A HADOOP-BASED APPROACH TO ACCELERATE THE TIME-CONSUMING MONTE CARLO SIMULATIONS IN ORDER TO COMPUTE THE SUITABILITY SURFACES FOR BIG RASTER DATA SETS IN S-MCDA PROBLEMS

by

Vitus Graule

RESEARCH PAPER FOR MARSHALL PLAN FOUNDATION

Carinthia University of Applied Sciences

Geoinformation an Environmental Technologies BSc Program

Supervisors

DI (FH) Christoph Martin Erlacher, MSc.

Prof. Dr. Piotr Jankowski

FH-Prof. Dr.-Ing. Karl-Heinrich Anders

FH-Prof. Dr. Gernot Paulus, MSc. MAS

Villach, October 2018

Abstract

The Spatial-Multicriteria Decision Analysis (S-MCDA) is a complex geospatial computation, which can be computationally intensive and memory consuming. Therefore, solving such a spatial decision analysis can be done on a computer cluster to provide the necessary computational requirements and compute the result in a relatively short time. Furthermore, the computational workload of the algorithm was parallelized and distributed amongst a cluster of comparatively low-cost computers. This approach was compared to a sequential model run on a single relatively powerful machine to get the acceleration of the whole model. Within the scope of this paper, a Hadoop-cluster and a parallel and distributed solution for a cluster system was adapted. The Hadoop-cluster is build up at Carinthia University of Applied Science located in Villach, Austria. The employed S-MCDA model incorporates the decision rule weighted linear combination (WLC) and features a simulation of the resource-intensive Monte Carlo Simulation (MCS). A useful extension for the S-MCDA model could be the variance-based Spatially-Explicit Uncertainty and Sensitivity Analysis (SEUSA) or changing the decision rule to Ideal Point. The algorithm was implemented with Java in the computational model of MapReduce and the used middleware was Hadoop version 2.7.2. This paper gives an overview on the research project, which included a research stay abroad at San Diego State University in California in the USA, describing the applied methodology and algorithm implementation on the Hadoop-cluster, and discusses future work with in this research project.

Table of Contents

1 Introduction

Multicriteria decision analysis (MCDA) is important in everyday life decisions to select the best of all possible alternatives. In Geographical Information Science location plays a huge part. Therefore, spatial multicriteria decision analysis (SMCDA) supports experts in difficult decision situations to select the best potential.

1.1 Motivation

A S-MCDA model assists experts in important decision-making by applying sensitivity and uncertainty analysis (SEUSA) techniques to improve the robustness and increase confidence in the predictions (Ligmann-Zielinska & Jankowski, 2014; Erlacher, et al., 2017). A S-MCDA is a time-consuming model to run, depending on different factors such as the spatial resolution of the study area, the number of criteria considered, the complexity of this model, and the number of model runs necessary. One way to increase the computation time of this model is to use parallel and distributed programming concepts like Hadoop or Dask (Desch, et al., 2018).

Erlacher et al. (2016; 2017) achieved reasonable speed-ups of the most time-consuming part of a S-MCDA model through the use of a Graphic Processing Unit (GPU)-based parallelization approach. The employed S-MCDA model incorporated a SEUSA using variance-based sensitivity analyses, which includes MCS, the latter being the part which was targeted for the speed-up. In 2018, even computer with high-end hardware have limitations with respect to data storage and computational capacity. Therefore, the workflow of the implementation of the same S-MCDA model was split among a collection of different computers. The collection of computers is called computer-cluster. Such a cluster was built on the Hadoop version 2.7.2 framework and it can be considered as a distributed memory system. Each computer has its own memory-processing unit and is connected by a communication network. The S-MCDA model incorporating the weighted linear combination (WLC) method is implemented in the programming language Java.

1.2 Problem Definition and Objective

With a lack of detailed information about the robustness of the model and uncertainty on the model results most of S-MCDA applications cannot provide conscientious support for experts during decision-making. Therefore, Ligmann-Zielinska and Jankowski (2014) created the sensitivity and uncertainty analysis for experts to evaluate the results. With this analysis experts can identify which criterion or alternative is uncertain and sensitive for the model run.

The most time-consuming part of the variance-based uncertainty and sensitivity analysis for this S-MCDA problem is the Monte Carlo Simulation (MCS). As the basis of performing SEUSA and producing a myriad of suitability surfaces the MCS showed be accelerated in this research project. To provide accurate results of the uncertainty and sensitivity high number of simulations are necessary. If one sticks with the conventional, non- parallelized and non-distrusted SEUSA approaches there always will be compromise solution in respect to the problem size and the number of simulations.

During decision-making processes for application domains such as landscape assessment, hazard risk assessment, environmental protection, land use planning, and sustainable regional development a lack of detailed information about the robustness and uncertainty of the model's result can cause a lot of problems. Therefore, S-MCDA models represents an opportunity to support experts in this procedure to select the best alternative. According to Malczewski (1999), the values and the weights of the input criteria refer to the main source of the uncertainty in multicriteria analysis. Hence, Spatially-Explicit Uncertainty and Sensitivity Analysis (SEUSA) is an important step in S-MCDA to increase the quality of the decision process. Wainright et al. (2014) categorized the variety of sensitivity analysis methods roughly into local- and global methods. Global Sensitivity Analysis (GSA) methods constitute dependencies among criteria in comparison to local sensitivity analysis such as on-at-time methods. The disadvantage of GSA methods such as the variance-based sensitivity analysis is the requirement of including many samples (e.g. weight samples). Besides that, the computational demand increases with the number of criteria, the aggregation method and the size of the project area corresponding the

number of locations. The Hadoop version 2.7.2 framework can handle large S-MCDA problems including millions of locations and several hundred thousand of simulations by the MapReduce computational concept for Hadoop (White, 2015, p. 19). A normal single computer could not manage such a problem except the workstation has a powerful GPU, which only be has physical limitations concerning the data storage and the computational capacity. Hence, parallel and distributed computing frameworks for clusters such as Hadoop 2.7.2 spreads the workload among nodes.

Consequently, the main objective of this research project focuses on the Hadoop migration for a specific raster-based S-MCDA problem that bears on Monte Carlo Simulations (MCS) in order to generate the stack of suitability surfaces that illustrates the input for conduction SEUSA. This stack of suitability surfaces is necessary to create average and standard-deviation maps, which represent the result. The result for this project is created once by the Hadoop-cluster and the other time by a sequential algorithm, to compare the performance of a parallel/distributed and sequential model run.

1.3 Methodology

With the three-staged concept – as shown in Figure 1– an on-time completion of this project was secured. In the preparation phase, the specifics of Hadoop 2.7.2 had to be learned. Therefore, it was necessary to perform steps like requirement analysis and a literature research. The implementation was the most time and labor-intensive part where the Hadoop-cluster and a functional parallel and distributed algorithm had to be developed. Eventually it led to the performance tests, where the different run times were compared and analyzed.

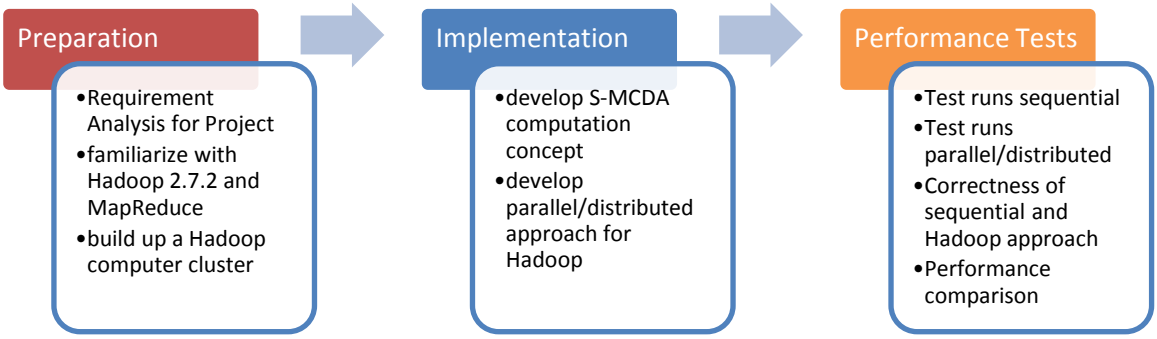

Figure 1: Three-step concept for the workflow of this project

It was necessary to build up a similar Hadoop-cluster like at the San Diego State University (SDSU) before starting to implement the S-MCDA problem in Villach at the Carinthia University for Applied Sciences (CUAS). The Hadoop-cluster at the SDSU, Department of Geography, is running on the operating system (OS) CentOS7 a Linux distribution and consists of 16 machines. These nodes are all equally equipped. The Hadoop-cluster also provides components like MapReduce, Hadoop Distributed File System (HDFS) and Spark, which runs on top of Hadoop. Compare to SDSU's Hadoop-cluster, the cluster in Villach has different equipped workstations and the cluster includes MapReduce and HDFS. The Hadoop-cluster will only include maximal 7 machines. On the master machine, it is necessary to set up different configurations than on worker nodes even, even though all machines have to be connected via a switch to communicate.

The most time-consuming part of the employed S-MCDA model is the MCS. Therefore, the focus in the implementation is on the parallelization and distribution of the MCS on the Hadoop cluster. This should lead to an increase of the performance time of the whole model. The algorithm to perform a S-MCDA problem with the decision rule weighted linear combination (WLC) is divided into the Mapper- and the Reducer-part, which are both necessary for the MapReduce computing concept (Figure 2). Both parts parallelize and distribute the workload along the Hadoop-cluster. The Mapper prepares the input-data for the Reducer by creating key/value pairs. These intermediate key/value pairs are the input for a computation in the Reducer. The creation of the intermediate key/value pairs depends on the splitting of the input-data. For example, the key could be the first token in a long String and the rest of the String is the value.

Therefore, it is necessary to split the key from the value and then create the key/value pair. The Reducer will run the computation with its input and creates the final output. A more detailed description and Figure on the algorithm can be found in Chapter 4.

Figure 2: Overview of the Map/Reduced based WLC computation on a Hadoop-cluster

The final results are average and standard-deviation map, which were computed out of the suitability surface stack. It is necessary to compare each pair of alternatives of the final result of the sequential and parallel/distributed model run. This check of correctness was done within the performance test step. Therefore, subtract the sequential result of the parallel/distributed result. If both are the same the subtraction should be zero for each location.

1.4 Expected results

The expected results of this research project comprise:

- a detailed analysis of the MCS procedure of the sequential SEUSA framework
- the conceptual development
- the implementation of the Hadoop-based parallel and distributed computing approach
- installation and set-up a Hadoop cluster in Villach

The outcome of this research study will contribute to the development of an expandable and adaptable framework for performing spatially-explicit uncertainty and sensitivity analysis and will be applicable for application domains such as landscape assessment, hazard risk assessment, environmental protection, land use planning and sustainable regional development.

1.5 Thesis Structure

This paper is structured as follows. It begins with the Introduction, which includes motivation, problem definition and research questions, methodology, expected results and the thesis structure. The theoretical background gives a first overview about SMCDA, parallel and distributed computing concept and the framework Hadoop 2.7.2. The methodology comprises the case study, concept, data and the tools and software. Furthermore, it includes the implementation with data/sources, implementation on the Hadoop cluster, testing and validation. After the results there is a discussion, summary, conclusion and to complete the paper the last is the future work.

2 Theoretical background

This section presents theoretical background for the research study on Spatial Multicriteria Decision Analysis, parallel and distributed computing concepts, and the Hadoop framework.

2.1 Spatial Multicriteria Decision Analysis

Multicriteria Decision Analysis (MCDA) and Multicriteria Decision Making (MCDM) are terms which are used equally (Malczewski, 1999). MCDM problems require often conflicting and incompatible criteria, which include both concepts attributive and objective. MCDA refers to a group of methods and procedures that help in the selection of the best possible choice while considering multiple decision alternatives. Both Multiattribute Decision Making (MADM) and Multiobject Decision Making (MODM) problems are further categorized and described in Malczewski (1999, p. 81). In this thesis MADM is used in the model. Different approaches to structure MCDA problems have been elaborated (Keeny et al. 1976, Saaty 1980, Chankong et al. 1983, Kleindorfer et al. 1993). Malczewski (1999, p. 82) puts up six components for a MCDA problem:

> *"(1) a goal or a set of goals the decision maker (interest group) attempts to achieve; (2) the decision maker or group of decision makers involved in the decision-making process along with their preferences with respect to evaluation criteria; (3) a set of evaluation criteria (objectives and/or attributes) on the basis of which the decision makers evaluate alternative courses of action; (4) the set of decision alternatives, that is, the decision or action variables; (5) the set of uncontrollable variables or states of nature (decision environment); and (6) the set of outcomes or consequences associated with each alternative-attribute pair (Keeney et al. 1976, Pitz et al. 1984)"*
> (Malczewski, 1999, p. 82).

The decision matrix (Table 1) is the leading element of a MCDA framework, it consists of columns and rows. The columns represent the input criteria and the rows the alternatives (Pitz et al., 1984). Every cell in the decision matrix represents a result

(decision outcome) for the given input criteria and alternatives. Furthermore, it includes weights for each criterion, which are assigned by decision makers.

Table 1: Decision Matrix where the alternatives are representing the rows and the criteria are representing the columns lean on the existing illustration (Malczewski, 1999).

Decision Matrix	Criterion 1	Criterion 2	...	Criterion n
Alternative 1	x_{11}	x_{12}	$x_{1...}$	x_{1n}
Alternative 2	x_{21}	x_{22}	$x_{2...}$	x_{2n}
...	$x_{...1}$	$x_{...2}$	...	$x_{...n}$
Alternative m	x_{m1}	x_{m2}	$x_{m...}$	x_{mn}
Preferences	Weight 1	Weight 2	...	Weight n

This decision matrix is the base of the MCDA framework (Figure 3), but it includes five more components for the whole MCDA framework. As listed above, the framework consists of a general goal, decision makers, objectives, alternatives and outcomes (Malczewski, 1999).

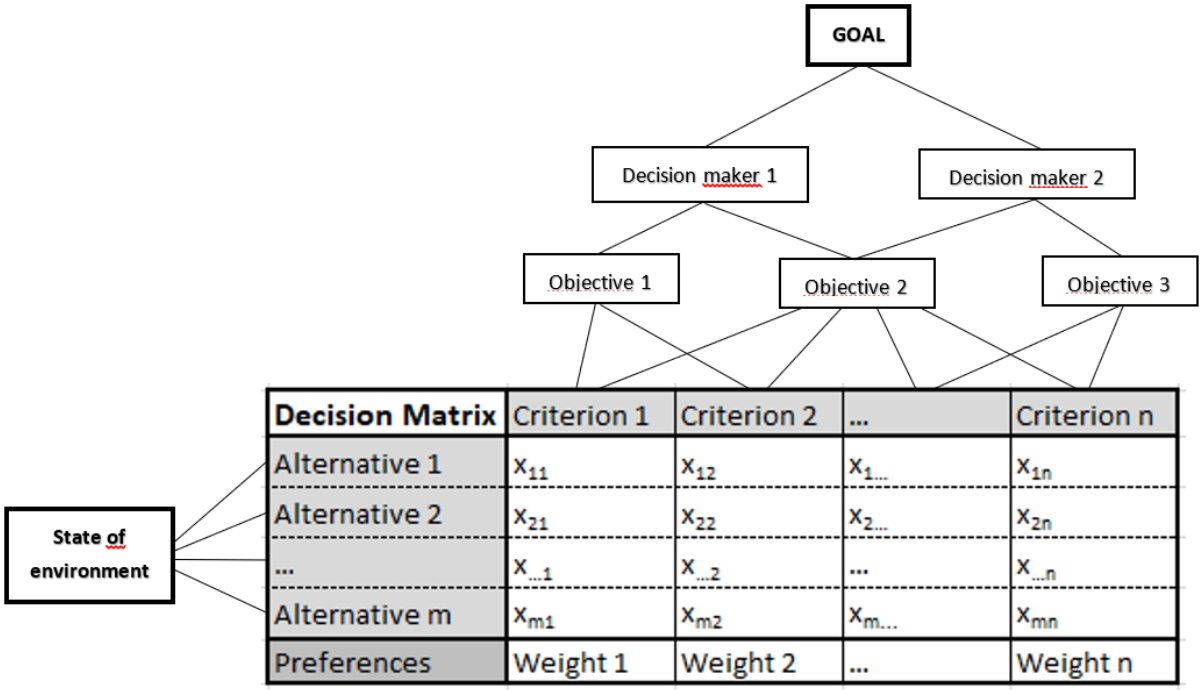

Figure 3: Framework for multicriteria decision analysis lean on an existing illustration (Malczewski, 1999, p. 82)

The goal or set of goals the decision maker attempts to achieve, is the top-level element of the framework for multicriteria decision analysis. For example, in the context

of urban planning, the goal may be to improve the air quality in a district. Decision maker can be a single person or a group of people with the same interest or preference. These persons are generally involved in complex decision problems. Their preferences are directly represented in the weights of each criteria. A criterion is a standard of judgement or a rule to test the desirability of alternative decisions (Hwang & Yoon, 1981; Malczewski, 1999).

In the last couple of decades there have been research and investigations on spatial multicriteria decision analysis techniques (Malczewski, 2006; Malczewski & Rinner, 2015; Malczewski, 1999). SMCDA specifies the application of multicriteria decision analysis with a spatial component as seen in Figure 4. It includes a set of geographically defined alternatives from which a choice of one or more alternatives is made with the respect to a given set of evaluation criteria. The spatial alternatives are a collection of point, line, and areal objects, attached to which are criterion values. Criterions can by of two different kinds: spatially explicit and spatially implicit. Malczewski & Rinner describe spatially explicit criteria as involving spatial characteristics of decision alternatives. According to Malczewski (1999), SMCDA needs both data on criterion values and the geographical locations of alternatives. GIS-based multicriteria decision analysis uses GIS and the traditional MCDM techniques to process the data and obtains information for making the decision. These solutions can be displayed with maps and additional expand by tables and graphs. Therefore, the result gets more comprehensive.

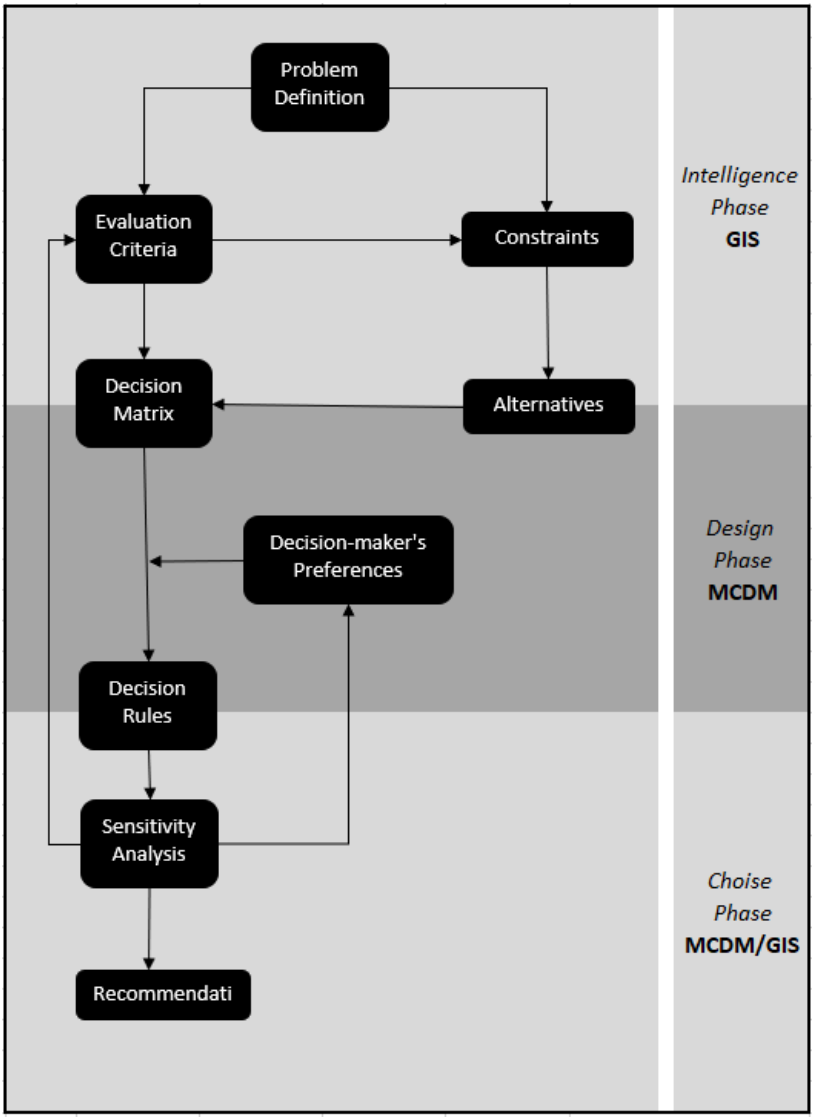

Figure 4: Framework for spatial multicriteria decision analysis lean on an existing illustration (Malczewski, 1999, p. 96)

2.2 Parallel and Distributed Computing Concept

The rapid development and change in the computer technology can be stunning. Till now the computer clock speed grow from megahertz to gigahertz but even more impressive is the increase of memory capacity (Hayes, 2007). Not only the technology achieved a great increase in computing speed and memory capacity also the problem size and complexity increased massive of the time. Therefore, parallel and distributed computing concepts have been developed. These often use a similar model as the divide and conquer concept and break the given job into sub-tasks that can be solved

simultaneous on a single machine with a multicore central processing unit (CPU) (Lescisin & Mahmoud, 2016).

2.2.1 Parallel Computing

Parallelism can be classified in three classes according to Rauber & Rünger (2013, p. 4) every class has a different dividing rate for the program. There is (1) fine-grained, which stands for a program broken down into small tasks, (2) coarse-grain is the opposite and stands for a program broken down into larger tasks and (3) medium-grained is in between these two categories. With a higher grain size, the communication and scheduling plays a higher importants in parallelism.

Rauber & Rünger (2013) differentiate between parallelism at (1) instruction level, (2) data parallelism, (3) loop parallelism and (4) functional or task parallelism. Instruction level parallelism is if certain instructions of a program can run at the same time. Doing the same operations to different elements on a large data structure is called data parallelism, when the elements are distributed evenly among the processing units and processed in parallel. Loop parallelism refers to loop operations in the algorithm and it can be exploited because usually the iterations can be executed in arbitrary order. Task parallelism is possible with sequential programs that contain independent parts.

Not only creating a parallel algorithm is important to run the program in parallel, one also has to take into considerations several software aspects. The specific Operation System (OS), the programming language and the compiler, as well as the run time libraries (Rauber & Rünger, 2013, p. 93). The OS controls the hardware and software resources on the computer and determines which program can run at what time and with which memory capacity. It also manages the allocation of the memory and the access to external devices. For every program initialized by the user, the OS creates a process which is an instance of the program that is appointed a specific block of memory and receives various information like descriptors of available resources the program can access. A parallel program can utilize multi-processing or multi-threading. With multi-processing, the application launches several processes for executing specific tasks in parallel, where each process runs independent because of separated resources. To avoid large (time consuming) necessary information exchange, multi-threading can be used instead.

Threading can help to divide a program into more or less independent subtasks. Threads are lightweight processes that are contained within a process, therefore sharing the same resources. Begin and end of a thread is defined and there can be a hierarchical relation. Within a process, multiple threads can run concurrently to improve the performance. This threads are provided by the runtime system or by the OS. OS threads have the advantage that the OS is aware of the existence of these threads and can ensure an efficient use of the cores of a multicore system (Rauber & Rünger, 2013).

Threads or processes need information exchange, which is why there has to be some form of organization of the address space. Threads have the benefit, that they use the same memory because they are part of a process. Handling the communication can be done synchronous or asynchronous. Asynchronous communication provides a higher parallelism because the sender process can continue its work while the message is buffered by the system and delivered when the receiver is ready to accept the message. This can result in buffer overflow, if a large number of messages are send in a short period (Rauber & Rünger, 2013).

2.2.2 Distributed Computing

A collection of computers communication over a network are often called cluster. Generally, a cluster refers to a collection of computers or processors cooperating to solve a problem. As these computers are physically separated, they not have an inherent shared memory but some sort of data in a common repository like a shared file system. The programmer has to be aware of the possibility of different hardware configurations and or operating systems. Therefore, some sort of software could help managing the heterogeneity in a distributed system. It is called middleware and adds an extra layer to the architecture between the distributed application and the individual OS of each node (Lescisin & Mahmoud, 2016). The main advantages of distributed systems are scalability and incremental expandability.

MapReduce is a programming paradigm to use Hadoop as a middleware which is recognized as a large data volume processing framework (Kang, et al., 2015). A detailed explanation about Hadoop and MapReduce follows in the chapter 2.3 in this paper.

Distributed computing concepts run distributed applications that are simply distributed across multiple operating systems processes. All running on the same physical uniprocessor, to integrated applications that run on multiple computers connected by a network. The concept and distributed applications are in use since the late 1970s and began to spread widely in the 1980s with the commercial networking technology. Furthermore, the accessibility of cheap computing power was in favor of the distributed computing concept spread (Taylor, et al., 2012, p. 414). According to Kshemkalyani & Shinghal (2008, p. 2) a distributed execution (program, application) is a computation of processes across a distributed system to collaboratively achieve a result. Distributed systems are a collection of entities working together on solving a problem. In a typical distributed system each computer has a memory- and processing-unit and these parts are connected by a communication network as shown in Figure 5 (Kshemkalyani & Singhal, 2008). Starting a computation (distributed application) on the master of a distributed system, it will broadcast the workload balanced to each accessible working node. Most of the nowadays used distributed computing concepts provide scheduler and compiler to help the developer/programmer.

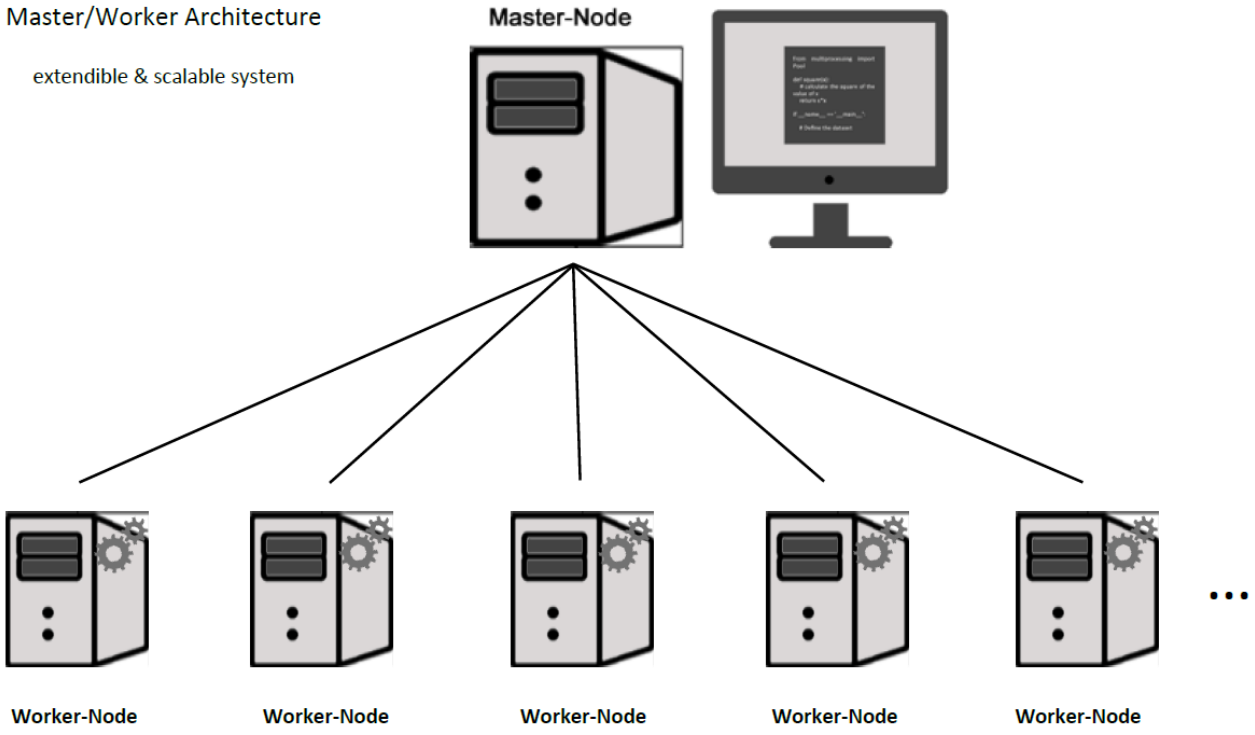

Figure 5: Illustration of a distributed system (computer cluster) and the master/worker architecture (Desch, et al., 2018)

In this project both parallel and distributed computing concepts will be important to achieve a significant performance increase on the model run time. Large SMCDA simulations require great computational capability, which normally cannot be satisfied by a standard single machine. A possible way to solve this problem is to use the parallel and distributed computing concept implemented in varies open source frameworks. The workload is split up on a computer-cluster and each node in this cluster can parallelize the given job into tasks on the central process unit. Hadoop is one example which implements the parallel and distributed computing concept with MapReduce.

2.3 Hadoop

Hadoop is an open-source software for reliable, scalable, distributed computing developed by the Apache™ Hadoop® project. Doug Cutting and Mike Cafarella created a previous version of what is now Hadoop Distributed File System and MapReduce in the year 2004. A brief history about the Hadoop development can be found in White (2015, pp. 9-12).

The Hadoop project includes different basic modules (Apache Hadoop), this modeles and additional models are pictured in Figure 6:

- ***Hadoop Common:*** *The common utilities that support the other Hadoop modules.*
- ***Hadoop Distributed File System (HDFS™):*** *A distributed file system that provides high-throughput access to application data.*
- ***Hadoop YARN:*** *A framework for job scheduling and cluster resource management.*
- ***Hadoop MapReduce:*** *A YARN-based system for parallel processing of large data sets.*

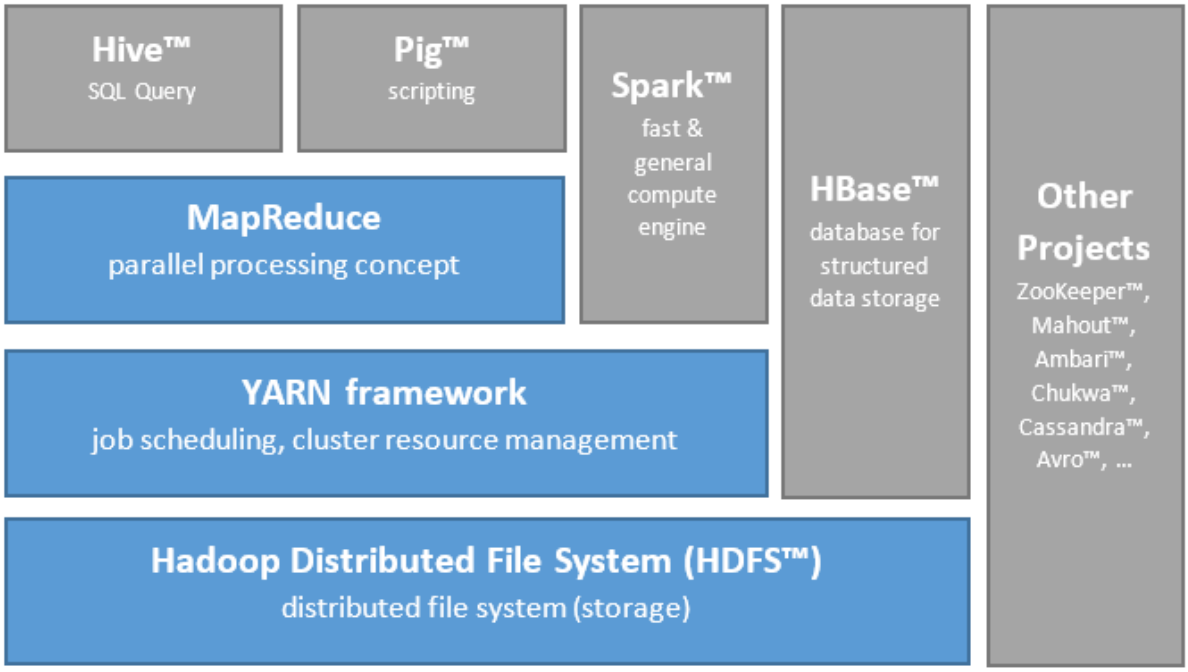

Figure 6: Illustration of the Hadoop architecture with main models (blue) and additional models/projects within the Apache group (grey). Information about the modules is from the official Apache Hadoop website (https://hadoop.apache.org/; access 2018/11/12)

A distributed file system (like HDFS) on a number of separated machines is necessary when datasets outgrow the capacity of a single physical machine. Distributed file systems are more complex than regular disk file systems due to network complications, for example, handling node failure without data loss. HDFS solves this problem with saving the same data block on different nodes. Hadoop provides HDFS as their distributed file system and it is built to run on commodity hardware. It is especially designed for storing very large files with streaming data access. Often the data is up to hundreds of gigabytes, terabytes or petabytes. HDFS is built around the idea of streaming data access that efficient data processing pattern is a write-once-read-many-time pattern (Figure 7). HDFS uses the same block concept for saving the data in the distributed file system as on a single disk. The only difference is the size of the blocks, in HDFS the default block size is 64MB compared to only 512 bytes on a disk. The larger block size has to do with the performance of seek and computing comparison (White, 2015, p. 47). Each file in HDFS is broken into block-sized chunks, which are stored as independent units. These blocks in HDFS have several benefits. First one is that large datasets can be saved on a HDFS even if one single disk in the network has not enough capacity. In fact, it is another benefit to split the file and save it among different nodes to prevent losing data by losing one node (Figure 7).

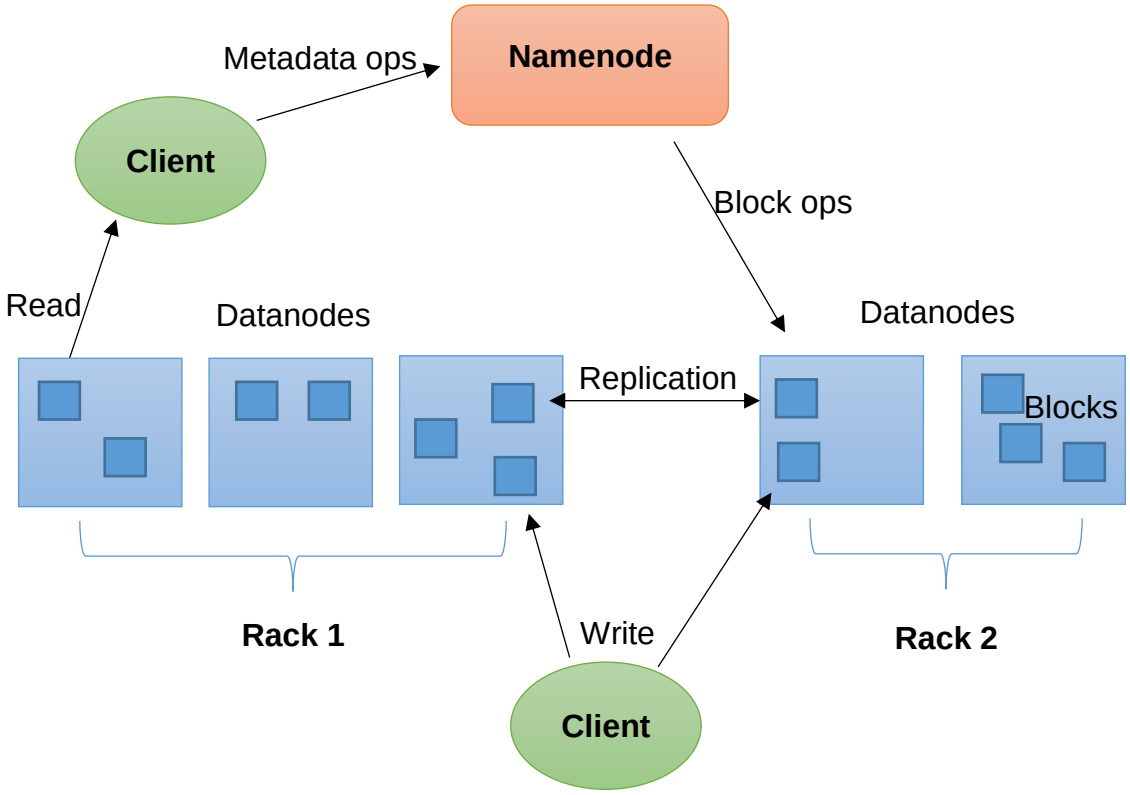

Figure 7: Illustration of the Hadoop Distributed File System (HDFS) (http://wikis.gm.fh-koeln.de/wiki_db/index.php?n=Datenbanken.HDFS, access 2018/10/31)

According to White (2015, p. 46) it is also worth to examine applications for which using HDFS does not work so well. For example, applications with a low throughput of data because HDFS is optimized for delivering exactly the opposite.

MapReduce is a simple programming model for data processing. Hadoop can run MapReduce programs written in various languages like C++, Java, and Python. MapReduce perfectly fits for processing large datasets (large volumes of data) in parallel by dividing the Job into a set of independent Tasks. A full MapReduce program consists at least of a Mapper-class with a map function and a Reducer-class with a reduce function (Figure 8). A Job executes either a Mapper or Reducer across a set of data, whereas a Task only executes a part of the data. The Hadoop MapReduce concept needs a single master node with a Job-tracker to control the Job requests form clients and computation nodes (slaves, worker). Each worker node has a TaskTracker instance to monitor the running Tasks. The JobTracker divides all the workload to the slave nodes and on those the TaskTracker schedules the given job in tasks, which can

be processed parallel. The JobTracker and TaskTracker in MapReduce represent the parallel and distributed computing concept. Furthermore, the JobTracker collects the status and diagnostic information about the Jobs to the client.

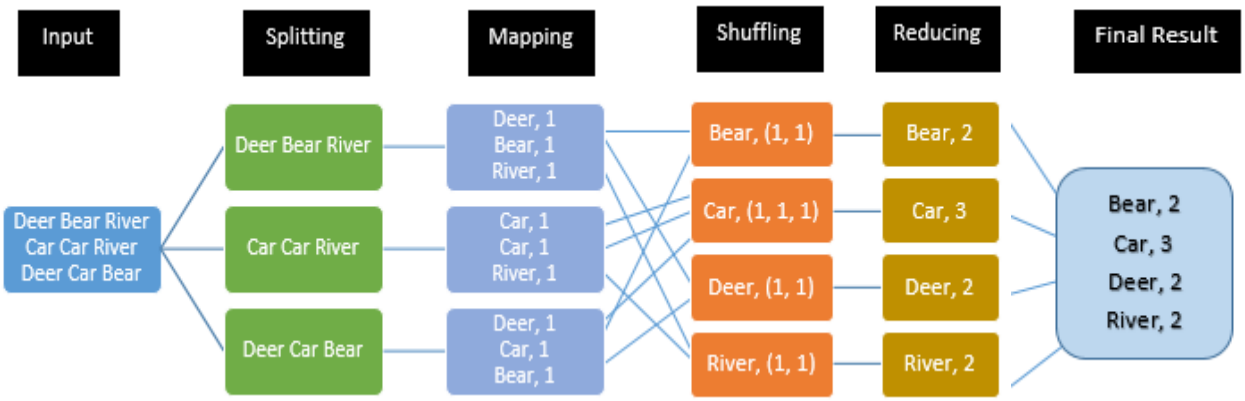

Figure 8: Illustration of Map/Reduce Concept with an example lean on an existing illustration (https://www.dezyre.com/hadoop-tutorial/hadoop-mapreduce-tutorial-, access 2018/10/31)

YARN stands for “Yet Another Resource Negotiator” and was developed in 2010 by a group at Yahoo!, it is sometimes called MapReduce 2 (White, 2015, p. 79). YARN splits up the responsibilities of the JobTracker into separate entities. The JobTracker takes care of job scheduling and task progress monitoring. Job scheduling means to match the different tasks with the TaskTracker and task progress monitoring means controlling tasks and restart failed or slow tasks. By separating the roles mentioned above, it is necessary to create daemons like the resource manager to manage the use of resources across the cluster. An application master manages the application running on the cluster. YARN follows the idea that both daemons (JobTracker, TaskTracker) communicate with each other and that the resource manager provides cluster resources for the application master with a number of containers. The node manager ensures that the application does not use more resources than it can provide.

Finally, White (2015, p. 6) summarized it like “In a nutshell, this is what Hadoop provides: a reliable, scalable platform for storage and analysis. What’s more, because it runs on commodity hardware and is open source, Hadoop is affordable.” and used a similar description as on the official website of apache Hadoop.

3 Methodology

This chapter describes the given tools/software and the workflow for the project. Further details on the implementation are given in chapter 4.

3.1 Concept

The S-MCDA problem model can be displayed like in Figure 9. Analyzing the most time-consuming part, the Monte Carlo Simulation, was the first step in this project. Furthermore, the sensitivity and uncertainty analysis (SEUSA) can be applied and analyzed. Ligmann-Zielinska and Jankowski (2014) proposed the original SEUSA framework, where the model builds a stack of suitability maps with the help of the MCS and the outputs are an average suitability surface, a standard deviation uncertainty surface, and a number of sensitivity surfaces. S-MCDA problems often go along with a huge computational demand therefore, it is useful to use the computing concepts parallelization and distributing. With this concept even, an increase of simulations or an increase of the project can be handled. Furthermore, the parallelizing and distributing of the implementation of a S-MCDA problem should increase the performance. The workload for such a problem is perfectly handled in a parallelized and distributed implementation and it differs this research project from the sequential implementation of the SEUSA framework on which Ligmann-Zielinska & Jankowski (2014) and Salap-Ayca & Jankowski (2016) were concentrated. Erlacher et al. (2016; 2017) already achieved an increase in the performance with his GPU-based concept compared to the sequential salutation. The GPU-based concept increased the most time-consuming part, the MCS computations, 150 times for a landscape assessment- and a land use planning application example. This concept does have limits in storage capacity as well as a limitation because of the read-compute-write functions, which have physical limits in a computer.

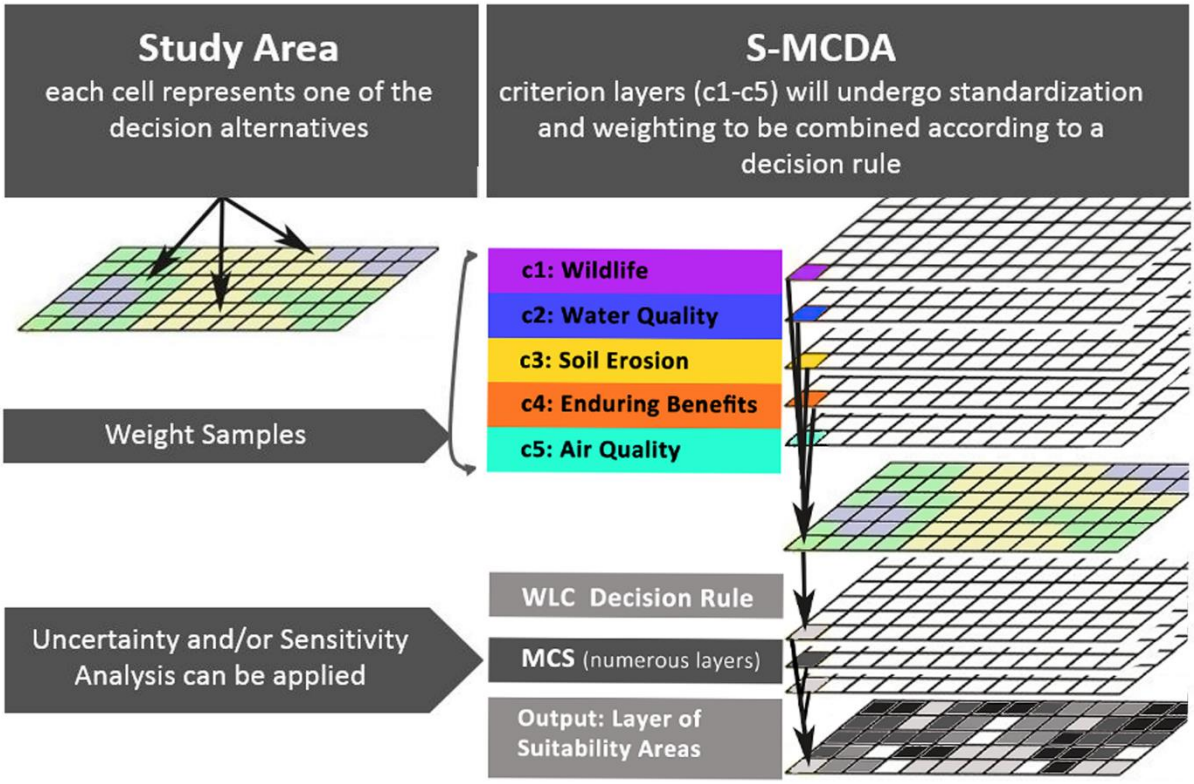

Figure 9: Illustration of the workflow of the simplified employed S-MCDA problem model

With the knowledge of a huge potential in parallelizing and probably in distributing as well, the second sub-task is to focus on the conceptual development of a parallel and distributed algorithm for a Hadoop-based cluster. Hadoop requires the algorithm with the computing concept MapReduce and provides lots of useful help in the background to parallelize and distributed the workload of the S-MCDA problem. Even so it is necessary to explore the capabilities, requirements and limitations of Hadoop 2.7.2 for the migration of the sequential MCS algorithm. The open source framework Apache Hadoop 2.7.2 includes the programming concept MapReduce, which is necessary for parallelizing and distribution, and the Hadoop Distributed File System. The HDFS represents a fault tolerant distributed file system to safe and provide data on the cluster, which is designed to run on a low-cost hardware. MapReduce indicates a programming model for large-scale data processing. It includes two parts, the first one is the Mapper-class with a mapper-function, which pre-processes the input data to key/value pairs and the Reducer-class with a reduce-function, which combines the intermediate key/value pairs to a result value (Polato, et al., 2014; Yao, et al., 2017).

With the implementation for the land-allocation S-MCDA problem of this research project it is possible to measure the computational performance (time) and compare it to other solution options. In advance it is necessary to prioritize agricultural land units within the S-MCDA problem based on environmental benefits for the Environmental Benefit Index (EBI) to perform the measurement. “Wildlife”, “Water-Quality”, “Soil-Erosion”, “Enduring-Benefits” and “Air-Quality” are the environmental factors and as the decision rule for this S-MCDA problem was used WLC. Those environmental factors will be the input data for the model. A detailed description about the methodology of the WLC aggregation method can be found in Malczewski (1999, pp. 199-204). This case study, which is also used by Erlacher et al. (2017), is located at Southwest Michigan (Şalap-Ayça & Jankowski, 2016) and each input criterion (ASCII file) consists of more than 12 million pixels for the raster-based approach. With various sample sizes the performance test performs different number of model runs for the MCS and the potential of the parallel and distributed approach should be pointed out. Furthermore, one can see how the cluster/this approach handles the increase of the workload and how this effects the performance of the implementation. In this research the sample sizes and the number of model runs are defined as follow:

1. Sample Size: 352; Simulation Runs: 2,464

Get the number of simulations with the formula R = (k+2)*N, where k indicates the number of criteria and N represents the sample size of the criterion weights. A more detailed description of the given S-MCDA problem implying the SEUSA workflow can be found in detail at Erlacher et al. (2017; 2016). With linear scale transformation methods raw data can be standardized to criterion scores. Malczewski (1999, p. 116f) referenced Voogd 1983 and Massam 1988 as a good source to look a number of linear transformations up. The most common used ones are *maximum score* and the *score range procedures.* All data, which are benefit criteria, in this study were standardized with the *score range procedure* method, but a detailed explanation about linear scale transformation gives Malczewski (1999, pp. 116-119). With all criteria being benefit criteria only equation 2 was necessary in this case. For cost criteria equation 1 will be used to standardize. This information is important for the standardization process in order to enable comparability of the criteria. For each location of criterion c, the index

i represents the row and the index j represents the column. Therefore, x'_{ijc} indicates the standardized criterion value for each alternative, where x_c^{max} and x_c^{min} are the minimum and maximum values for the corresponding criterion.

$$x'_{ijc} = \frac{x_{ijc} - x_c^{min}}{x_c^{max} - x_c^{min}} \quad (1)$$

$$x'_{ijc} = \frac{x_c^{max} - x_{ijc}}{x_c^{max} - x_c^{min}} \quad (2)$$

In a SMCDA it is necessary to select one of the decision rules, for example Weighted Linear Combination (WLC) or in this case Ideal Point (IP). The WLC method is also named Simple Additive Weighting (SAW) in the literature. With two components the criterion weight (w_j) and the value score for the criterion attribute value (x_{ij}), which are evaluated for each alternative (A_i) by the following equation, the WLC calculates the weighted Sum:

$$A_i = \sum_j w_j x_{ij} \quad (3)$$

The IP method calculates the performance of a set of alternatives based on their separation from the ideal point (Malczewski, 1999).

$$rc_{ij+} = \frac{s_{ij-}}{s_{ij+} + s_{ij-}} \quad (4)$$

The separation is measured in this model with the Euclidean distance. Therefore, the variable p is set to two. This measured distance represents the separation between the uniform criterion of one specific location and the ideal value of the criterion on this pixel. With the following two equations the separation from the positive (5) and the negative ideal point (6) can be measured.

$$s_{ij+} = \left[\sum_c w_c^p \left(v_{ijc} - v_{+c}\right)^p\right]^{1/p} \quad (5)$$

$$s_{ij-} = \left[\sum_c w_c^p \left(v_{ijc} - v_{-c}\right)^p\right]^{1/p} \quad (6)$$

The result of the simulation runs will be a stack of suitability surfaces which is the input for the uncertainty analysis as seen in Figure 10. The stack of suitability surfaces depends on the number of simulation runs, which can be obtained by the formula R = (k + 2) * N, where k indicates the number of criteria and N represents the sample size of the weights. In the course of the uncertainty analysis, an average suitability map and a standard deviation map are computed in order to quantify the variability of each alternative. Locations with a high average suitability value and a high standard deviation value represent choice candidates, which should be further investigated in order to identify criterion weights that influence high standard deviation. In this model the uncertainty are the weight samples see Figure 10.

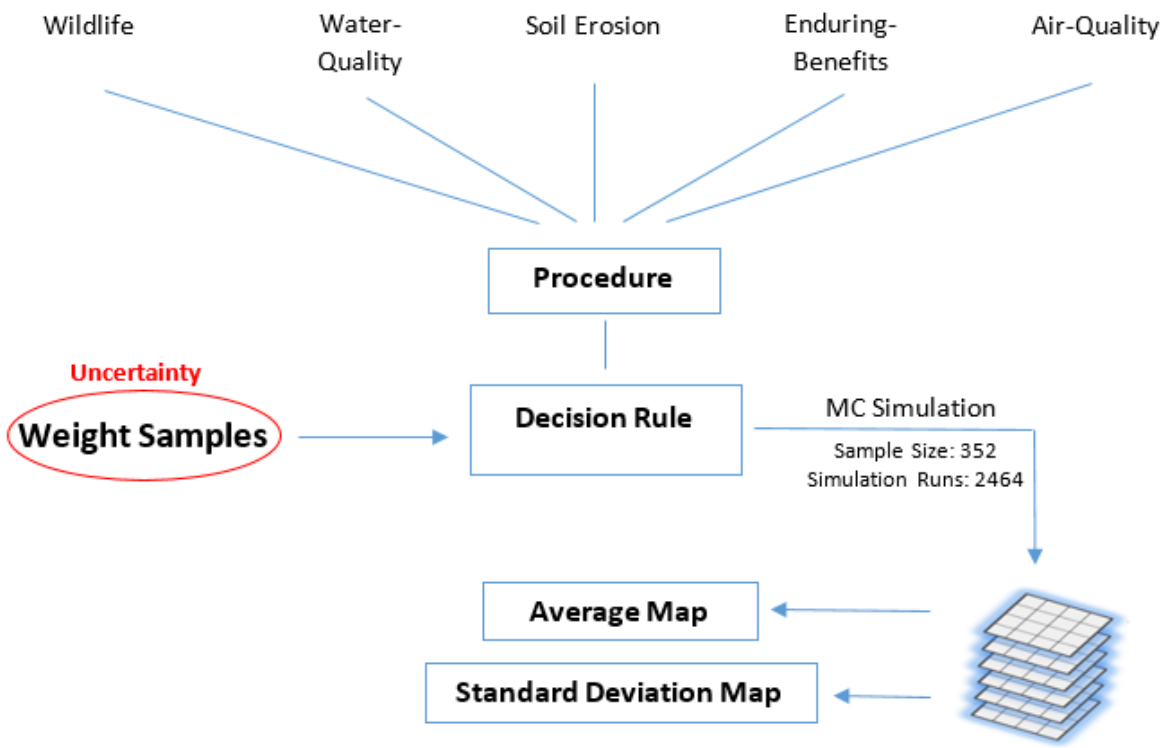

Figure 10: Illustration of the SMCDA workflow model

3.2 Tools/Software

The cluster was built up in a laboratory of the Department of Geoinformation and Environmental Technologies at the Carinthia University for Applied Sciences (CUAS) in Villach, Austria. The cluster leans on the Hadoop-cluster at the San Diego State

University (SDSU), Department of Geography. It runs on the open source operating system CentOS 7 (Linux-Distribution) and consists of one master- and fifteen slave nodes. All those nodes are equipped equally with E5520 2.26 GHz processors and have a total storage capacity of 4.38 TB. The middleware Hadoop on this cluster provides following components MapReduce, HDFS and Spark.

The built cluster in Villach includes up to seven computers with different hardware, one computer is the master and the rest are working nodes as seen figuratively in Figure 5. The whole cluster is based on the open source OS CentOS7 (64 bit), which is a Linux/Unix version. The middleware to run a parallel/distributed program on the cluster is Hadoop 2.7.2, which requires Java (Version 1.8). More detail on the workstation can be found in chapter 4.1.

4 Implementation

This section outlines the implementation approach by describing the data/sources, implementation on the Hadoop-cluster at CUAS and the testing/validation. Note, that the results of the S-MCDA problem are demonstrated in chapter 5.

4.1 Data/Sources

The given resources for this project include python code, ASCII input-files, which represent the case study, regarding the previous work of Erlacher et al. (2016; 2017) and Şalap-Ayça & Jankowski (2016).

The employed S-MCDA model addresses a land-allocation problem for a case study located in the USA in Southwest Michigan (Şalap-Ayça & Jankowski, 2016). It prioritizes agricultural land units based on environmental benefits and the measurement of the performance of those land units, is given as the Environmental Benefit Index (EBI). This index has been formulated by the United States Department of Agriculture (USDA) and contains environmental factors rated with a certain point score. Within this project the environmental factors “Wildlife”, “Water-Quality”, “Soil-Erosion”, “Enduring-Benefits”, and “Air-Quality” include as input data for the model.

These input criteria maps provide as five two-dimensional ASCII files, which is the American Standard Code for Information Interchange (ASCII) and it is a standard way for character encoding using numeric codes. This files are already standardized and each pixel represents a location, which can be identified by row and column. No-data locations are marked with the value of '-9999'.

The actual slimmed down S-MCDA model employed in this project incorporates the decision rule WLC and features a simulation of the resource-intensive MCS technique used for SEUSA see Figure 10. The used simulation run size is 2464 model runs.

For building up the Hadoop-cluster CUAS provided seven workstations, most had different RAM boards and CPUs inside. Furthermore, CUAS suppled a switch to connect all computers within the cluster. The master-node has two DDR4 RAM boards with four gigabytes and an Intel® Core™ i5-6500 with 3.60 gigahertz. The worker-node differed in RAM from DDR2 to DDR4 and a capacity of two to four gigabytes. The CPUs in the workstations were mostly Intel® Core™ i5 generation but also an Intel® Pentium® D and an Intel® Core™2 Dou. The cluster was built with the operation system (OS) CentOs7 an open source Linux version, which is the same OS as on the Hadoop-cluster at SDSU. The software Apache Hadoop Version 2.7.2 was used to set up the Hadoop-cluster and Hadoop is used as the middleware in this project. Further required software on each node is Java version 1.8.

4.2 Implementation on Hadoop Cluster

The Hadoop-cluster in Villach was built with the help and instruction of Mr. McKinsey from the Department of Geography at San Diego State University in a number of development steps. At first a single computer Hadoop-cluster was built up on a machine with CentOS 7 (Linux-Distribution) and Java 1.8. Therefore, the single machine had to take care of all the workload and all organisation/management. With the possibility of communication/connection through a switch a 'real' cluster with master and worker node was feasible. The single computer Hadoop-cluster machine was used as the master machine. Therefore, it was necessary to change configurations. Furthermore, worker nodes had to be set up. In the second step of the

building up of the Hadoop-cluster a two node cluster was set up with one master and one worker node. In this phase of the cluster the master is still used as worker node as well. Starting the cluster is possible from the master and all nodes have to be online before to get started as well.

At first coding was done with Eclipse on a separate computer with Windows 10 (64 bit), after setting up a two nodes cluster with one master and one worker-node, coding was done on the master node. The master node has CentOS 7 as operating system and needed a different Eclipse version for Linux.

The basic structure of the developed algorithm for the employed S-MCDA model is depicted in Figure 11.

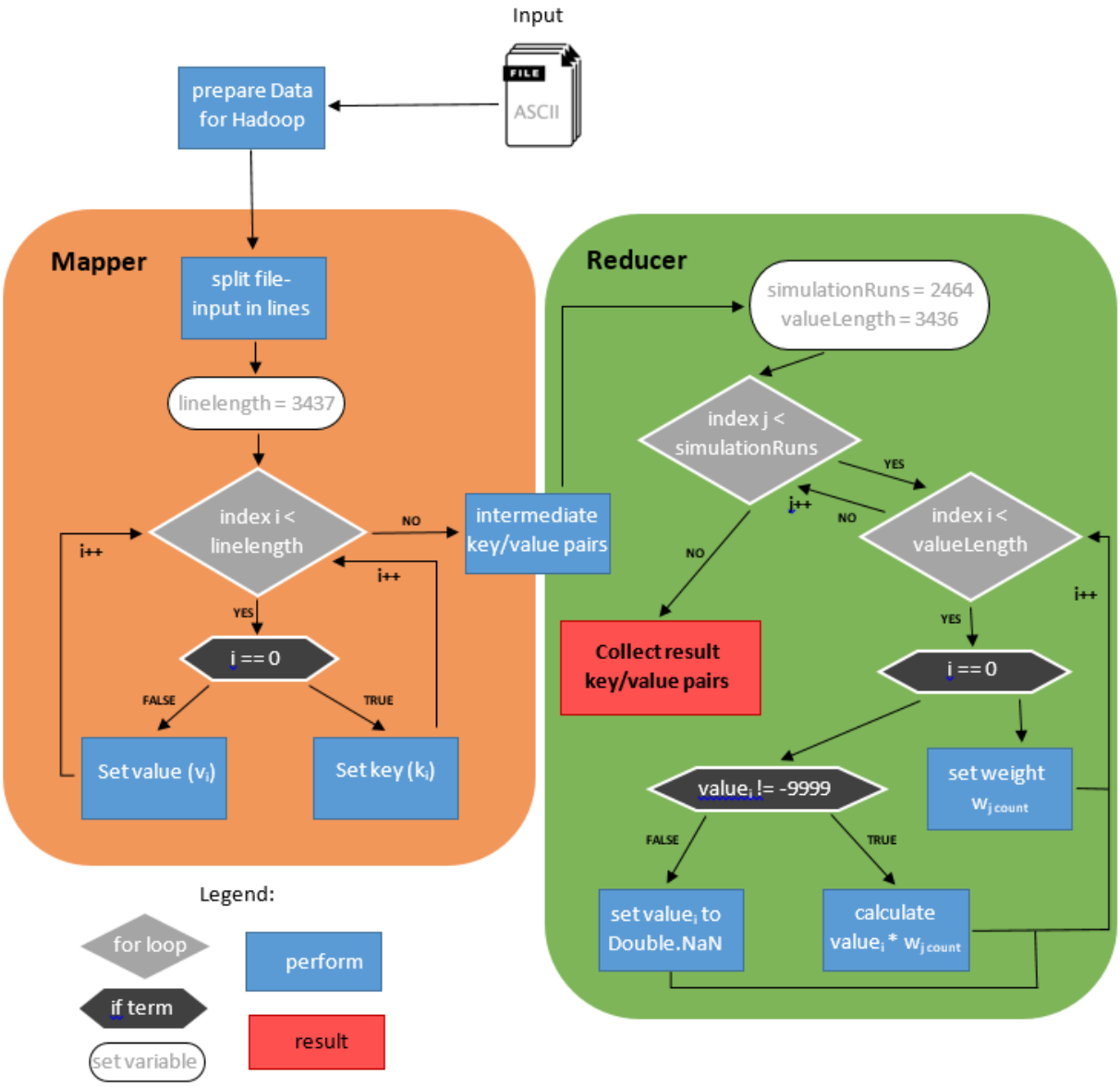

Figure 11: Illustration of the Hadoop-cluster-based (CPU) acceleration approach to perform a simplified S-MCDA model.

After analysing the provided algorithm, it became clear that with a Hadoop-cluster the computational go to concept was MapReduce. Running an algorithm requires the initialization of the Hadoop-cluster and the upload of the ASCII input files on the HDFS. The input ASCII files had to be prepared before uploading those on HDFS. For the used algorithm it was necessary to insert two extra columns. The first one is the row index and the second extra column is an index for the weight. Starting the algorithm on the Hadoop-cluster is possible by creating a runnable jar file of the java-classes. Afterwards, the first task with in the algorithm 'split files-input in lines' in the mapper is submitted. It creates the intermediate key/value pairs for the reducer, while going

through each line and indicate the key. After the creation of the intermediate key/value pairs the reducer starts to run the decision rule WLC on each alternative. Therefore, it is necessary to create weights and get a stack of suitability maps out of the computation. With this stack the final key/value pairs can be calculated by taking the average of each location over the whole stack of suitability maps.

4.3 Testing and Validation

For the testing and validation two excel documents were created to document the different parameters, which are important for each run and to decide if one can compare two runs with each other. The first document includes information about the cluster and each node. The second document lists information about single model runs on the Hadoop-cluster.

Table 2: Illustration of the specification of each node in the cluster

Cluster									
computername	nodename	IP-address	OS	Java version	Hadoop version	RAM	RAM model	CPU cores	CPU model
						possible / provided		physical / virtual in cluster	
VPC128	master	10.1.1.201	CentOS7	1.8.0_171	2.7.2	8 / 4 GB	2 * 4 GiB DIMM DDR4 Synchronous Unbuffered	4 / 8	Intel® Core™ i5-6500 CPU @ 3.20GHz
VPC837	slave1	10.1.1.200	CentOS7	1.8.0_171	2.7.2	8 / 4 GB	2 * 4 GiB DIMM DDR4 Synchronous 1333 MHz	4 / 8	Intel® Core™ i5-2400 CPU @ 3.10GHz
VX12001	slave2	10.1.1.202	CentOS7	1.8.0_171	2.7.2	2 / 0,5 GB	2 * 1 GiB DIMM DDR2 Synchronous 533 MHz	2 / 8	Intel® Pentium® D CPU 3.20GHz
VPC632	slave3	10.1.1.203	CentOS7	1.8.0_171	2.7.2	8 / 4 GB	1 * 8 GiB DIMM DDR3 Synchronous 1600 MHz	4 / 8	Intel® Core™ i5-4590 CPU @ 3.30GHz
VPC118	slave4	10.1.1.204	CentOS7	1.8.0_171	2.7.2	8 / 4 GB	1 * 8 GiB DIMM DDR3 Synchronous 1600 MHz	4 / 8	Intel® Core™ i5-4590 CPU @ 3.30GHz
VPC179	slave5	10.1.1.205	CentOS7	1.8.0_171	2.7.2	8 / 4 GB	2 * 4 GiB DIMM DDR3 Synchronous 1600 MHz	4 / 8	Intel® Core™ i5-3570 CPU @ 3.40GHz
VPC426	slave6	10.1.1.206	CentOS7	1.8.0_171	2.7.2	4 / 0,5 GB	2 * 2 GiB DIMM DDR2 Synchronous 667 MHz	2 / 8	Intel® Core™2 Dou CPU E6750 @ 2.66GHz

The first excel about the cluster includes hardware information as the ‘computername’, ‘nodename’, IP-address, OS, installed Java version, installed Hadoop version, equipped and provided RAM, RAM model, physical and virtual CPU cores and the CPU model (Table 2). For example the total RAM capacity of the master is 8GiB but only 4GiB are provided for the cluster. This information about the cluster and each node is important for the second document, because the included machine in the cluster are a factor in different model runs.

Table 3: Illustration of the simulation run document of the Hadoop-cluster

Simulation Runs:								
date	start time	end time	run time	decision rule	simulations	cluster size	included machines	provided RAM/cores
								virtual cores
2018-11-06	15:07	19:52	4h 41min	WLC	2464	7 nodes	master, slave 1-6	21GiB / 56
2018-11-07	10:26	15:09	4h 43min	WLC	2464	5 nodes	master; slave 1,3,4,5	20GiB / 40
2018-11-07	15:49	20:24	4h 35min	WLC	2464	5 nodes	master; slave 1,3,4,5	28GiB / 40

The second excel holds the information about the model runs on the Hadoop-cluster. This includes information like date, start time, end time, run time, decision rule, simulations, cluster size, included machines and provided RAM/cores. For example the first documented run on the cluster with 2464 simulations took 4 hours and 41 minutes.

Two model runs are only comparable if the prerequisites are the same. In the case of performance comparison the model runs need to run on the cluster with the same requisites, which can be identified with the parameters decision rule, simulations, cluster size, included machines and provided RAM/cores. If all this parameters are the same, on can compare the run time of the two model runs.

The approach to compare the final results with the one simulation run, which means the sequential and manually created result with ArcMap from Esri and the other result with the Hadoop-cluster and the algorithm. Subtracting those results in ArcMap with the raster calculator, the optimal case result is a raster with zero at every location. Because of computational issues like rounding mistakes and displaying values in different data sets, the subtracted result raster has really small values, which can be considered as zero, see Figure 12. Therefore, the subtraction result is considered as a successful result and proofs the concept.

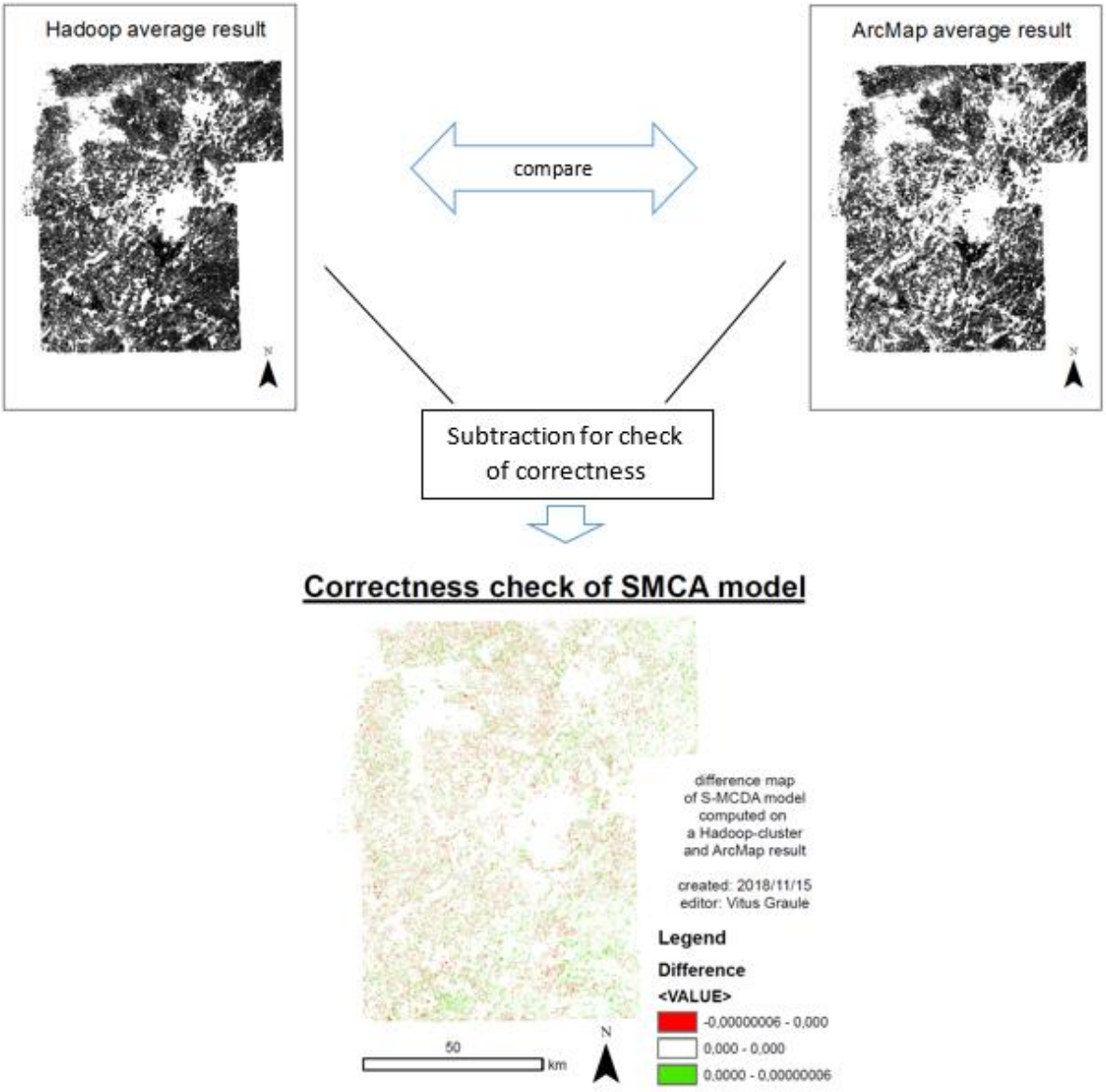

Figure 12: Correctness check of the AVG-result. Subtraction of Hadoop average result and ArcMap average result

5 Results

The aim of this research was building up a Hadoop-cluster and the development of a parallel and distributed algorithm for the employed S-MCDA model. The illustration of the developed algorithm is depicted in Figure 11.

The Hadoop-cluster in Villach at CUAS includes seven machines, one is the master- and also a worker-node in one machine and the other six workstations are worker-nodes. The cluster is based on the Hadoop version 2.7.2 and it is extendable as well as each node can get update random access memory (RAM).

The implemented parallel and distributed algorithm resulted in a decrease of the overall computation time of the S-MCDA model compare to the manual run on ArcMap. The result of the employed S-MCDA model is an average map with the decision rule WLC (Figure 13). Within the scope of this project small numbers of simulation runs were made to compare the results of the Hadoop-cluster and manually created results with the software ArcMap from Esri. The comparison is also made in ArcMap. Therefore, it was necessary to use the same weights in ArcMap as which were used in the computation on the cluster. The manually approach for one simulation run includes five random generated weights and the five ASCII input files. It is possible to compute the same model in ArcMap with the tool ‘weighted Sum‘ in ArcMap as with the Hadoop-cluster.

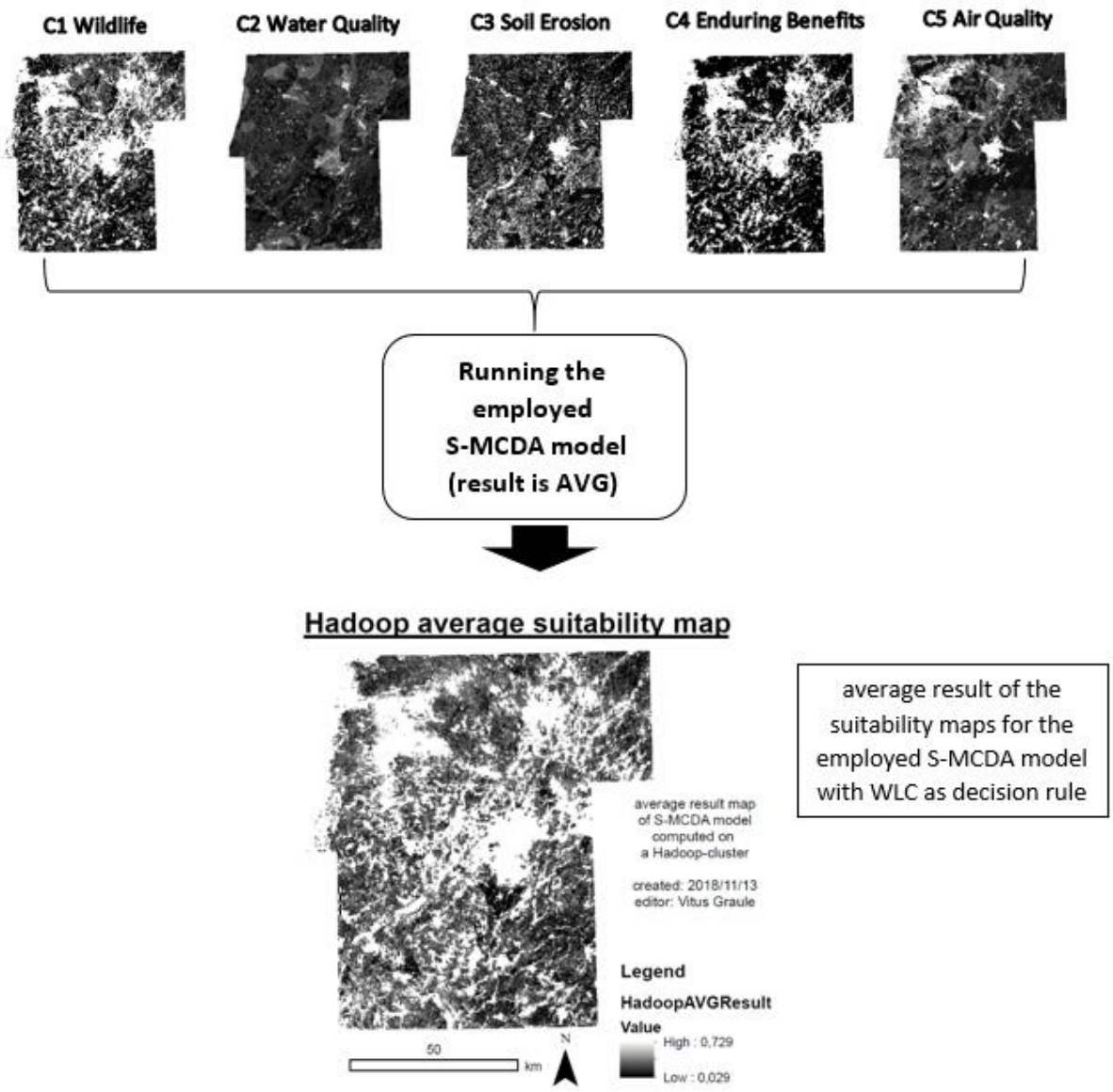

Figure 13: Illustration of S-MCDA approach with pictured input criteria and average result map

Running the model with 2464 simulation runs on the Hadoop-cluster at CUAS gives a result like pictured in Figure 14. This result map illustrates the average of the stack of suitability maps, which is created within the algorithm run.

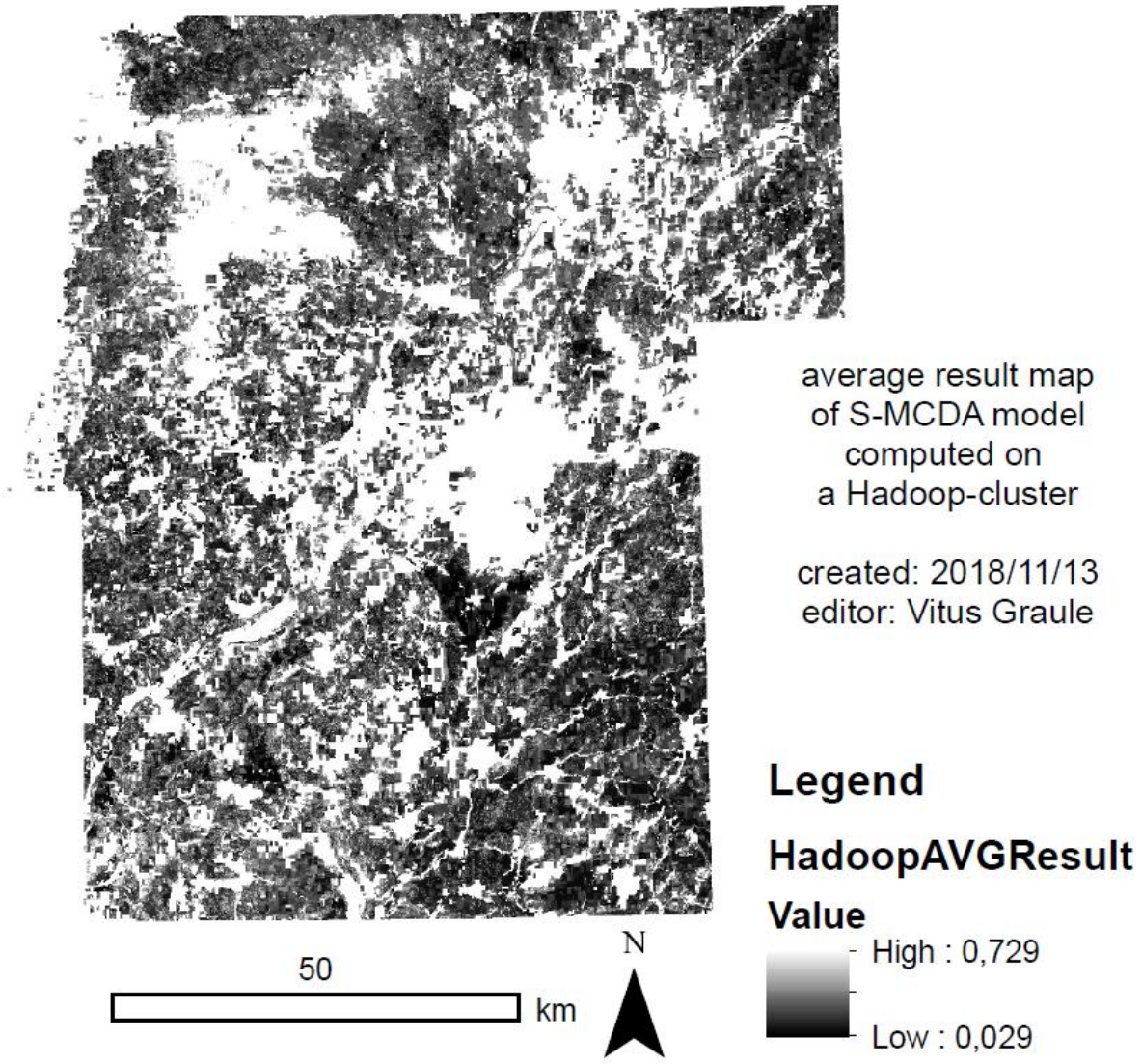

Figure 14: result of average suitability map, computed on the Hadoop-cluster and 2464 simulation runs (random generated weight values)

6 Discussion

The potential for reasonable computational speed-ups is given by the Hadoop-cluster and the developed parallel and distributed version of the provided algorithm but the speed-up is not as great as expected right now.

Building up the Hadoop-cluster lead to some difficulties. The first one was because of the administration password to change the operating system from Windows to CentOs7. Ing. Ulf Erich Scherling had all the administration right and passwords and helped to get access on each node for the OS setup. The OS setup automatically installed a Java version on the workstation. Therefore, it was necessary to check for preinstalled Java versions and remove those. After that it was possible to install Java 1.8 and set the environmental variables. Java is a requirement for Hadoop therefore, it is necessary to install Java before continuing with the installation of Hadoop Version 2.7.2 on a node. Using machines with different hardware requisite lead to different configurations within the Hadoop setup. One had to match the configurations to the hardware specification of the workstation. Another difficulty in the cluster right now is getting log files of computations, job runs and the scheduler. With the more information of this log files the cluster and algorithm could be developed for further speed up.

Currently, the proposed algorithm solution is limited by the size of the Hadoop-cluster and by the lack of performance optimizing the code. The developed algorithm should be seen more as a prototype and there is great potential to adapt different things. The algorithm should be optimized regarding to speed-up and time consumption. A difficulty at this point is that Hadoop in general provides good workload balance and therefore speed up in the computation time. Right now it is a problem to figure out which part of causes the limits. Is it the cluster or is it within the implemented code. In a next step one could change the decision rule from WLC to Ideal Point (IP) in the S-MCDA model. IP is a different decision rule, which basically measures the distance between non ideal and ideal point to the computed value. A detailed descripted can be found in Malczewski (1999, pp. 197-204). Furthermore, the algorithm can be extended with a full SEUSA instead of only employing a simulation of the MCS. An overview on its robustness and find potential sources of overhead could help the algorithm as well.

7 Conclusion

The implemented parallel and distributed version of a sequential algorithm for a S-MCDA problem has resulted in a speed-up over a contrastable sequential implementation. The employed S-MCDA model incorporates the decision rule named WLC and features a simulation of the resource-intensive MCS technique used for variance-based SEUSA. The algorithm was implemented with the use of the given computing model MapReduce by Hadoop. The latter computational model had to be used to go with the middleware Hadoop 2.7.2. The implemented approach included to upload the ASCII input data files to the HDFS.

The expected goals of this project have been partially reached. The implemented algorithm should be seen as a prototype that needs further development for the full potential. First implied performance tests give a hint about the potential speed-up with higher simulation runs and a higher workload as well as the comparison of the Hadoop-cluster and a sequential algorithm.

8 Future work

Following steps are planned till the finish of this research project. The Hadoop-cluster should include a directory with log files, which give more detailed information about existing problems and exceptions while running the implemented algorithm on the cluster. With this information it will be easier to solve problems and improve the algorithm and overall computation time of the model. This information can also include details about the distribution on the cluster and can give hints how to optimize the distribution on the cluster and also the parallelization on each node. Another possible step could be an increase in the cluster size to get a greater acceleration in the model run time. CUAS can possibly provide some more machines. This machines will be newer than the nodes in the cluster right now, therefore, this workstations should supply great computational power for the cluster. Another way to get more power out of the existing cluster would be to update each node and use all the possible slots for RAM boards on the included nodes in the cluster.

On the Hadoop-cluster:

- Create and writing log-files while running the implemented algorithm for better problem/exception reading and handling
- Increase the number of working nodes for the CUAS cluster

Further steps on the implemented algorithm are to create separated result files with an average and a standard deviation result map. This result files should automatically be converted to ASCII files for further test and checks in ArcMap. Therefore, the no-data value has to be set to '-9999' and the files do need the ASCII-header with further information. The header handling also has to be treated, that the input files do not have to be edited manually before the computation or the upload on the HDFS. Another step would be to change the decision rule in the algorithm from WLC to IP. For the IP decision rule are more information of the whole dataset necessary as for the decision rule WLC. For example the maximum and minimum value of the dataset have to be computed. Before the change of the decision rule one should address the simulation run number and increase this number. This is just useful after an acceleration of the current simulation run number of 2464. For the correctness check a sequential algorithm has to be implemented. Furthermore, this sequential approach is necessary for the performance comparison. The acceleration in the performance can just be measured if there is a similar sequential approach. The time comparison between the sequential and parallel/distributed run is as important as the comparison of the parallel/distributed run with different simulation runs.

On the algorithm:

- Create AVG- and STD-result in one single algorithm run
- Handle with the header of the ASCII file
- Change the decision rule from WLC to IP
- Run algorithm with different simulation runs
- Create sequential algorithm for time comparison
- Correctness check of the results between sequential and Hadoop computed result (AVG and STD-map)

For a better comparison the final algorithm should run on CUAS cluster as well as on the SDSU cluster because the computational power at SDSU cluster is higher. This will give an overview on the difference of both clusters and the ability to accelerate a computing intensive model. Running different numbers of simulation runs a couple of times and taking the average and standard deviation would help as well to get a more comparable result. Finally, the long-term vision is the development of a computationally-efficient algorithm for a complete S-MCDA model and a useable Hadoop-cluster at CUAS for further projects.

9 Acknowledgement

Financial support for this work was provided by the Austrian Marshall Plan Foundation. Scientifical support for this project was provided by Prof. Dr. Piotr Jankowski and Mr. Dave McKinsey from the Department of Geography at San Diego State University. My supervisor at Carinthia University of Applied Science are DI (FH) Christoph Erlacher, MSc., FH-Prof. Dr. Gernot Paulus, MSc. MAS and FH-Prof. Dr.-Ing. Karl-Heinrich Anders from the Department of Geoinformation and Environmental Technologies. My supervisor from CUAS supported me as well in the project. The international office at CUAS location Villach supported me with the organization of the research stay abroad.

10 Literaturverzeichnis

Apache Hadoop, n.d. *Apache Hadoop.* [Online] Available at: http://hadoop.apache.org/ [Accessed 17 08 2018].

Desch, A., Erlacher, C. & Anders, K.-H., 2018. *Parallel and Distribtued Geoprocession with Python: A S-MCDA Approach,* Villach: s.n.

Erlacher, C. et al., 2017. A GPU-base Parallelization Approach to conduct Spatially-Explicit Uncertainty and Sensitivity Analysis in the Application Domain of Landscape Assessment. *GI_Forum Journal*, Issue 1, pp. 44-58.

Erlacher, C. et al., 2016. GPU-based Solution for Accelerating Spatially-Explicit Uncertainty- and Sensitivity Analysis in Multi-Criteria Decision Making. In: J. Baily, D. Griffith & D. Josselin, eds. *Proceeding of Spatial Accuracy.* Montpellier: s.n., pp. 305-312.

Hayes, B., 2007. Computing in a Parallel Universe. *American Scientist,* (November-December), Volume 95, pp. 476-480.

Hwang, C.-L. & Yoon, K., 1981. *Multiple Attritbute Decision Making.* Berlin, Heidelberg: Springer.

Kang, S. J., Lee, S. Y. & Lee, K. M., 2015. Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems. *Advances in Multimedia,* Volume 2015.

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

Lescisin, M. & Mahmoud, Q. H., 2016. *Middelware for Writing Distributed Applications on Physical Computing Devices.* Austin, TX (USA), s.n., pp. 21-22.

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

Malczewski, J., 1999. *GIS and Multicriteria Decision Analysis.* New York: John Wiley and Sons.

Malczewski, J., 2006. GIS-based Multicriteria Decision Analysis: A Survey of the Literature. *International Journal of Geographical Information Science,* 7 August, 20(7), pp. 703-726.

Malczewski, J. & Rinner, C., 2015. *Multicriteria Decision Analysis in Geographic Information Science.* Heidelberg: Springer-Verlag Berlin.

Polato, I., Ré, R., Goldman, A. & Kon, F., 2014. A comprehensive view of Hadoop research-A systematic literature review. *Journal of Network and Computer Applications*, 01 08, pp. 1-25.

Rauber, T. & Rünger, G., 2013. *Parallel Programming - for Multicore and Cluster Systems.* 2nd ed. Berlin Heidelberg: Springer-Verlag.

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

Taylor, R. N., Medvidovic, N. & Dashofy, E. M., 2012. *Software Architecture.* s.l.:John Wiley & Sons, Inc..

Wainwright, H. et al., 2014. Making sense of global sensitivity analyses. *Computers & Geosciences*, Issue 65, pp. 84-94.

White, T., 2015. *Hadoop: The definitive Guide, fourth Edition.* 4 ed. s.l.:O'Reilly Media.

Yao, X. et al., 2017. Spatial coding-based approach for partitioning big data in Hadoop. *Computer & Geoscience*, pp. 60-67.