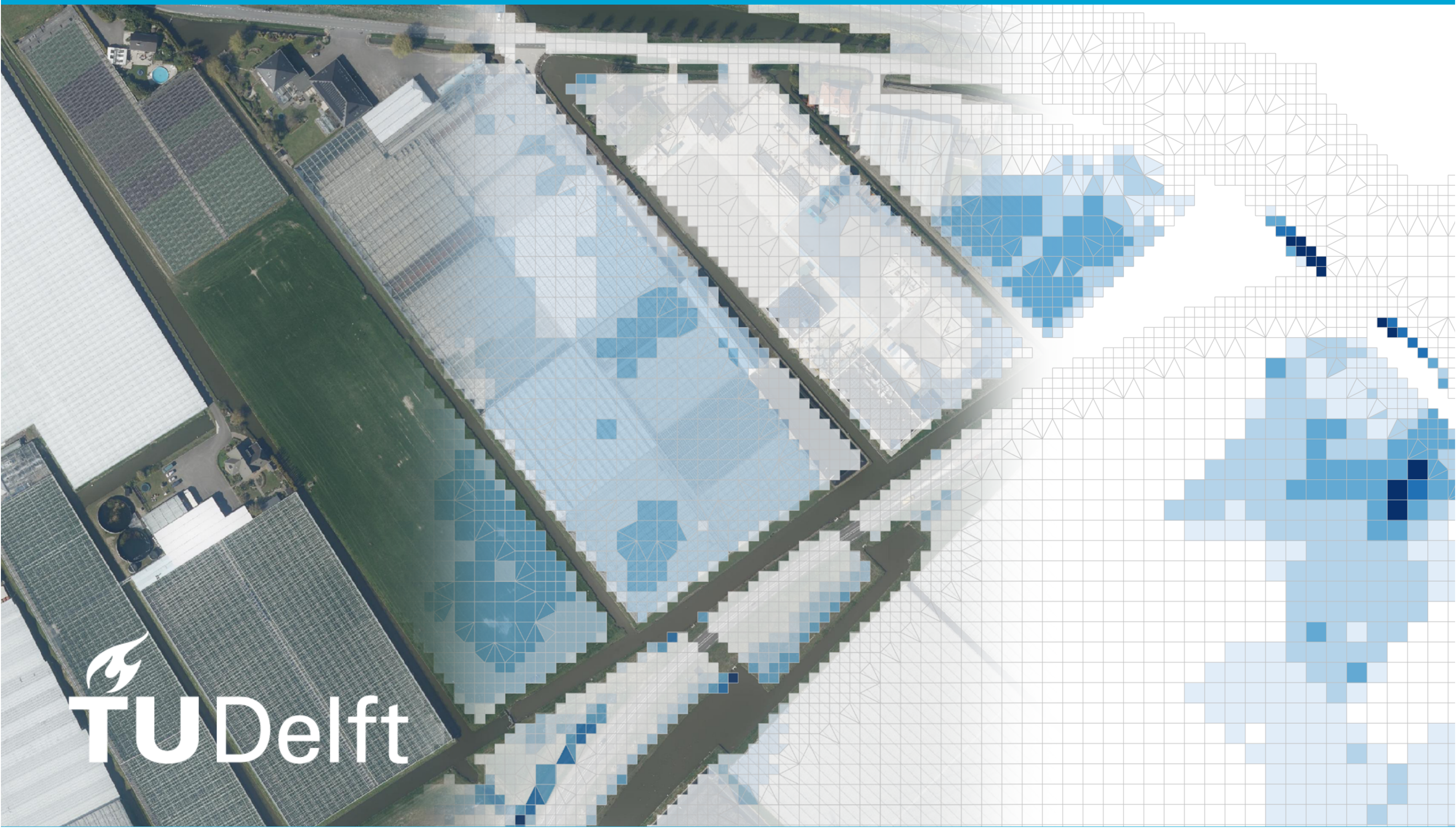
MSc thesis in Hydraulic Engineering

High Performance Computing and Information Theory with D-HYDRO 1D2D on Cloud Infrastructure

Demi de Rijke

2023



MSc thesis in Hydraulic Engineering

High Performance Computing and Information Theory with D-HYDRO 1D2D on Cloud Infrastructure

