GI_Forum 2019, Issue 1
Page: 69 - 86
Full Paper
Corresponding Author:
c.erlacher@cuas.at
DOI: 10.1553/giscience2019_01_s69

Parallel and Distributed Computing for large raster-based Spatial Multicriteria Decision Analysis Problems: A Computational Performance Comparison

Christoph Erlacher[1,2], Angelika Desch[1], Karl-Heinrich Anders[1], Piotr Jankowski[3] and Gernot Paulus[1]

[1]Carinthia University of Applied Sciences, Villach, Austria
[2]University of Salzburg, Austria
[3]San Diego State University, USA

Abstract

This article focuses on a cluster-based parallel and distributed approach for large raster datasets in the context of Spatial Multicriteria Decision Analysis (S-MCDA). The research addresses a land-prioritization model with respect to conservation practices. The reliability of the model results is examined using a variance-based Spatially-Explicit Uncertainty and Sensitivity (SEUSA) framework. The original case study area to which we applied the model was located in southwest Michigan, USA, and incorporated millions of mapping units (pixels). As part of the model sensitivity analysis, several thousand intermediate raster datasets representing suitability surfaces are generated by means of a Monte Carlo Simulation (MCS). The creation of the suitability surfaces represents the most time-consuming and memory-intensive step within the SEUSA framework. Sequential computational approaches to implementing SEUSA often have to accept a compromise with respect to problem size and the number of simulations, resulting in the low quality of the model sensitivity measures. This article presents the concept and implementation of a distributed and parallel solution based on the Python-Dask framework in order to improve the quality of SEUSA results for computationally-intensive spatial models.

Keywords:

parallel and distributed computing, Python Dask framework, Monte Carlo Simulation, spatially-explicit uncertainty and sensitivity analysis, spatial multi-criteria decision analysis

1 Introduction, Motivation and Problem Definition

Multicriteria Decision Analysis (MCDA) methods and their applications in various spatial and non-spatial domains have been explored for decades (Hwang & Yoon (1981); Malczewski (1999); Malczewski (2006); Malczewski & Rinner (2015); Tzeng & Huang (2011); Penadés-Plà et al. (2016)). In this article, we propose a raster-based parallel and distributed

solution in the context of Spatial Multicriteria Decision Analysis (S-MCDA) that comprises the Spatially-Explicit Uncertainty and Sensitivity Analysis Framework (SEUSA-Framework) suggested by Ligmann-Zielinska & Jankowski (2008, 2014). The S-MCDA-related terminology used here is based on Malczewski (1999), who described the framework and the components embedded in its 'Intelligence Phase', 'Design Phase' and 'Choice Phase' (p. 96).

S-MCDA can support experts during the decision-making process for application domains such as landscape assessment, hazard risk assessment, environmental protection, land-use planning and sustainable regional development. The majority of S-MCDA applications do not provide detailed information about the robustness of model results. Malczewski & Rinner (2015, p. 192) identify criterion scores and weights as the main sources of uncertainty in S-MCDA. Therefore, performing uncertainty and sensitivity analysis is an important step in S-MCDA towards improving the decision-aiding process. Within this research, uncertainty regarding the criterion weights is addressed in order to allow a comparison of the sequential SEUSA-Framework (Ligmann-Zielinska & Jankowski, 2008, 2014) and the parallel and distribution approach with respect to improved runtimes.

The available sensitivity analysis methods can be broadly categorized as local and global methods (Wainright et al., 2014). Global Sensitivity Analysis (GSA) methods account for interdependencies among model input factors, in contrast to local sensitivity analysis methods that focus on first-order effects – one factor at a time in isolation from interactions with other factors. A disadvantage of GSA methods such as variance-based sensitivity analysis is that in order to obtain meaningful sensitivity values, a large number of weight samples need to be included to perform the time-consuming Monte Carlo Simulations (MCS). Additionally, the number of criterion maps, the aggregation methods and the size of the project area specified by the number of pixel locations influence the computational demand. Each location represents an alternative that has to be considered for a defined S-MCDA use case. For example, large S-MCDA problems incorporate millions of alternatives and several hundred thousand simulations which are likely to be beyond the capacity of a single personal computer. Although there are powerful GPU (Graphic Processing Units) workstations that can process massive computations very fast, these workstations are subjected to limitations concerning data storage and data flow capacity, especially between the computer's GPU and CPU. As an alternative to GPUs, parallel and distributed frameworks for computer clusters offer the possibility of spreading the workload among several cluster nodes. Hence, the main objective of this research is to develop a scalable and adaptable approach for performing parallel and distributed spatially-explicit uncertainty and sensitivity analysis. The findings represent a fundamental contribution to the development of a parallel and distributed SEUSA-Framework, answering the need for extensibility concerning various S-MCDA decision rules and sensitivity analysis methods that is suitable for various application domains.

2 Methodology, Concept and Implementation

The focus is on the conceptual development of the parallel and distributed approach to speed up the computationally- and memory-intensive steps of the SEUSA-Framework, and on the procedure for selecting a suitable parallel and distributed Python framework according to the requirement analysis. The study site is located in Southwest Michigan, USA and includes Allegan, Barry, Cass, Kalamazoo, St Joseph and Van Burren counties. The site was adopted from Şalap-Ayça and Jankowski (2016) because it represents an attractive area for testing the parallel and distributed approach with regard to large raster datasets.

2.1 Experimental Setup

Our focus was on a land-prioritization model that refers to the Environmental Benefit Index (EBI). The model represents a simple scoring procedure and comprises the criterion raster maps 'Wildlife', 'Water Quality', 'Soil Erosion', 'Enduring Benefits' and 'Air Quality'. According to Hellerstein (2017), the EBI represents perhaps the most crucial design element for prioritizing agricultural land units with respect to greater environmental benefit. A more detailed description of the original case study and criteria can be found in Şalap-Ayça and Jankowski (2016, p. 114).

Within the scope of this research, each criterion is represented by a raster map consisting of approximately 13 million pixels. The case study therefore presents a sound experimental setup for a parallel and distributed computing approach incorporating hundreds of thousands of simulations. Following the S-MCDA approach (Malczewski, 1999), each raster layer was standardized using the score range procedure (see Figure 1, which includes the formula).

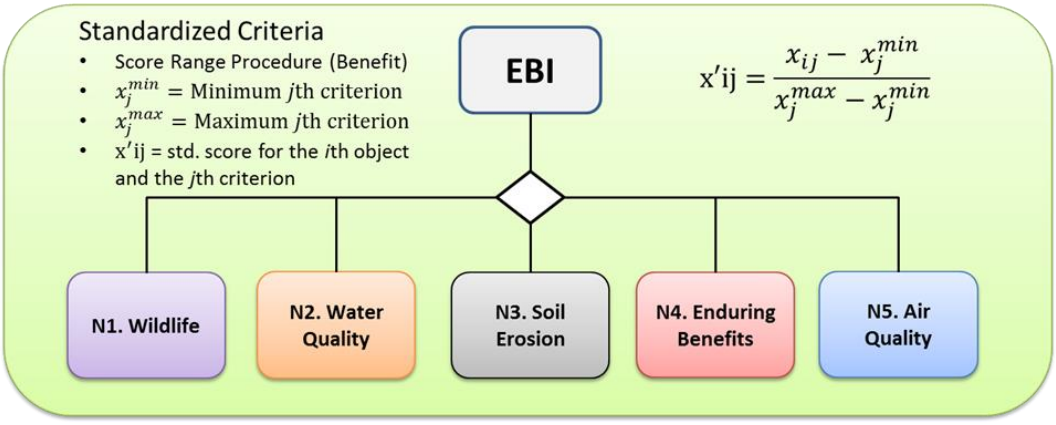

Figure 1: Standardizing the main criteria 'Wildlife', 'Water Quality', 'Soil Erosion', 'Enduring Benefits' and 'Air Quality' by using the score range procedure (Malczewski, 1999, p. 118).

2.2 The SEUSA Framework

For this research, a spatial variance-based sensitivity analysis approach that draws on Ligmann-Zielinska & Jankowski (2008, 2014) was used. The approach incorporates the following steps:

(I) generation of the weight samples for the criteria of the S-MCDA model, carried out using SimLab 2.2 software, which incorporates the quasi-random radial sampling method proposed by Sobol' (1993). The total number of simulation runs can be expressed by equation (2), where j is the number of criteria and N the number of weight samples:

$$R = (j + 2) * N \tag{2}$$

(II) running the Monte Carlo Simulation (MSC) incorporating the previously-generated weight samples and a specific decision rule such as the Simple Additive Weighting (SAW) or the Ideal Point (IP) method, resulting in the generation of suitability surfaces. As mentioned by Malczewski (1999, p. 199), the SAW method, also called the Weighted Linear Combination scoring procedure, is frequently used in the context of S-MCDA for its transparency and simplicity. Each standardized criterion value x_{ij} is multiplied by the associated weight value w_j and followed by the product summation (Drobne & Lisec, 2009, p. 464), where the index *i* represents the specific pixel location within the study area, and the index *j* represents the specific standardized criterion map (see equation 3). In contrast, the IP decision rule calculates the relative closeness rc_{i+} to the IP (see equation 4), where s_{i+} denotes the IP and s_{i-} indicates the negative IP (Malczewski, 1999, p. 225). This decision rule is also known as the Technique for Order Preference by Similarity to the Ideal Solution (TOPSIS) and was originally introduced by Hwang and Yoon (1981). As stated by Malczewski (1999, p. 226), the IP method avoids *some of the difficulties associated with the interdependence-among-attributes assumption*. Several studies have integrated TOPSIS in the context of S-MCDA (Ligmann-Zielinska & Jankowski, 2014; Feizizadeh et al., 2014; Şalap-Ayça & Jankowski, 2016; Erlacher et al., 2017).

$$s_i = \sum_j w_j x_{ij} \tag{3}$$

$$rc_{i+} = \frac{s_{i-}}{s_{i+} + s_{i-}} \tag{4}$$

(III) the maps (i.e. raster surfaces) for the uncertainty (standard deviation) and sensitivity indices are computed from the stack of suitability maps obtained in step II in order to identify criteria that cause the variability. A more detailed description of the SEUSA Workflow can be found in Ligmann-Zielinska & Jankowski (2014).

The most crucial part of the SEUSA-Framework concerning the computational demand and CPU memory requirements is the generation of the suitability surfaces incorporating the weight samples and the decision rule. **Fehler! Verweisquelle konnte nicht gefunden werden.** illustrates the steps for creating the suitability and uncertainty surfaces that are pertinent for the performance comparisons.

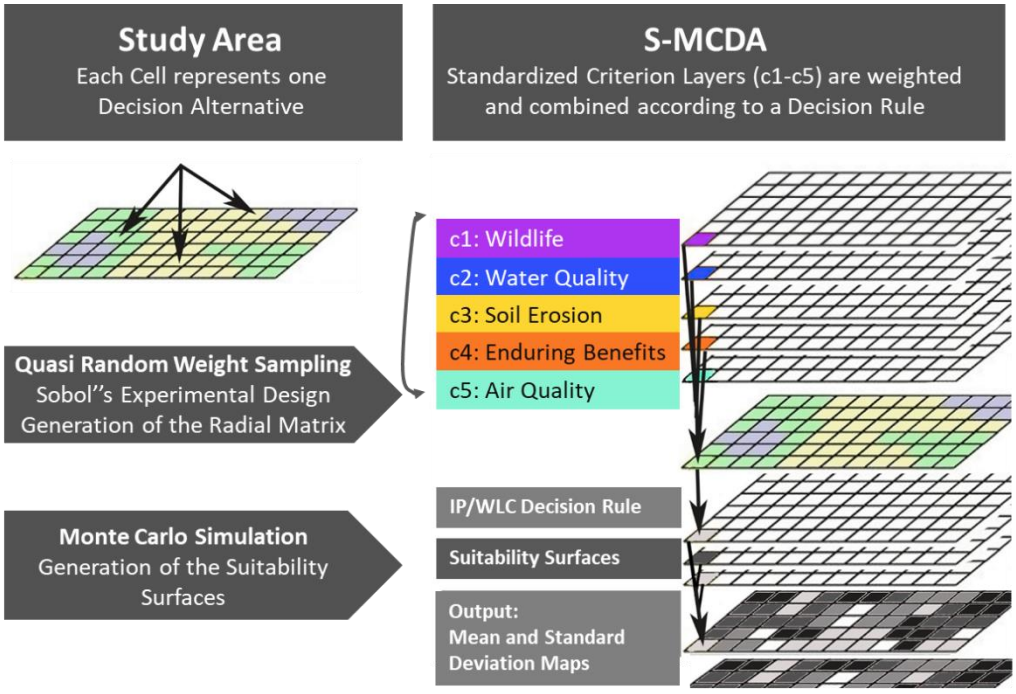

Figure 2: Workflow for generating the suitability and uncertainty surfaces that are relevant for the performance comparison.

2.3 Selection of a Parallel and Distributed Python Framework

This section provides an overview of parallel and distributed computing relevant for the Python framework selection. For a comprehensive introduction to parallel and distributed computing, the reader is referred to Duncan (1990), Foster (1995), Yang et al. (2011), Rauber & Rünger (2013), Singh (2013), Rizvi (2016), Gu et al. (2017), and Trobec et al. (2018). The computational performance indicators identified by Desch (2018) (including load balancing, fault tolerance, debugging instruments and scalability) that are relevant for the selection of parallel and distributed Python framework selection were integrated into a simple multicriteria decision analysis model.

Serial computations subdivide tasks into a discrete set of instructions and are executed by a single processor unit one after another. The original SEUSA-Framework proposed by Ligmann-Zielinska and Jankowski (2008, 2014) refers to the sequential Python approach,

which compromises between the number of pixel locations and the number of simulations. The studies conducted by Feizizadeh et al. (2014) and Şalap-Ayça and Jankowski (2016) also refer to the sequential SEUSA-Framework solution and are limited by computational capacity. In contrast to sequential computing, parallel computing carries out a set of instructions simultaneously by using multiple processor units (e.g. CPUs [Central Processing Units], and/or GPUs [Graphic Processing Units]). Erlacher et al. (2017) presented a GPU-based concept (parallel computations) that achieved a respectable computational acceleration of the time-consuming MCS but was limited by the available memory capacity.

