Complex Event Processing of Market Data through Data Modeling in Big Data

Shiny A, Rikhil G R, Namita Vagdevi Cherukuru, Alen Mammen Ivan, Sunil Prakash J

Abstract: ***Mathematical Finance utilizes advance refined mathematic models and advanced computer techniques to forecast the movement of worldwide markets. To possess an ability to react intelligently to the fast-paced changes in the business is a winning factor. Complex event processing with advanced toolchains plays a crucial role in the explosive growth and diversified forms of market data. To resolve such issues, we have developed a model based on Big Data that processes the intricate tasks to assess the market data. The model executes complex events in a data-driven mode in parallel computing on copious data sets, this model is known as StatCloud. To implement StatCloud, we have used datasets from the Bombay Stock Exchange to determine the performance. We execute the model with the help of Data analysis techniques and Data Modelling. The experiment results show that this model obtains high throughput and latency. It executes data dependent tasks through a data-driven strategy and implements a standard style approach for developing Mathematical Finance analysis models. This integrated model facilitates the work process of complex events in a financial organization to enhance the efficiency to implement the right strategies by the financial engineers.***

Index Terms: ***Mathematical Finance, Big Data Analysis, Complex Event Processing, Data parallelization, Parallel Computing, Statistics.***

I. INTRODUCTION

Mathematical Finance uses mathematical models to predict the outcome and future of the market. It identifies the potential risks the ongoing projects can face in the future, and determine the value of the derivatives and other external assets. In the business industry, the rate of exchange of data has surpassed the existing range in terabytes. The market analysis reports show that financial proceedings have increased by 65 percent in the past 15 years in the Indian market. The data consumed for trade is around 300MB and 40 times larger in case of quote data. Besides the challenge of processing copious data sets, the varied forms of data encountered toughens the task.

Unstructured data unlike structured data is not organized and makes it difficult to implement data analysis techniques.

Revised Version Manuscript Received on July 20, 2019.

Shiny A, Computer Science and Engineering, SRM IST, Chennai, India.
Rikhil G R, Computer Science and Engineering, SRM IST, Chennai, India.
Namita Vagdevi Cherukuru, Computer Science and Engineering, SRM IST, Chennai, India.
Alen Mammen Ivan, Computer Science and Engineering, SRM IST, Chennai, India.
Sunil Prakash J, Computer Science and Engineering, SRM IST, Chennai, India

Though the amount of information retrieved from sources is enormous, the financial companies lack the insightful data necessary. The main aim of the financial engineers is to extract the insightful information helpful for the gaining profits at different stages. To yield desired information, we integrate Data analysis methods such as a) data cleaning and aggregating and b) to build a data-driven mathematical model. Most of the statistical models developed comprises uniform spaced time series. These models can process data in the specified limit, for example, a finance engineer might require one structured dataset to determine the mid prices instead of processing all the data available. In the first stage of data retrieval, the data is insightful with irrelevant information. Data cleaning aids in deciphering the improper data and obtain insightful thoughts. This robust data leads to finding the profitable frequency. Later, the clean data is structured into advanced data models with the help of algorithms and computing mathematics. This model helps to extract streams of data from large aggregated sets of data. For instance, we could design Bayes method to understand Artificial intelligence.

The statistical models are designed to represent features such as trend following, co-integration and Pairs trade. The companies base the predictions on the price fluctuations within market limits. A finance manager can forecast a gain and loss at the horizon. The portfolio can be enhanced and tuned based on the predictions received. The manager can decide the rate of returns to the company and sequence the workflow. The time-series data at a certain frequency is used to make profitable decisions; this is known as data-driven strategy. Marketing firms yearn for a single solution to handle the data analysis strategies and implementing in their workflow. To facilitate this, we have developed a strategic paradigm that uses Data analysis techniques and mathematical models and is termed as StatCloud.

Our project measurement based on parameters such as latency and throughput. The experiment performed shows the enhanced capability and surpasses the efficiency of the existing model. The prototype can simultaneously perform number of data-dependent tasks. The contributions of the project are:

a) Through parallel computing in a data-parallel mode, multiple tasks on data are performed efficiently. The series of data is structured into smaller modules and helps in processing intricate events in Mathematical Finance modelling.

b) The information is stored in the form of a series of tasks for the business market, the strategies proposed are executed in a data-parallel mode and the model is an integration of Big

Data and Statistics.

c) The proposed system uses data from Bombay Stock Exchange of India to compute the log feedback from the ping data. The log feedback is optimized, inconsistencies eradicated and the log feedback is integrated with the Autoregressive integrated moving average model.

d) The performance of the model is determined by the parameters like throughput and latency. It is measured from user's point of view and input. On the whole, the project is a multi-disciplinary model that has sophisticated models that help Finance engineers process intricate models with high frequency and predictability.

II. STATCLOUD SYSTEM MODEL

The StatCloud system model is a data-driven model that uses multicore architecture unlike the previous existing model, which executes the process in series manner. The StatCloud system model is a data-driven model that uses multicore architecture unlike the previous existing model, which executes the process in series manner. The extended version of the existing system will enhance the proposed StatCloud platform for developing Finance applications based on big data. The parallel computing platform, the model can perform tasks at a high frequency with higher feedback.

A. Overview

The StatCloud consists of i) a database to store and retrieve data, it also acts as a server where it indexes the data and provide database service, ii) a parallel computing platform that acts as a client and iii) a cloud platform where the application services are provided to the user. The server plays a crucial role in managing the financial data; it stores the information in the form of symbols. The data is stored in hashed and compressed format and assigned blocks of storage, each block has a unique identification mark that can be used to retrieve data from the database. Saving data by hashing secures the data from breaching and compression opens up room for more space in the disk.

Each hash is stored with the relative date of the data created, these algorithms helps in segregating the data into different blocks for security. In addition, the server conducts data queries and follows client-server model. The server separates requests from the data provider and client, the communication happens over internet. The tasks demanded by the user, which is also a cloud portal. The user sends prompt in the form of parameters, which are executed by the client portal. The user specifies the data required for the algorithm. Client proceeds with querying the data as per the demands for user and forward it to the server. The interconnected process happens in a series: Server sends data to the client portal and the client executes the tasks and give feedback to the user.

The process follows the technique of separation data by an algorithm, this model supports "Anything-as-a-service" protocol in the cloud platform. Storing the information in cloud can increase the efficiency of the user model and process tasks at a faster pace. SaaS platform helps the user to interface with the big data model on a subscription basis and IaaS provides physical computing resources to the users to access high level APIs.

B. Big Data Analytics

Big data can weave insightful information from streams of data. Each dataset has a different story that is essential for a company. In our model, the data analytics techniques are performed in data parallelization mode. Through parallel computing in the client platform, data service module and complex event processing module is implemented. The data service modules obtains the necessary information and decompress for fetching results and forwarding it to the user. It also does required dehashing to parse the data and understand the deciphered symbols for the requested information. The complex event processing analyses the data and produces an outcome, multicore architectures are used to create such mode in data-driven model. Synchronization errors can be minimized through parallel computing and focus on the efficiency of the model. The tasks are split and distributed among the processors, which is executed in a parallel computing environment.

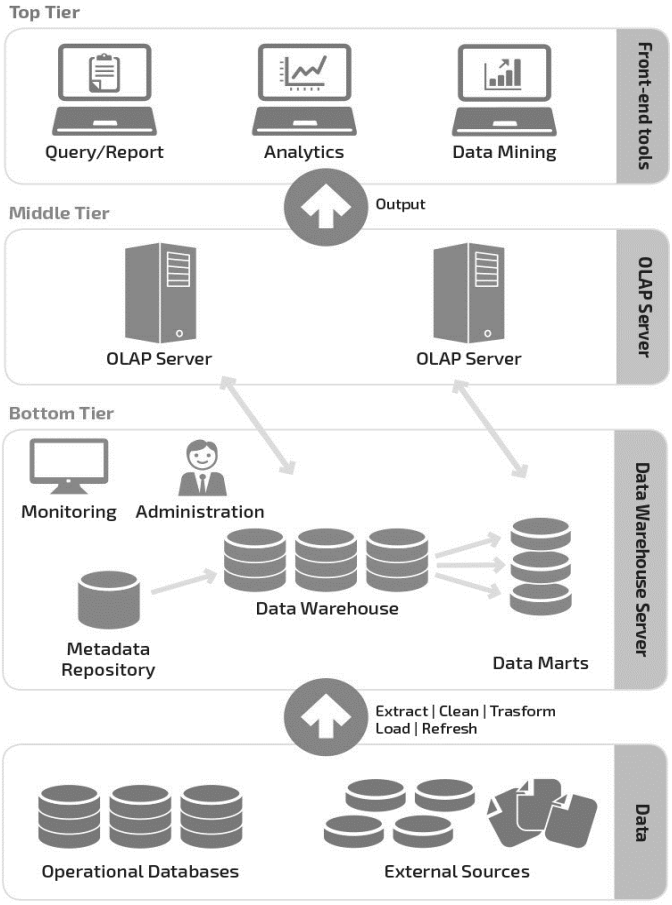

Fig 1: Architecture of interconnection between different levels of the proposed model.

III. DATA DRIVEN EXECUTION MODEL

In the following sections, multiple tasks are processed in the parallel computing platform. We define the variables used in this section and provide definitions and values used for the data-driven model. Models

created for mathematical operations are mentioned in the following sections. The definitions in this section help understand the context in the following sections and the how these definitions are used in determining the mathematical models.

A. Definitions

In the data-driven model, the tasks are simultaneously processed without blocking in the intervals. The availability of the data determines the scheduling process. Let us consider that

a) An algorithm features numerous tasks, which is streams of information or instructions and the input, is does not have boundaries.

b) The mathematical models are built to take in at least two instances of input and can have any number of inputs given to the model.

1st Definition: The tasks which performed simultaneously are represented in the forms of graphs where P=G (T, E), where T is the node and E is the edge that represent the relation between the data and the nodes formed.

2nd Definition: The task can be represented in the form of $a_n = a(n; T_n, e_n)$, n is the node of the graph and e_n is the output of the node n and $e_n \in Z$. Z is the superset of T_n, which defines the data dependencies of the model. For instance, if $e_n \in Z$, all of the outputs and inputs given to and by the model is Z. We can say that $T_n = \emptyset$, when there this no outputs, discrepancies in the process, when no input is given to the paradigm and root node is empty.

3rd Definition: In the group $A = \{a_n\}$, s represents the nodes and in the group $Z = \{e_n\}$, f defines the edges. Considering the nodes and edges in this group, we can say that $s \geq f$.

B. Time and Data Scheduling

In this segment, we will discuss about the three algorithms based on time, data and the hybrid between the both. These algorithms are used for scheduling strategies where the data is split into tasks and implemented based on the priorities. These algorithms are performed at different stages of processing the data, the paradigm uses any number of inputs and outputs to determine the processing of the data. We will compare how each of these algorithms work.

In the Time-dependent algorithm, we try to analyze the number of threads that can be processed in a certain time limit. The information is distributed to the set of tasks. The information is represented in the form of huge number of threads that have tasks spilt to process simultaneously at different rates. A pool of threads are structured in such a way that it can be implemented simultaneously and on a parallel platform. The tasks in the model are implemented in a first-in first-out basis where tasks are given at one end and the other end pushes out the processed tasks. Usually, in this algorithm, the worker threads are executed in the order in which the model receives the tasks. To keep a track of what is happening throughout the process, few flags are used to send and receive the status of the task and the process. Following are the statuses of the process:

- *Idling*: The task is yet to be assigned.
- *Scheduled:* The tasks to be implemented have been determined and has not started executing yet. It is scheduled for the next step.
- *Running:* The thread has been running throughout the process and has not been completed yet.
- *Completed*: The task has been completed successfully.
- *Faulted*: The process has been halted due to errors or bugs in the model.

At the beginning, the process is marked as idle as there might be some other process being executed in the background. As soon as the worker thread is ready to be implemented, it is scheduled based on when it has been received. Once the task is out of the scheduling status, the model starts analyzing the task and status is changed to running. Running is the important phase of task process, after it has been analyzed successfully, a completed status is sent as a feedback to the user. At times, the model might prompt a message stating that the process has been stopped due to some reason, this status flag is known as faulted.

