iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 9 **Issue:** VI **Month of publication:** June 2021

**DOI:** https://doi.org/10.22214/ijraset.2021.36049

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429*
*Volume 9 Issue VI June 2021- Available at www.ijraset.com*

# Design of Optimized Trading Strategies with Web Assembly

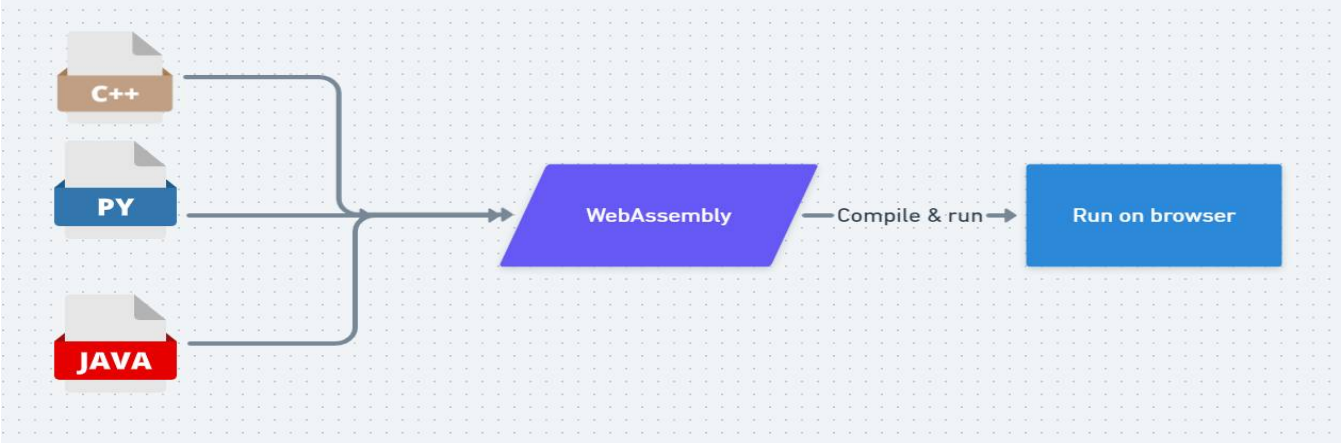
Shubham R Rahate
*Department of Computer Application, Bharati Vidyapeeth's Institute of Management and Information Technology, Navi Mumbai, Maharashtra, India*

***Abstract*— Web Assembly is growing and also the most widely studied area which interests many developers when it comes to performance and speed to make web development fast as ever. When it comes to speed and performance algorithms can perform faster computations. Algorithmic trading executes trade at a faster speed. It can buy and sell stocks within a fraction of milliseconds. However, selecting the right tools and technologies is extremely important in algorithmic trading. There are trading strategies which we can use to optimize our trade and increase the return gained on buying and selling stocks. But, choosing an efficient programming language is substantially important. A programming language with a low latency can leverage the trade. Most commonly used languages for algorithmic trading are C/C++, Java, C#, Python. Speed and performance are an essential factor in algorithmic trading. The main purpose of introducing web Assembly in trading as discussed above is speed and performance. Web Assembly is a low-level binary instruction which can execute any program on the web and it can deliver native like performance on the internet. Using Web Assembly, we can compile any code written in languages like C/C++, C#, Java, and python to wasm (Web Assembly executable file) and run on the browser. Web Assembly was developed by W3C, Mozilla Corporation, and Google.***

**Keywords**— ***Web Assembly, Algorithmic Trading, Design, Web Technology, Finance, Stock Market, Computational Finance.***

## I. INTRODUCTION

This research paper introduces the concept of Web Assembly with optimized trading strategies. Since, Web Assembly can provide native-like performance on the browser. We develop algorithmic trading strategies in our native environment using languages like C/C++, C# or Python and compile it into the .wasm module to generate the wasm file, then finally run on the browser. The basic flow of execution of modules in Web Assembly is shown below.




### *A. Benefits of using Web Assembly:*

1) Web applications run faster.
2) Provides huge performance for running web assembly.
3) No language obstacles, any programming language can be compiled to Web Assembly. This means it supports multiple languages.
4) Knowledge of a new programming language is not required.
5) Follows W3C open-source standards.
6) Heavier applications can run smoothly on all contemporary browsers and devices using Web Assembly.
7) Web applications can be scaled up to a large number of users. Hence, there are no limitations on the user.

