ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

OPTIMIZATION OF SALES DATA FORECASTING COMPUTATION PROCESS USING PARALLEL COMPUTING IN CLOUD ENVIRONMENT

I Kadek Susila Satwika[1], I Putu Susila Handika[2]

[1]Program Studi Sistem Komputer, Institut Bisnis dan Teknologi Indonesia
[2]Program Studi Teknik Informatika, Institut Bisnis dan Teknologi Indonesia

email: susila.satwika@instiki.ac.id[1], susila.handika@instiki.ac.id[2]

Abstract

The Holt-Winters Exponential Smoothing algorithm optimised using the Modified Improved Particle Swarm Optimization (MIPSO) algorithm is an algorithm that is able to provide good sales data forecasting results. However, there is a problem that when the iteration process is carried out using 1 computer, it takes a long time to finally get the test results. It is necessary to optimise the computational process to get more optimal and efficient results. This research will combine parallel computing technology and cloud computing technology to help speed up the computing process. The results of this research show that the more server used, the greater the reduction in execution time that occurs, because heavy computing tasks can be distributed more efficiently to many machines. This is evident from the comparison between single server and parallel server. Then the combination of more cores and servers produces the most optimal configuration in accelerating computation.

Keywords : Holt-Winters Exponential Smoothing, Modified Improved Particle Swarm Optimization, Parallel Computing, Cloud, Server

Received: 24-09-2024 | **Revised:** 08-11-2024 | **Accepted:** 21-11-2024
DOI: https://doi.org/10.23887/janapati.v13i3.85278

INTRODUCTION

Technology plays a very important role in the world of business and sales, allowing companies to increase efficiency, expand market reach, and improve customer experience. Data analytics and big data technologies enable companies to gain deep insights into market trends and customer behaviour [1]. With these technologies, companies can collect, store, and analyse large and varied amounts of data, including data from sales transactions, customer interactions, social media, and more. Through careful data analysis, companies can identify emerging market trends, anticipate customer demand, and adjust their sales strategies in real-time. In addition, by understanding customer behaviour based on historical data, companies can make more accurate segmentations, personalise services, and optimise marketing strategies to achieve higher levels of customer satisfaction. This is often referred to as the sales data forecasting process.

Sales forecasting is a critical element of business strategy that enables companies to effectively plan production, manage inventory, and make informed decisions. Accurate sales forecasts allow organizations to align their production schedules with anticipated demand, thereby minimizing excess inventory and associated costs [2][3]. Sales data forecasting has undergone significant development and become a common practice in various industries. Various companies use forecasting to plan production, inventory management, and marketing strategies. Technological advances, especially in the fields of data analytics and artificial intelligence, have expanded forecasting capabilities. Various methods are used to predict future sales with a higher degree of accuracy. In addition, the use of big data in forecasting is also becoming common, allowing companies to process and analyse large and varied amounts of data. With accurate forecasting, companies can avoid excessive stock costs, improve customer satisfaction with the right inventory, and optimise marketing strategies. Based on this, research on the sales data forecasting process is very important to do.

There are many algorithms used in forecasting data [4][5][6]. Of the many algorithms, the Holt-Winters Exponential Smoothing algorithm was chosen in this study because the data processed is sales data with seasonal data patterns. Furthermore, the

Modified Improved Particle Swarm Optimization (MIPSO) algorithm is added to optimise Holt-Winters parameters. In previous research, testing has been carried out for this algorithm [7]. Previous tests were carried out on local computers. The result of the test is that the Holt-Winters Exponential Smoothing algorithm with optimisation using the MIPSO algorithm is able to provide good sales data forecasting results. However, to get forecasting results there are problems that arise during testing.

The problem that arises in previous research is that it takes a long time to finally get the test results. This is because the testing was done only on a local computer. The data processed when doing computing is very large, while 1 computer is not able to process large data efficiently and optimally. So it takes a long time to do the computation process.Â In addition, currently th application has been widely applied to the cloud, and the previous algorithm was only tested on a local computer, so it is not yet known how the performance of the algorithm in the cloud environment. It is necessary to do a test to get more optimal and efficient results.

Based on the problems previously described, to forecast sales data accurately and efficiently, parallel computing technology is needed because serial processing methods are unable to fulfil the requirements of processing large amounts of data [8]. In the forecasting process, there is a data mining process that uses algorithms to process large amounts of data, where the result of the process is valuable information [9][10]. So in this study, computational testing of the Holt-Winters Exponential Smoothing algorithm will be carried out with optimisation using the MIPSO algorithm using parallel computing technology.

