International Journal on Advanced Science Engineering Information Technology

Vol.12 (2022) No. 4
ISSN: 2088-5334

An Investigation of Computation Time Based on Domain Size in WRF Model

Aisya Nafiisyanti [a,*], Ibnu Fathrio [a]

[a] *Center of Athmospheric Science and Technology,National Institute of Aeronautics and Space, Bandung, 40173, Indonesia*
Corresponding author: [*]aisya.nafiisyanti@lapan.go.id

***Abstract*— Estimating WRF model computation time is necessary because of the need for domain expansion for weather prediction. However, the optimum computation time is not simply gained by enlarging the domain and adding processor numbers. Thus, an investigation was carried out to determine the correlation between computation time on domain size, number of grids, and number of processors used to run the WRF model. This study uses a collection of computation time as the data input from running the WRF model with two domain group ratios, 2: 1 and 1: 1, and various processors. Negative Exponential Function (NEF) and Power Function (PF) as exponential decay functions are evaluated to represent the curve formed from the computation time against the number of processors in one domain case. This study also evaluates the speed up and efficiency of the use of processor numbers against the tested domains. NEF represents the decrease in computation time curve in a domain case better than PF since this function has a steeper slope, better initial value, and k constant that keeps the computation time falling below 0. The computation time can be optimally saved by adding approximately eight processors at the same domain ratio with four times larger grid size and the same amount of grid number. Investigations for other domain ratios need to be carried out to determine the characteristics of computation time on the number of processors, grid size, and domain size.**

***Keywords*— Computation time; NEF; ratio; processors; speed up.**

Manuscript received 4 Mar. 2021; revised 6 Jun. 2021; accepted 3 Aug. 2021. Date of publication 31 Aug. 2022.

I. INTRODUCTION

Simulating atmospheric weather at various scales requires adequate computational resources, and the process should be carried out efficiently. This is because these simulations generally act as a prediction process for weather conditions. The computation speed to produce the results is very important because these results must be known before the predicted time occurs. One of the ways to optimize computation is to perform workload sharing management. The parallel computation process can be done with data parallelism and task parallelism [1]. This research analyzes computational performance with a parallelism data approach to the WRF model.

Weather Research Forecast (WRF) is an atmospheric simulation model based on Numerical Weather Prediction (NWP) [2], [3]. In addition, the model, which has been built since 2000, is widely used for research. This model is used because of the increasing level of user confidence, which can be seen from the increase in the number of registrations, with a total of 3400 registrations from 2000 to 2014. The number of users increases in this model due to support from NCAR in the form of a website that continuously releases version updates and bug fixes and provides tutorials and a forum for the community to share knowledge and experience [4]. Moreover, the number of publications is 3200 in total until mid-2015 [5]. The community formed also shows that the model has been a promising tool and has contributed to the field of weather prediction [6]. Besides, a comparative study between WRF and other Numerical Weather Predictions (NWP) shows that this model has modules that can be coupled loosely. Thus, it can provide reliable and accurate results [7].

WRF provides two dynamic solvers: Advanced Research Weather (ARW) and the Nonhydrostatic Mesoscale Model (NMM). In addition, other simulation systems are extensions of this model: 1) WRF Data Assimilation System (WRFDA), 2) WRF Atmospheric Chemistry Model (WRF-Chem), and 3) WRF Hydrological Modelling System (WRF-Hydro). This model provides customizable features and schemes according to the research needs [8], where four different Planetary Boundary Layer schemes are reviewed to find the best configuration. The tool can model severe weather as typhoons, tornadoes, and extreme rain on a large scale. Other than that,

it is also capable of simulating local phenomena on a smaller scale that requires higher resolution, such as wind farms or airport areas. WRF accuracy in several studies has yielded satisfactory results because the model can simulate several examples of specific phenomena that impact the area being studied [9], [10].

Basically, the more cores used will speed up the computation process, but at some point, the computation will experience overhead, that is the ineffective runtime due to the need for communication between cores and redundant computing [11], [12]. This is caused by using too many cores. Thus, the computation speed is not always determined by the number of cores used. So, it is important to have the ability to estimate the number of cores needed based on specific simulation settings.

This study aimed to determine the effect of domain settings in the pre-processing and processing stages. The author evaluated the computations time and exhibited patterns formed during the study. Thus, model users can estimate the time required to perform computations based on specific domain settings.

II. Material and Method

A. Material's Configurations

The computation in this study uses one Supermicro server node with AMD Opteron 6320 processor, which consists of a logical 16 CPUs. That is built from 2 sockets, four cores per socket, and 2 threads per core. This device operates with 2800 MHz cycles, x86_64 architecture, and 2048 MB cache size per CPU.

