DISTRIBUTED DATA MINING - AN APPLICATION FOR THE WOSTM

ARMIN R. MIKLER, JOHN T. MAYES
Department of Computer Science
University of North Texas
Denton, Texas 76203, USA
mikler|mayes@cs.unt.edu

Abstract
The task of analyzing large volumes of information with the goal to find patterns or relationships that can be exploited in decision processes is referred to as data mining. Given that the data to be mined is located at remote sites, this process will tax both, the communication as well as the computational resources of the data mining infrastructure. In this paper, we focus on the efficient retrieval of information that is available on the WWW and present a distributed approach to data acquisition and cleaning, which is a pre-cursor to the actual data mining process. Since distributed data acquisition may require a number of loosely coupled remote hosts, we have identified the WOS infrastructure as an ideal environment for distributed data mining.

Keywords: Data Mining, Agents, World Wide Web, WOSTM.

1. INTRODUCTION

Data Mining is the process of searching for information in large volumes of data[1]. The information obtained through data mining is often used in support of decision processes that attempt to predict future behavior based on past performance. Intuitively, the quality of such predictions depends (among other things) on the quality of the raw data as well as the outcome of the mining process itself. While most research in the area of data mining has focused on the analysis and exploitation of existing data sets, this paper addresses some of the issues related to the acquisition process, which is in fact a pre-cursor to the actual exploitation of the data. Our exploration of data mining tools in general and the design and implementation of a distributed approach to data acquisition in particular, is driven by the need to create and maintain a comprehensive database containing daily price information for stocks traded on US stock markets.

On one hand, it is the increasing availability of electronic trading and discount brokerages that enables investors to perform their own research to find attractive investment opportunities, on the other hand, it entails the need to acquire, manage, and analyze large amounts of web-based data. Therefore, temporal complete and comprehensive stock market data has become a valuable commodity that has been made available via the World Wide Web. Clearly, for well managed and complete market information, the process of data mining has added value to the original raw data. While such information is available at some monetary cost, its content is generally limited by the particular properties of the data mining procedure. In addition,
users may be bound by a legal policy, which can restrict the further exploitation of the market information.

In this paper, we are proposing an agent-based distributed data mining approach, which allows users to individualize the data mining and acquisition process. In what follows, we will give the rationale for distributing the data acquisition process across an environment of loosely coupled hosts.

Section 2 describes the problems associated with mining stock market data from the Internet. Section 3 deals with the design of our system. Section 4 concludes this paper with a summary of the project and directions for future work.

2. Model for Distributed Data Mining

Distributed data mining extends data mining concepts by dividing the work across several processors or machines. A potential benefit of distributed data mining is increased execution speed. The problem is broken down into smaller sub-problems, distributed to several processors or machines and executed in parallel. Many implementations of data mining lend themselves to a distributed approach, however, they still require that data be located at a centralized resource or data warehouse[1]. The manner in which data is collected and stored in the data warehouse can create some interesting challenges.

2.1. Mining Financial Data

Stock market data has some interesting properties that make it difficult to mine. These difficulties are due to time sensitive data, limited access to databases on the Web, small amounts of data in larger files, and limited network resources.

Price information for stocks is time sensitive, i.e. there is only a limited amount of time collect data. Stock price information is updated every time a stock is bought or sold. To retrieve a snapshot of the day’s activity, it is necessary to wait until the markets are closed to collect data and finish before the markets open the following day.

Companies that offer free access to stock market data limit the amount of retrievable data. They do so because stock market data is a proprietary commodity. They are either bound by legal contract by their data supplier or, more likely, preventing the general public from collecting and reselling the data. As a result, data can only be retrieved in small amounts. The price information for one to five stocks is the average limit permitted per retrieval.

Access is granted to free stock market price information through Common Gateway Interface (CGI) and Active Server Page (ASP) scripts. The machines that run these scripts are commonly known as Quote Servers. Scripts that are executed on these machines return dynamic Web pages with the necessary information embedded in Hypertext Markup Language (HTML). The amount of usable data returned is small compared to the total size of the script output. A majority of the text returned is hypertext formatting tags and embedded Web page scripting.