8) Using wasm high level code is converted into low level binary instructions. Hence, it can also be obtained in text format.

*B. Benefits of Algorithmic Trading*

1) Uses algorithms (step by step instruction) or automated tools to place order in the exchange.
2) Performs trading without human intervention. Hence, no human emotions involved while trading. Therefore, it can buy and sell a large volume of stocks specified by certain conditions and time-interval.
3) Makes decisions based on the analysis and time interval.
4) Trading is faster since it uses electronic systems to buy and sell the stocks.
5) It can increase profit or ROI (Return on Investment) on the stocks.
6) Monitors market volatility.

## II. LITERATURE SURVEY

We have conducted a survey and found that C/C++ is the most commonly used languages for trading . Other commonly used languages are Java, C#, and python. C/C++ provides high performance because of its memory management and strong implementation of data structure. On the other hand, Java performs well due to its object-oriented nature. C# based on C++ and Java also do a better job and we can build faster native apps with the help of it. When it comes to python, it is slightly slower than C/C++. But it can be used for performing technical analysis based on the historical datasets. Big trading firms perform trading on FPGA which can provide ultra-low latencies for High Frequency Trading (also a subset of algorithmic trading). No matter which languages we choose for algorithmic trading. We can compile and run the strategies in Web Assembly using any languages.

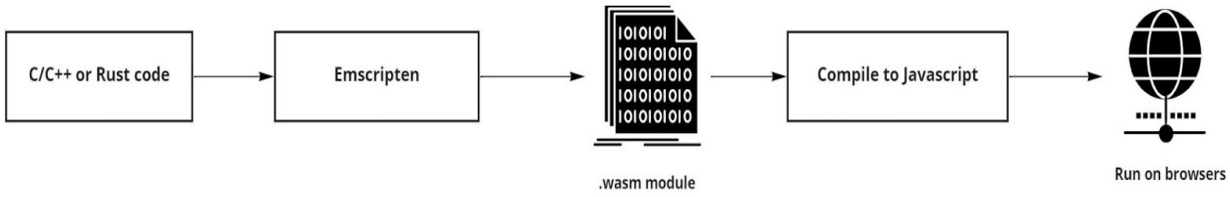
## III. PROPOSED WORK

Web Assembly is a low-level binary set of code combined with high level programming languages alongside either with JavaScript runtime or Rust to provide faster native-like performance on the browser.

*A. Working of Web Assembly*

Web Assembly also known as wasm uses emscripten library. Emscripten is an LLVM to JavaScript compiler. It takes C/C++ or Rust code and converts them into low level binary instruction which generates a program.wasm file then finally compiled to a JavaScript to run on the contemporary browsers [7]. Fig.4.1.1. Illustrates the standard working of Web Assembly.

*B. Web Assembly Architecture*




*C. Algorithmic Trading Working*

Algorithmic trading is a computerised trading or automated trading tool which uses algorithms (a step by step set of instructions for executing a task) for execution of orders (buying and selling stocks) within predefined timings based on market conditions. It uses a set of strategies and time intervals for either buying or selling of securities. The order takes place at a securities exchange. It uses advanced mathematical models and complex algorithms for analysis, decision making for buying and selling of securities.

For. e.g. In a moving average of a 50 day stock price above 200, then buy the stocks. (Moving average is a historical data point that has a stock price data on average of 50 days). For a moving average of 50 days if the stock price is below 200, then sell the stocks.

The algorithm for buying and selling the stock in an trade exchange:

1. Function buy_stocks:
    a. IF(moving_avg > 200):
        i. purchase_stock;

b. ELSE IF(moving_avg < 200):
   i. sell_stock;
c. ELSE:
   i. STOP;

*D. Requirement for Algorithmic Trading*

*1)* Powerful Computer for executing trading strategies.
*2)* Network connection for placing the order into the exchange
*3)* Knowledge of basic stock market concepts.
*4)* Algorithmic analysis for developing optimized strategies.
*5)* Historical dataset for analysing the stock prices.

