International Journal of Advances in Computer Science and its Applications – IJCSIA
Volume 4 : Issue 1 *[ISSN 2250 – 3765]*
Publication Date : 09 January 2014

An Integrated Platform for System Trade using the Cloud

Takahiro KOITA* and Hiroaki TSUJIMOTO*

***Abstract*—Recently, algorithmic trading, which is an automatic trade by means of a computer and a trading rule, has become popular. To use the algorithmic trading effectively, its trading rule is quite important because the trading rule is an algorithm to determine the timing of using market parameters. Investors are eager to find a good trading rule, which can be a good indicator to trade stocks, from among enormous numbers of rules. However, the existing trading-rules evaluation systems for virtual dealing cannot be used for full search of trading rules. The causes of the problem are the compatibility of virtual dealing software and the computing resource available for trading-rules evaluation. In this research, we have designed and developed an integrated platform for trading rules search to cope with these two issues.**

Keywords—algorithmic trading, cloud computing, Hadoop

I. Introduction

With the growing popularization of the stock market through the Internet, system trade has recently been attracting attention[1-5]. System trade is an investment method that deals in equities or the like in accordance with a certain number of trading rules, but without discretion during the performance of the investments. Investors involved with system trading are searching for trading rules that are effective in gaining profit by real operations while repeating many rounds of trial and error. The trading rules used for system trades are generally evaluated by using a trading rules evaluation system, but there is a problem in that it is difficult to discover effective trading rules with existing trading rules evaluation systems.

To solve this problem, this study designs and implements an integrated platform that makes it possible to execute an evaluation of all trading rules within a realistic timeframe by using large-scale distributed processing. We then perform evaluation experiments using the integrated platform and demonstrate its usefulness by evaluating its performance.

Faculty of Science and Engineering,
Doshisha University, Japan

II. Evaluation and current situation of trading rules

A. *Trading rules*

The trading rules used for system trades are expressed by combinations of technical indices and parameters. A technical index is an indicator created for analyzing current stock situations and forecasts of share price movements, from information such as share prices, earned values, and time. One example is an indicator called the relative strength index (RSI). RSI is an indicator that expresses the sensation of overheating of the market, calculated from the share prices over the past n days. As a general guideline, if the value of this RSI is 30% or less, the stock in question is said to have been oversold so we should buy; if it is 70% or more, the stock has been overbought and we should sell. If this guideline is adapted to be a trading rule, it is expressed as a conditional statement such that the buy condition is if RSI < 30 and the sell condition is if RSI > 70. These two numbers become RSI parameters. If the values of these parameters change, the timing and results of trading will also change, so each combination of values is handled as a single independent trading rule. In other words, the number of trading rules is governed only by the number of combinations of indicators and parameters.

Since there are 50 RSI parameters for each pairing of 0 to 49 with 51 to 100, there are 2,500 trading rules that use RSI. There are various technical indices other than RSI, but since there are several hundred of the main technical indices alone, the total number will be extremely large if we also take into account trading rules that handle a number of technical indices.

B. *Evaluation of trading rules*

The trading rules used for system trade are generally evaluated by using a trading rules evaluation

system. This trading rules evaluation system consists of virtual trading software, which has a backtesting function, and computational resources that runs that virtual trading software. Backtesting involves performing a simulation as to what performance the trading rules would achieve with respect to historical market data. However, it is difficult to discover effective trading rules with the existing trading rules evaluation system. Two major problems are considered to be the reasons for this.

The first problem is that there is no compatibility between different virtual trading software programs. At the moment, the trading rules of virtual trading software are in various different description formats, but since there is no compatibility between these formats, the existing trading rules evaluation system can only handle one specific virtual trading software program. However, since each virtual trading software program has different characteristics, the details of backtesting that can be executed will also differ greatly. From this, we see that it is impossible to do all the backtesting with the existing trading rules evaluation system, which can only handle a virtual trading software program, making it difficult to discover effective trading rules.

The second problem is that there is insufficient computational capability for evaluating the huge volume of trading rules. Since the total number of trading rules is extremely high, in order to discover effective trading rules it is necessary to evaluate larger numbers of trading rules, using large-scale computational capabilities provided by a large amount of computational resources. However, since the existing trading rules evaluation system is designed to use only a single computational resource, it is not possible to utilize sufficient computational capability with the existing system.

III. Implementation of integrated platform

To solve the above problems, this study designed and implemented an integrated platform that makes it possible to do all the backtesting by large-scale distributed processing within a realistic timeframe. We implemented compatibility between virtual trading software programs by using an integrated interface, and large-scale distributed processing by using distributed computing.

A. *Integrated interface*

The integrated interface consists of three parts: a settings file written in a unified description language, language conversion classes, and a template program for the virtual trading software. Trading rules take various different description formats, depending on the virtual trading software. However, although description formats differ, the information itself is common. An outline of information on these common parts is a settings file in a unified description language. The template program for the virtual trading software is a program which describes the parts of the template, in a manner of speaking, since it is necessary to configure all of the trading rules when generating the trading rules by the virtual trading software.

We took a method of generating the trading rules by combining this template program with settings file information that has been converted into any desired description format by language conversion classes. This makes it possible to evaluate any desired trading rules indirectly by a number of virtual trading software programs, and thus establish compatibility between virtual trading software programs.