Combining limited access to Web databases and script output size creates additional problems with available network resources. Collecting stock price information from databases on
the Web requires the capture of CGI or ASP script output and extraction of the data from the captured text. Because access to price data is limited to a few stocks per retrieval, it is necessary to perform this task several thousand times to collect the price action for all stocks.

2.2. Workload

Currently, there are approximately 11,000 to 12,000 different stocks traded on New York, NASDAQ and American Stock exchanges. Collecting price data for each stock has the potential of generating demands on local network resources that cannot be met.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Url 1</th>
<th>Url 2</th>
<th>Url 3</th>
<th>Url 4</th>
<th>Url 5</th>
<th>Url 6</th>
<th>Url 7</th>
<th>Url 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKLM</td>
<td>6353</td>
<td>17360</td>
<td>11797</td>
<td>16563</td>
<td>15443</td>
<td>16024</td>
<td>12713</td>
<td>30423</td>
</tr>
<tr>
<td>ADPT</td>
<td>6300</td>
<td>17219</td>
<td>11794</td>
<td>27105</td>
<td>15443</td>
<td>15669</td>
<td>12717</td>
<td>29084</td>
</tr>
<tr>
<td>ADMS</td>
<td>6401</td>
<td>16132</td>
<td>11791</td>
<td>15794</td>
<td>15458</td>
<td>15593</td>
<td>12722</td>
<td>27652</td>
</tr>
<tr>
<td>ARVX</td>
<td>6369</td>
<td>15536</td>
<td>11788</td>
<td>27082</td>
<td>15433</td>
<td>17793</td>
<td>12701</td>
<td>27989</td>
</tr>
<tr>
<td>ALSK</td>
<td>6412</td>
<td>16153</td>
<td>11800</td>
<td>15807</td>
<td>15450</td>
<td>15659</td>
<td>12730</td>
<td>27773</td>
</tr>
<tr>
<td>AHPI</td>
<td>6336</td>
<td>15635</td>
<td>11804</td>
<td>15666</td>
<td>15445</td>
<td>15986</td>
<td>12727</td>
<td>28013</td>
</tr>
<tr>
<td>APCX</td>
<td>6424</td>
<td>16090</td>
<td>11820</td>
<td>15804</td>
<td>15459</td>
<td>15644</td>
<td>12738</td>
<td>28446</td>
</tr>
<tr>
<td>AMAT</td>
<td>6348</td>
<td>17073</td>
<td>11797</td>
<td>27082</td>
<td>15456</td>
<td>15655</td>
<td>12723</td>
<td>29435</td>
</tr>
<tr>
<td>BMET</td>
<td>6373</td>
<td>17019</td>
<td>11801</td>
<td>20641</td>
<td>15445</td>
<td>15619</td>
<td>12724</td>
<td>28541</td>
</tr>
<tr>
<td>BOCI</td>
<td>6394</td>
<td>16317</td>
<td>11791</td>
<td>15785</td>
<td>15441</td>
<td>15698</td>
<td>12708</td>
<td>28071</td>
</tr>
</tbody>
</table>

**Table 1: Script Output in Bytes**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Url 1</th>
<th>Url 2</th>
<th>Url 3</th>
<th>Url 4</th>
<th>Url 5</th>
<th>Url 6</th>
<th>Url 7</th>
<th>Url 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKLM</td>
<td>325</td>
<td>1092</td>
<td>1156</td>
<td>903</td>
<td>5008</td>
<td>976</td>
<td>596</td>
<td>3300</td>
</tr>
<tr>
<td>ADPT</td>
<td>334</td>
<td>1121</td>
<td>5342</td>
<td>1916</td>
<td>704</td>
<td>1098</td>
<td>550</td>
<td>3353</td>
</tr>
<tr>
<td>ADMS</td>
<td>342</td>
<td>1155</td>
<td>2621</td>
<td>840</td>
<td>4340</td>
<td>995</td>
<td>554</td>
<td>2677</td>
</tr>
<tr>
<td>ARVX</td>
<td>333</td>
<td>1040</td>
<td>3128</td>
<td>802</td>
<td>658</td>
<td>1152</td>
<td>615</td>
<td>2847</td>
</tr>
<tr>
<td>ALSK</td>
<td>342</td>
<td>1050</td>
<td>2560</td>
<td>940</td>
<td>680</td>
<td>948</td>
<td>529</td>
<td>3335</td>
</tr>
<tr>
<td>AHPI</td>
<td>353</td>
<td>1067</td>
<td>10674</td>
<td>816</td>
<td>687</td>
<td>1000</td>
<td>536</td>
<td>3057</td>
</tr>
<tr>
<td>APCX</td>
<td>350</td>
<td>1071</td>
<td>1282</td>
<td>822</td>
<td>665</td>
<td>1028</td>
<td>562</td>
<td>3100</td>
</tr>
<tr>
<td>AMAT</td>
<td>337</td>
<td>4531</td>
<td>4263</td>
<td>877</td>
<td>646</td>
<td>1097</td>
<td>546</td>
<td>3487</td>
</tr>
<tr>
<td>BMET</td>
<td>357</td>
<td>1030</td>
<td>8697</td>
<td>789</td>
<td>660</td>
<td>1032</td>
<td>495</td>
<td>3314</td>
</tr>
</tbody>
</table>

**Table 2: Transmission Time in Milliseconds**

Table 1 shows the output in bytes for ten stocks from eight different Quote servers. Minimum, maximum and average script output for the 50 stocks in our original test set are also given.
While the sizes are not particularly significant per retrieval, performing this task 12,000 times and an additional 12,000 times to check for data consistency adds up to substantial amount of raw data to collect per day. The following example demonstrates the total resource requirement in terms of connection speed.

In the experiment, the size of the average script output for our test set of 50 stocks was found to be around 16,474 bytes. Using this average, the total amount of unprocessed data to retrieve the price action for each stock and check each stock for consistency would be approximately

$$2 \times 12,000 \text{ stocks} \times 16,474 \text{ bytes} = 395,375,712 \text{ bytes} = 377 \text{ megabytes}$$ (1)

Assuming that we have 8 hours to collect data, it would require a sustained throughput of

$$\frac{395,375,712 \text{ bytes} \times 8 \text{ bits}}{8 \text{ hours} \times 3600 \text{ seconds}} = 109,827 \text{ bits per second}$$ (2)

This could be achieved with an ISDN or higher connection, however, it does not factor in transmission delays and processing time for each page. Table 2 shows the time in milliseconds to request and collect each page with minimum, maximum, and standard deviation values given for the 50 stocks in our test set. As the data shows, the times collected from a single site can vary drastically. The average time to collect a page is 1.66 seconds, which gives us a total time of approximately

$$\frac{2 \times 12,000 \text{ stocks} \times 1.66 \text{ seconds}}{3,600 \text{ seconds}} = 11 \text{ hours}$$ (3)

Realistically, the time required to collect data should be as short as possible to allow time to apply traditional data mining techniques to the updated database. Results could then be available for review before the opening of the stock markets on the following day. However, limiting the time frame and collecting data on a single network can potentially create a bottleneck as depicted by Figure 1.

As stated earlier, the script output for each Quote server is largely hypertext formatting tags and embedded Web page scripting. The majority of the output does not contribute to the required data and in fact, only creates a substantial amount of overhead for the system. The amount of processed data for each page is approximately 50 bytes. Therefore, the total amount of processed data per day is

$$12,000 \text{ stocks} \times 50 \text{ bytes} = 600,000 \text{ bytes} = 586 \text{ kilobytes}$$ (4)

Unfortunately, there is no efficient way to separate the data from its transport medium. This gives rise to a distributed approach to data mining.

3. System Design

In this section, we describe the various components of our system. As described earlier, the problem of collecting stock market information is comprised of a small set of operations repeated numerous times. Our proposed system is agent based which moves some of the computational components, i.e. text processing and database pre-filtering, closer to the data source
or, more importantly, moved away from local computational and communication resources. The potential benefits of an agent-based approach are decreased execution time and decreased network resource requirements.

In general, an agent is a computer program (thread) that acts autonomously on behalf of a person or organization. Agents can be stationary (executes only on the system where it began execution) or mobile (can move to a remote system for execution[2]). Agents can be classified according to four characteristics, namely, intelligence, cooperative behavior, autonomy, and mobility[3]. We define these characteristics as follows:

- **Intelligence** is the ability to adapt to circumstances brought upon by the dynamics of the system.