Then this research also uses cloud technology to implement parallel computing technology. The application of parallel computing technology in the cloud provides significant advantages in the sales data forecasting process. Cloud computing significantly enhances the sales data forecasting process by providing scalable resources and advanced analytical capabilities. The integration of machine learning on cloud platforms, such as Microsoft Azure, allows for the automation of sales predictions, reducing reliance on subjective human evaluations and improving accuracy through data-driven methodologies [11][12]. By utilizing cloud infrastructure, organizations can access vast datasets and employ sophisticated algorithms that enhance predictive capabilities, ultimately leading to better-informed decision-making and strategic planning [13][14]. This shift towards cloud-based solutions not only streamlines the forecasting process but also supports real-time data processing, which is crucial for adapting to market changes swiftly [15].

In previous research, several studies have been conducted on data processing using parallel computing [16][17][18]. The research shows only tested using 1 computer by using variations in the number of CPUs. Not testing with more than 1 computer. Then, no one has done it for the sales data forecasting process. So this research focuses on the optimisation of a computational process of the Holt-Winters Exponential Smoothing algorithm and the MIPSO algorithm for sales data forecasting.

Based on the explanation above, this research was conducted to optimise the iteration process carried out by the Holt-Winters Exponential Smoothing algorithm and the Modified Improved Particle Swarm Optimization (MIPSO) algorithm. By using parallel computing technology in the cloud environment, it is expected to get optimal and efficient sales data forecasting results.

METHOD

In general, the sales data forecasting process using parallel processing in a cloud environmentis depicted in Figure 1.

1. Data Normalisation

Data normalisation is the process of transforming raw data into a more uniform scale, usually in the range of 0 to 1 [19]. This process is important to ensure that every feature or variable in the dataset has the same scale, so that no variable dominates or affects the optimisation process more than the others. In the context of optimisation algorithms such as MIPSO, normalisation helps speed up the convergence process as all variables are treated equally. Without normalisation, the algorithm may focus more on variables that have larger values, thus affecting the overall forecasting results. The normalisation process used in this research is Min-Max Scaling, where each value is converted into a certain range based on the minimum and maximum values in the dataset. The Min-Max Scaling equation can be seen in Equation 1.

$$\hat{X} = \frac{x - x_{\min\_global}}{X_{\max\_global} - X_{\min\_global}} \quad (1)$$

In forecasting using Holt-Winters Exponential Smoothing, normalisation also helps in avoiding calculation errors caused by large variations between variables, so that forecasting

results become more accurate. Once the data is normalised, the algorithm can work more efficiently in finding the optimal parameters to be used for forecasting.

2. Determination of MIPSO Algorithm Parameters

This step involves setting the parameters that will be used by the MIPSO algorithm, including the number of particles (population size), the number of iterations, as well as other parameters such as particle velocity. The number of particles used in this study are two scenarios, namely 100 particles and 500 particles. The more particles used, the more potential solutions explored by the algorithm in finding the optimal parameters for Holt-Winters Exponential Smoothing. However, more particles also means that the execution time will increase significantly, as each particle requires complex forecasting calculations [20]. With 500 particles, the algorithm will be able to explore a wider solution space, increasing the chance of finding a more optimal solution, but on the other hand, this process requires more computational resources

and time. Therefore, the number of particles should be chosen carefully, considering the balance between the desired solution quality and acceptable execution time. MIPSO uses particle positions and velocities to guide the search for an optimal solution based on a combination of the best solution an individual particle has ever found (personal best) and the best solution of the entire population (global best).

