# Energy Consumption Studies of WRF Executions with the LIMITLESS Monitor

A. Bustos[1], A. Cascajo[2], A.J. Rubio-Montero[1], E. García-Bustamante[3], J.A. Moriñigo[1], D.E. Singh[2], J. Carretero[2], and R. Mayo-Garcia[1]

[1] Technology Department, CIEMAT, Av. Complutense 40, 28040 Madrid, Spain.{andres.bustos, antonio.rubio, josea.morinigo, rafael.mayo}@ciemat.es

[2] Computer Science and Engineering Department, Universidad Carlos III de Madrid, Avenida Universidad 30, Legans, 28911 Madrid, Spain{cascajo, dexposit, jcarrete}@inf.uc3m.es

[3] Energy Department, CIEMAT, Av. Complutense 40, 28040 Madrid, Spain. elena.garcia2@ciemat.es

**Abstract.** We present a study of the performance of the Weather Research and Forecasting [WRF] code under several hardware configurations in an HPC environment. The WRF code is a standard code for weather prediction, used in several fields of science and industry. The metrics used in this case are the execution time of the run and the energy consumption of the simulation obtained with the LIMITLESS monitor, which is the main novelty of this work. With these results it is possible to quantify the energy savings of WRF run configurations, which include variations in the number of computing nodes and in the number of processes per node. It is found out that a slight increase in the computing time can drive to a noticeable reduction in the energy consumption of the cluster.

**Keywords:** HPC optimization · WRF · LIMITLESS

## 1 Introduction

The optimal use of hardware resources is one of the major issues in Computer Science and in particular in High Performance Computing (HPC). Usually, the optimization of code executions targets the minimization of the computing time, setting aside other criteria. However, the study of the energy consumed in the computations is of particular interest from the environmental and economical points of view. Such an evidence has become cornerstone with the advent of the exascale infrastructure in which a kind of trade-off between computational and energy efficiencies is pursued. Examples of this trend nowadays can be found in the literature. Thus, a study of the performance and energy consumption of HPC workloads on a cluster can be found in [20], an auction mechanism model for energy-efficient HPC is detailed in [6], or even theoretical approaches for achieving a sustainable performance while reducing energy consumption [11]

demonstrate the actual necessity for successfully combining both performance and energy issues.

A timely research line such as artificial intelligence is being used for improving this trade-off as well. Driven holistic approaches to reduce the power usage effectiveness (PUE) [31] or swarm optimised greedy algorithm [14], just to mention a few, are recent developments on this topic.

However, there is a lack of understanding of the power consumption characteristics of HPC jobs which run on production HPC systems [24]. The gap is being filled by studies applied to different codes widely used, see for example [7] for an energy-efficiency tuning of lattice Boltzman simulations, [30] for stencil-based application on Intel Xeon scalable processors, or [12] for molecular dynamics calculations executed on hybrid and CPU-only supercomputers with air and immersion cooling.

Following this line of work, we study both the computing time and the energy consumption of two different simulations of the WRF (Weather Research and Forecasting) code, widely used in forecasting predictions and climate evolution. There is in the literature a vast number of references related to the evaluation of the WRF code in what respects their physical results applied to different areas in the world, but scarce information about its computational and energy efficiencies in ultimate processors to the authors' knowledge.

To carry out this study, we test WRF together with a new performance monitor for HPC clusters called `LIMITLESS` already developed by the authors. `LIMITLESS` presents an easy deployment and configuration to gather data with a low overhead from any hardware element in the HPC cluster.

The reminder of this paper is organized as follows. We start with a brief introduction to WRF and the simulations carried out in the experiments in Sec. 2. Then we describe the main characteristics of `LIMITLESS` and the methodology followed in the performance studies in Secs. 3 and 4. We show the results in Sec. 5 and present our conclusions in Sec. 6. The main results are that diverse deployments of WRF with small variations in the execution time can lead to important variations in the power consumption.

## 2 `WRF`: the standard for weather simulations

The Weather Research and Forecasting [WRF] simulation code is a well-established tool in weather prediction since 2007 [19]. It is a massive parallel code that solves the atmosphere dynamics, considering many physical and multiscale phenomena. The lowest 1-3 km region of the atmosphere within the troposphere, characterized by friction and turbulent mixing is called *planetary boundary layer* (PBL) [29]. The PBL plays an important role in the transportation of energy (including momentum, heat and moisture) into the upper layers of the atmosphere and acts as a feedback mechanism in wind circulation. Mesoscale models, as WRF, include different schemes for convection, planetary boundary layer turbulence, radiation, cumulus and land-surface processes which is a complete description of the behavior of the atmosphere both in hindcast and forecast. The PBL parametrization

implemented in the mesoscale models is important for accurate simulations of turbulence, wind, wind power, air quality and, in general, any process occurring in the lower layers of the atmosphere. WRF is used in many industrial and research activities, including weather forecasting and wind power.