- **Cooperative behavior** describes the ability to share knowledge among agents and/or negotiate a common strategy that yields actions that lead to an overall acceptable performance.

- **Autonomy** allows agents to execute without the interference of users.

- **Mobility** describes an agent's ability to transfer between different hosts.
The facilities needed to collect stock market data from the Internet include communication, agent initialization, agent control, database manipulation and text processing. Our system is comprised of four distinct components, namely, the Coordinator, Executor, Dispatcher and Harvester. Common to all modules is the ability for each component to communicate and cooperate with one another. A basic diagram of the system is shown in Figure 2.

![System Layout Diagram](image)

**Figure 2: System Layout**

The Coordinator Agent is the core of the system. The main purpose of the Coordinator is to provide a secure interface to the data base. It is also responsible for initializing the two static agents of the system. To initialize each agent, the Coordinator maintains a list of participating hosts and checks the availability of each host. If the host is available, a message is sent requesting that a particular agent be retrieved and executed. These messages are in the form $ABC$ where $A$ is the task, $B$ is the name of the required agent and $C$ is security parameters.

The Executor Agent controls the retrieval and execution of the agents. A service request is
sent over the local network or Internet and granted access if the requesting hostname is found in a list of known hosts. Contained within the service request is the name of a particular agent needed to perform some task. This agent is then retrieved from the requesting host via Hypertext Transfer Protocol (HTTP) and executed. Upon completion of its task, the agent returns control to the executor module to perform any necessary cleanup. As with any system capable of remote execution, security is a primary concern. However, security is beyond the scope of this paper and will be discussed elsewhere.

The Dispatcher Agent is a mediator between the Coordinator and the Harvesters. The role of the Dispatcher Agent is to forward the ticker symbols of each stock to the Harvesters, check the results for data consistency, and send the results to the Coordinator. Each Dispatcher is given its own portion of the overall retrieval list and an address list of available Harvesters. The retrieval list is processed serially by sending messages to two different Harvesters requesting the price information for one stock. To ensure that no two Harvesters perform all of the work, a sliding window over the list of available Harvesters is used. When results are received, the data is checked for consistency. If the data from the two Harvesters does not match, requests are sent to additional Harvesters until a consensus over the price is made.

The Harvester Agent performs a majority of the work of the system. It is responsible for requesting and processing stock price information from Quote servers. In order to make the Harvester agent uniform for all Quote servers, templates are used. These templates contain parameters necessary to issue the request and process the captured output. When created, each Harvester is assigned a particular Quote server and must obtain the template for its assigned Quote server. The Harvester then processes Dispatcher requests until it receives a message to terminate.

An environment such as the WOS, where resource sharing is intrinsic, provides the opportunity to exploit the inherent parallelism of the Internet by distributing communication resource requirements across a global pool of machines[4]. We view distributed data mining as an ideal application for the WOS infrastructure.

4. SUMMARY AND DIRECTION FOR FUTURE WORK

In this paper, we have described some of the problems associated with collecting stock price information from free sources on the Internet. Using a distributed approach, it is possible to distribute some the workload across remote hosts. This allows us to take advantage of the inherent parallelism of the Internet and minimize the requirements for local computational and communication resources. Incorporating agents into the system, allow us to control the collecting process and minimize the install base and administration requirements on remote hosts. Considering that we are using resources that have been voluntarily added to a global resource pool, it is in the interest of both the owner and the user to keep executing modules small. Agents generally do not consist of large code components and are therefore considered lightweight.

There are a number of enhancements and modifications to be made to the design of the system. The project described in this paper focuses on the collection of stock price information. To broaden the knowledge base of each stock, an additional module could be introduced to the system that is capable of collecting corporate fundamental information, i.e. cash, assets
and liabilities. Also, we have focused on U.S. stock markets. Extending the system to include international stock markets may provide insight into individual markets and the strength of the global economy. As with any complex system, visual management tools can aid system administration. A graphical user interface for system configuration, status and error tracking would be particularly helpful. Last but not least, the issue of security must be addressed. Mechanisms must be in place to prevent the execution of agents from unauthorized hosts.

REFERENCES