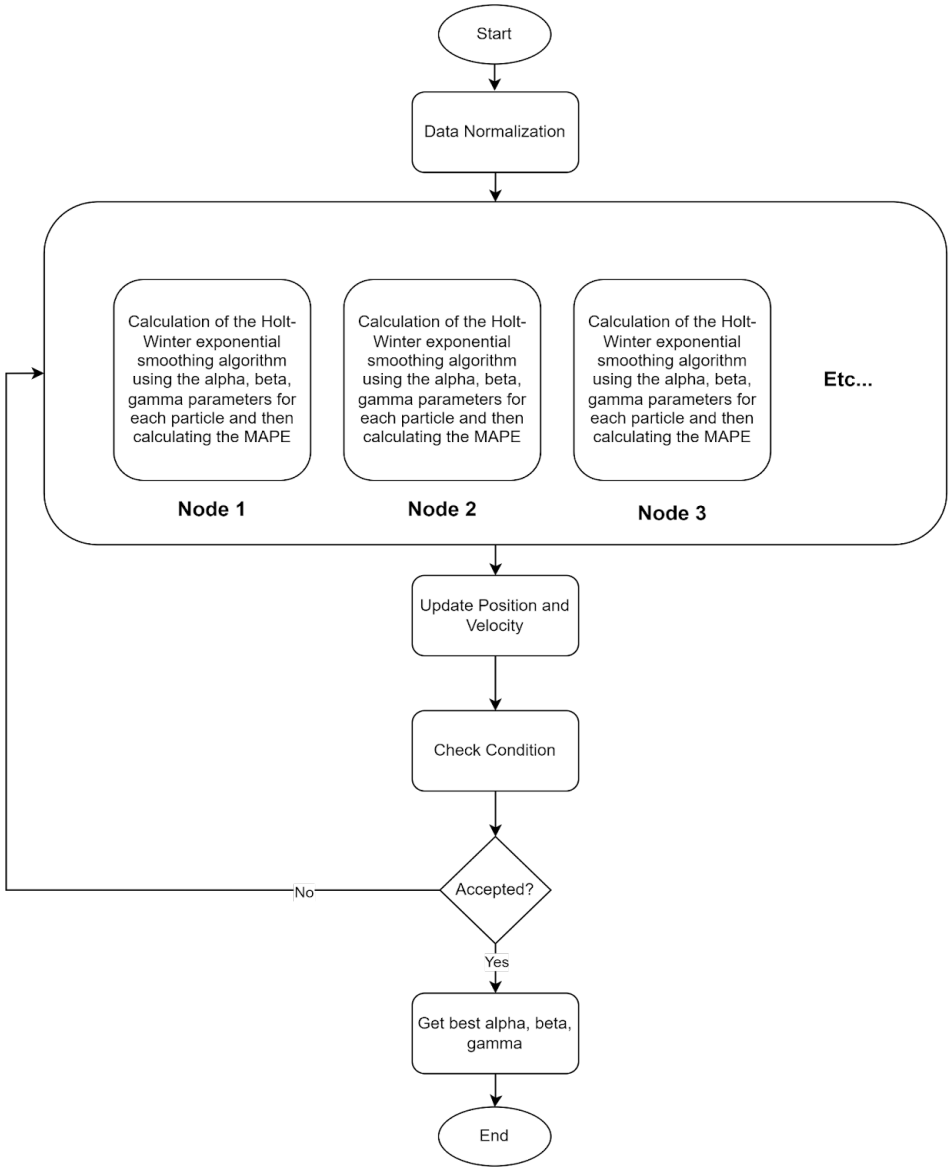

Figure 1. The Sales Data Forecasting Process Using Parallel Processing in a Cloud Environment

3. Implementation of Parallel Processing

Parallel processing is a computing technique that divides a calculation task into several processing units so that they can be run simultaneously, thus speeding up the overall execution time. Some research indicates that parallel processing enables attention to be distributed across several words at the same time, in contrast to serial processing models, which suggest a step-by-step approach [21]. In this research, parallel processing is performed using six Virtual Private Servers (VPS). Each VPS is assigned to process a subset of the particles generated by the MIPSO algorithm. The main task of each VPS is to perform forecasting using the Holt-Winters Exponential Smoothing method and calculate the forecasting error value in the form of Mean Absolute Percentage Error (MAPE). The parameters used in Holt-Winters Exponential Smoothing include alpha, beta, and gamma, which determine how level, trend, and seasonal data are treated in forecasting. With parallel processing, these tasks are divided across multiple machines that work independently but simultaneously, so the overall computation time can be shortened. Each VPS runs the calculation process for a certain number of particles, then the results are aggregated and sent back to the main machine. This technique not only increases the processing speed but also allows for more efficient use of computing resources, especially in handling large numbers of particles.

This research conducted 2 tests to find the length of execution time, namely:

- The first test is to compare the execution time between machines without parallel processing (1 single VPS) and machines with parallel processing. The machine with parallel processing uses variations in the number of VPS from 2 to 7 VPS. The number of particles used in the first test is 100 particles. The specifications of each VPS in experiment 1 can be seen in table 1.

Table 1. VPS Specifications Experiment 1

VPS	CPU (core)	Memory (MB)
1 VPS	1	2
VPS (Parallel)	1	2

- The second test is to compare the processing speed on a parallel VPS where the number of CPUs is made to vary by 1 core, 2 cores, and 4 cores. As for the number of particles used in the second test is a total of 500 particles. For the specifications of each VPS in experiment 2 can be seen in table 2.

Table 2. VPS Specifications Experiment 2

Testing	CPU (core)	Memory (MB)
1	1	2
2	2	2
3	4	2

4. MIPSO Particle Position and Velocity Update

After the VPS calculates the MAPE value for each particle, the next step is to update the position and velocity of the particles in the search space. At this stage, the MIPSO algorithm works to move the particles to a position closer to the optimal solution. The position update is based on two main factors: (1) **personal best**, which is the best position ever achieved by that particle, and (2) **global best**, which is the best position ever achieved by the entire population of particles. Particle velocities are updated using a combination of these two factors, adding random elements to ensure variety in the solution search. In this way, particles that are underperforming will move towards better positions, while particles that are already close to the optimal solution will maintain or improve their positions. This process ensures that the MIPSO algorithm will gradually approach the optimal solution from iteration to iteration. The equations for updating speed and position in MIPSO are shown in equations 2 and 3. Once the position and speed are updated, the algorithm is ready to start the next iteration.