The two kind of WRF simulations presented in this work differ in the mesoscale model:

- The non-local-K scheme from Yonsei University, denoted by *YSU*. [17]
- The model with local turbulence kinetic closure from Mellor-Yamada Nakanishi and Niino, Level 2.5, denoted by *MYNN*. [23]

Some recent studies were reported in literature regarding the use of these two settings e.g. [18], [32], [15], [16], [13]. In particular, the WRF model is widely used to generate wind resource maps. These wind atlases are usually made by simulating several years at high horizontal and vertical resolution, for which the computational cost is very high. Therefore, the WRF model is usually run in HPC in a distributed way, i.e., a long simulation is run in each group of nodes. In the *New European Wind Atlas* (NEWA) project [1], it was decided to run simulations of 7 days plus a 24h spin-up period, which overlaps with the last day of the preceding weekly run. An advantage of the weekly runs is that the simulations are independent of each other and can be integrated in parallel. This reduces the total wall clock time needed to complete a multi-year climatology at a decent computational overhead. Fig. 1 shows the temperature at 2 meters above ground level in the whole Mediterranean Sea using the WRF model with weekly run. The PBL *MYNN* parametrization was used in this case.

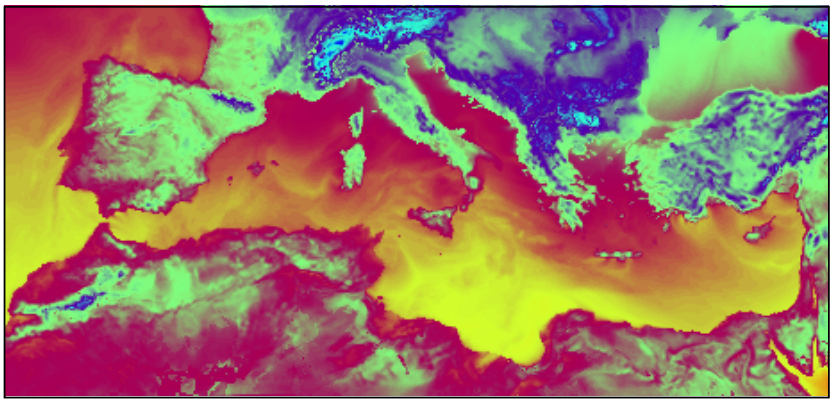

**Fig. 1.** Temperature at 2 meters above ground level for a weekly run simulation in the Mediterranean area.

For each weekly run, in the domain of the Fig. 1 (6,213,460 grid points), 4 nodes in Xula (160 cores, see Sec. 4) are used and the computational time is

around 6 hours, i.e., 960 core hours. With 52 week/year for 30 years simulated, 1,497,600 core hours will be needed if 4 nodes are used (just the half when 8 nodes are used and so on). As for many medium-size supercomputers, the amount of nodes which are accessible by a user is limited, the logical consequence is that the same simulation is performed several times in a kind of parameter sweep calculation. In this point is where a trade-off between the simulation walltime and the energy consumed is interesting to be analysed (see below in the results section). Even more when the WRF scalability is not optimal, i.e., a computational and energy cost-effective execution is of even greater importance.

In the present work we have executed short test WRF simulations with the two parametrizations aforementioned for our performance studies. Depending on the number of computing nodes involved and the MPI configuration, those simulations take 20-40 minutes to complete in a standard HPC infrastructure and produce 53 GB of output data. Because the problem size is always constant for each input configuration (*YSU/MYNN*), all scaling studies presented here refer to strong scaling.

## 3 The LIMITLESS monitor

In this work, data from the system have been collected using the LIMITLESS monitor. LIMITLESS [10, 8] is a highly-scalable framework for monitoring and application scheduling that is able to work under near-to-second sampling rates with low overheads. However, this sampling interval can be established from hours to less than a second. LIMITLESS is fully integrated with other system software like the scheduler, and other runtimes that allow it to enhance some goals such as application-level monitoring, I/O interference awareness, and scalability. In a previous work [9], a description of the monitoring architecture as well as a practical example of its use for enhancing the application scheduling are shown.

LIMITLESS includes a monitoring tool designed to provide performance information for generic purposes in large scale systems. It consists of the following componentes: one *LIMITLESS Daemon Monitor* (LDM) per node, that periodically collects the performance metrics; a set of *LIMITLESS Daemon Aggregators* (LDAs), that are responsible for forwarding the information from the LDMs to other aggregators or servers, and the *LIMITLESS Daemon Server* (LDS), which gathers and stores the monitoring information in an ElasticSearch database. The LDS is also able to send the information to other processes involved in the monitoring or to store it locally for future local exploitation like, for instance, in-transit processing.

The monitoring information collected by LIMITLESS includes, but is not limited to, different metrics related to CPU, main memory, I/O and communication network utilisation, as well as temperature and energy consumption. The monitor collects many of these metrics directly from the kernel performance counters. However, the energy consumed is measured by means of the *Intelligent Platform Management Interface* (IPMI).