Demi de Rijke

May 2023

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Hydraulic Engineering

Demi de Rijke: *High Performance Computing and Information Theory with D-HYDRO 1D2D on Cloud Infrastructure* (2023)

The work in this thesis was carried out in the:

Hydraulic Engineering
Delft University of Technology

Supervisors: Assistant Prof.dr. Juan Pablo Aguilar-Lòpez
Assistent Prof.dr. Oriol Colomés Gené
Dr. Mattijn van Hoek

Acknowledgements

I want to express my gratitude to several individuals who have contributed to completing this thesis. First and foremost, I would like to thank Assistant Professor Juan Pablo Aguilar-Lòpez, for his guidance and support throughout the research process. Furthermore, I would also like to thank Oriol Colomés Gené for the interesting discussions. Their expertise and insights have been invaluable in shaping this work.

I am also grateful to Mattijn van Hoek for his valuable contributions and support, including weekly meetings, feedback, and suggestions, which have greatly enriched my research experience. Furthermore, I would like to thank everyone at HKV Lijn in Water, for a great time during my graduation internship and for the support and ideas that they have shared.

I would like to sincerely thank my parents and sister for their unwavering support, encouragement, and patience during my academic journey. I would also like to thank Lennart, who has provided essential support and assistance throughout the thesis writing process. Finally, I would like to thank my friends for their interest and support throughout my studies. Their encouragement and understanding have been essential sources of motivation and inspiration.

Thank you all for your contributions.

Abstract

This study focuses on optimizing the use of high-performance computing on public cloud infrastructure, along with information theory, for assessing water systems. These assessments are computationally intensive and can benefit from parallel computing and the evaluation of the collected data with information theory. A case study of a water system analysis for the Vlietpolder was conducted to test various cloud configuration settings using an embarrassingly parallel batch computation. The hydrodynamic simulations involved D-HYDRO 1D2D models with different precipitation events and model resolutions. The modelling results were quantified using normalized Shannon's Entropy to facilitate the comparison of system configurations, evaluating the batch computation process to determine whether enough simulations have been performed and comparing individual simulations.

The study showed that public cloud infrastructure provides comparable computational performance to local computers and servers, and offers opportunities for vertical and horizontal scaling for parallelization. The study also provides insight into the impact of allocated resources, node size, and node type on cloud infrastructure performance. Furthermore, the quantified information derived from the simulations can be utilized to evaluate the batch computation output and support cost-benefit analyses for selecting configuration settings and model decisions given modeling scenarios.

The study concludes that combining cloud infrastructure and information theory can enhance hydrodynamic modelling for batch computations in water system analysis. The findings provide insights into the potential benefits of utilizing public cloud infrastructure for large-scale computations of hydrodynamic simulations.

Contents

List of Figures

List of Tables

1 Introduction

Hydrodynamic simulations, particularly those based on the two-dimensional shallow water equations, require numerically intensive schemes to be solved, resulting in significant computational demands Cea and Costabile [2022]. High-performance computing involves using parallel computing on large-scale computer clusters to solve complex problems. This can involve using supercomputers, or cloud infrastructure Morales-Hernández et al. [2020]. With cloud infrastructure approaching on-premise cluster-level performance, it has become an appealing platform for high-performance computing, as noted in Matsuoka et al. [2023] and De Sensi et al. [2022].

In the cloud, the capacity of the CPU, memory and storage resources can be adapted to the demands of the user. This is called cloud elasticity. This enables both vertical and horizontal scaling, which entails selecting appropriate computing instances that match the computational requirements. This includes choosing the type and size of instances used, as well as the degree of parallelization of simulated processes. Embarrassingly parallel workloads, that do not require communication between the parallel tasks, may be accelerated as fast as cloud resources permit Fox [2008]. To take advantage of cloud infrastructure and parallelize processes, decisions about vertical and horizontal scaling must be made. In this research, different strategies are tested and analysed to provide insight into these processes.

The cloud infrastructure's elasticity facilitates the parallelization of processes, decreasing computation time but introducing a new parameter, the computation cost. While the cost of performing computations on on-premise systems is often not explicitly determined, each model run on cloud infrastructure is explicitly charged. Even on the ”unlimited” cloud infrastructure, it is desirable to limit the number of simulations.

Information theory, particularly Shannon's Entropy, is utilized to quantify the collected information. This measure assesses the amount of uncertainty or surprise of data and is utilized to evaluate the information produced by hydrodynamical modelling output. At the batch computation level, the comparison of the entropy of individual events is used to determine whether enough simulations have been performed. The collection of information is combined with the costs of the cloud infrastructure and other parameters of the computational process of different cloud configurations to make for different scenarios trade-offs. Indicators are introduced that could help the modeller for different scenarios make a weighed decision by evaluating the potential benefits and costs of a proposed decision.

1.1 Objective

This study focuses on utilizing high-performance computing on public cloud infrastructure, together with information theory in water system assessments to determine the most time, memory and cost-efficient HPC configuration. These assessments are computationally intensive because a large number of input combinations need to be hydrodynamically modelled, which process could be sped up by running parallel and evaluating the collection of information from the inundation maps. The main research question to be answered to meet this objective is as follow:

Research Question:
How could high-performance computing on cloud infrastructure and information theory enhance batch computations for hydrodynamic modelling, such as in water system analysis?

The research has been divided into four sub-questions to enhance the process of answering. Firstly, the possibilities and limitations of the configuration on public cloud instances have been reviewed, which can be done by a literature study since bottlenecks of current computations have been researched yet. Secondly, more detail is being paid to the amount of information produced from the hydrodynamic modelling. This will be followed by focusing on the actual performance of the computation by considering the different configurations. Lastly, information theory will be used to quantify the amount of information of the outcomes of the model simulations. The sub-questions are as follow:

Sub-Questions:

- What are the possibilities and limitations of HPC on cloud infrastructure for hydrodynamic modelling?
- How to quantify the information obtained with hydrodynamic modelling?
- How to optimize the configuration of the cloud infrastructure for HPC with hydrodynamic modelling?
- How could a cost-benefit analysis be performed of HPC on cloud infrastructure for hydrodynamic modelling?

1.2 Thesis Outline

In this section, an outline will be provided of the contents of this report and in what order topics are presented and discussed.

In chapter 2, the literature review is performed to collect and analyse the existing literature on the possibilities and limitations of the public cloud infrastructure. Furthermore, literature has been considered about Shannon's Entropy and the applications of this concept of information theory in water resources management.

In chapter 3, the methodology is described of the experiments that have been performed. The case study of the Vlietpolder is introduced, combined with the modelling process within the water system analysis. For the evaluation of the computational processes on cloud configuration settings, the variants are discussed, and an overview is given of the tests that were performed. This is followed by the method that is used to quantify the collected information and how this relates to the scenarios that are considered for the cost-benefit analysis.

In chapter 4, the results are presented of the tests that have been performed. This considers an analysis of the computational processes on the cloud configurations that include tests about the pod resource allocation and the node size and type definition. The evaluation of the hydrodynamic modelling results with Shannon's entropy definition is presented, together with the indicators for the scenarios of the cloud configuration.

In chapter 5, the discussion of the literature, methodology and results is shown. In this section, the results are explored and combined with the related literature and how these relate to utilising high-performance computing and information theory on the cloud infrastructure.

In chapter 6, the conclusion and recommendations for further research are given. The main findings of this research are combined with answering the sub-questions and main research question, followed by three recommendations for follow-up research.

2 Literature Review

2.1 Hydrodynamic modelling

Hydrodynamic modelling is a simulation technique used to study the behaviour of water bodies, such as oceans, lakes, rivers, and estuaries. It involves the use of mathematical models and numerical methods to simulate the flow of water and the transport of materials within these systems. The models used in hydrodynamic modelling allow to estimate various physical factors, such as water depth, velocity, and pressure, to describe the behaviour of water bodies over time. Hydrodynamic modelling could be used for various purposes. Examples are; providing a deeper understanding of the complex processes that take place within water bodies, such as tides, waves, and currents, to evaluate the impact of various human activities, such as coastal development and pollution, and assessing water systems to analyse the flood risk of the built environment.

With the different types of utilization purposes, there are also different levels of complexity of hydrodynamic modelling techniques. The floodplain flow determines the spatial representation that is required and based on this the models can be dimensionally grouped into 1D, 2D and 3D, with an additional combination of 1D2D Teng et al. [2017]. The application types of the 3 model representations and the combination will be described and discussed.