$$v(i+1) = wv(i) + c_1.rand\big(pbest(i) - p(i)\big) + c_2.rand\big(gbest(i) - p(i)\big) \quad (2)$$

$$p(i+1) = p(i) + v(i+1) \quad (3)$$

Information:

$v(i+1)$: The velocity of the particle at iteration (i+1)
w : Inertia Weight
$v(i)$: The velocity of the particle at iteration i.
$c1, c2$: The cognitive and social acceleration factors
pBest : Individual best at iteration i
gBest : Global best at iteration i
$p(i+1)$: The position of the particle at iteration i + 1
p : The position of the particle at iteration i

5. Loop Back with the Latest Parameters

The looping process is restarted using the particle positions and velocities that have been updated in the previous stage. In the next iteration, the VPS again performs forecasting calculations using the Holt-Winters Exponential Smoothing (HWES) parameters generated by the updated particles. This process is repeated periodically until one of two conditions is met: (1) the algorithm reaches convergence, which is when the change in the optimal solution is very small or nonexistent, or (2) the maximum number of iterations set in the initial configuration is reached. Each time an iteration is performed, parallel processing is again applied to speed up the calculation process. Thus, the entire computational process runs more efficiently as the parameter updates are performed in parallel across multiple VPSs. The ultimate goal of this process is to find the optimal alpha, beta, and gamma parameter values to be used in the Holt-Winters Exponential Smoothing method, so as to produce the most accurate forecasting based on the data used.

RESULT AND DISCUSSION

The data used in this research is daily sales transaction data for 1 year. In this study, two tests were carried out, the first test was to compare the processing execution time of algorithms run without parallel processing, and algorithms run with parallel processing. The number of particles used in the first test is 100 particles. In the first scenario, the algorithm is run without parallel processing, where all calculations are performed sequentially on one machine. The execution time is measured based on how long it takes a single machine to complete the entire forecasting process using the Holt-Winters Exponential Smoothing method for all particles. The second scenario uses parallel processing, where the calculation process is distributed across multiple Virtual Private Servers (VPS). The particles generated by MIPSO are divided into several VPSs, with each VPS running the forecasting calculation simultaneously. The results from each VPS are then combined by the main engine to update the position and velocity of the particles. This test includes variations in the number of VPS from 2 to 7 VPS. The purpose of the test is to compare the execution time between the two scenarios and determine the impact of parallel processing on computational efficiency. The results of the first test are shown in Table 3. While Figure 2 shows a graph of the average execution time for each test.

The test results in Table 3 show the execution time of the algorithm with and without parallel processing, using various numbers of Virtual Private Servers (VPS). In the test without parallel processing (1 single VPS), the average execution time reaches 23.36 seconds, which is the longest time because all calculations are performed sequentially on one machine. When parallel processing was applied with 2 VPS, the average execution time decreased to 17.58 seconds, indicating an increase in efficiency. However, a more significant improvement starts to be seen when the number of VPS is increased. At 3 VPS, the average execution time drops to 16.12 seconds, and continues to decrease to 7.17 seconds at the use of 7 VPS, which is the fastest execution time.

Table 3. Test Results Between Single VPS and Parallel VPS

VPS	1	2	3	4	5	AVG Wall Time
	Wall Time	Wall Time	Wall Time	Wall Time	Wall Time	
1 (Single)	18,356430	21,780789	24,581319	25,121887	26,962149	23,360515
2 (Parallel)	18,451455	17,556244	17,537082	17,130937	17,245379	17,584220
3 (Parallel)	9,370723	23,841294	9,008284	18,695359	19,733433	16,129819
4 (Parallel)	6,924187	6,254065	18,267599	10,950443	13,402526	11,159764
5 (Parallel)	12,819219	5,498435	9,740178	4,897491	9,284691	8,448003
6 (Parallel)	8,178772	4,715277	4,630536	8,656173	10,438637	7,323879
7 (Parallel)	3,814405	8,400412	7,772227	8,240462	7,668662	7,179234

ISSN 2089-8673 (Print) | ISSN 2548-4265 (Online)
Volume 13, Issue 3, December 2024

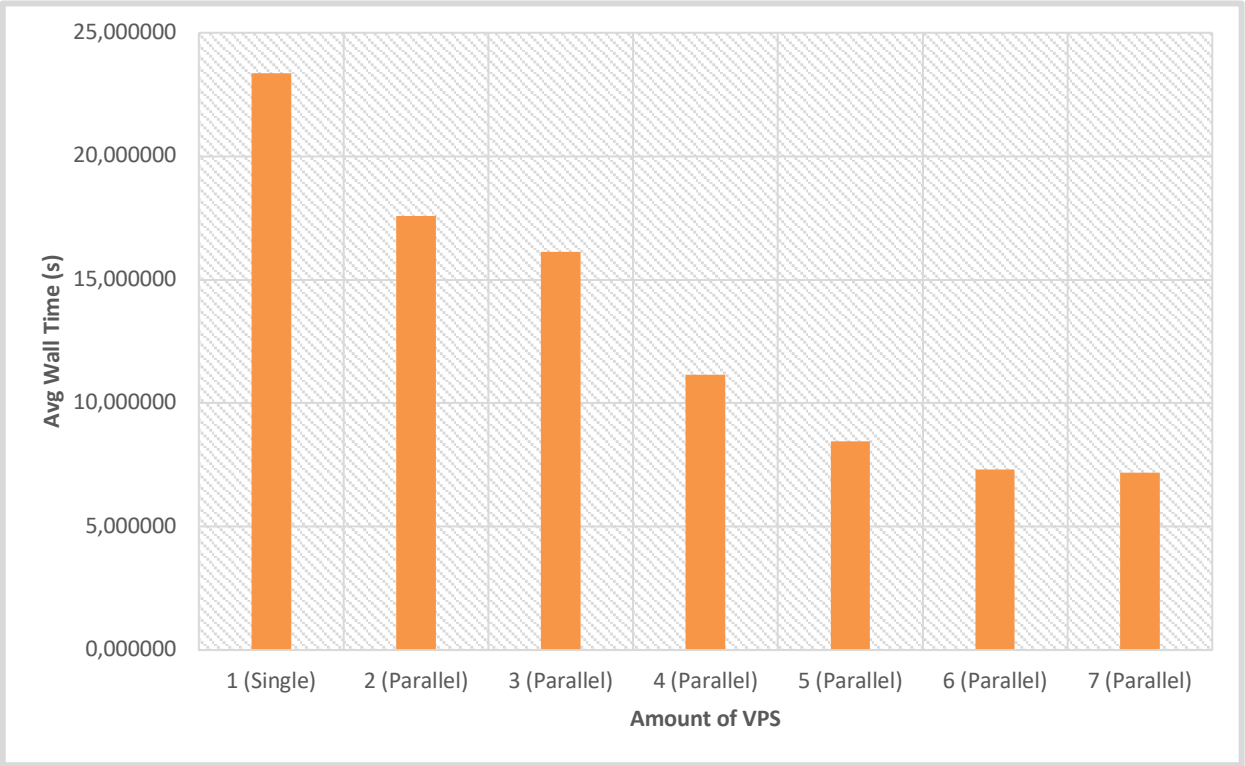

Figure 2. Graph of Test Results Between Single VPS and Parallel VPS

Figure 2 shows that the greater the number of servers used in parallel, the less execution time is required. This decrease in execution time reflects that the assignment of tasks to more VPS effectively speeds up the computation process, as the workload is shared across multiple machines working simultaneously. The use of 7 VPS gives the best results, indicating that parallel processing with more VPS significantly speeds up computation time. However, after reaching a certain number of VPS, the speed increase tends to be smaller due to the optimality limit of parallel processing. Overall, parallel processing proved to be very effective in reducing execution time, especially when more VPS are used, with the best results achieved at 7 VPS.

In the second study, researchers focused on exploring the effect of the number of CPU cores in each VPS as well as the number of VPS used on execution time and computational efficiency. To achieve maximum results, the number of particles used in the algorithm is 500 particles. The use of 500 particles aims to provide a large enough computational load on each VPS, so as to maximise the use of resources and test the ability of VPS to handle intensive calculation processes. Researchers wanted to find out whether the addition of cores to the VPS would have a significant impact on execution speed. By testing this scenario, we hope to find the optimal combination between the number of VPS and the number of cores that can speed up the execution time without compromising the quality of the results. This research also aims to identify the optimal limits of the number of cores and VPS, where further additions may no longer provide significant efficiency improvements. As such, this research provides greater insight into the effect of computing resource configuration on algorithm performance in the context of parallel processing. The results of the second test are shown in Table 4 to Table 6.

Table 4. Test Result Parallel VPS Using 1 CPU

VPS	1 Wall Time	2 Wall Time	3 Wall Time	4 Wall Time	5 Wall Time	AVG Wall Time
2 (Parallel)	188.481903	210.487925	199.835315	186.967507	187.936219	194.741774
3 (Parallel)	119.641999	120.004928	123.616700	113.330921	118.700213	119.058952
4 (Parallel)	78.700213	75.066566	78.600602	76.928911	78.803164	77.619891
5 (Parallel)	57.042980	53.617649	53.102503	55.466980	55.654282	54.976879
6 (Parallel)	26.513063	29.077122	27.214981	27.207473	28.979605	27.798449
7 (Parallel)	20.014084	22.253458	22.689154	19.090868	20.829710	20.975455

Table 5. Test Result Parallel VPS Using 2 CPUs

VPS	1 Wall Time	2 Wall Time	3 Wall Time	4 Wall Time	5 Wall Time	AVG Wall Time
2 (Parallel)	167.448885	165.821469	159.581198	162.374531	164.992291	164.043675
3 (Parallel)	90.254399	98.878628	100.984940	103.984932	94.192831	97.659146
4 (Parallel)	70.878313	71.989660	69.189284	72.992894	71.298386	71.269707
5 (Parallel)	51.872990	52.112349	50.783793	49.398280	53.928942	51.619271
6 (Parallel)	24.928933	25.923894	25.389288	23.983933	22.928935	24.630997
7 (Parallel)	19.928939	18.938478	18.983945	17.983398	18.197392	18.806431

Table 6. Test Result Parallel VPS Using 4 CPUs

VPS	1 Wall Time	2 Wall Time	3 Wall Time	4 Wall Time	5 Wall Time	AVG Wall Time
2 (Parallel)	143.277241	161.211375	145.217200	159.599451	160.432064	153.947466
3 (Parallel)	71.522495	70.288507	69.532607	70.259877	68.569875	70.034672
4 (Parallel)	60.948760	61.156850	59.659891	62.236588	61.154984	61.031415
5 (Parallel)	45.897660	42.125837	41.265930	44.246984	43.154873	43.338257
6 (Parallel)	20.258370	19.458797	21.125487	20.256480	21.468971	20.513621
7 (Parallel)	16.563223	17.125695	15.568637	15.125499	14.169855	15.710582

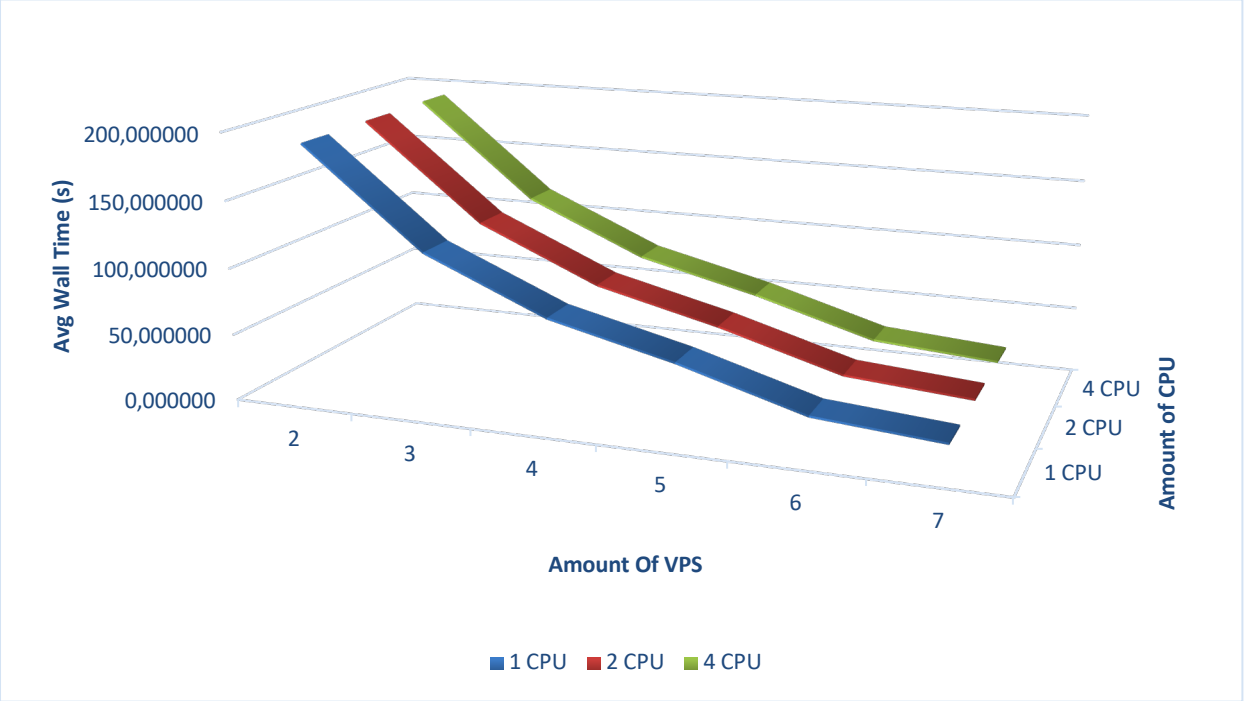

Figure 3. Average Time Comparison Between 1 CPU, 2 CPUs, and 4 CPUs

The test results show how the number of cores and the number of VPSs affect the execution time of the algorithm with 500 particles. In the test with 1 core per VPS, the execution time decreased significantly as the number of VPS increased. Using 2 VPS takes an average of 194.74 seconds, which is the longest time in this configuration. When the number of VPS is increased to 3 VPS, the execution time drops dramatically to 119.05 seconds. This drop continued with 4 VPS recording an average time of 77.61 seconds, and with 5 VPS, the execution time reduced again to 54.97 seconds. Better results were achieved with 6 VPS, where the average execution time was 27.79 seconds, while the fastest time was achieved by 7 VPS, which was 20.97 seconds. In the test with 2 cores per VPS, similar results were seen with execution times decreasing as the number of VPS increased. Using 2 VPS requires an average time of 164.04 seconds, while with 3 VPS, the execution time decreases to 97.65 seconds. This decrease continued with 4 VPS recording 71.26 seconds, and with 5 VPS, the average execution time reached 51.61 seconds. Using 6 VPS resulted in an average time of 24.63 seconds, while 7 VPS recorded the fastest execution time of 18.80 seconds. The last test using 4 cores per VPS showed the same pattern, where the more VPS used, the faster the execution time. With 2 VPS, the average execution time was 153.95 seconds. The use of 3 VPS reduced the time to 70.03 seconds, and 4 VPS further decreased the time to 61.03 seconds. The execution time with 5 VPS was 43.33 seconds, and with 6 VPS it was 20.51 seconds. The fastest time was achieved by 7 VPS, with an average time of 15.71 seconds.

Figure 3 shows the overall test results. The figure shows that increasing the number of VPS significantly reduces the execution time, especially when the number of cores per VPS is increased. In all configurations, the more VPS and the more cores used, the faster the calculation process can be completed. The use of 4 cores per VPS and 7 VPS resulted in the fastest execution time in all tests, with an average of 15.71 seconds. This decrease in execution time occurs because workload sharing becomes more efficient when more cores and VPS are used. Each VPS can handle a smaller portion of the total particles to be processed, so computationally heavy tasks can be completed in parallel by multiple machines at once. As the number of cores in each VPS increases, each VPS has higher processing capabilities, allowing more particles to be processed simultaneously. This drastically reduces the number of iterations that must be run sequentially, thus speeding up the overall computation process. Moreover, the addition of VPS also reduces the waiting time in the system, as the load distributed among multiple VPS reduces queues and bottlenecks in data processing. With many VPS and cores working simultaneously, the use of computing resources becomes more optimised, allowing the MIPSO algorithm to perform calculations in a

much shorter time. The combination of increasing the number of VPS and cores on each VPS simultaneously increases the efficiency of parallel processing, which is the main reason for the significant decrease in execution time in this test.

CONCLUSION

This research examines the effect of parallel processing in optimising the MIPSO algorithm for forecasting using the Holt-Winters Exponential Smoothing method. The two main aspects tested are the effect of the number of VPS and the number of cores used in each VPS on execution time.

The results show that increasing the number of VPS significantly speeds up the calculation process. In the first study, using 1 core per VPS, increasing the number of VPS resulted in a gradual decrease in execution time. The more VPS used, the greater the reduction in execution time that occurs, because heavy computational tasks can be distributed more efficiently to many machines. In the second study, in addition to the number of VPS, the effect of the number of cores on each VPS was also investigated. The results show that using more cores per VPS gives a significant acceleration in execution time. By increasing the number of cores from 1 to 2 and 4, the execution time is reduced substantially as more particles can be processed in parallel within each VPS. The combination of more cores and more VPSs resulted in the most optimal configuration for accelerating computation.

REFERENCES

[1] S. Septa and H. Hoirul, “Peran Big Data pada Sektor Industri Perdagangan: Tinjauan Literatur pada Perusahaan Bidang Perkantoran,” *J. Off. Adm. Educ. Pract.*, vol. 2, no. 3, 2022, doi: 10.26740/joaep.v2n3.p198-210.

[2] B. E. and N. K., “Sales Forcasting and Organizational Performance in Bakery Industry in Nigeria,” *Br. J. Manag. Mark. Stud.*, vol. 6, no. 1, 2023, doi: 10.52589/bjmms-mcg2fuin.

[3] A. Fauzan, D. G. Rahayu, A. Handayani, I. Tahyudin, D. I. S. Saputra, and P. Purwadi, “Sales Forecasting Analysis Using Trend Moment Method: A Study Case of a Fast Moving Consumer Goods Company in Indonesia,” *J. Inf. Technol. Cyber Secur.*, vol. 1, no. 1, 2023, doi: 10.30996/jitcs.7572.

[4] I. D. N. A. Manuaba, I. B. G. Manuaba, and M. Sudarma, “Komparasi Metode Peramalan Grey dan Grey-Markov untuk mengetahui Peramalan PNBP di Universitas Udayana,” *Maj. Ilm. Teknol. Elektro*, vol. 21, no. 1, 2022, doi: 10.24843/mite.2022.v21i01.p12.

[5] S. Sofiana, S. Suparti, A. R. Hakim, and I. Triutami, “PERAMALAN JUMLAH PENUMPANG PESAWAT DI BANDARA INTERNASIONAL AHMAD YANI DENGAN METODE HOLT WINTER’S EXPONENTIAL SMOOTHING DAN METODE EXPONENTIAL EMOOTHING EVENT BASED,” *J. Gaussian*, vol. 9, no. 4, 2020, doi: 10.14710/j.gauss.v9i4.29448.

[6] J. N. A. Aziza, “Perbandingan Metode Moving Average, Single Exponential Smoothing, dan Double Exponential Smoothing Pada Peramalan Permintaan Tabung Gas LPG PT Petrogas Prima Services,” *J. Teknol. dan Manaj. Ind. Terap.*, vol. 1, no. I, 2022, doi: 10.55826/tmit.v1ii.8.

[7] I Putu Susila Handika and I Kadek Susila Satwika, “Enhancing Sales Forecasting Accuracy Through Optimized Holt-Winters Exponential Smoothing with Modified Improved Particle Swarm Optimization,” *J. Nas. Pendidik. Tek. Inform.*, vol. 12, no. 2, 2023, doi: 10.23887/janapati.v12i2.65462.

[8] Y. H. R. Kang, A. Löffler, D. Jeurissen, A. Zylberberg, D. M. Wolpert, and M. N. Shadlen, “Multiple decisions about one object involve parallel sensory acquisition but time-multiplexed evidence incorporation,” *Elife*, vol. 10, 2021, doi: 10.7554/eLife.63721.

[9] M. H. Ahmed, S. Tiun, N. Omar, and N. S. Sani, “Short Text Clustering Algorithms, Application and Challenges: A Survey,” *Applied Sciences (Switzerland)*, vol. 13, no. 1. 2023, doi: 10.3390/app13010342.

[10] I. T. Julianto, D. Kurniadi, M. R. Nashrulloh, and A. Mulyani, “Data Mining Algorithm Testing For SAND Metaverse Forecasting,” *J. Appl. Intell. Syst.*, vol. 7, no. 3, 2022, doi: 10.33633/jais.v7i3.7155.

[11] A. Rezazadeh, “A Generalized Flow for B2B Sales Predictive Modeling: An Azure Machine-Learning Approach,” *Forecasting*, vol. 2, no. 3, 2020, doi: 10.3390/forecast2030015.

[12] Q. Ma and A. Xie, “Application of Big Data Analysis in Sales Forecasting,” *Adv. Econ. Manag. Polit. Sci.*, vol. 64, no. 1, 2023, doi: 10.54254/2754-1169/64/20231465.

[13] R. Ravitilova, D. Amelia, I. Saputra, and

R. Rinda, "Forecasting Sales: A Comprehensive Analysis of Forecasting Techniques for Sales Budget Determination," *J. Audit. Pajak, Akunt. Publik*, vol. 2, no. 2, 2023, doi: 10.32897/ajib.2023.2.2.3217.

[14] M. A. Khan *et al.*, "Effective Demand Forecasting Model Using Business Intelligence Empowered with Machine Learning," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3003790.

[15] J. Liu, Q. Chen, and Y. Qiu, "The design of ERP intelligent sales management system," in *Frontiers in Artificial Intelligence and Applications*, 2020, vol. 331, doi: 10.3233/FAIA200720.

[16] N. I. S. Baldanullah, N. Mulyarizki, I. Permatasari, I. P. Naufal, and D. C. Pratama, "Parallel Processing Pada Pemodelan Machine Learning Menggunakan Random Forest," *J. Informatics Adv. Comput.*, vol. 4, no. 1, 2023.

[17] I. K. S. Satwika and I. D. P. Gede Wiyata Putra, "ANALISIS PERFORMANSI KINERJA SERVER MENGGUNAKAN TERMINAL SERVER BERBASIS WINDOWS DAN LINUX (Studi Kasus STMIK STIKOM Indonesia)," *Netw. Eng. Res. Oper.*, vol. 5, no. 1, p. 30, 2020, doi: 10.21107/nero.v5i1.144.

[18] M. H. P. Swari, I. K. S. Satwika, and I. P. S. Handika, "Performance Analysis of Sales Big Data Processing using Hadoop and Hive in Cloud Environment," in *2020 6th Information Technology International Seminar (ITIS)*, Oct. 2020, pp. 162–166, doi: 10.1109/ITIS50118.2020.9320964.

[19] A. M. Elshewey *et al.*, "Optimizing HCV Disease Prediction in Egypt: The hyOPTGB Framework," *Diagnostics*, vol. 13, no. 22, 2023, doi: 10.3390/diagnostics13223439.

[20] Q. Zhao and C. Li, "Two-Stage Multi-Swarm Particle Swarm Optimizer for Unconstrained and Constrained Global Optimization," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3007743.

[21] D. Dotan and S. Dehaene, "Parallel and serial processes in number-to-quantity conversion," *Cognition*, vol. 204, 2020, doi: 10.1016/j.cognition.2020.104387.