*E. Algorithmic Trading Strategies*

Trading strategies are incorporated to identify the trends and the volatility of the market. There are lots of trading strategies such as moving average trading strategies which can be used to monitor the stock prices of a particular stock and initiate the order if the certain specified conditions are satisfied. Trading strategies are mathematical formulas used to perform calculations and analysis to identify trends and make decisions for either buying or selling of securities [1]. Fig 2. Illustrates the working flow of algorithmic trading.

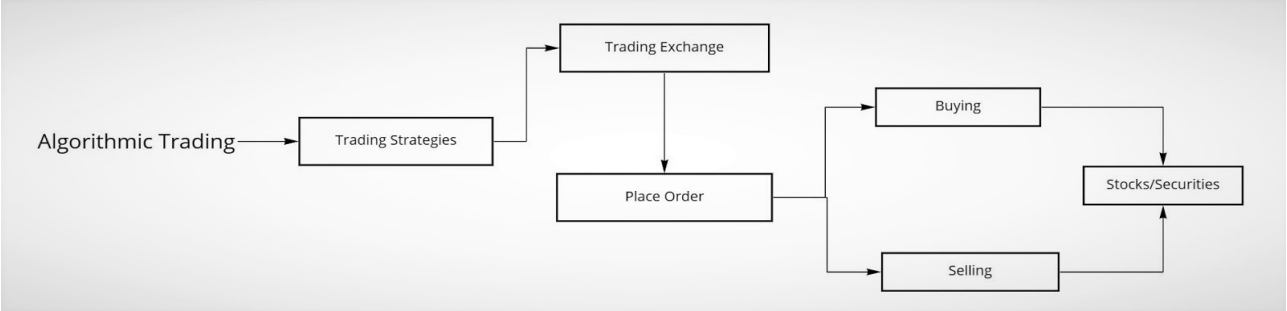



Fig.2 Algorithmic Trading working flow.

In this paper, we have conducted an experiment on the moving average. Moving average uses a statistical trading strategy that uses an average of 20-day or 50-day average. It analyses the average data points from the full dataset.

*F. Types of Moving average*

*1) Simple Moving Average:* The simple moving average is a moving average in simplest form. It takes the mean of all sets of values specified by the moving average time interval of the historical dataset. In other words, it calculates the sum of numbers or prices in financial instruments divided by no. of time period.

The mathematical formula for SMA(Simple Moving Average) is:

$$\mathbf{SMA}=\frac{A_1 + A_2 + A_3 + \ldots + A_n}{n}.$$

**Where:**
A = numbers or prices
n = no. of time period

*2) Cumulative Moving Average:* While SMA is a technique that gives output on the past observation data. There is another technique known as Cumulative Moving Average. As per the observation, it has found that CMA is not the best technique

to be used for analysing trends. The reason for being this, it takes all the weighted average values from the dataset until the current one. It considers all prior observations.

The equality weighted average of the sequence of 'n' values up to the current time is given by

$$\mathrm{CMA}=\frac{x_1+x_2+x_3+\ldots+x_n}{n}$$

For every new values that comes can be calculated by

$$\mathrm{CMA}_{n+1}=\frac{x_{n+1}+n*CMA_n}{n+1}$$

3) *Exponential Moving Average:* The Exponential moving average is a moving average that gives more weight to the recent prices. As a result of this, it can give better models or capture movement of the trend in a much faster way. EMA calculates the latest price trends better than SMA. EMA is better as compared to CMA because it outputs latest trends based on the higher weighted points in the datasets.

$$\mathrm{EMA}=\left[V_t*\left(\frac{s}{1+d}\right)\right]+\left[EMA_y*\left(1-\frac{s}{1+d}\right)\right]$$

**Where:**

EMAy = EMA yesterday
S = smoothing
D = no .of days
Vt = Value today
EMAt = EMA Today

*Advantage of Moving Average*

- Analyzes the latest trends in the stock market
- Initiates the decision for buying and selling of stocks based on weighted average values
- Moving average helps in smoothing out the data
- As it analyses latest trends it can determine the demand of commodities or goods in the market.
- Simplicity of data points makes it easier to plot several data lines at a same time.