This study uses WRF version 4.0, which was released in June 2018. First, this study conducted configuration settings to determine the area to be reviewed and the input data to be used. Configuration is done on the WRF Pre-processing System (WPS), a collection of three tasks: geogrid, ungrib, and metgrid. Those are purposed to prepare input data to be used in the main simulation [13]. Global Forecast System (GFS) hourly data on June 9th to 10th 2020 (24 hours) is used as data input. The domain configuration is set on the namelist.wps file. This study uses Mercator as map projector. The latitude and longitude center are set to 8° 7' 24.672" S and 117° 2' 13.056" E respectively, the geog data resolution is set to 5 minutes and the area size follows four scenarios: 1000 x 500 km^2, 2000 x 1000 km^2, 1000 x1000 km^2, 500 x 500 km^2.

There are possible resolutions to interpolate the geographic input data into static terrestrial data: 10 minutes (~19 km), 5minutes (~9 km), 2 minutes (~4 km), and 30 seconds (~0.9 km). Although it is recommended to choose a slightly better resolution than the grid resolution for the interpolation [14], where the domain resolution is set to the closest option, this study chose the resolution at 5 minutes for all grid sizes because the accuracy of the simulation is not the focus on the research. This configuration is also supported by part of the results of this study which is explained in the results and analysis section. The after-mentioned reason is also applied to the data and coordinate selection.

As aforementioned, the study performed four scenarios of two dimensions simulation area. Furthermore, to see the pattern of required time computation for each area, different grid settings are applied but still follow the size of the area determined. Each grid setting has a different resolution to follow the condition previously mentioned. The settings are described in Table 1.

TABLE I
DOMAIN SIZE SCENARIO WITH DIFFERENT RATIO

2:1			1:1		
Grid set	**Grid resolution (each side)**		**Grid set**	**Grid resolution (each side)**	
	1000 x 500 km^2	**2000 x 1000 km^2**		**1000 x 1000 km^2**	**500 x 500 km^2**
20x10	50	100	10x10	100	50
40x20	25	50	20x20	50	25
50x25	20	40	30x30	33.3	16.67
80x40	12.5	25	40x40	25	12.5
100x50	10	20	50x50	20	10
160x80	6.25	12.5	80x80	12.5	6.25
200x100	5	10	100x100	10	5
250x125	4	8	125x125	8	4
300x150	3.33	6.67	160x160	6.25	3.125
320x160	3.125	6.25	200x200	5	2.5
350x175	2.857	5.71	250x250	4	2
400x200	2.5	5	300x300	3.33	
450x225	2.22	4.4	320x160	3.125	
500x250	2	4	350x350	2.857	

The scenario set in Table 1 runs on sets of processors: 1, 2, 4, 8, 16, 32, 40, and 48. This setting must follow WRF data distribution provision on a parallel scheme. WRF applies grids decomposition in the parallelization process. First, the domain is divided into tiles, which depend on the total used processors number. Each tile has minimum 5 grids on each side, both rows, and columns. This is called the "halo regions", which function as an information channel to the neighboring tile. The region with 5x5 grids is a full halo region and must be avoided since it does not have space for computation. It can cause a crash or unrealistic output. The concept of the halo region is applied in WRF and in every parallel computation where the input/data is spatial and divided into neighboring data chunks [15]. A calculation between grid size and processors number is performed to work on this. The grid length in the west-east direction is divided by the number of tiles to get the tile's x-direction length. The same calculation applies to the grid length in the north-south direction to get the tile's y-direction length. The resulting number must be greater than 10 to achieve a successful run.

The next step is decomposing the number of processors. It is determined by the closest 2 factors of the processor number. For example, if 8 processors are used, the decomposition would be 2x4, and 16 processors would have 4x4 decomposition. Using 13 processors is not recommended since the decomposition would be 1x13. It is suggested to have squared decomposition. However, it is still possible to have un-squared ones. According to that, some domains can only be run using 1 processor, and in the contrary, there are domains that can be run using 32 processors or more.

The node runs simulations using Simple Linux Utility for Resource Management (SLURM). SLURM is a cluster workload manager for the Linux operating system [16]. This facility implements three main functions. Firstly, allocating jobs to resources. Secondly, providing a parallel job framework for the cluster to start, execute and monitor work, and thirdly parsing jobs when pending conditions are needed. This service is used since it is open-source, has a large community, and is used worldwide [17]. There is a possibility

that a process may have a waiting state if the computer resources are insufficient because there are a number of other processes run at the same time. If a process is on hold, the waiting time is counted as part of the total processing time. Therefore, the running process must be carried out without sharing resources with another job. SLURM used Process Management Interface (PMI) to allow process managers to interact with the MPI library [18].

