessive requirement of bandwidth and latency. In client/server paradigm, a transaction between the client and the server may require many
round trips over the network, and each trip will create network traffic and consumes bandwidth and latency.

Limitation of server capacity is the main problem in web-based paradigm. When a massive amount of request signals have been sent to the server simultaneously, the server cannot handle the workload and is likely to crash, resulting in the URL of the website cannot be retrieved until system administrators fix the problems. For example, airline companies would like to increase the hits of the website and so they try to promote the website by offering special price in a limited number of air tickets that can only be booked online. For example, Qantas Airway used to provide special price for passengers flying between Melbourne and Sydney (only AS2 for such a return ticket!). People were then rushing to visit the website to book the air tickets within the same period of time because of seat limitation. Eventually the server crashed because too many people were attempting to visit the website at the same time.

Is there any technique that is easy to eliminate those disadvantages? Mobile agent technique is a very promising candidate.

Thus far, there are some successful applications of mobile agents in database accessing. Papastavrou, Samaras, and Pitoura 4 proposed a new framework called “DBMS-Aglet Framework” for web-based distributed access to database systems based on Java-based mobile agents. The idea was to use DBMS-applet for creating mobile agent that travels directly to the remote SQL server. Once the mobile agent arrives at the SQL server, it initiates a local JDBC driver, connects to the database and performs any queries specified by the sending client. When the mobile agent finishes its task, it dispatches itself back to the DBMS-applet in the client machine. These mobile agents were not equipped with capability to further process the retrieved data on the server side. This is the main difference between the mobile agents in [4] and the mobile agents described in this paper.

Brewington, Gray, and Moizumi et al. developed a distributed information-retrieval applications called “technical-report searcher” 5. The application uses mobile agent technology to retrieval information from different hosts across the network. The idea was to let the mobile agent decides whether to spawn child or only interacts with the individual document collections across the network.

The emphasis of this paper is to find out in which situation mobile agents should be used by conducting a series of experiments. We have found out from the experiment results that the execution environment of mobile agents does affect the performance of mobile agents. Therefore, it is important to test the performance of mobile agents under different execution environments. However,
there is no experiment being conducted for testing the performance of the mobile agent application under different execution environments in [5].

The rest of the paper is organized as follows. Section 2 describes the database access models using mobile agents for accessing databases on the Internet. Section 3 presents the testing conditions. The experimental results and evaluation are reported in Section 4. Finally, Section 5 is the concluding remarks.

2. Database Access Models Using Mobile Agents

There are two situations to retrieve information from databases at remote servers by using mobile agents: (1) Mobile agents without further processing capability. In this case, after executing the query and retrieving the results, the mobile agent will bring back all the retrieved results home. (2) Mobile agents with further processing capability. In this scenario, the mobile agent will process the retrieved data locally at the server side, and only bring back a small portion of retrieved data home. Two mobile agents were developed, one for each case respectively.

To construct mobile agents for database access on the Internet, the following five components are required: database server (The MySQL database server was chosen for experiments), mobile agent server (Tahiti mobile agent server was used in the experiment, which is part of the Aglet SDK), JDBC driver (It acts as a bridge that binds mobile agents and database together), user interface applet, and mobile agent.

IBM's Aglet Software Development Kit (Aglets SDK) was chosen to be the mobile agent development platform. Aglet is a mobile Java-based autonomous agent that is developed by using Aglets SDK platforms. Aglet can dispatch itself, travel itself to another host on the network, and resume execution at the destination host.

The models used in the two cases have some difference. For the first case—mobile agents for database access without further processing capability; the model is shown in Figure 1. In this model, when the client needs to retrieve information from the database at the remote server, the client needs to use the user interface applet to create a mobile agent at the local mobile agent server. After that, the user interface applet will send the mobile agent to the remote server. Once the mobile agent arrives at the remote server, it will initialize a local JDBC driver and uses it to connect the database, and then performs any queries. When the mobile agent finishes its task, it brings back the results and displays them through the user interface applet.

There are a few advantages using this model. Firstly, the mobile agent does not rely on network connection since mobile agents can perform normally even
the network connection is unstable. Secondly, the client does not need to download the JDBC driver in advance as the mobile agent can use the remote JDBC driver to connect to the database.

The database access model using mobile agents with further processing capability is shown in Figure 2. In this case, mobile agents have the capability to process the query results on the server side. Once the mobile agent obtains the query results, it will process them locally at the server based on user preference (e.g., filtering the results). After processing the query results, it usually brings back fewer amounts of data back to the user interface applet.

![Figure 1 Database Access Model Using Mobile Agents (Without Further Processing)](image1)

For this model, in addition to the advantages mentioned in the first case, there is one more advantage. Since mobile agents can perform some processing after the data is being retrieved, the data that will be sent back to user will become less. Therefore the execution time and the usage of network will be less since mobile agents send less data than what it supposed to send.

![Figure 2 Database Access Model Using Mobile Agents (With Further Processing)](image2)

3. Experimental Environments

The performance of two mobile agents for database access was tested by calculating the execution time for them to accomplish a task. Each test result
provides two sets of data: execution time of first query and subsequent queries. The execution time for mobile agents is defined as follows:

\[
\text{Execution time} = \text{data transfer time} + \text{communication time} + \text{migration time}
\]

Data transfer time is the time for mobile agents to send the results to user interface applet over the networks. Communication time is the time for the user interface applet to establish the communication channel to the local mobile agent server in order to create and activate the mobile agent to another server. Finally, migration time is the time for mobile agents to dispatch themselves to any servers through the networks.