*Disadvantages of Moving Average*

- It is more vulnerable to false signals
- It is used by intraday traders for trading in short interval time frame.

## IV. IMPLEMENTATION OF ALGORITHMIC TRADING IN WEB ASSEMBLY

We used C++ language to implement a SMA (Simple Moving Average) strategy. To run a C/C++ program in web assembly first we need to use the Emscripten library [3]. To run the C/C++ program in the browser we also required a JavaScript environment. In Web Assembly, we run C/C++ programs under the JavaScript environment. Trading strategies are implemented in C/C++ and then compiled to Emscripten to generate a wasm file(a binary instruction) and then combined with JavaScript to run on the contemporary browsers. For implementing the code into python, we can use the Wasmer library to convert our python code into Web Assembly. Wasmer uses Rust engine for Compilation of python into a Web Assembly [8].

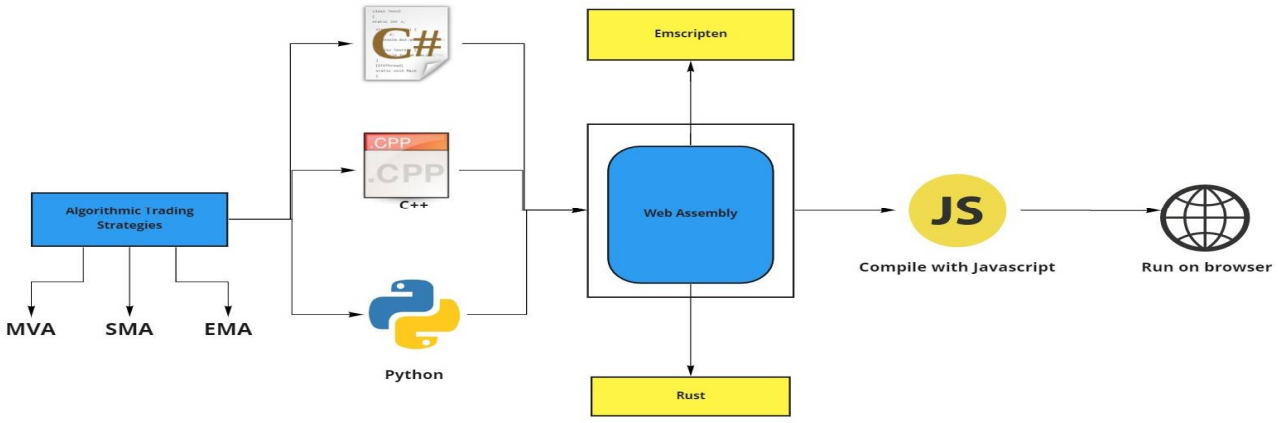


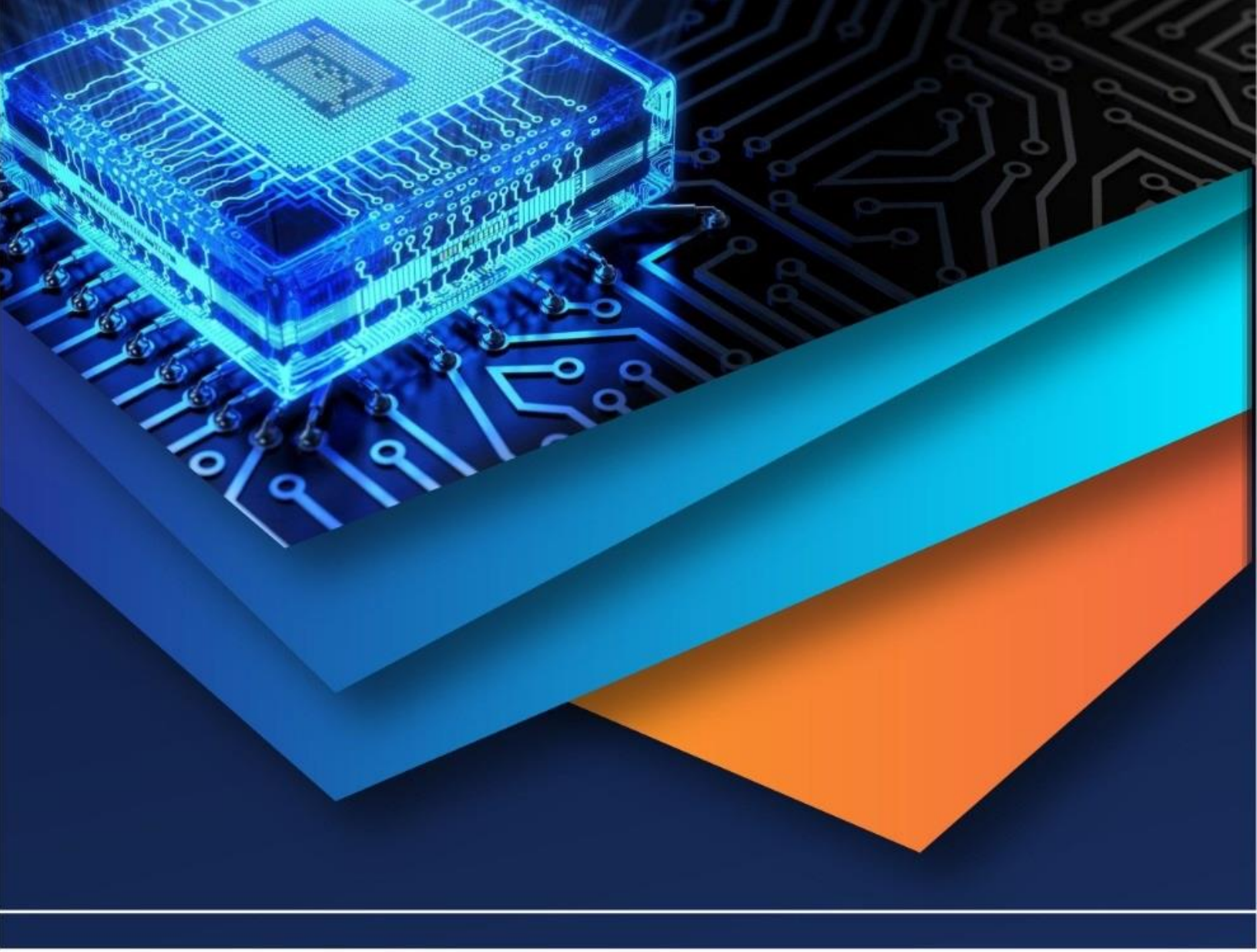

Fig 3. Implementation of Algorithmic Trading in Web Assembly.

## V. CONCLUSIONS

In this paper, we conclude that using C/C++ with Web Assembly boosts the performance of the web applications in the browsers. Running native code with Web Assembly has given native-like performance on the browser. Running heavier applications with Web Assembly is much faster. It is also found that C/C++ gives 20% better performance compared with other languages like C#, Java, and Python. C# gives optimum performance when