B. Methods

1) Negative exponential function (NEF): This function plots a graph of computation time towards the number of processors. NEF is chosen because the growth of computation time tends to decay exponentially. The modified NEF proposed by [19] has the equation 1

$$y(p) = a \times exp^{-bp} + k \quad (1)$$

Where a is the initial increment value, b is the slope of the curve, k is the minimum increment width, p is the number of processors and $y(p)$ is the computation time towards number of processors.

2) Power Function (PF): Along with the Negative Exponential Function, Power Function is also chosen because it is able to describe the exponentially decayed graph. Besides, it is also widely used in analyzing scheduling algorithms [20], where the processors' computation time is reviewed and is in line with this study. The traditional power function has the following formula

$$y(p) = ap^b \quad (2)$$

Where $y(p)$ is the processing towards the processor's number, p is the number of processors, a and b are initial value and constant that defines the slope of the curve, respectively.

3) Linear Interpolation: Linear interpolation is applied to expand the computation time data. This function estimates computation time in between two known values from two computed domains in a scenario from a processor set. This interpolation type is chosen since the trend of computation time of a domain tends to grow linearly towards the total number of grids. The equation is described in equation 3

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \quad (3)$$

Where y is the unknown computation time, y_1 and y_2 are the known computation time where y lies in between, x is the untested domain, and x_1 and x_2 are the tested domains. Although this method is firstly used for ancient astronomy, linear interpolation remains relevant to providing solutions to modern problems [21], [22].

Assume a data grid is defined in $m \times n$. Firstly, we increment m by 20 grids. So that, if there are 120x60 and 200x100 scenarios with a 2: 1 ratio, the added scenario extensions are 140x70, 160x80, and 180x90. The same treatment is applied in other domain scenarios with different ratios.

4) Speed up and Efficiency: We investigate the speed up of each domain scenario parallel performance. The method is denoted in equation 4 [23]

$$S_{m\times n} = \frac{t_{serial\ m\times n}}{t_{parallel\ m\times n}} \quad (4)$$

Where $t_{serial\ m\times n}$ is the serial computation time for domain $m \times n$ and $t_{parallel\ m\times n}$ is the parallel computation time for domain $m \times n$. The efficiency of each domain from the scenarios is evaluated to see the most optimum number of processor usage. The formula is denoted in equation 5

$$E_{m\times n} = \frac{t_{serial\ m\times n}}{t_{parallel\ m\times n} \times p} = \frac{S_{m\times n}}{p} \quad (5)$$

Where p is the number of processors and $S_{m\times n}$ is the speedup of p processors.

III. Result and Discussion

A. Pre-processing

The WRF running process for gaining computation time against a set of data scenarios has been done on a number set of processors. As mentioned in the Material Configuration section, the pre-processing step is performed on each data set. Table 2 is the example of WPS configuration computation time on data in which grid length is set to 50x25, and the size of each grid is 200 x 200 m^2.

TABLE II
WPS COMPUTATION TIME IN SECONDS

Proses/static terrestrial data resolution	10m	5m	2m	30s
geogrid	23.92	17.62	19.09	20.61
ungrib	1503.71	1460.22	1308.95	1342.88
metgrid	48.14	36.44	36.8	35.980
real	8.72	4.36	4.37	5.05
wrf	421.88	415.75	423.61	416.41

There are no significant time differences among different static terrestrial data resolutions in the geogrid process. However, the computation time varies in ungrib process. The highest difference is 194.76 seconds, or equal to 3.5 minutes, which is between 10m and 2m data resolution. The difference is tolerable since the biggest time difference is not in a matter of hours. Resolution in 5m is chosen because it has to cover all scenario data size.

B. Selection of Representative Function

Negative Exponential Function and Power Function are applied to gain computation time from untested processors. Since the processor numbers range between 1 to 48, the functions are applied to set of untested number of processors up = {3, 5, 6, 7, 9, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 45, 46, 47}.

The initial values in all domain groups and sizes have the same tendency, and the value increases as the grid numbers increase. However. NEF has a wider slope range rather than PF both in the domain group with a 2:1 ratio and 1:1 ratio. Other than that, the NEF slope values tend to increase. In contrast to PF, where the slope value tends to decrease regularly as the number of grids increases. Both initial values and slope in NEF shift irregularly as the grid numbers increase, as shown in Fig. 1(a) to 4(d).

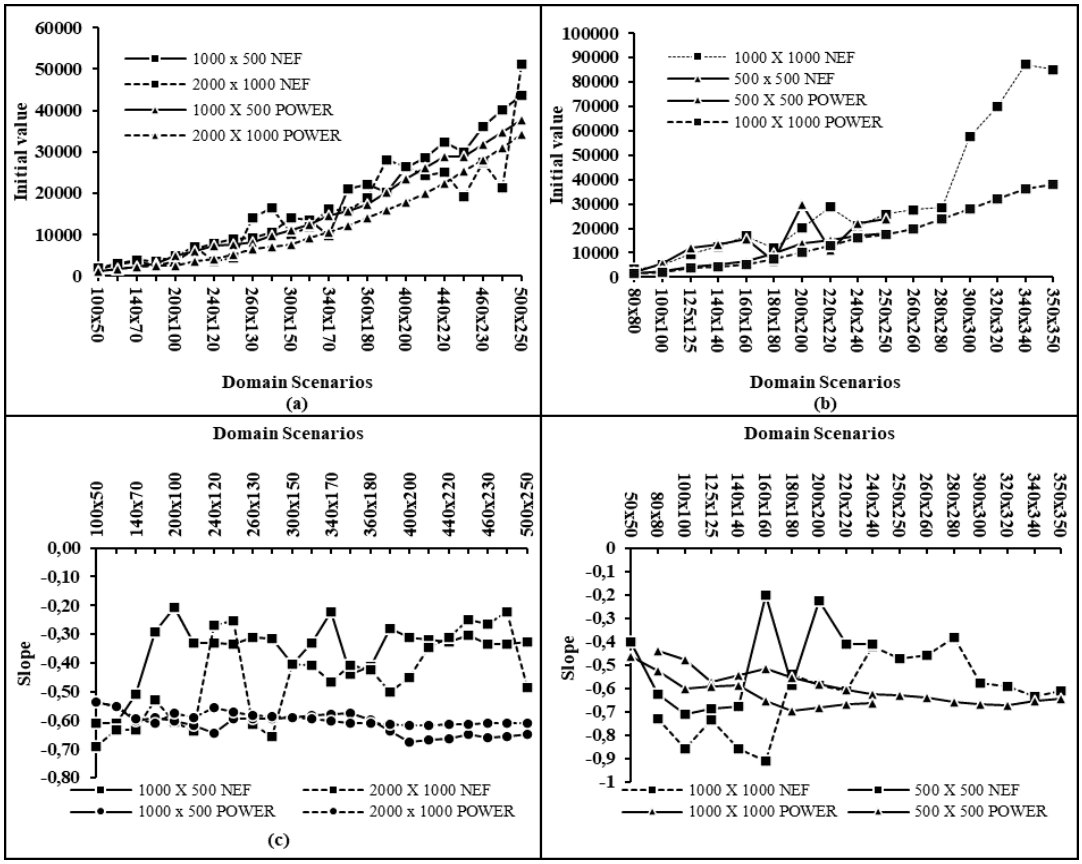

Fig. 1 (a) Initial value for domains group in 2:1 ratio scenario; (b) Initial value for domain group with 1:1 ratio; (c) Slope value for domains group in 2:1 ratio; (d) Slope value for domain group with 1:1 ratio

This difference occurs even though these two functions represent the same curve. This is because the NEF value determination uses the probability of the smallest difference between the actual and the generated computation time by the function. So that the results of value determination of variables a, b, and k are uncertain. On the contrary, the variable's value in PF tends to be stable because the values determination process does not use probability. Thus, the slopes of curves in PF (b) tend to be more stable than NEF.

This section divided the number of processors into 3 parts: initial, middle, and final. The initial, middle and final parts are represented by the set of processor number 1 to 3, 4 to 10, and 16 to 48, respectively. Table 3 shows the performance of the two functions in all data scenario. Columns 3 and 4 show the number of functions that give the approximate value of computation time with less difference to the original value. NEF shows better performances in approximating initial values in all domain group. This function also performs better in the middle part, except for 500x500 data group. Also, in the final part, NEF performs better except for 1000x1000 data group. Even though both functions start at not much different initial values, PF slope values are generally smaller than NEF, resulting in a gentler curve for PF computation time estimation. This produces a smaller initial value than the original value and a wider gap between PF curves and original curves in the middle part. This slope value factor makes a significant difference so that most of the PF computation time estimation results produce greater error than NEF.

TABLE III
NEGATIVE EXPONENTIAL FUNCTION AND POWER FUNCTION PERFORMANCE ON DIFFERENT DOMAIN GROUP

Domain Group	Processor	PF	NEF	Choices
1000 X 500	Initial	27	40	NEF
	Middle	30	59	
	End	16	60	
2000 X 1000	Initial	27	40	NEF
	Middle	25	65	
	End	34	41	
1000 X 1000	Initial	12	34	NEF
	Middle	11	50	
	End	51	13	
500 X 500	Initial	13	14	NEF
	Middle	22	15	
	End	0	36	

In addition. the steeper slope of the NEF brings the NEF curve closer to the original curve, which indicates a smaller error in the middle part. This is because NEF has a constant k which prevents the computation time from falling below 0 (computation time cannot be negative). Generally, constant k keeps the estimated value at the end close to the original value so that the difference between the NEF value and the original's is better than PF. The power function does not have this constant, so the greater the number of processors tested, the farther the difference will be. Because the exponential function is intended to generate the untested computation time, most of the untested data is data from the number of processors, 11 to 31, located in the middle and final curve.

The NEF function is chosen to represent the computation time formula towards the number processor.

C. Speed Up and Efficiency

Assume we have a domain with 100x50 grids. According to WRF decomposition setup, where a tile at least has 10 grids at both x and y directions, the maximum processors to run the process should be [x-grids/10] x [y-grids/10] which is 50 processors. However, this amount is not necessarily the optimal number to use. This study examines the computation time to the number of processors used based on the ratio and area size of the domain.

The speed up on domain with 2:1 ratio is reviewed. As shown in Fig. 2(a) and 2(b), the speed up on domains 1000x500 with large grid size (100x50 to 200x100) in average reach a maximum on 16 processors. Meanwhile, domains with smaller grid sizes (220x100 to 500x250) reach maximum on 30 to 32 processors. Increasing the number of processors does not make the computation time faster. In some cases, the computation time actually slows down. On domain 2000x1000 which is 4 times bigger than domain 1000x500, the speed up still seems be able to be optimized by increasing the number of processors, as shown in Fig. 2(b). The domains with smaller size have not reached the optimal speed up because the difference between speed up to 40 processors and 48 processors is still more than 0.5. Likewise, the optimum average speedup is at 24 processors in larger domains.

The same pattern is shown in the calculation of speed up in the domains with a ratio of 1: 1 as shown in Fig. 2(c) and 2(d). In the 500x500 scenario size, domains that have a smaller grid size reach a maximum speed up at 24 processors. While in 1000x1000 scenario size, they can reach a maximum at 32 processors.

For example, on two domains with 250x125 grids, one has 4km^2 and the other has 16km^2 grid size. Domain with a larger grid size has more spaces to perform calculations even though the number of halo regions to transfer information to the neighboring tiles are the same. This makes the computation process faster and less time-consuming for information exchange. Besides, MPI has tendencies to experience bottleneck when a processor has small amount of workload but large amount of communication process [24]. The same pattern is shown on two domains with same grids number in 1:1 ratio.

Some cases in all scenario sizes show unsteady increase in speed up value. This happened because the computing process is affected by the condition of the device. At certain times, the device experiences queue to get resources because the simulated model runs simultaneously with other processes, so that at certain running results, the time needed is greater than the running result with fewer processors.

Although there is a slight increase in the speed-up value in the figure, the time saved in actual computation time is around 30 seconds to 1.5 minutes. The computation time to extract the model results is reduced, but the time consumed by the halo regions increases, then the time saved is used for information exchange. So, increasing the number of processors is not recommended if a scheme has achieved its peak speed. The use of up to 48 processors only provides an average efficiency of 0.23 in all scenario sizes.

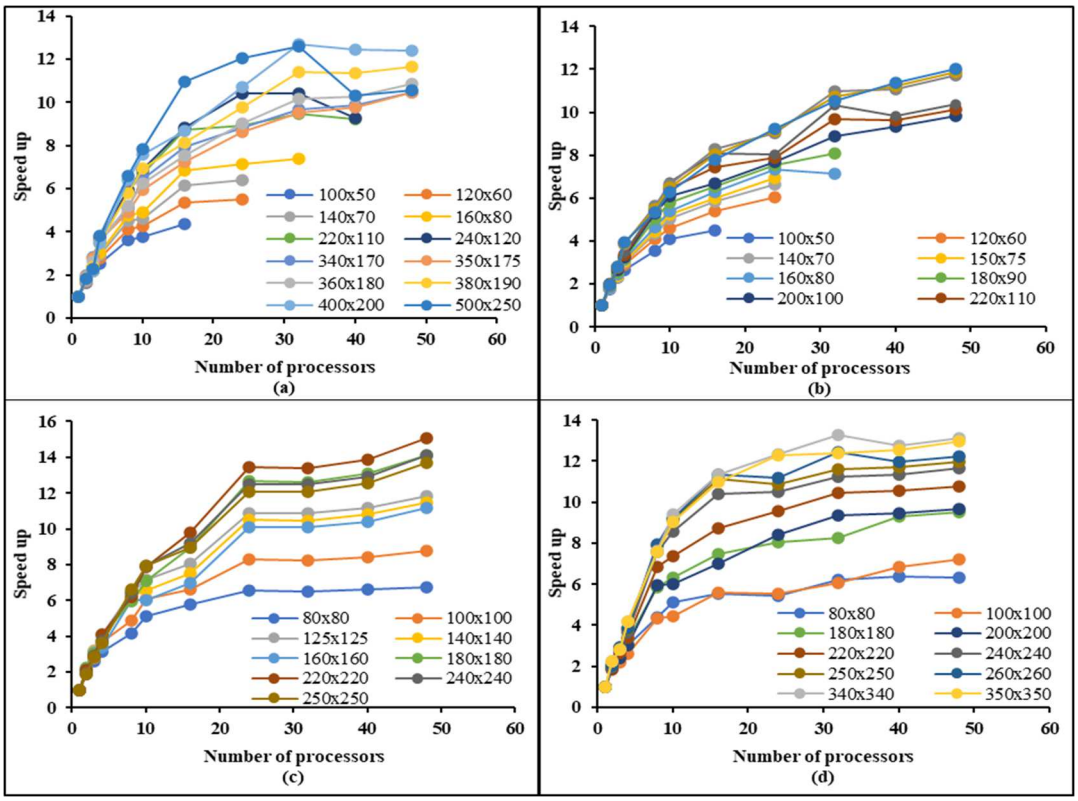

Fig. 2 (a) Speed up for 1000x500 domain group; (b) Speed up for 2000x1000 domain group; (c) Speed up for 500 x 500 domain group; (d) Speed up for 1000 x 1000 domain group

The model may perform with different computation times when applied to multiple computation devices, especially on devices with fewer processors than in this study, because additional time will be required to exchange information between devices. This research also evaluates the computation time in grid scenarios at the same ratio with the same length on *x* and *y*-direction; and in different sizes, for example, a 250x125 grid with a grid size of 16 km^2 and 64 km^2 (See Table 2). Mean Average Percentage Error (MAPE) is used to perform the evaluation. The results then act as the difference average between two domains plotted in Figures 3 and 4.

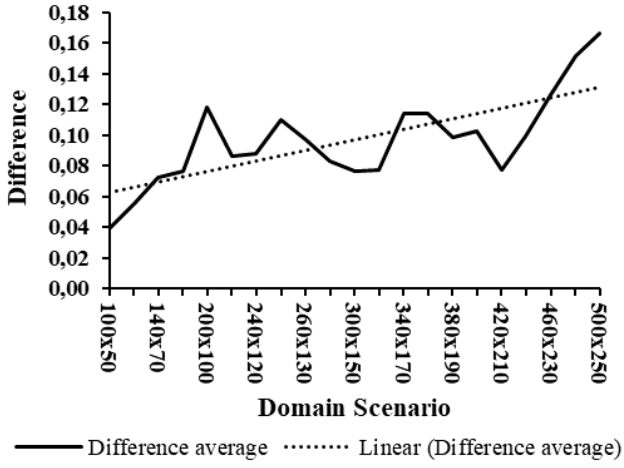

Fig. 3 Difference average on a domain with 2:1 ratio

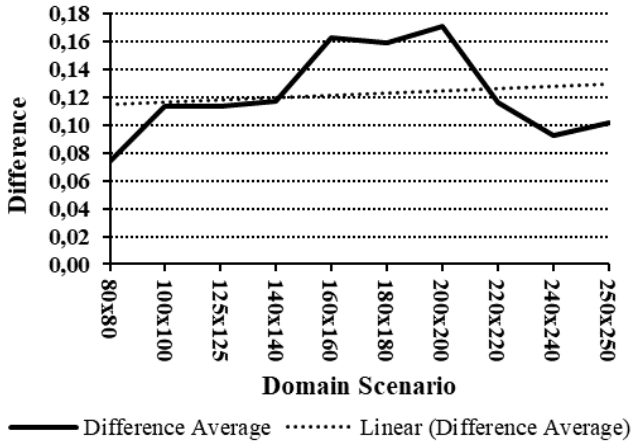

Fig. 4 Difference average on a domain with 1:1 ratio

As seen in Fig. 3, the calculation results show the average difference variation, ranging from 0.04 to 0.17 with an average difference of 0.10. Even though the calculation gives varying results, the average difference tends to go up with the growth of the grid number—for example, two different domain groups at the same ratio. The first group has two domains with a large grid size, which means fewer grid numbers. For example, in domains with 100x50 grids, the first and second domains have 10,000 km2 and 40000 km2 grid sizes, respectively. Then the second group has two domains with small grid sizes, which means more grid numbers. For example, in domains with 250x125 grids, the first and second domains have 4 km2 and 8 km2 grid sizes, respectively. It is shown that the computation time differences in the first group are smaller than in the second group. This is in line with the statement above that the smaller the grid size, the higher the computation time needed. Since the first group has a larger grid size, the number of grids inside halo regions is fewer; thus, less time is needed for information exchange on both domains. Also, the enlargement of the value of the difference is in line with the increase of computation time for the domain that the second group represents.

Meanwhile, in Fig. 4, the difference average slightly increases, although the characteristics of the average difference trend toward the increase in the number of grids are the same as in the domain group with 2: 1 ratio.

IV. Conclusion

The computation time curves obtained from the study decay exponentially as the number of processors used increases in each scenario domain. The curves have slopes represented well by the NEF since this function has a steeper slope, better initial value, and k constant that keeps the computation time falling below 0 in general. The computation time can be optimally saved by adding approximately 8 processors at the same domain ratio with 4 times larger grid size and the same amount of grid number. The results obtained in this study are limited to the domain group with a ratio of 2: 1 and 1: 1, so investigations for other domain ratios need to be carried out to determine the characteristics of computation time on the number of processors, grid size, and domain size. Also, it would be better if the object of study was expanded by increasing the number of devices and the size of the domain.

Acknowledgment

The authors thank the Center of Atmospherics Science and Technology- Indonesian National Institute of Aeronautics and Space (LAPAN) for providing a High-Performance Computing server for simulating this study.

References

[1] P. Pacheco, "Why Parallel Programming?," in *An Introduction to Parallel Programming*, P. S. Pacheco, Ed. San Francisco: Elsevier, 2011, pp. 1–14.

[2] W. C. Skamarock *et al.*, "A Description of the Advanced Research WRF Model Version 4," Boulder, CO, USA, 2019. doi: http://dx.doi.org/10.5065/1dfh-6p97.

[3] B. C. Ancell, A. Bogusz, M. J. Lauridsen, and C. J. Nauert, "Seeding chaos: The dire consequences of numerical noise in NWP perturbation experiments," *Bull. Am. Meteorol. Soc.*, vol. 99, no. 3, pp. 615–628, 2018, doi: 10.1175/BAMS-D-17-0129.1.

[4] NCAR|UCAR, "WRF scaling and timing _ Computational and Information Systems Laboratory," *cisl.ucar.edu*, 2020. https://www2.cisl.ucar.edu/resources/wrf-scaling-and-timing#scaling (accessed Oct. 26, 2020).

[5] J. G. Powers *et al.*, "The weather research and forecasting model: Overview, system efforts, and future directions," *Bulletin of the American Meteorological Society*, vol. 98, no. 8, pp. 1717–1737, 2017.

[6] Y. Li *et al.*, "High-resolution regional climate modeling and projection over western Canada using a weather research forecasting model with a pseudo-global warming approach," *Hydrol. Earth Syst. Sci.*, vol. 23, no. 11, pp. 4635–4659, 2019, doi: 10.5194/hess-23-4635-2019.

[7] A. Z. Abualkishik, "A comparative study on the software architecture of WRF and other numerical weather prediction models," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 24, pp. 8244–8254, 2018.

[8] Y. Kim, K. Sartelet, J. C. Raut, and P. Chazette, "Evaluation of the Weather Research and Forecast/Urban Model Over Greater Paris,"

Boundary-Layer Meteorol., vol. 149, no. 1, pp. 105–132, 2013, doi: 10.1007/s10546-013-9838-6.

[9] H. Duan, Y. Li, T. Zhang, Z. Pu, C. Zhao, and Y. Liu, "Evaluation of the Forecast Accuracy of Near-Surface Temperature and Wind in Northwest China Based on the WRF Model," *J. Meteorol. Res.*, vol. 32, no. 3, pp. 469–490, 2018, doi: 10.1007/s13351-018-7115-9.

[10] S. Sharma, R. Siddique, S. Reed, P. Ahnert, and A. Mejia, "Hydrological model diversity enhances streamflow forecast skill at short-to medium-range timescales," *Water Resour. Res.*, vol. 55, no. 2, pp. 1510–1530, 2019, doi: 10.1029/2018WR023197.

[11] S. Höfinger, T. Ruh, and E. Haunschmid, "Fast Approximate Evaluation of Parallel Overhead from a Minimal Set of Measured Execution Times," *Parallel Process. Lett.*, vol. 28, no. 1, pp. 1–12, 2018, doi: 10.1142/S0129626418500032.

[12] W. Wu, L. He, W. Lin, R. Mao, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 1–16, 2019, doi: 10.1109/tc.2020.2994391.

[13] D. Meyer *et al.*, "WRF-TEB: Implementation and Evaluation of the Coupled Weather Research and Forecasting (WRF) and Town Energy Balance (TEB) Model," *J. Adv. Model. Earth Syst.*, vol. 12, no. 8, p. 18, 2020, doi: 10.1029/2019MS001961.

[14] A. Golzio, S. Ferrarese, C. Cassardo, G. Adele, and D. Manuela, "Land-Use Improvements in the Weather Research and Forecasting Model over Complex Mountainous Terrain and Comparison of Different Grid Sizes," *Boundary-Layer Meteorol.*, p. 33, 2021, doi: 10.1007/s10546-021-00617-1.

[15] H. Schmitz, "Schnek: A C++ library for the development of parallel simulation codes on regular grids," *Comput. Phys. Commun.*, vol. 226, pp. 151–164, 2018, doi: 10.1016/j.cpc.2017.12.023.

[16] D. Koo *et al.*, "An empirical study of I/O separation for burst buffers in HPC systems," *J. Parallel Distrib. Comput.*, vol. 148, pp. 96–108, 2021, doi: 10.1016/j.jpdc.2020.10.007.

[17] C. Hollowell, J. Barnett, C. Caramarcu, W. Strecker-Kellogg, A. Wong, and A. Zaytsev, "Mixing HTC and HPC Workloads with HTCondor and Slurm," *J. Phys. Conf. Ser.*, vol. 898, no. 8, p. 8, 2017, doi: 10.1088/1742-6596/898/8/082014.

[18] N. Malitsky *et al.*, "Building near-real-time processing pipelines with the spark-MPI platform," in *2017 New York Scientific Data Summit (NYSDS)*, 2017, pp. 1–8, doi: 10.1109/NYSDS.2017.8085039.

[19] E. R. Cook, S. G. Shiyatov, V. S. Mazepa, A. Ecology, and U. Branch, *Treering standardization and growth-trend estimation . In .: Cook E . Kairiukstis L . (eds .)*, no. April 2016. Springer-Science+Business Media, B. V., 1990.

[20] P. Singh, B. Khan, A. Vidyarthi, H. H. Alhelou, and P. Siano, "Energy-aware online non-clairvoyant scheduling using speed scaling with arbitrary power function," *Appl. Sci.*, vol. 9, no. 7, 2019, doi: 10.3390/app9071467.

[21] Y. Monno, D. Kiku, M. Tanaka, and M. Okutomi, "Adaptive residual interpolation for color and multispectral image demosaicking," *Sensors (Switzerland)*, vol. 17, no. 12, pp. 1–21, 2017, doi: 10.3390/s17122787.

[22] E. V. Konopatskiy and A. A. Bezditnyi, "Geometric modeling of multifactor processes and phenomena by the multidimensional parabolic interpolation method," *J. Phys. Conf. Ser.*, vol. 1441, no. 1, 2020, doi: 10.1088/1742-6596/1441/1/012063.

[23] Z. Ye, W. Zhou, L. Zhang, Y. Ge, K. Xiao, and Y. Deng, "Multi-user mobile sequential recommendation: An efficient parallel computing paradigm," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 2624–2633, doi: 10.1145/3219819.3220111.

[24] R. Moreno *et al.*, "Analysis of a New MPI Process Distribution for the Weather Research and Forecasting (WRF) Model," *Sci. Program.*, vol. 2020, no. i, p. 13, 2020, doi: 10.1155/2020/8148373.