B. *Distributed computing*

In implementing distributed computing, it is necessary to have computational resources to do the actual processing, and distributed computing middleware. For the computational resources, we acquired computational resources that utilize cloud services. There are currently various different forms of cloud service[7-11], but for this study we employ Amazon EC2 because of its impressive record of performance as computational resources for distributed computing. We employ Hadoop for the distributed computing middleware. Since Hadoop specializes in high-speed data processing with no contraints on the computer used, we can expect it to demonstrate a high level of performance in the evaluation of trading rules. We also introduced the elastic load balancer (ELB), which is an EC2-dedicated load balancer provided by Amazon, in order to support the reduction of bottlenecks in the distributed processing.

IV. Evaluation

We performed evaluations of each of the integrated interface and the distributed computing, as evaluation of the integrated platform.

A. *Integrated interface*

For the integrated interface, we performed conversions in practice using the virtual trading software programs Tactico[12] and Kabu Robo, to verify compatibility between the virtual trading software programs. We also verified that trading rules with the same details could be created from the same settings file by two different virtual trading software programs. A comparison of the integrated platform that utilized the integrated interface and the existing trading rules evaluation system is shown in Table 1.

Table 1 Comparison with existing system

	Existing system	Integrated platform
Target software	Specific software	All software
Software expandability	None	Expandable
Trading rules generation	Manual	Automatic

B. *Distributed computing*

For the distributed computing, we first investigated scale-out performance. We used the evaluation environment described below, measured evaluation time targeted at 1000 trading rules when the number of computational nodes was varied from one to 16, and compared with round-robin results.
Evaluation results are shown in Fig. 1. Numbers of computational nodes are plotted along the horizontal axis with execution times along the vertical axis. As a result, we could verify that execution time is shortened with the proposed system roughly as a

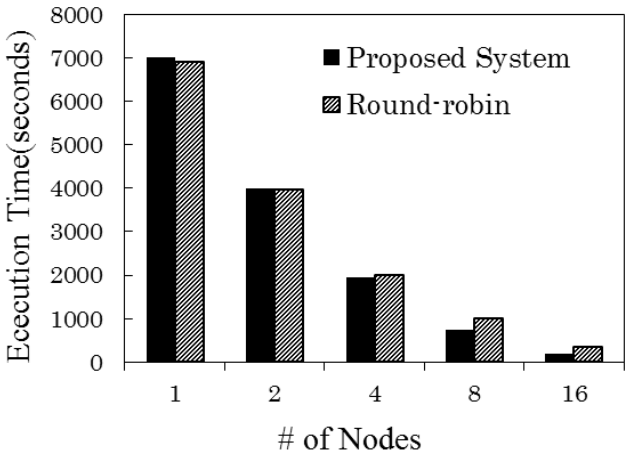

Fig. 1 Numbers of computation nodes and execution times

Table 2 Result of the best trading rule

Number of trades	3841
Average trade performance per trade	1.30%
Average trade period per trade	12.11 days
Number of successes	1407 (36.63%)
Average profit when successful	14.80%
Maximum profit	120.73%
Number of failures	2434 (63.37%)

function of increases in computation nodes, as expected. The comparison with round-robin processing also verified that a difference in processing time between the two methods appeared as the number of nodes increased. If we calculate the time required for evaluating one trading rule when the number of computation nodes is 16, it is approximately 0.5 seconds. This means that the time required for evaluating 6 million trading rules that are provisionally calculated to be the main trading rules is34 days, which also confirmed that there were no practical problems with the evaluation efficiency of the proposed system.

Detailed results of dealing with the most highly evaluated trading rules discovered by the processing done by the above performance evaluation are shown in Table 2. From Table 2, it has been verified that calculations of the operating performance for these trading rules show approximately 50 times, and the proposed system makes it possible to discover highly respected trading rules.

V. Conclusion

This study designed and implemented an integrated platform that makes it possible to execute an evaluation of all trading rules in system trading within a realistic timeframe, by using large-scale distributed processing. We also demonstrated the

validity of the proposed system by verifying that there was compatibility between virtual trading software programs, that there was sufficient scale-out performance for large-scale distributed processing, and that trading rules that were highly respected in practice were discovered by the system.

References

[1]F. Toriumi, K. Izumi, and H. Matsui : "Evaluation of influence of automated trading programs using artificial market", Proceedings of the Annual Conference of The Japanese Society for Artificial Intelligence, Vol. 22, 1E2-05, 2008.

[2]Y. Kodama and M. Xie: "Construction of the Stock Trading Agent Using Reinforcement Learning" Information Processing Society of Japan Research Report, MPS, Mathematical Modeling and Problem Solving Research Report, Vo. 19, pp. 57-60, 2007.

[3] Omega Chart: http://www.omegachart.org/download.html/

[4] Sistore Damshii: http://kabu-trading.com/sts-download.html/

[5] Minkabu: http://minkabu.jp/

[6] Google App Engine: http://code.google.com/intl/ja/appengine/

[7] Windows Azure: http://microsoft.com/windowsazure/

[8] Amazon EC2: http://aws.amazon.com/jp/ec2/

[9] BOINC: http://boinc.berkeley.edu/

[10] Hadoop: http://hadoop.apache.org/

[11] Amazon ELB: http://aws.amazon.com/jp/elb/

[12] Tactico: http://www.lagarto.co.jp/tactico/

[13] Kaburobo: http://www.kaburobo.jp/