For each case, the mobile agents access the database on the remote server via a 10 Mb/s Local Area Network (LAN) and a 28,000 bps dial-up connection to an Internet Service Provider. Therefore, all the experiments were categorized into two cases: experiments conducted on LAN and experiments conducted on the Internet. In each case, the performance of the mobile agent with processing and without processing was tested. All the tests were repeated ten times. Before every set of test begins, user interface applet will be restarted and the mobile agent server will clear the class cache for server rejuvenation.

4. Experimental Results and Evaluation

To evaluate the performance of mobile agents and find out in which situations mobile agents should be used, some experiments were conducted. The performance of mobile agents was compared with an applet application in typical client/server paradigm. The applet has the same functionality as the mobile agents so that their performance in the experiments is comparable. The experiments are divided into cases.

4.1. Experiments Conducted on the LAN

Mobile agent without further processing capability vs applet: In this experiment, both mobile agent and applet are used for retrieving information from a remote database. Since the mobile agent and the applet will not process any data on the server side, thus they will bring back the same numbers of records retrieved from query to the client. Since mobile agents need to transport themselves to the remote host and execute the query locally at the remote server, therefore, it generates migration time. On the other hand, the applet only needs

\[\text{Note that applet application is different from user interface applet in previous sections}\]
to send request signals over the networks. As a result, the mobile agents are more likely to take longer time to accomplish a task.

**Mobile agent with further processing capability vs applet:** In this experiment, functionality has been added to both mobile agent and applet. Once they retrieve information from remote database, further data processing will be taken place. However, the mobile agent and the applet use different strategy to process data. Mobile agents will filter the query results locally at the remote host and only send back the most relevant data to the user over the network. On the other hand, the applet will process the queried results on the client after it receives the query results through the network. That is, the mobile agent only brings back a small portion of data to the client, while the applet needs to transfer all the retrieved data. As the data transfer rate is very high under LAN, there is no obvious performance difference between mobile agent and applet in this case.

### 4.2. Experiments Conducted on the Internet

**Mobile agent without processing vs. Applet:** This experiment is similar to the first experiment in Section 4.1 except the testing environment is different. This experiment was conducted on the Internet using the local Internet Services Provider (ISP). The result of this experiment (Figure 3) shows that the mobile agent used 19.969 seconds to execute the first query while the applet used only 3.615 seconds, which is 6 times faster. For subsequent query, the applet only used 3.088 seconds for execution, which is also 6 times faster than the mobile agent. It has a significantly large different in execution time of two approaches because the migration time in the mobile agent is costly under the Internet environment.

**Mobile agent with further processing vs. Applet:** This experiment is similar to the second experiment in Section 4.1, but this time the experiment was conducted on the Internet. The result of this experiment (Figure 4) shows that the mobile agent used 16.985 seconds to execute the first query while the applet used 31.238 seconds, which is nearly 2 times slower than the mobile agent. For subsequent query, the mobile agent used 16.451 seconds for execution, which is 2 times faster than the applet. In this experiment, mobile agent performed better than the applet. It is because the migration overhead can be recovered by the time gained in data transfer.
4.3. Discussion

From all the experiments conducted, we have observed that the data transfer rate of the execution environment and the capability of mobile agents are the main factors that affect the performance of mobile agents developed in this application scenario.

By using mobile agent technology, receiving massive amount of request signals by the server are no longer needed. Instead, mobile agents will carry the task and dispatch to the host of the website to perform the task locally. Nevertheless, the server will still be overloaded by receiving massive number of mobile agents being sent to the same server simultaneously. Solutions can be easily found for this issue when using mobile agents. One of the possible solutions is to setup a queue in the mobile agent server in the server side. When mobile agents arrive at the server, they will be put in a queue in the mobile agent server and the mobile agent server will use the first come first serve method to allow mobile agents access resources at the server. The size of the queue is set in advance to prevent too many mobile agents dispatching to the server at the same time. If too many mobile agents have been sent to the same host at the same time and the queue is full, the mobile agent server will reject any late coming mobile agents until the queue is not full again. By using the queue method in mobile agent technology, the server will never be overloaded because the mobile agent server is able to control the number of mobile agents accessing the server simultaneously.
5. Conclusions

Traditionally, accessing databases on the Internet is based on the client/server paradigm (including web-based paradigm). With the introduction of mobile agent paradigm, mobile agent technology is adapted to improve the performance of existing paradigms. The results presented in this paper have shown that mobile agent technology is able to eliminate some drawbacks in the traditional paradigms.

Two mobile agents were developed for database access on the Internet. The performance of mobile agents and applets was compared under different environments. By analyzing the experimental results, it is evident that mobile agents should be used as a tool for accessing databases on the Internet when they are enhanced to provide the capability to filter the query results. This finding can provide guidance for the application developers of mobile agents in this field.

References

2. Tobias Butte; Based Technologies for the Development of Agent-based Distributed Applications, Department of Computer Science; *Darmstadt University of Technology*, D-64283 Darmstadt.