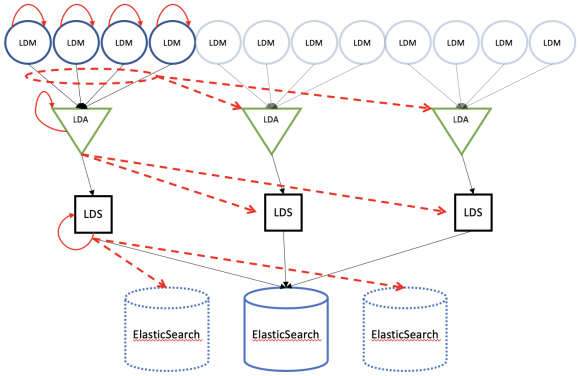


**Fig. 2.** Example of deployment with Triple Modular Redundancy and watchdog processes in the first monitoring branch.

Figure 2 shows a generic deployment of `LIMITLESS` with fault-tolerance mechanisms enabled for the LDMs, LDAs and LDSs. The purpose of replicating these components is to enhance the monitor scalability and resilience. The techniques applied are *Triple Modular Redundancy* (TMR) and *Watchdog processes* (WD). The first one allows each component to forward the information to three other components (instead of one). TMR includes two configurations: with replication and without it. When replication is enabled, each component sends the same information to the three linked components. Alternatively, if replication is not enabled, the component sends the information to the first linked component and waits for confirmation. If there is no confirmation, it sends the information to the second linked component and waits again. If there is no confirmation, it repeats the process with the third option. Note that in this last approach, once a confirmation has been received, the data replication is avoided in order to reduce the communication overhead. On the other hand, WD is a process that executes an application as a service, checking if the managed components are running or not. In case of failure, WD restarts the component with the same configuration. Note that Figure 2 shows a generic topology that does not necessarily fit with the monitoring framework topology used in this work.

`LIMITLESS` monitor has been selected to provide the performance information due to its reduced overhead (shown in Table 1) with values less than 1 %. This overhead does not include the overhead of collecting energy consumption through Impitool and the state of the InfiniBand network. Table 2 shows the overheads of monitoring with those two features, that are higher for those sampling intervals because of the syscall related overheads. The columns identified by $SI$ indicate the sampling interval; $TT$ shows the total execution time while the monitor is running; $Ctime$ is the number of CPU seconds that the monitor dedicates

to collect the information; finally, the $O$ columns show the CPU overhead in percentage.

**Table 1.** `LIMITLESS` monitor overhead under different sampling intervals, without measuring the power consumption.

| $SI$ [s] | $TT$ [h] | $Ctime$ [s] | $O(\%)$ |
|---|---|---|---|
| 1 | 24 | 144 | 0.166 |
| 5 | 24 | 24 | 0.027 |
| 10 | 24 | 11 | 0.013 |

**Table 2.** `LIMITLESS` monitor overhead under different sampling intervals (including power consumption).

| $SI$ [s] | $TT$ [h] | $Ctime$ [s] | $O(\%)$ |
|---|---|---|---|
| 1 | 24 | 2350 | 2.72 |
| 5 | 24 | 475 | 0.55 |
| 10 | 24 | 243 | 0.28 |

We can observe that the monitor overhead is very low, which is important in production clusters to keep monitor interference with other applications as small as possible.

The topic of monitoring in distributed systems has been extensively addressed in previous works. However, monitoring is a crucial component that has to be adapted to new technology improvements related to HPC platforms. There are solutions based on frameworks that combine various components to offer global views (for example Ganglia [21], [4], Nagios [3] or Slurm [33]). Those frameworks are well known and used in HPC. However, simpler and lighter monitors also have a place, as in the case of Collectd [5], which works as a local monitoring daemon.

The advantage of using `LIMITLESS` is that is easily adaptable and reconfigurable to multiple platform configurations. `LIMITLESS` is also able to select different metrics of interest and process the results with a minimum impact on the platform performance.

## 4 Methodology

The simulations presented in this work were run on the Xula facility, located at the data center at CIEMAT, Madrid, Spain. This HPC cluster is in production since November 2019 and it is currently adhered to the Spanish Supercomputing Network (RES)[2]. The cluster is divided into several Infiniband islands with

different hardware installed, which correspond with partitions in the batch system. In this work, the first partition was used, which is precisely the offered to the external users belonging to the RES. This allows better reproducibility and the profiting of the results, because many of them make use of WRF in their research. The island is composed of computing nodes with double Intel(R) Xeon(R) Gold 6148. In total there are 40 cores at 2.40 GHz per node (hyper-treading is disabled). Inside the island, the connectivity is based in 4xEDR Infiniband (100Mbps), without blocking among nodes, but with 2:1 blocking against the LUSTRE storage servers.

Moreover, the nodes used in this work were reserved and removed temporally from the execution queues to improve the accuracy in the measurements. Thus, no other software was using the hardware that might influence on our experiment, with exception of the shared storage. Similarly, turbo-boost is disabled in the island. Thus, allocating in the nodes the maximum of 40 MPI tasks will not disrupt the measurements.