According to Flynn's Taxonomy (Flynn, 1972), multiprocessor units can be characterized by the dimensions of instructions (tasks) and data. The architectures for multiple instruction streams and multiple data streams represent the most common types of parallel computing nowadays and are applicable for clusters of workstations that incorporate multicore processors (nodes) (see Figure 3), grid computing, or supercomputers, for example. In this article, we refer to networked parallel computer clusters, where a specific control unit, known as a master scheduler, distributes the workload. A detailed description of the local cluster used can be found in section 2.5, 'Design of the Performance Comparison'.

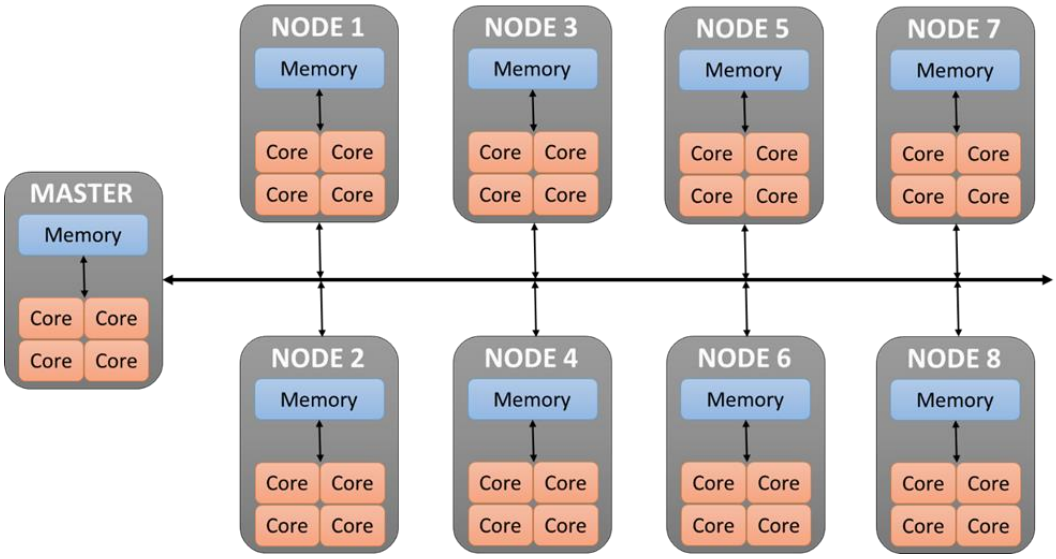

Figure 3: Illustration of a parallel cluster: workstations (nodes) connected via a network.

Nodes are connected via Ethernet. They communicate by passing messages incorporating the Message Passing Interface (MPI) standard and use guidelines for data exchange within local networks (e.g. Transmission Control Protocol). Challenges concerning parallel and distributed computing might arise because of communication constraints (network bandwidth and latency dependency), source code migration (from sequential solution to parallel solution), maintenance and debugging difficulties. For example, package inspection,

depending on the package size, can slow down communication within the local network. In order to select the most suitable parallel and distributed Python framework, the following indicators were considered: supported platforms (Windows, Linux, Unix) and hardware (CPU and GPU), applicable for local clusters; load balancing; support of NumPy data structure; fault tolerance (continue operating properly); scalability (ability to add nodes easily); debugging instruments; detailed documentation and support (e.g. learning materials, developer forum, user activity); installation; maintenance issues. A comprehensive explanation of all indicators, along with the parallel and distributed Python frameworks in the context of the multicriteria evaluation model, can be found in Desch (2018).

2.4 Parallel and Distributed Approach based on Python-Dask

The Python-based parallel and distributed solution selected in this research for generating the suitability and uncertainty surfaces incorporates the open source Dask-Framework, which outperformed the other frameworks investigated. The most important advantages of the Dask-Framework are the ease of installation and maintenance, the support of *NumPy* objects and diagnostic tools (e.g. real-time and responsive dashboard), the applicability for local clusters as well as their easy scalability, the moderate migration effort concerning decision rules, and its fault-tolerant behaviour. Furthermore, Dask also supports libraries including Pandas and Scikit-Learn (machine learning), and is being continually developed. A more detailed and very recent description of the advantages and examples of applications can be found in Daniel (forthcoming, July 2019).

Several steps were taken to adapt the sequential approach to the proposed parallel and distributed solution. Figure 4 highlights the workflow differences between the sequential and the parallel/distributed solutions. The Dask-Framework offers dynamic task-scheduling and provides schedulers for single-threaded, multi-threaded, multi-process and distributed execution (Rocklin, 2015, p.129). In our project, the distributed Dask scheduler was integrated for the performance of simulation runs. The workstations within the local cluster used in our research communicate with each other via TCP in a peer-to-peer data-sharing model. Each node has access to the input dataset over the Network File System (NFS), in order to minimize network communication. For the input dataset, the Hierarchical Data Format (HDF5) is used. This contains the criterion maps, with metadata such as the minimum and maximum criterion values. Both implementations (the sequential and the parallel solutions) use N-dimensional *NumPy* arrays for storing the input raster datasets, the intermediate suitability surfaces, and the uncertainty surfaces (average and standard deviation map).

In contrast to the sequential implementation, the Dask-Framework creates *dask.arrays* that represent a parallel array library and comprise the *NumPy* interface (Rocklin, 2015, p.126). The *dask.array* submodule utilizes *Dask Graphs*, a Python dictionary that maps keys to tasks or values, by using all resources (e.g. cores and memory) of the local cluster, in order to operate on large-size datasets (Rocklin, 2015, p.127). The whole *dask.array* is shaped into

blocks (*NumPy* arrays) and incorporates a tuple of integer values, which have to be considered in respect of different simulation sizes. The first index indicates the criterion map and remains static, whereas the criterion maps' indices for row and column are dynamically determined. The higher the number of model runs, the smaller the chunk sizes for row and column. Additionally, the decision rule used is a further important indicator for generating the blocks. Therefore, the total memory required for the whole computation has to be incorporated in order to avoid memory overload for a specific node.

The nodes' capabilities regarding the system's memory and threads per core are further relevant aspects for defining the chunk sizes, which are important properties for the scheduler concerning a balanced distribution of the workload among the workers. Each task, such as the generation of the suitability surfaces (including various decision rules) and the creation of the average and standard deviation map, represents a Python function and is mapped across all blocks of the *dask-array*. The tasks are submitted on *futures*, which extends the *concurrent.futures* Python interface and tracks the status of the tasks among the various workers.

The execution is triggered by calling the *NumPy* function *asarray()*. This function converts the *dask.array* into a *NumPy* array and stores the blocks (incorporating the suitability values) in the memory. Each block of the suitability surfaces consists of a three-dimensional array in which the first two indices indicate the subarea of the study site and the third represents the EBI values for each simulation run. These suitability values represent the input for calculating the average and standard deviation map, which is written to a new ASCII dataset.

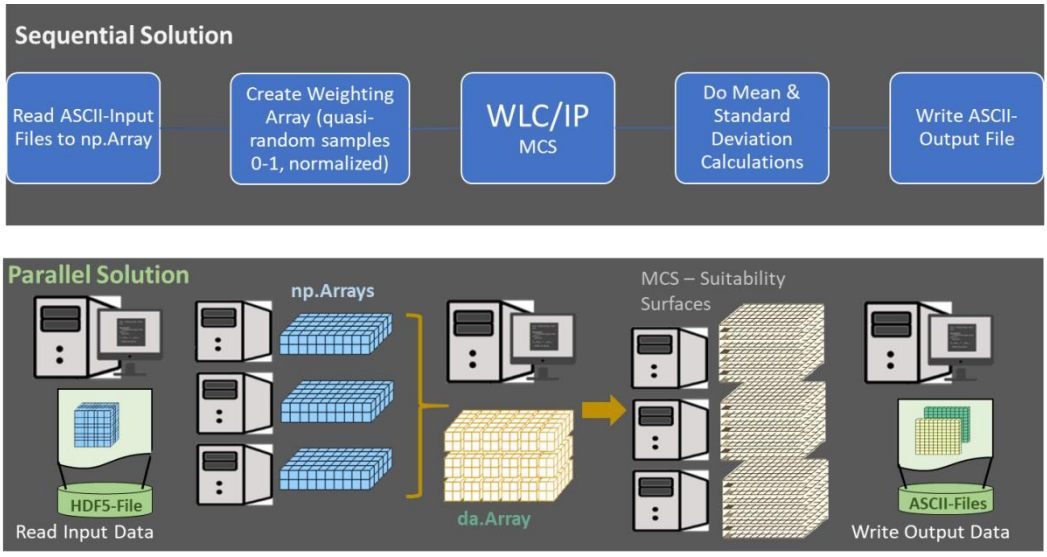

Figure 4: Comparison between the sequential processing workflow (top) and the Dask-Python-based parallel and distributed workflow (bottom). np.Arrays refers to NumPy arrays; da.Array refers to dask.array.

2.5 Design of the Performance Comparison

The performance tests conducted referred to a local cluster that included 16 nodes and one master acting as scheduler. Each node (HP-Z420 Workstation) incorporated the Intel® Xeon® CPU E5-1603 v3 2.8 GHz processor (four cores and one thread per core), 16 GB CPU memory, an NVIDA® Quadro K2000 with 2 GB GPU memory, and ran on the Windows 10 (64-bit version) platform. The workstations were configured for GIS-based education and were frequently in use. Therefore, the performance tests were conducted during non-lecture periods only, especially during weekends, holidays and semester breaks, when the utilization rate of the local network was at its lowest. The GIS laboratory was equipped with 24 workstations, but for the final performance comparisons, we used a set of pre-selected workstations that demonstrated less runtime variability. For the performance comparison the following settings were defined:

(I) fixed simulation size (2,464 model runs) and variable cluster size (1, 2, 4, 8, 12 and 16 nodes).
(II) fixed cluster size (16 nodes) and variable simulation size (2,464; 4,928; 9,856; 19,712 and 39,424 model runs).

The same settings were applied for both decision rules, i.e. the weighted linear combination (WLC) and the IP methods. Each simulation setting was performed 30 times in order to retrieve meaningful runtime measurements.

Simulation runs that were affected by node failure still generated correct suitability values, but were not included in the time measurements. Additional uncontrollable aspects like background process, virus scanning tools, firewall and package inspection should be considered uncertainties in the performance comparison. For the performance analysis, the speed-up and efficiency metrics (Rauber & Rünger 2013, pp. 180, 182; Trobec et al., 2018, p.10) were considered.

3 Proof of Concept and Interpretation of the Performance Comparisons

This section focuses on the results of the performance comparison for different cluster and simulation sizes, including both decision rules (IP and WLC). In order to carry out the comparisons, the scheduler on the master node has to be initialized via a command line; each worker node is allocated to the scheduler according to an Internet Protocol Address. The box plots in Figures 9–12 (see Appendix) illustrate the distributions of the measuring sequences for the performance comparison.

3.1 Fixed Simulation Size and variable Cluster Size

The analysis setting is for a fixed simulation size of 2,464 model runs on cluster sizes of 1, 2, 4, 8, 12 and 16 nodes. Thirty repetitions were conducted for each cluster size and decision rule. Figure 5 illustrates the measurement sequence for different numbers of nodes, where the runtime is given in seconds. Using the IP method, the runtime for one worker node running in parallel is approximately 18 minutes and 31 seconds; for the WLC method, it is approximately 7 minutes and 50 seconds. The runtime differences are due to the significantly higher computational complexity of the IP method.

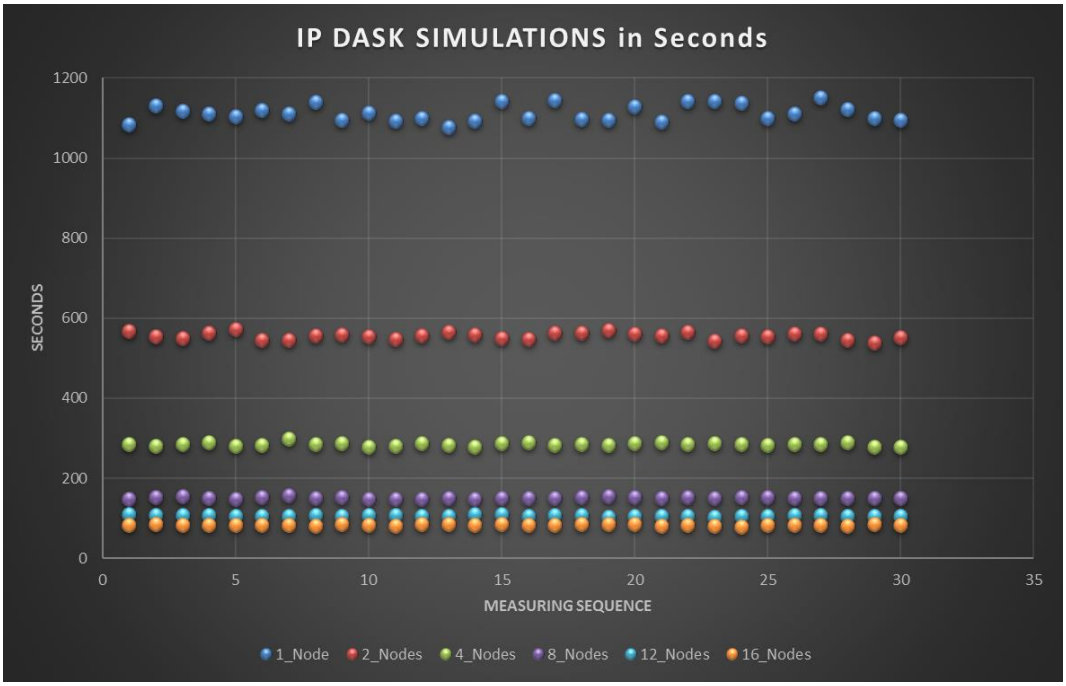

Figure 5: IP Dask measuring sequence for 2,464 model runs, with different numbers of nodes incorporated in the local cluster.

Table 1 compares the performance metrics for runtime in seconds (T(n)), speed-up (S(n)) and efficiency (E(n)), for both the IPC and WLC decision rules, where *n* indicates the number of worker nodes. Figure illustrates the speed-up for the IP decision rule for different cluster sizes: there is a linear increase in speed-up as the local cluster is increased in size. The speed-up and the efficiency are based on the comparison of one worker running the computations in parallel on the one hand, with other numbers of worker nodes which are also running the computations in parallel. The efficiency decreases to approximately 83% by expanding the local cluster to include 16 worker nodes. The speed-up for two nodes using the WLC method is marginally higher than expected, which might be caused by small

differences in the computational capacity of the nodes. It should be noted that the WLC method uses a simple aggregation technique with lower computational complexity. Consequently, reading and writing operations account for a greater proportion of the overall runtime.

Table 1: Performance metrics for different numbers of worker nodes: T(n) – runtime in seconds (median), S(n) – speed up, and E(n) – efficiency, for the IP and WLC methods.

Metrics	1 Node	2 Nodes	4 Nodes	8 Nodes	12 Nodes	16 Nodes
T(n) | IP	1111.342	556.695	284.593	151.412	106.572	83.619
S(n) | IP	1.000	1.996	3.905	7.340	10.428	13.290
E(n) | IP	1.000	0.998	0.976	0.917	0.869	0.831
T(n) | WLC	469.657	228.668	118.387	63.537	46.061	34.963
S(n) | WLC	1.000	2.054	3.967	7.392	10.196	13.433
E(n) | WLC	1.000	1.027	0.992	0.924	0.850	0.840

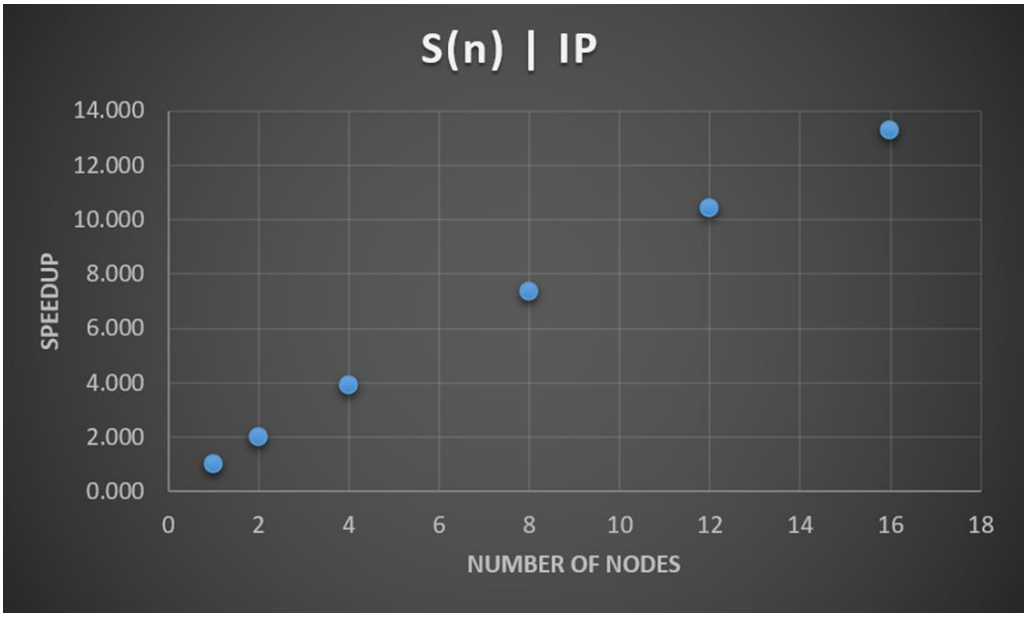

Figure 6: Speed-up S(n) for the IP method using different cluster sizes and 2,464 model runs.

A direct comparison between the original SEUSA-Workflow and the parallel and distributed solutions was not possible: the number of simulations in conjunction with the number of locations and the method employed exceeded the combined memory capabilities of the workstations. When comparing one worker in parallel mode using four cores with one worker using just one core (*Dask LocalCluster*), speed-ups of approximately 2.54 (WLC method) and 1.63 (IP method) were achieved. It should be mentioned that the block sizes for the IP simulations are smaller than those for the WLC simulations. The IP requires more memory capacity because of generating the ideal and anti-ideal intermediate surfaces. Additionally, reading and writing operations, background processes of the operating system, and scheduling the computations within the local cluster detract from the parallelization gains. As stated by Rauber & Rünger (2013, p.182), the number of cores represents an upper bound concerning the theoretical speed-up. This speed-up is limited by the degree of parallelism and can be quantitatively captured by applying Amdahl's law.

3.2 Fixed Cluster Size and variable Simulation Size

In order to test the influence of the number of simulation runs on the performance indicators, a fixed cluster size of 16 nodes was used. The increasing simulation sizes ranged from 2,464 through 4,928, 9,856, 19,172 to 39,424 runs. **Fehler! Verweisquelle konnte nicht gefunden werden.** shows the runtimes for both decision rules. In comparison to the IP method, when the number of model runs is increased the computational demand for the WLC method indicates a more linear relationship. Figure clearly illustrates the trend of the non-linear relationship between the runtime and the number of model runs for the IP method. The runtimes for the IP simulations incorporating 78,848 and 157,696 model runs were almost 2 and 7.5 hours respectively. It should be noted that for both simulation runs (see the red dots in Figure) only a few iterations were conducted. The runtimes achieved should therefore be treated as approximate values only.

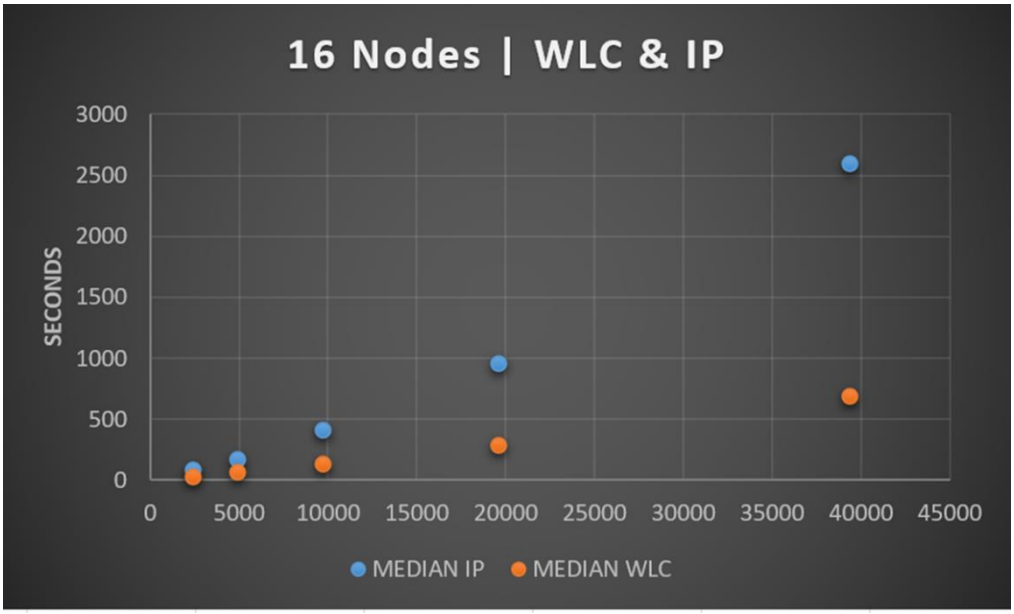

Figure 7: Computation times for the IP and the WLC methods for different numbers of model runs (x-axis).

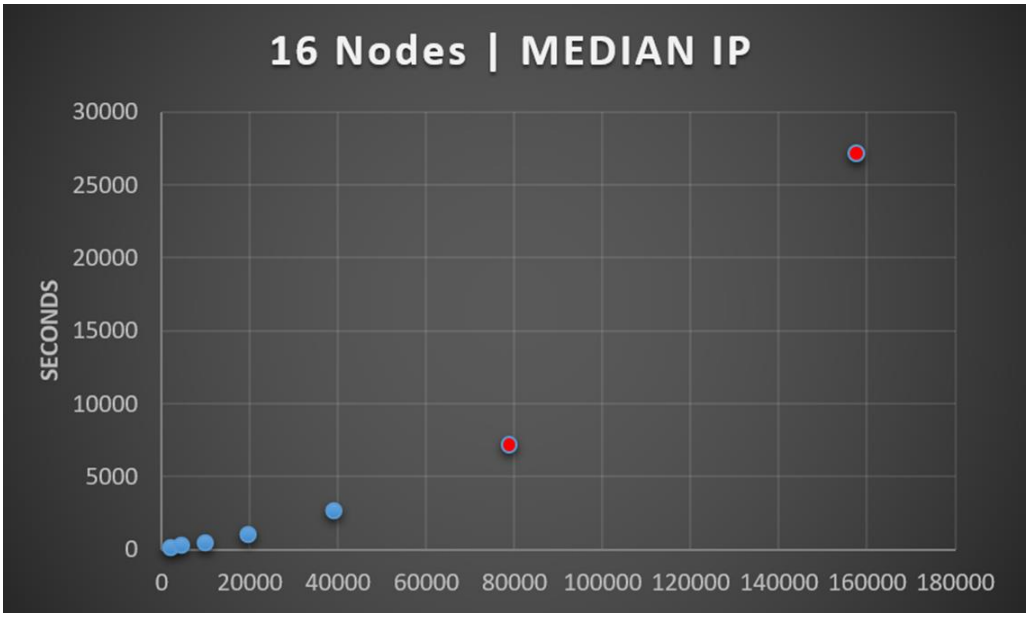

Figure 8: Non-linear relationship with respect to runtime (y-axis) for the IP method, achieved by increasing the number of model runs (x-axis). The red dots indicate the runtimes for 78,846 and 157,696 runs respectively.

4 Summary, Discussion and Future Prospects

This paper has presented a Python-based parallel and distributed approach for Spatial Multicriteria Decision Analysis (S-MCDA) for a use case that focused on land prioritization in an area of southwest Michigan (US), the map for which incorporated almost 13 million pixels. The research addresses a variance-based Spatially-Explicit Uncertainty and Sensitivity Analysis Framework (SEUSA-Framework) based on Ligmann-Zielinska & Jankowski (2008, 2014) and emphasizes the conceptual and computational performance differences between the sequential approach on the one hand, and the parallel and distributed approach on the other. A notable acceleration was achieved for the parallel and distributed solution based on the Dask-Framework. For example, the performance comparison between the distributed cluster including 16 worker nodes and the local cluster utilizing one core of a worker node revealed significant speed-ups for the former configuration, of 21.66 and 34.12 for the IP and WLC methods respectively for 2,464 model runs. The direct comparison between the sequential SEUSA-Framework, the GPU-based solution proposed by Erlacher et al. (2017), and the parallel and distributed solution presented in this paper requires additional coding effort to standardize the comparison bases.

The proposed parallel and distributed approach represents a runnable solution for large raster datasets that incorporate thousands of simulations. This parallel and distributed SEUSA approach can be used for different raster-based S-MCDA problems that incorporate multiple benefit and cost criteria as well as multiple competing objectives and interests (various stakeholder views). The concept of the parallel and distributed solution is currently limited to global S-MCDA techniques and will have to be adapted in order to incorporate local weight changes within neighbourhoods, a requirement that could be addressed by further development of the approach we have presented here.

Additionally, the approach can be extended by the integration of further decision rules, such as Ordered Weighted Averaging, to allow trade-off and comparative risk levels to be dealt with. Furthermore, an S-MCDA problem that includes alternatives presented in a vector-based format (e.g. points, polylines and polygons), where columns express the criteria, can be converted to multidimensional *NumPy* array objects.

The approach can be further developed to more effectively accommodate the generation of sensitivity surfaces (first-order and total-order sensitivity maps), by adding an additional task to the dask.array. Additionally, in the parallel and distributed approach proposed here, the use of worker nodes equipped with higher-end GPUs is expected to result in performance gains.

In summary, the proposed parallel and distributed approach offers a simple and scalable means to increase the applicability of the original SEUSA framework. Furthermore, this Dask-based approach is an appropriate solution for existing clusters within local networks, common in academic institutions and private companies, because it does not disturb ongoing operations and does not cause conflicts with regard to software installation dependencies.

Acknowledgement

Special thanks go to our colleagues Dr. Adrijana Car and Melanie Erlacher, MSc for their constructive feedback concerning the structure and the linguistic editing of this contribution.

References

Daniel, C.J. (in press). *Data Science with Python and Dask*. Manning Publications.

Desch, A. (2018). Parallel and Distributed Geoprocessing with Python: A S-MCDA Approach (Unpublished bachelor thesis). Carinthia University of Applied Sciences.

Drobne, S., & Lisec, A. (2009). Multi-attribute Decision Analysis in GIS: Weighted Linear Combination and Ordered Weighted Averaging. *Informatica* 2009, 33, 459-474.

Duncan, R. (1990). A Survey of Parallel Computer Architectures. *Computer*, 23(2), 5-16. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=44900

Erlacher, C., Jankowski, P., Paulus, G., & Anders, K.-H. (2017). A GPU-Based Parallelization Approach to Conduct Spatially-Explicit Uncertainty and Sensitivity Analysis in the Application Domain of Landscape Assessment. *GI_Forum 2017, Journal for Geographic Information Science*, 5(1), 44-58. Retrieved from https://doi.org/DOI:10.1553/giscience2017_01_s44

Feizizadeh, B., Jankowski, P., & Blaschke, T. (2014). A GIS based Spatially-explicit Sensitivity and Uncertainty Analysis Approach for Multi-Criteria Decision Analysis. *Computers and Geosciences*, 64, 81-95.

Flynn, M.J. (1972), Some Computer Organizations and their Effectiveness. *IEEE Transactions on Computers*. 21(9), 948-960. Retrieved from https://ieeexplore.ieee.org/document/5009071

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Bosten, USA: Addison-Wesley Longman.

Gu, Z., Wang, Y., Hua, QS., & Lau, F.C.M (2017). Distributed Computing. In: Gu, Z., Wang, Y., Hua, QS., & Lau, F.C.M (Eds.), *Rendezvous in Distributed Systems*. Singapore: Springer.

Hellerstein, D. M. (2017). The US Conservation Reserve Program: The evolution of an enrollment mechanism. Land Use Policy, 63, 601-610.

Hwang, C. L., & Yoon, K. (1981). *Multiple Attribute Decision Making Methods and Applications: A State of the Art Survey*. Springer-Verlag, Berlin.

Kshemkalyani, A. D., & Singhal, M. (2008). *Distributed Computing Principles, Algorithms, and Systems*. Cambridge, UK: Cambridge University Press.

Ligmann-Zielinska, A., & Jankowski, P. (2008). A Framework for Sensitivity Analysis in Spatial Multiple Criteria Evaluation. In: Cova, T. J., Miller, H. J., Beard, K., Frank, A. U., Goodchild, M. F., (Eds.). (2008). *Proceedings from the Fifth Intentional Conference on Geographic Information Science*, LNCS 5266, Springer-Verlag Berlin, 217-333.

Ligmann-Zielinska, A., & Jankowski, P. (2014). Spatially-Explicit Integrated Uncertainty and Sensitivity Analysis of Criteria Weights in Multicriteria Land Suitability Evaluation. *Environmental Modelling & Software*, 57, 235-247.

Malczewski, J. (1999). *GIS and Multicriteria Decision Analysis*. New York, Wiley.

Malczewski, J. (2006). GIS-based multicriteria decision analysis: a survey of literature. *International Journal of Geographical Information Science*, 20(7), 703-726. doi: 10.1080/13658810600661508

Malczewski, J., & Rinner, C. (2015). *Multicriteria Decision Analysis in Geographic Information Science: Advances in Geographic Information Science*. New York, Springer.

Penadés-Plà, V., Garcia-Segura, T., Martí, J. V., & Yepes, V. (2016). A Review of Multi-Criteria Decision-Making Methods Applied to the Sustainable Bridge Design, *Sustainability 2016, 8(12)*, 1295-1315. Retrieved from https://www.mdpi.com/2071-1050/8/12/1295/pdf

Rauber, T. & Rünger, G. (2013). *Parallel Programming for Multicore and Cluster Systems* (second edition). Berlin-Heidelberg, Germany: Springer-Verlag.

Rizvi, M.A.K. (2016). *Simulation of Parallel and Distributed Computing: A Review*. IOSR Journal of Computer Engineering, 18(2), 5-11. doi:10.9790/0661-1802030511

Rocklin, Matthew. (2015). Dask: Parallel Computation with Blocked algorithms and Task Scheduling. In *Proceedings of the 14th Python in Science Conference* (SciPy 2015). 126-132. doi: 10.25080/Majora-7b98e3ed-013

Şalap-Ayça, S., & Jankowski, P. (2016). Integrating Local Multi-Criteria Evaluation with Spatially Explicit Uncertainty-Sensitivity Analysis. *Spatial Cognition & Computation*, 16(2), 106-132.

Singh, I. (2013). Review on Parallel and Distributed Computing. *Scholar Journal of Engineering and Technology*, 1(4), 218-225. Retrieved from https://pdfs.semanticscholar.org/b640/0ededf1b500f04ce05efcf2c0c8863a15cdb.pdf

Sobol', I.M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiment*, 1(4), 407-414.

Trobec, R., Slivnik, B., Buli´c, P., & Robič, B (2018). *Introduction to Parallel Computing: From Algorithms to Programming on State-of-the-Art Platforms*. Cham, Switzerland: Springer Nature Switzerland AG.

Tzeng, G. H., & Huang, J. J. (2011). *Multiple Attribute Decision Making Methods and Applications (first edition)*. Boca Raton, USA: Taylor & Francis.

Wainwright, H., M., Finsterle, S., Jung, Y., Zhou, Q., & Birkholzer, J. T. (2014). Making sense of global sensitivity analyses. *Computers & Geosciences*, 65, 84-94.

Yang, C. T., Huang, C. L., & Lin, C. F. (2011). Hybrid CUDA, OpenMP, and MPI parallel programming on multicore GPU clusters. *Computer Physics Communications*, 182, 266-269.

Appendix

The box plots below illustrate the distribution of the measuring sequences for both performance comparison settings (fixed simulation size and variable node size).

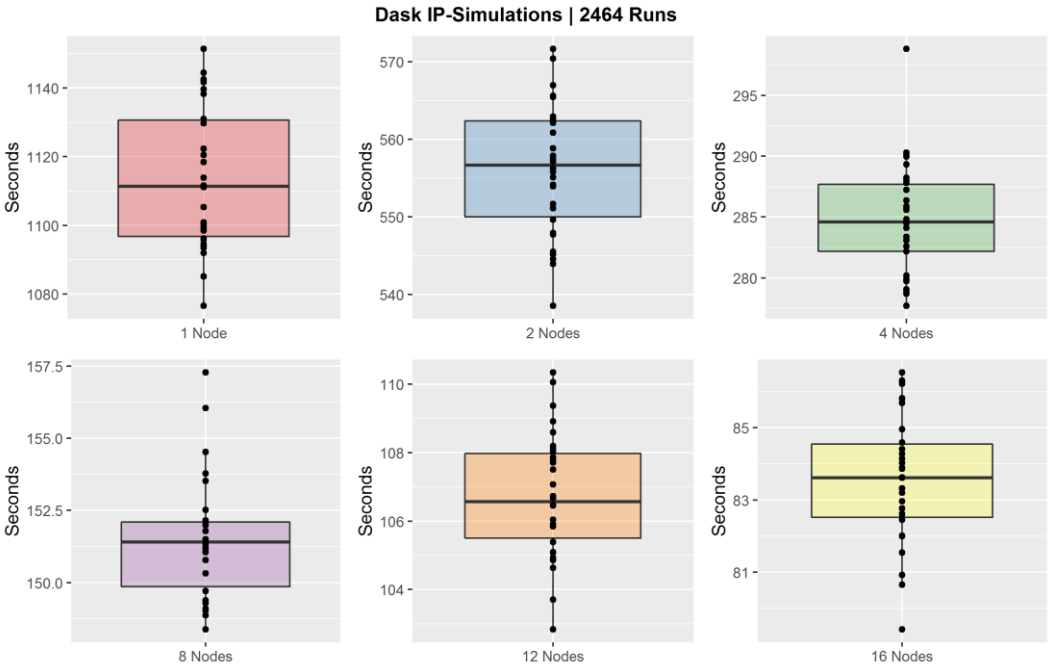

Figure 9: Box plots for 2,464 IP simulation runs and variable node sizes.

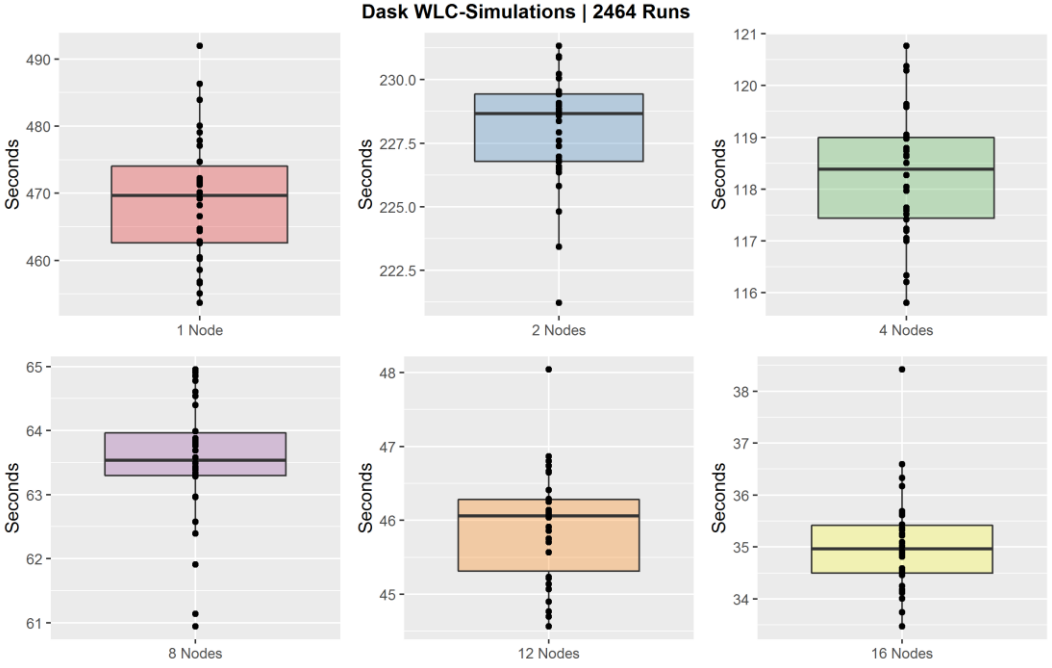

Figure 10: Box plots for 2,464 WLC simulation runs and variable cluster sizes.

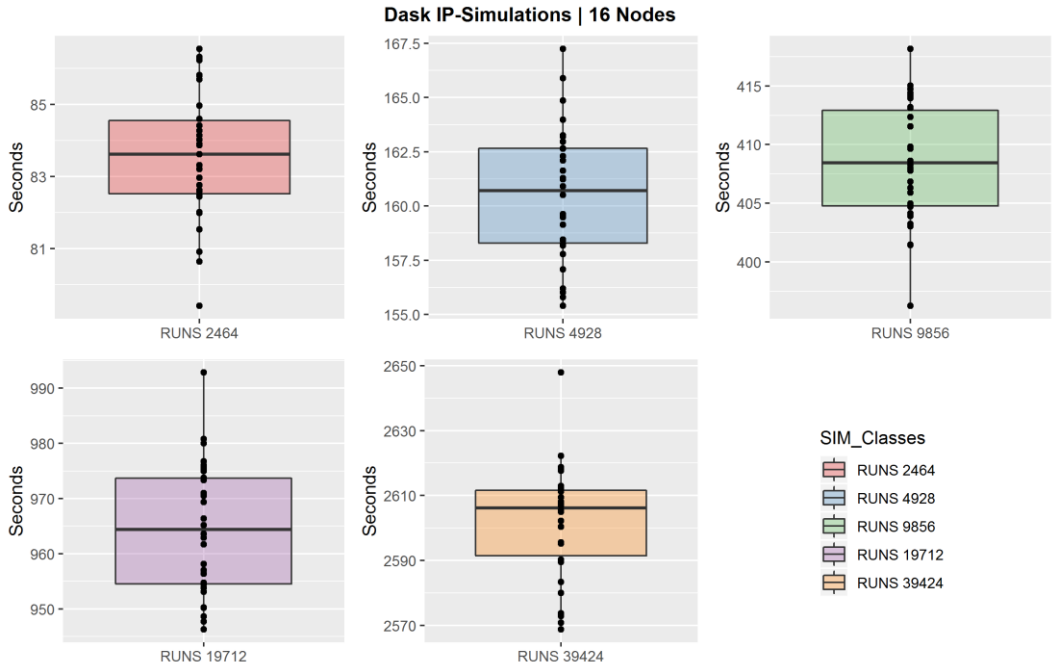

Figure 11: Box plots for a 16-node cluster and a variable IP simulation size.

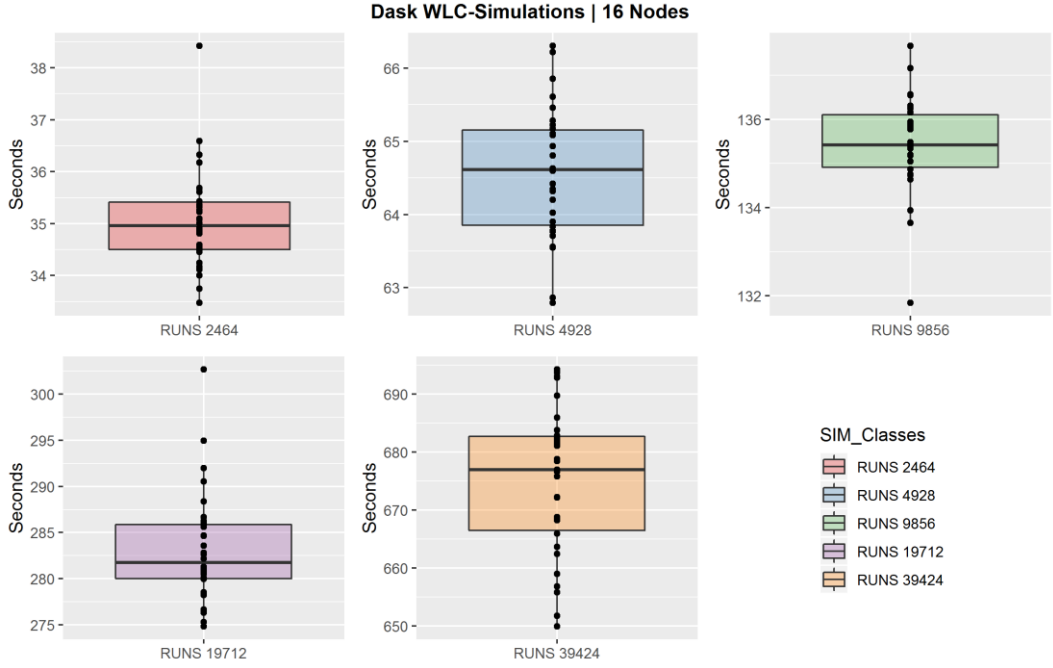

Figure 12: Box plots for a 16-node cluster and variable WLC simulation size.