Algorithm 1: Time-Dependent

```
1   Data: Task tuples {D(k, m)} where k is task id and m
2   is task status. A built-in task-based thread pool.
3   procedure TSched (D(k, m), tc)
4   initialization (set all task status to idling);
5   while the stopping criteria have not been met
6   for all task k
7   if task k is idling or completed, then set status of task
8   k as scheduled; insert task k to the thread pool.
9   end-for
10  sleep for tc seconds.
11  end-while
12  await thread pool to complete
13  end-procedure
```

Fig 2: The Snippet for Time-dependent scheduling

The task is performed only if the data required is received. A number of tasks are processed concurrently and the flow of data is continuous. Fig.3 shows the snippet used for implementing the algorithm. The worker threads help in categorizing the tasks in the queue based on when and the order in which they are received. The results of these tasks are added to the end of its own output queue, which keep stacking up with each output appended. The downstream nodes further receive tasks that were not processed earlier, these tasks are first analyzed by the worker threads which checks whether these have been executed or not. The downstream nodes marks the results received from the worker thread as "dirty" and these tasks are queued for processing later by the worker thread. to summarize, the scheduler thread sends the extracted task to the worker thread and the processing of the task is started. The results collected from the data is pushed to the scheduler thread. The results are recursively forwarded to the nodes, which is later checked if it has been forwarding the results or not. If the data is pushed successfully to the scheduler thread, it either moves to the next task or the process is ended. The Fig.3 represents the recursive running of the algorithm to process the tasks.

Algorithm 2: Data-Dependent

Data: Task tuples {S(*m, l*)} where *i* is task id and *s*
is task status. A built-in task-based thread pool.
procedure DSched (S(*m, l*))
initialization (set all task status to *idling*);
while the stopping criteria have not been met
for all task *m*
if task *m* is *idling* or *completed*, **and** it was marked as "dirty", **then** set status of task *m* as *scheduled*; insert task *i* to the thread pool.
end-if end-for
end-while
await thread pool to complete
end-procedure

Fig. 3: Snippet for Data-dependent scheduling

The Time-dependent and Data-dependent algorithm can be used together as a hybrid without the loss of a generality.

IV. MODULAR DESIGN APPROACH FOR MATHEMATICAL MODEL

The raw data obtained is coined as dirty data which mostly consists of inaccurate data or irrelevant data. Cleaning the data and structuring it is the first step while processing the data which is followed by data aggregation. The data crunching method give the output in the form of bar data which consists of close price, open low and high. These parameters are determined within a certain interval of time and the data obtained as an output is further compressed to make it feasible to work under high pressure or work load. To prepare and process data under different parameters and conditions, we have developed a model to ease the execution of large datasets by detecting outliers and removing or truncating the data not relevant. We have implemented autoregressive integrated moving average model to standardize the whole process and perform different use cases used in testing. Different levels of computing is used to perform data analysis with such humongous amount of data. All of these in short depict the models, algorithms and methods used in processing the data.

A. Modules

In this project, we have used eight modules and all of these have been discussed further. The first node is the root node which used to initiate a graph in the model. It is used to retrieve the user demands in given time intervals such as the source name, database information and stock details. Pipe-in node is followed by latter which splits the messages and rearranges them based on the type of message and stock notations used. A create-event node is used to process ping messages as per the given time periods by the user. A ping-match-event node is used to cross check the data flow with the ping-processing events and obtains data flowing to the downstream nodes.

Based on the criteria of the data given as an input, the link node filter the data and reformats the data in the form of symbols and remove or add data if necessary. To find the means and medians of the data, aggregation node is being used. The final result of the processed data is given through the output node, the output data can be of any format or as specified by the user. In additional to these, miscellaneous nodes are created to perform arithmetic operations. These modules are individually programmed and tested for their efficiency, all the modules are interdependent and cannot work properly without the whole set together.

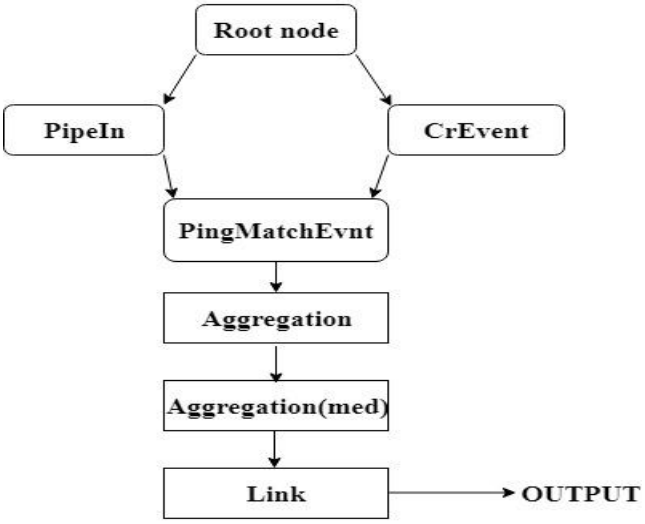

Fig. 4: Flowchart to understand the ping messages simultaneously sent as feedback from the database

B. Testing Cases

The test begins with *Case 1: Calculation of volatility of log returns from data using ping messages.* The raw data is aggregated into bar prices from which the log returns are computed. This calculation exhibits a periodic pattern in the volatility. The flow of the process begins from the root node and advances downstream. A sequence of requests are initiated by the client and during the runtime, the request is sent to the server to work on.

Case 2: Calculation of seasonal volatility of returns from a database, the data crunching technique is time consuming thus the work is split among the processors. The database platform is split to create two instances, which make the process complex but efficient in processing it faster.

In *Case 3: Calculation of volatility of grouped data using ping messages*, a sequence of returns are considered as an equal set. The sample variance is tested in a moving window. The time series information is programmed to the root node and is formatted by the pipe-in node. A personal estimator is created with the help of a weighted volatility. The volatility is sent as an output in the format that the user specifies.

Case 4: Autoregressive integrated moving average model is used when the data is non-stationary and the regression is on its own lag.

V. EXPERIMENTS AND TESTING

The tests are performed on information and huge datasets retrieved from Bombay stock exchange dated from 2017 to 2018. The experiment setup uses a Dell PowerEdgeR720 and an Intel Xeon E5-2602 at 1.9GHz, the drive used is SATA hard drive of 500GB.

In this prototype, two parameters namely throughput and latency is used to test the efficiency and compare with the existing model in the market. Throughput gives us the number of ping messages received per second or in milliseconds. The throughput and latency for typical workload is tested and observed, the same has been implemented at different levels of the workflow. The latency shows the average readings of the processing done to get the data in the track.

The data received from BSE India can be categorized into trade data and best bid and offer data, each of these messages require 35 to 60 bytes of memory. The testing of this model can be done by different methods such as black box, white box and integration testing. Testing can help in finding the errors and anomalies in different modules before completely executing the prototype.

Black-box testing is performed to check that the graph links have been associated properly and there is equivalent partitioning done for the nodes. Black-box helps in finding the boundary values of the graph and compare it with the counterparts. White-box testing is done to find the flow of the graph and how the data has been distributed among the nodes, this further leads to deriving the necessary test cases. We can also find the cyclometric complexity and determine the graph matrices control. The integration test is done while creating the program and simultaneously checking for the errors.

Type	# of Stocks	Latency (µs/tick)	Throughput (Gbps)	ARIMA Calc Time (second)	Process Time Per Stock (second)
Trade	500	4.32	0.073	263.12	0.52
	100	3.80	0.083	63.54	0.59
	10	4.44	0.075	10.58	1.00
BBO	500	0.75	0.629	698.94	1.54
	100	0.63	0.745	151.92	1.73
	10	0.67	0.709	25.06	3.26

Fig 5: Performance between BBO and Trade data

VI. RESULTS AND DISCUSSIONS

The conducted experiments had worked on all the four cases that were defined in the earlier sections. The results of the first case show that a several millions of ping messages were retrieved from an in-house server as the throughput and the frequency steadily increases from the latter. We can infer that BBO has higher frequency than the trade data.

In the second case, the results have been similar to the first case of volatility of log returns except that the process has been split between the processors and the work is done much faster and efficiently. The characteristics of this group show that frequency of BBO in terms of frequency per second is constantly increasing at a higher rate than the trade data. This data helps in finding the feedback ping messages that were retrieved at the certain timestamp.

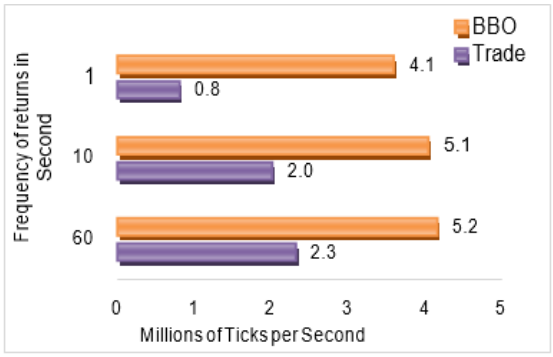

Fig. 6: Throughput in million messages per second

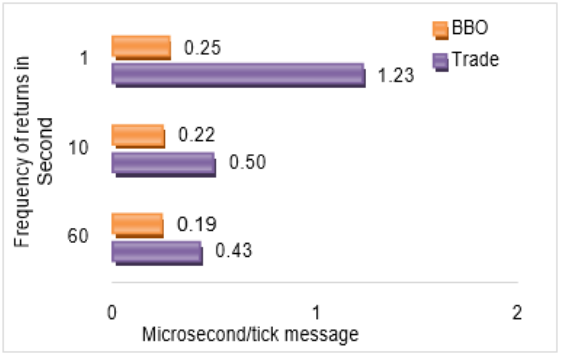

Fig. 7: Latency in microsecond per message

The impact of stock symbols can be seen in third case through the latencies and throughput received with the help of ping messages. A high latency of few million messages is experienced and low rates of latency is developed over the intervals. In this test, many stock datasets were implemented and split between the processors to execute they heavy load datasets faster. There is high scalability of performance when the processing amount of data is large.

The fourth case is implemented with the help of Autoregressive integrated moving average model which has the number of stocks varied for the test.

The ARIMA models takes account of 80% to 90% of the total processing time, also with an increase of stock the throughput and latency changes irregularly but the wall clock time increases clearly as a result of intensive computing. The models can also be compared with the Matlab R2017 and LiClipse to study the lag of time and number of ping messages retrieved and sent.

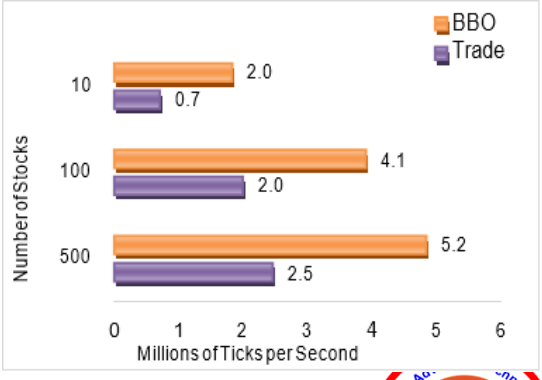

Fig. 8: Throughput in millions of ticks per second for ARIMA

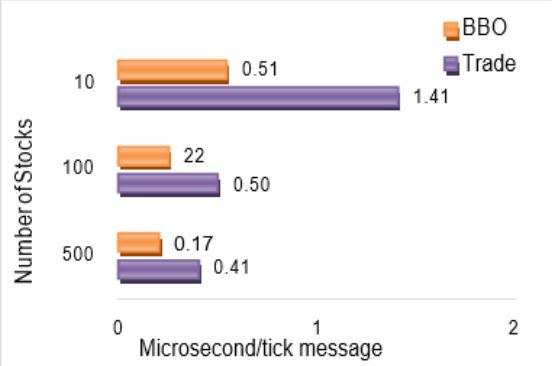

Fig. 9: Latency in microsecond per tick message for Trade and BBO

VII. FUTURE ENHANCEMENTS AND CONCLUSION

We have discussed about StatCloud, which uses a combination of Data analytics tactics with a data-driven mode of executing the model. Parallel computing facilitates data parallelization, which aids in processing multiple data tasks at the same time that is crucial when working with huge amounts of data. To make this process possible and less intricate, we have built a model based on Mathematical finance that helps in seamlessly working with data in business firms or any industry that solely relies on the insights from the data that they receive and produce.

The prototype that we worked on has exhibited high throughput in the form of millions of ping messages received as a feedback and a sub-microsecond latency is received. We have also demonstrated how data-driven execution takes place in a parallel computing environment, importantly, this model has a modular design designated to support data crunching methods and data modeling in Mathematical finance. Overall, our project design fulfills the need to have a prototype that is able to handle intricate events faced by the big data engineers in the field of Mathematical finance. The project provides ways in which the statistical engineers and enthusiasts can use such models to develop insights and weave stories from the data they possess.

Currently, this project works with the help of Python, LiClipse and SQL. It is important to expand the working of this model in different platforms and support multi-language processing. Many leading companies such as IBM and Kaggle use Python as their primary language to process data and build modules. MATLAB is another such popular programming platform that has been used to perform arithmetic operations on the data directly from the source file without importing. StatCloud consists of template libraries present in Python such as SciPy, MatLib and NumPy and selective functions from LiClipse.

There is a lag between the ping messages from LiClipse and the StatCloud, in future models there has to be consistent feedback of data from all the modules, this can happen by using fast computing principles. From the testing and the experiments performed, we can say that the processing of the StatCloud is much faster than the traditional method of using LiClipse or Matlab. Computer accelerators are evolving and it is important to update the module periodically to support computer accelerators such as Xeon Phi by Intel. With a pursuit of improvement and development of the StatCloud, the model produces new horizons and insights to accelerate the growth of their workflow.

REFERENCES

1. W. Härdle, T. Kleinow, and G. Stahl, Applied Quantitative Finance: Springer Berlin, 2002.
2. E. Thorp, "A perspective on quantitative finance: Models for beating the market," The Best of Wilmott, p. 33, 2005.
3. B. Fang and P. Zhang, "Big Data in Finance," in Big Data Concepts, Theories, and Applications, S. Yu and S. Guo, Eds., ed Cham: Springer International Publishing, 2016, pp. 391-412.
4. X. Shi, P. Zhang, and S. U. Khan, "Quantitative Data Analysis in Finance," in Handbook of Big Data Technologies, A. Y. Zomaya and S. Sakr, Eds., ed Cham: Springer International Publishing, 2017, pp. 719-753.
5. J. Chevalier and G. Ellison, "Risk taking by mutual funds as a response to incentives," Journal of Political Economy, vol. 105, pp. 1167-1200, 1997.
6. R. A. Brealey and E. Kaplanis, "Hedge funds and financial stability: An analysis of their factor exposures," International Finance, vol. 4, pp. 161-187, 2001.
7. T. J. Chemmanur and P. Fulghieri, "Investment bank reputation, information production, and financial intermediation," The Journal of Finance, vol. 49, pp. 57- 79, 1994.
8. B. Brown, M. Chui, and J. Manyika, "Are you ready for the era of 'big data'," McKinsey Quarterly, vol. 4, pp. 24- 35, 2011.
9. BSE (2016-2018). BSE India Market Data. Available: https://www.bseindia.com/market_data.html
10. P. Brandimarte, Numerical methods in finance and economics: a MATLAB-based introduction: John Wiley & Sons, 2013
11. X. Dong, "New development on market microstructure and macrostructure: Patterns of US high frequency data and a unified factor model framework," State University of New York at Stony Brook, 2013.
12. O. E. Barndorff-Nielsen, "Econometric analysis of realized volatility and its use in estimating stochastic volatility models," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 64, pp. 253-280, 2002.
13. K. Vijayakumar and C. Arun, "Continuous Security Assessment of Applications in Cloud Environment", International Journal of Control Theory and Applications, ISSN: 0974-5645 volume No. 9(36), Sep 2016, Page No. 533-541.
14. R.Joseph Manoj, M.D.Anto Praveena, K.Vijayakumar, "An ACO–ANN based feature selection algorithm for big data", Cluster Computing The Journal of Networks, Software Tools and Applications, ISSN: 1386-7857 (Print), 1573-7543 (Online) DOI: 10.1007/s10586-018-2550-z, 2018.
15. K. Vijayakumar and V. Govindaraj, "An Efficient Communication Technique for Extrication and Cloning of packets on cloud", International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.66 May 2015

AUTHORS PROFILE

Shiny A, Assistant professor at SRM IST Chennai. Research interests are on Network security, Data Mining and Digital System Design. She has published papers in IJDCN. Shiny is a university rank holder in M.E CSE.

Rikhil G R, student at SRM IST Chennai in the department of Computer Science and Engineering. He has worked on projects in mobile computing and analytics platform.

Namita Vagdevi Cherukuru, student at SRM IST Chennai in the department of computer science and engineering. Namita has been working on research projects in the field of Machine Learning and Computer vision.

Alen Mammen Ivan is a student of SRM IST Chennai in the department of Computer Science and Engineering. Alen is proficient in DBMS and has published a paper on Database servers.

Sunil Prakash J, a student of Computer science and Engineering in the department of Computer Science and Technology. Sunil has great interest in software development and gaming Technology.