We configured the LDM in `LIMITLESS` to measure every 5 seconds and dump the data from all involved nodes into a text file in the central server. The information is sent via Ethernet to the LDA/LDS, which run in an independent server in the cluster, so the monitor does not overload the MPI communications. We used Python to integrate, analyze and plot all the results.

In the experiments we carried out several WRF simulations with the two different input configurations (*YSU* and *MYNN*), described at the end of Sec. 2. For each input file, we measured the execution time and energy consumption in different deployments: varying the number of nodes involved and the number of processes per node [*np*]. Every simulation case, defined by the input file, the number of nodes and *np*, is executed at least three times in order to have a more robust estimation of the computing time and energy consumption (see below for a deeper explanation about reproducibility and errors). In this way we can estimate the variation in our results due to the use of the cluster by other users that can occupy neighbouring nodes or make intensive use of the storage system. Thus, we present the mean values and all error bars correspond to the standard deviation.

We completed a total of 166 simulations in the Xula cluster and found that typically the the dispersion in the measurements is very small. We obseve those the errors correspond to a standard deviation below 3% in almost 90% of the cases in the results of the next Section (figures 3, 4 and 5 ), showing the previously mentioned good reproducibility. In this sense, it should be pinpointed that taking a t-Student confidence interval of 95% and the previous standard deviation value, moving from a set of three repetitions to six would simply reduce the final error associated to the measurement from 1.14% to 0.07%, with the consequent additional energy consumption and $CO_2$ emission made by the cluster. Hence we can say that the experiments are reproducible in most cases.

## 5 Results

The main objective of these experiments if to find out the relation between the strong scaling of an MPI application, such as WRF, with its energy consumption. This is shown in the following figures of this Section, starting with a generic scaling study, increasing the number of cores/nodes, and later relating them to the energy consumption.

In Fig. 3 we plot the execution time of WRF using the *YSU* input file using a single node with 40 CPUs and varying the number of processes, *np*. We can see that the minimum execution time is obtained using $np = 20$, indicating that the scaling is not very good above those *np*. This is a known behavior of a certain type of MPI applications [34, 28, 25]. MPI operations are optimized for process communication between nodes, not for efficient intra-node communications, where context changes, protocols, and intermediate libraries overload the CPUs and increase the latency. For those reasons, if the application cannot use threads (e.g. OpenMP), the MPI processes are distributed in the cluster [22] to measure both the strong and weak scaling.

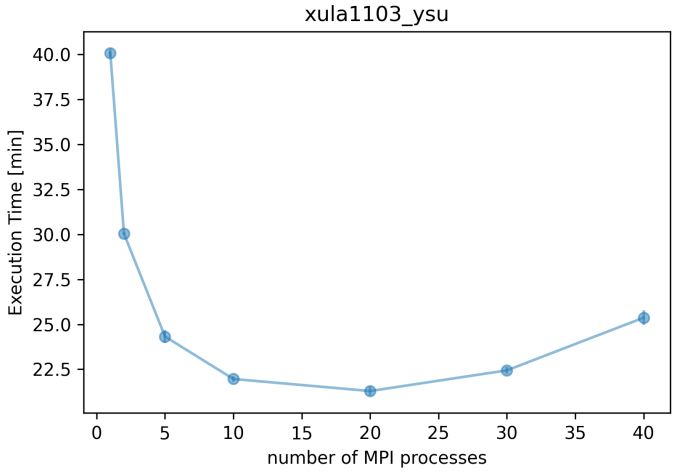


**Fig. 3.** Execution time of WRF with *YSU* input file, using a single Xula node [named xula1103], as a function of the number of MPI processes.

We chose to compare the maximum distribution of MPI processes between CPUs ($np = 2$, one process per CPU) and the optimal concentration ($np = 20$), varying the number of nodes and consequently the total number of MPI tasks. This is shown in Fig. 4, where we plot the execution time versus *np*. As can be seen, although for $np = 2$ each process has all hardware resources available,

WRF does not scale well for more than 4 nodes. On the other hand, there are no improvements if we use more than one node if they execute 20 processes.