running with Web Assembly. It is also noted that python performs slower than C/C++, however, it can still boost 20% performance than native applications. Python can be best used for performing technical analysis and gaining insights into the data. From this, we also conclude, Web Assembly gives better performance. For better performance and speed, selecting an efficient programming language in a native environment is essential.

## REFERENCES

[1] Michael L. Halls-Moore, PhD., *Successful Algorithmic Trading: A step by step guide to developing systematic trading strategies using the python programming language.*
[2] Edward Leshik and Jane Cralle, *An introduction to Algorithmic Trading basic to advance.*
[3] Abhinav Jangda, Bobby Powers, Emery D. Berger, and Arjun Guha., *Not So Fast: Analyzing the Performance of Web Assembly vs. Native Code,*https://www.usenix.org/conference/atc19/presentation/jangda, University of Massachusetts Amherst. (2019).
[4] Web Assembly Design, (2020). https://github.com/WebAssembly/design
[5] Andreas Rossberg(editor) Web Assembly Community Group, Web Assembly *Specification Release* 1.1 (Draft 2021-06-01) .
[6] Micha Reiser and Luc Bläser., Accelerate JavaScript Applications by Cross-Compiling to Web Assembly, (2017)
[7] Compiling C/C++ module into Web Assembly, (2021) https://developer.mozilla.org/en-US/docs/WebAssembly/C_to_wasm
[8] Wasmer in python, (2021). https://github.com/wasmerio/wasmer-python



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY