Project Synopsis: Evaluating STRIP *

Brad Adelberg[†] Hector Garcia-Molina[‡]

Abstract

Ths paper describes preliminary efforts at evaluating the performance of the Stanford real-time information processor (STRIP v2.0). We desribe a benchmark for active real-time databases based on a program trading application and report STRIP's performance on this benchmark.

Keywords: derived data, view maintenance, active database system, transaction scheduling, program trading, real-time database system.

1 Introduction

The Stanford real-time information processor (STRIP) is a soft real-time main memory database system with special facilities for importing and exporting data as well as handling derived data. Data can be imported into (and exported from) STRIP using the stream interface, which allows STRIP to maintain materialized views of remote and very rapidly changing data. In addition, a sophisticated rule system supports the efficient maintenance of derived data over rapidly changing base data. This paper summarizes our efforts to benchmark the performance of STRIP v2.0. It also presents a sample of the preliminary performance results we are obtaining, focusing on the performance of STRIP's import and active data capabilities. For more details on the STRIP project, the reader is directed to [AKGM96].

One of the challenges of designing a database system with new functionality is to find a suitable benchmark to test it with. For STRIP, we have designed a benchmark based on a program trading application. Program trading is the use of a computer to automatically identify and act on market imperfections to make money. A program trading system is built from many components, including a source of information about market activity, a database, an expert system, and a system to execute electronic trades, as shown in Figure 1. We are primarily concerned with the function of the database component of this system, although we consider the other components where they impact the requirement on the database. As shown in Figure 1, a program trading application uses three kinds of data. First, it must maintain the current prices of all of the financial instruments it can trade.[1] As the market changes, this data can be changed as well using price reporting feeds

*This work was supported by the Telecommunications Center at Stanford University, by Hewlett Packard and by Philips.

[†]Stanford University Department of Computer Science. e-mail: adelberg@db.stanford.edu

[‡]Stanford University Department of Computer Science. e-mail: hector@db.stanford.edu

[1]It might also store additional information about each instrument, such as historical prices or trading volume, but those are details that will be abstracted away for now.

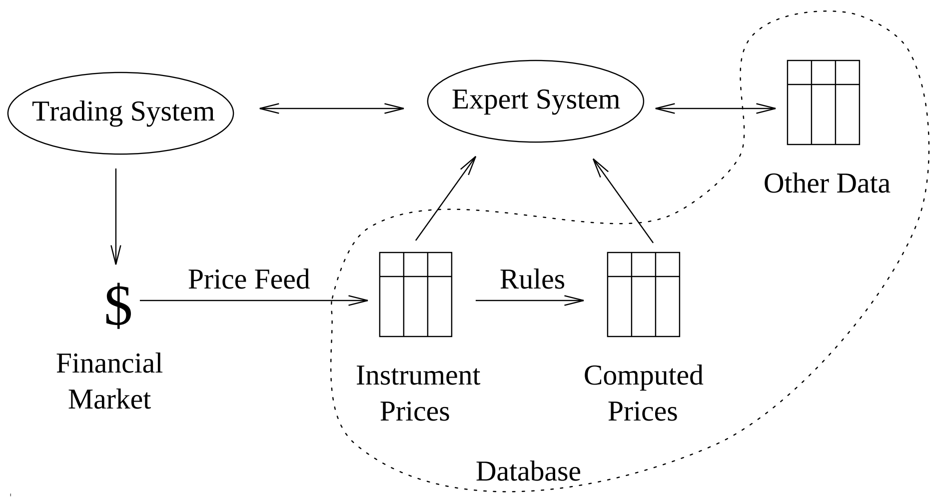

Figure 1: A high level model for a program trading application.

available from several different vendors. Next, data computed from the instrument prices must be maintained. This can include the theoretical prices of derived instruments, predictions for future prices, etc. Often, trading decisions can be made more easily on these types of derived data rather than on the underlying price data. Finally, the trading system will keep additional information not derived in any way from the external environment, such as the current holdings in its portfolio.

The database component of such a system needs to support both active rules and real-time scheduling. Active rules are necessary so that the expert system can be triggered when pre-specified price changes occur. The alternative, polling, is not feasible. Real-time scheduling is required since trading opportunities last only a short time and naive scheduling algorithms may result in too many missed deadlines. Currently, no published system uses real-time scheduling because the overhead of sophisticated scheduling is greater than the available gains. The challenge then of building a database system to support program trading or like applications, then, is not just to support active rules and real-time scheduling, but to do so in a way that does not degrade the system performance.

2 A simple stock benchmark

In this section we describe the program trading "application" used to benchmark STRIP. In practice, program trading systems are custom built by each trading firm and their market models and trading algorithms are closely held secrets. Thus the benchmark we present here is simplified both out of necessity, since very little information is publicly available, and also to focus our attention on the important issues of data management without getting lost in the details of financial modeling. Still, we feel that the application model that results captures the important features of the real problem and will point out how to extend it where appropriate.

The benchmark requires the database to maintain three types of prices: stock prices, composite index prices, and theoretical option prices. The stock prices can be simply updated in the database according to the market feed but the composite and option prices must be computed from the stock prices. In fact, the current trend of feed providers is to send more than stock prices with the feeds, including popular composite prices (e.g. DJIA) and other derived values. Still, additional derived data, such as that related to proprietary market models, will always have to be computed by the database. Because composite averages and theoretical option prices have known functions, are easy to understand, and reasonably reflect the types of data that need to be computed, we choose to compute them as part of the benchmark rather than some purely fictional data.

The formula to compute a composite average composed of n stocks is $comp\_price = \sum_{i=1}^{n} w_i p_i$ where p_i and w_i are the price and weighting of the ith stock respectively. For theoretical option prices, we use the Black-Scholes pricing model [BS73], which although known to under value options, is still commonly used and reported. The model computes the price of an option as a function of five parameters:

- the current price of the underlying stock,
- the exercise (strike) price of the option,
- the annualized continuously compounded risk-less rate of return,
- the time remaining before expiration expressed as a fraction of a year,
- the standard deviation of the annualized rate of return of the underlying stock.

The database for the benchmark contains the following six tables:

stocks(<u>symbol</u>,price) - contains the price of every stock as reported by the wire service.

stock_stdev(<u>symbol</u>,stdev) - contains the standard deviation of the annualized rate return of every stock. This information is required to compute theoretical option prices.

comp_prices(<u>comp</u>,price) - contains the computed price of every composite average (i.e., Dow Jones Industrial Average (DJIA)). Refer to the formula above.

comps_list(<u>comp,symbol</u>,weight) - describes the many-many relation between stocks and composites.

option_prices(<u>option_symbol</u>,price) - contains the computed price of every listed option. (See [BS73] for the full equation).

options_list(<u>option_symbol</u>,stock_symbol,strike,expiration) - describes the one-many relation between stocks and options.

The tables comp_prices and option_prices are actually materialized views with the following definitions:

```
create view comp_prices as
select comp,sum(price*weight)
from stocks,comps_list
where stocks.symbol = comps_list.symbol
group by comp
```

create view option_prices **as**
select option_symbol,$f_{black-scholes}$(price,strike,expiration,stdev) **as** price
from stocks,stock_stdev,options_list
where stocks.symbol = options_list.stock_symbol **and** stocks.symbol = stock_stdev.symbol

A program trading application is driven by changing stock prices as reported by a market feed. For the benchmark, we use the consolidated quote file provided as part of the New York Stock Exchange's TAQ database [New94]. The quote file lists all of the stock price quotes made during actual days of trading on all of the major U.S. exchanges. Each entry contains the symbol of the stock being quoted, the bid and ask prices, and the time of the quote to the nearest second. As the stock prices in the database change as dictated by the quote file, the database must recompute the prices of the composite indexes and the options. In STRIP, this is done by writing rules that are triggered by changes to the attribute price in table stocks.

3 Performance model

The stocks table is populated from data provided by the New York Stock Exchange as part of their TAQ database [New94] and contains 6600 stocks. The experiment is driven by the actual price changes recorded during trading in January of 1994. The trace is loaded into memory before the experiment begins to remove the I/O costs from the results. Each run lasts for 30 minutes during which the price changes that occurred during real trading on the exchange are applied to the system in real-time. The quote rate varies significantly over the 30 minute run but the average rate is 24 changes per second. The table comp_prices contains 400 composites, each calculated from 200 stocks. The component stocks in each composite are chosen randomly but in direct proportion to their trading activity as measured by the number of price changes in a day of trading. We feel that this is indicative of the distribution of real stocks in real composites: the stocks of large companies which trade frequently are most often used in composites since they tend to be good indicators of entire industries. The table option_prices contains 50,000 entries. As with composites, these are randomly generated but in relation to the frequency of trading of the underlying stocks. Hence the expected number of listed options for a particular stock is the total number of options times the fraction of the entire trace due to the particular stock.

The performance numbers reported here are for STRIP running on a HP-735 under HP-UX 9.03 with 144Mb of main memory. No other user processes were run on the system during the experiments. Response times are measured using the Unix function *gettimeofday*. The fraction of the CPU required for an experiment was measured using the Unix call *times*. In order to understand the reported results, some basic timing measurements on STRIP v2.0 are reported in Table 1.

Action	Time (μsec)
begin task	10
end task	5
begin transaction	1
commit transaction	10
open cursor	60
fetch cursor	15
update cursor	10
close cursor	25
get lock	10
release lock	30
run scheduler	5

Table 1: Basic performance number for STRIP 2.0

Task	# of times run	External Queue Time (ms)	Internal Queue Time (ms)	Execution Time (ms)
Import	44710	11.3 (40.6)	0.026 (0.008)	3.0 (4.9)
Compute composite	40695	34.4 (67.8)	0.026 (0.014)	10.9 (10.2)
Compute option	41270	25.2 (66.3)	0.026 (0.013)	8.4 (9.2)

Table 2: Average task timing results for STRIP v2.0 (s.d. in parenthesis)

4 Results

Maintaining the 400 composite averages and the 50,000 option prices used 82% of the cpu capacity. The timing results for each type of task in the system are reported in Table 2. The import task changes the prices of stocks based on the market feed which is connected to STRIP through its stream interface (see [AKGM96]). Rule condition checking occurs within this task which may trigger other tasks to recompute the composites or options derived from the altered stock. For each type of task, we report the number of times it executed as well the average amount of time it spent (and standard deviation of the time) in the external queue, internal queue (process pool), and executing.

The early performance results of STRIP v2.0 are promising. The system is able to handle large amounts of derived data at the full market feed rate. The external queueing times are relatively long but that is mainly due to the system being so highly loaded. In addition, the experiments reported here do not use the *unique transaction* facilities of STRIP which can greatly reduce the recomputation load. The one area of concern is that the variation in execution times is high. This is primarily due to interference from HP-UX which we hope to reduce with further tuning of the STRIP scheduler.

References

[AKGM96] B. Adelberg, B. Kao, and H. Garcia-Molina. Overview of the STanford Real-time Information Processor (STRIP). *SIGMOD Record*, 25(1):34–7, 1996.

[BS73] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–54, 1973.

[New94] New York Stock Exchange, Inc. *The TAQ database*, 3.0 edition, June 1994.