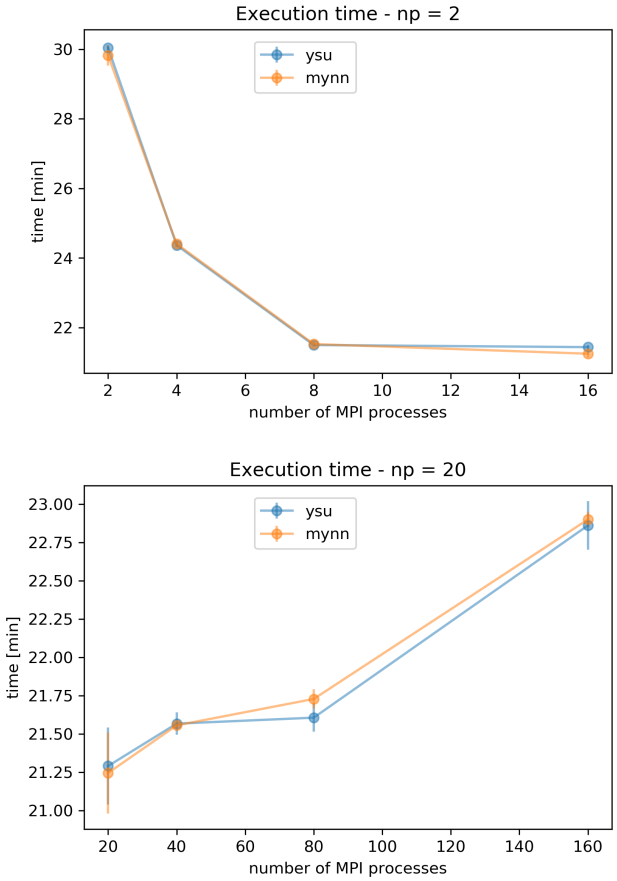


**Fig. 4.** Execution times of WRF runs for *YSU* and *MYNN* input configurations. *np* is the number of processes per node, so 1, 2, 4 and 8 nodes are used to obtain the total number of MPI processes in each execution.

The energy consumption in the previous cases does not show promising results. As can be observed in Fig. 5, increasing the number of nodes produces a monotonously increase, close to linear, in the energy consumption. Concentrating processes in nodes implies, in principle, some energy savings, but it occupies all the cores that can be eventually utilized by other users. So, from the power consumption point of view, it is advisable neither to use more than one node nor launching few MPI processes per node. As it is known, policies regarding

processes location, spreading, and/or consolidation must be properly evaluated in the context of the Resource Management System (Slurm in the present work) at the same time of considering the utilization profile (CPU utilization, average number of cores, average requested memory, etc.) by users. In this sense, the current work for characterizing the WRF execution profile opens the door to perform dynamic allocation of tasks that will benefit this evaluation. Such a dynamic allocation can be performed by means of the integration of ckeckpointing mechanisms into the Resource Management System [26].

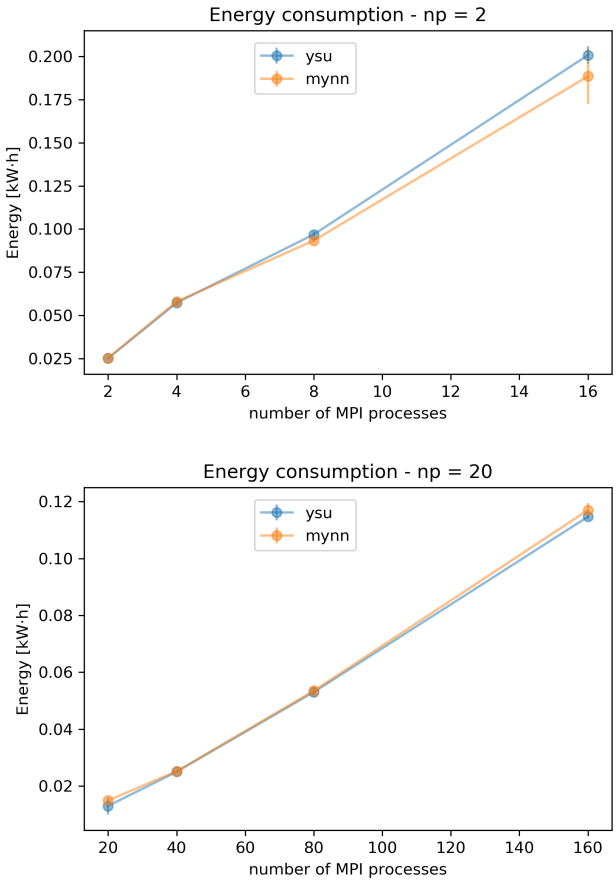


**Fig. 5.** Energy consumption of WRF runs. *np* is the number of processes per node, so 1, 2, 4 and 8 Xula nodes are used.

In Fig. 6 we show a scatter plot of the energy consumption versus the execution time for all 166 simulations considered. In this set of simulations, we include

those from Figs. 4 y 5 and additional ones performed with different *np*. Since, as we previously said, we repeat every setting at least three times, we observe in the small clusters the typical dispersion of the measured quantities. We also observe a huge variability in the energy consumption, whereas the computing time is more bounded.

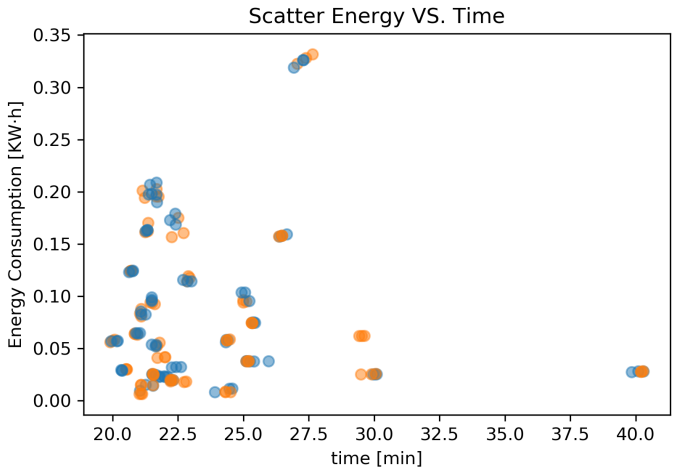


**Fig. 6.** Scatter plot of energy consumption and execution time for all numerical experiments (blue=*MYNN*, orange=*YSU*).

Similarly, in Fig 7 we show another scatter plot with the number of MPI processes versus the execution time, confirming the reduced scaling of the WRF run with the number of MPI processes.

Finally, in Table 3 we show the best cases, those that minimize the energy consumption or execution time, of all WRF executions. We can observe that:

- With *MYNN* input, increasing computing time 6% reduces energy consumption a factor of 9.
- With *YSU* input, increasing computing time 20% reduces energy consumption a factor of 7.

The use of the fastest settings indicated in Table 3 imply an excessive occupation of hardware resources, that is unaffordable because of the small gain in execution time.

Nevertheless, and taking into account that the presented results are obtained in a shared cluster used by a wide set of different users and computational requirements, there are several combinations of *np* and the number of nodes grouped altogether below 0.1 kWh in Fig. 7 and also below 50 MPI processes in

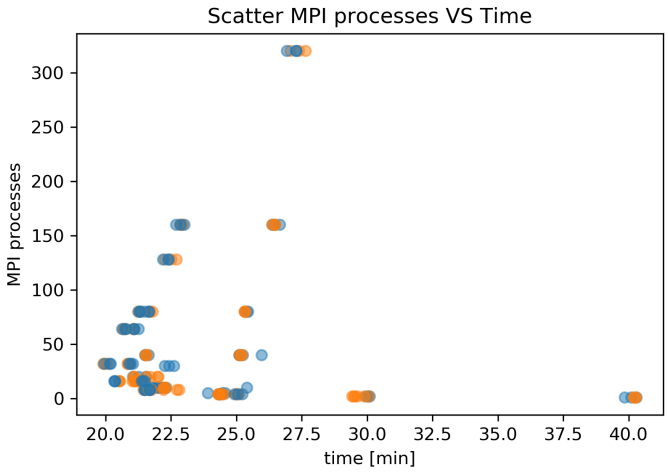


**Fig. 7.** Scatter plot of the number of MPI processes and execution time for all numerical experiments (blue=*MYNN*, orange=*YSU*).

| Input | Nodes | *np* | Ex. time | Energy |
|---|---|---|---|---|
| *MYNN* | 8 | 32 | **19.9 min** | 0.0560 kWh |
| *MYNN* | 1 | 16 | 21.0 min | **0.0063 kWh** |
| *YSU* | 8 | 32 | **19.9 min** | 0.0569 kWh |
| *YSU* | 1 | 5 | 23.9 min | **0.0081 kWh** |

**Table 3.** Cases with minimum execution time and energy consumption. See the text above for estimation of the errors.

Fig. 6, with execution times smaller than 25 minutes. Consequently, it is possible to define executions of WRF that balance the cluster occupation, the duration of the run, and the power cost, i.e., to find out say sub-optimal configurations that obtain a close-to-optimal trade-off for these three variables.

## 6 Conclusions and future work

It is not unusual that codes such as WRF are run on a daily basis, in particular to execute weather forecasting jobs in several geographical locations. This represents significant expenses in terms of human, material, and energy resources, that also are difficult to quantify. This is due to the nature of WRF and other similar codes, that scale differently in execution time and energy consumption in relation to the total number of processes generated and occupied nodes.

We have performed a preliminary study of the execution time and energy consumption for two input configurations of WRF, changing the number of nodes

and the number of processes in each node. We have characterized the strong scaling of WRF, its consumption and resource usage in a modern HPC cluster.

In the WRF cases here documented, the energy consumption grows linearly with the number of nodes used, whereas the computing time is barely reduced, or even increases when each nodes executes the maximum number of MPI processes allowed. However, reducing the number of processes per node does not imply a smaller consumption, but only a smaller cluster occupation.

Hence, it has been found that in this dedicated infrsatructure there are several sub-optimal deployments that compensate execution time, energy usage, and cluster occupancy. Among them, we see that is possible to drastically reduce the energy consumption of the nodes, with the corresponding refrigeration savings, at the expense to slightly increase the computing time. As previously commented, this outcome can profit from a dynamic allocation of tasks mechanism (based on checkpointing capabilities) that will optimize this trade-off. Even when of much interest, to find out the algorithm that rules this dynamic allocation of tasks taking also into account the infrastructure being exploited is out of the scope of this work. Such a endeavor will require of the application of artificial intelligence methodologies to design a scheduling algorithm that would optimize the use of a shared cluster as Xula is accordingly to the users' demands and historic behaviour.

This article is then the first step for such a future work in which a deeper study must be performed. Hence, an increase in the number of executions, code configurations, number of iterations even when a good reproducibility has been already obtained, consideration of the idle power consumption on additional nodes (when executing all processes in the same node), etc. ought to be performed. This bunch of experiments could be then complemented with the design of a new scheduling algorithm for an optimal use of the cluster that could profit from the available study of the researchers exploiting Xula [27].

At the same time, the achieved experience paves the way for future studies with other codes suitable in HPC job planning in the context of energy efficiency. In this sense, the previous conclusions and their potential extension to codes with a near perfect performance scaling should be confirmed or not in order to check if there are less possibilities of energy savings as the power consumption will follow a linear behaviour with the use of resources used.

## Acknowledgements

This work was partially funded by the Comunidad de Madrid CABAHLA-CM project (S2018/TCS-4423), the ADMIRE project Grant Agreement number 956748 (H2020-JTI-EuroHPC-2019-1), and the ENERXICO project Grant Agreement number 828947 (H2020–FETHPC-2018).

## References

1. New European Web Atlas. https://www.neweuropeanwindatlas.eu , accessed on June 2021.

2. Red Española de Supercomputación. https://www.res.es/, accessed on June 2021.
3. Nagios - the industry standard in it infrastructure monitoring. https://www.nagios.org/ ((2018-10-25)), accessed on June 2021.
4. Ganglia Monitoring System. http://ganglia.sourceforge.net/ ((2018-10-29)), accessed on June 2021.
5. Collectd - The system statistics collection daemon. https://collectd.org/ ((2018-10-30)), accessed on June 2021.
6. Ahmed, K., Tasnim, S., Yoshii, K.: Simulation of auction mechanism model for energy-efficient high performance computing. In: Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. p. 99104. SIGSIM-PADS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3384441.3395991, https://doi.org/10.1145/3384441.3395991
7. Calore, E., Gabbana, A., Schifano, S.F., Tripiccione, R.: Energy-efficiency tuning of a lattice boltzmann simulation using meric. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K. (eds.) Parallel Processing and Applied Mathematics. pp. 169–180. Springer International Publishing, Cham (2020)
8. Cascajo, A.: Daemonuser manual. https://www.arcos.inf.uc3m.es/acascajo/daemon/, accessed on June 2021.
9. Cascajo, A., Singh, D.E., Carretero, J.: Performance-aware scheduling of parallel applications on non-dedicated clusters. Electronics **8**, 982 (09 2019). https://doi.org/10.3390/electronics8090982
10. Cascajo, A., Singh, D.E., Carretero, J.: LIMITLESS - LIght-weight MonItoring Tool for LargE Scale Systems. In: Proceedings - 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2021. pp. 220–227. Institute of Electrical and Electronics Engineers Inc. (mar 2021). https://doi.org/10.1109/PDP52278.2021.00042
11. Cerf, S., Bleuse, R., Reis, V., Perarnau, S., Rutten, É.: Sustaining performance while reducing energy consumption: A control theory approach. In: Sousa, L., Roma, N., Tomás, P. (eds.) Euro-Par 2021: Parallel Processing. pp. 334–349. Springer International Publishing, Cham (2021)
12. Dlinnova, E., Biryukov, S., Stegailov, V.: Energy consumption of md calculations on hybrid and cpu-only supercomputers with air and immersion cooling. Advances in Parallel Computing **36**, 574–582. https://doi.org/10.3233/APC200087, https://ebooks.iospress.nl/volumearticle/53966
13. Dörenkämper, M., Olsen, T.B., Witha, Hahmann, A.N., Davis, N.N., Barcons, J., Ezber, Y., García-Bustamante, E., Gonzĺez-Rouco, J.F., Navarro, J., Sastre-Marugn, M., nd W. Trei, T.S., agar, M., Badger, J., Gottschall, J., Rodrigo, J.S., Mann, J.: The making of the new european wind atlas - part 2: Production and evaluation. Geosci. Model Dev. (13), 5079–5102 (2020). https://doi.org/DOI: 10.5194/gmd-13-5079-2020
14. Dupont, B., Mejri, N., Da Costa, G.: Energy-aware scheduling of malleable hpc applications using a particle swarm optimised greedy algorithm. Sustainable Computing: Informatics and Systems **28**, 100447 (2020). https://doi.org/https://doi.org/10.1016/j.suscom.2020.100447, https://www.sciencedirect.com/science/article/pii/S2210537920301712
15. Garrido, J., González-Rouco, J.F., Vivanco, M.G., Navarro, J.: Evaluation of surface temperature regional climate projections over the iberian peninsula. Clim. Dyn. (55), 3445–3468 (2020). https://doi.org/10.1007/s00382-020-05456-3

16. Hahmann, A.N., Sile, T., Witha, B., Davis, N.N., Dörenkämper, M., Ezber, Y., García-Bustamante, E., Rouco, J.F.G., Navarro, J., Olsen, B.T., Söderberg, S.: The making of the new european wind atlas - part 1: Model sensitivity. Geosci. Model Dev. (13), 5073–5078 (2020). https://doi.org/10.5194/gmd-13-5053-2020
17. Hong, S.Y., Noh, Y., Dudhia, J.: A new vertical diffusion package with an explicit treatment of entrainment processes. Monthly Weather Review (134, Issue 9), 23182341 (2006), https://doi.org/10.1175/MWR3199.1
18. Jiménez, P.A., Dudhia, J.: Improving the representation of resolved and unresolved topographic effects on surface wind in the wrf model. J. Appl. Meteor. Climatol. (51), 300316 (2012)
19. Kerbyson, D., Barker, K., Davis, K.: In: Analysis of the Weather Research and Forecasting (WRF) Model on Large-Scale Systems. vol. 15, pp. 89–98 (01 2007)
20. Mantovani, F., Garcia-Gasulla, M., Gracia, J., Stafford, E., Banchelli, F., Josep-Fabrego, M., Criado-Ledesma, J., Nachtmann, M.: Performance and energy consumption of hpc workloads on a cluster based on arm thunderx2 cpu. Future Generation Computer Systems **112**, 800–818 (2020). https://doi.org/https://doi.org/10.1016/j.future.2020.06.033, https://www.sciencedirect.com/science/article/pii/S0167739X19309781
21. Massie, M.L., Chun, B.N., Culler, D.E.: The ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing **30**(7), 817–840 (2004)
22. Moríñigo, J.A., García-Muller, P., Rubio-Montero, A.J., Gómez-Iglesias, A., Meyer, N., Mayo-García, R.: The Journal of Supercomputing **76**(9), 68346859 (2020). https://doi.org/10.1007/s11227-019-03142-8
23. Nakanishi, M., Niino, H.: An improved mellor-yamada level-3 model with condensation physics: Its design and verification. Boundary-Layer Meteorol. (112), 1–31. (2004)
24. Patel, T., Wagenhuser, A., Eibel, C., Hnig, T., Zeiser, T., Tiwari, D.: What does power consumption behavior of hpc jobs reveal? : Demystifying, quantifying, and predicting power consumption characteristics. In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 799–809 (2020). https://doi.org/10.1109/IPDPS47924.2020.00087
25. Rodríguez-Pascual, M.A., Moríñigo, J.A., Mayo-García, R.: Effect of mpi tasks location on cluster throughput using nas. Clust. Comput. **22**(4), 11871198 (2019). https://doi.org/10.1007/s1058 6-018-02898-7
26. Rodríguez-Pascual, M., Cao, J., Moríñigo, J.A., Cooperman, G., Mayo-García, R.: Job migration in hpc clusters by means of checkpoint/restart. The Journal of Supercomputing **75**, 6517–6541 (2019). https://doi.org/- 10.1007/s11227-019-02857-y, https://doi.org/10.1007/s11227-019-02857-y
27. Rodríguez-Pascual, M., Rubio-Montero, A.J., Moríñigo, J.A., Mayo-García, R.: Execution data logs of a supercomputer workload over its extended lifetime. Data in Brief **28**, 105006 (2020). https://doi.org/https://doi.org/10.1016/j.dib.2019.105006, https://www.sciencedirect.com/science/article/pii/S2352340919313629
28. Shainer, G., Lui, P., Liu, T., Wilde, T., Layton, J.: The impact of inter-node latency versus intranode latency on hpc applications. In: Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems. p. 455460 (2011). https://doi.org//10.2316/p.2011.757-005
29. Stull, R.B.: An introduction to boundary layer meteorology. Kluwer Academic Publishers, Dordrecht, Boston, London (1988)

30. Szustak, L., Wyrzykowski, R., Olas, T., Mele, V.: Correlation of performance optimizations and energy consumption for stencil-based application on intel xeon scalable processors. IEEE Transactions on Parallel and Distributed Systems **31**(11), 2582–2593 (2020). https://doi.org/10.1109/TPDS.2020.2996314
31. Tracey, R., Hoang, L., Subelet, F., Elisseev, V.: Ai-driven holistic approach to energy efficient hpc. In: Jagode, H., Anzt, H., Juckeland, G., Ltaief, H. (eds.) High Performance Computing. pp. 267–279. Springer International Publishing, Cham (2020)
32. Vegas-Cañas, C., González-Rouco, J.F., Navarro-Montesinos, J., García-Bustamante, E., Lucio-Eceiza, E.E., García-Pereira, F., Rodríguez-Camino, E., Chazarra-Bernabé, A., Alvarez-Arévalo, I.: An assessment of observed and simulated temperature variability in the sierra de guadarrama. Atmosphere (11), 985 (2020)
33. Yoo, A.B., Jette, M.A., Grondona, M.: Slurm: Simple linux utility for resource management. In: Workshop on Job Scheduling Strategies for Parallel Processing. pp. 44–60. Springer (2003)
34. Zhang, C., Yuan, X.: Processor affinity and mpi performance on smp-cmp clusters. In: IEEE International Symposium Parallel and Distributed Processing. pp. 1–8. Atlanta,USA (2010). https://doi.org/10.1109/IPDPSW.2010.5470774