The simplest representation is to treat the flow as one-dimensional along the centre line of the river channel. For many hydraulic simulations, a 1D assumption can be used, simply because a detailed model solution is unnecessary or because the flow is remarkably 1D, such as a pipe or in a confined channel. With assumptions such as cross-section averaged velocity and parallel flow on the floodplains with respect to the main channel, 1D models can also be used to model open surface floodplain flow Teng et al. [2017], the 1D conservation of mass and momentum that are solved in the D-HYDRO 1D2D Suite are 2.1 and 2.2.

$$\textit{Conservation of mass 1D} \qquad \frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = 0 \tag{2.1}$$

$$\textit{Conservation of momentum 1D} \qquad \frac{1}{A}\frac{\partial Q}{\partial t} + \frac{1}{A}\frac{\partial\left(\frac{Q^2}{A}\right)}{\partial x} + g\frac{\partial h}{\partial x} - g(S_0 - S_f) = 0 \tag{2.2}$$

The 2D models represent a two-dimensional field with the assumption that the water depth as a third dimension is shallow in comparison to the other two dimensions, which means that the velocity variation in the z-direction is assumed as average. The D-HYDRO 1D2D suite of Deltares solves the two-dimensional shallow water equations, which represent the mass and momentum conservation in a place Deltares [2022]. These equations can be derived by depth averaging the Navier-Stokes equations Teng et al. [2017], and are presented in 2.3 and 2.5. For many scales, a two-dimensional shallow water approximation is adequate. However, in certain events, insight into vertical features such as vertical turbulence, vertices and spiral flow at bends could be relevant. For the representation of these features, 3D models could be used. There are 2 common methods that 3D models are solved, the first one being the 2D shallow water equations for horizontal flow with a quasi-3D extension to model velocity in vertical layers and the second model three-dimensional Navier-Stokes equations Teng et al. [2017].

Conservation of mass 2D
$$\frac{\partial h}{\partial t} + \frac{\partial (hu)}{\partial x} + \frac{\partial (hv)}{\partial y} = 0 \tag{2.3}$$

Conservation of momentum 2D
$$\frac{\partial (hu)}{\partial t} + \frac{\partial \left(hu^2 + \frac{1}{2} g h^2 \right)}{\partial x} + \frac{\partial (huv)}{\partial y} = 0 \tag{2.4}$$

$$\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial \left(hv^2 + \frac{1}{2} g h^2 \right)}{y} = 0 \tag{2.5}$$

Besides these three models, the combination of models is receiving wider recognition as it allows to maximize the benefits of modelling approaches Teng et al. [2017]. The most significant combination is the coupling of 1D and 2D models. The most suitable type of model depends on the characteristics of the hydrodynamic problem and needs to be evaluated per scenario. The general rule of thumb is that the more detailed and more extensive the model, the more (computationally) expensive the simulation will be. In essence, 2D hydrodynamic models are the most widely used models in flood extent mapping and flood risk estimation studies Teng et al. [2017]. In the case of a water system analysis, the combination of 1D and 2D models is used, to simulate the channels one-dimensionally and the floodplains in two dimensions.

The simulations are performed with the D-Hydro 1D2D Suite, which consists of three modules: the D-Flow Flexible Mesh, D-Rainfall Runoff, and D-Real Time Control. The Rainfall-Runoff model is used for the simulation of the hydrological processes. This involves including the precipitation as external forcing, the soil type and seepage to determine the balance of the ingoing and outgoing flows for the catchment areas. These catchment areas are linked to the 1D component of the model, the waterways, and depending on the groundwater level and water level in the waterways, this flow can be in both directions. The 1D component of the model simulates the channel flow with the 1D conservation of mass and momentum as shown in 2.1 and 2.2, based on the predefined computational points along the 1D waterway. In the waterways, structures such as gates, weirs and pumps are located that are controlled in the modelling using the D-Real Time Control module. The 2D component simulates the overland flow, without a predetermined flow path with the use of flexible mesh that is finer towards the waterways. These two components are coupled with 1D2D links. At every computational point of a 1D waterway a link is placed between the waterway and the nearest 2D grid cell. The number of computational points along the 1D waterway, the grid size and the number of 1D2D links present a trade-off between accuracy and computation time. More links and grid cells could increase the accuracy but could also increase the computational time and cost.

Hydrodynamic simulations, and two-dimensional shallow water equations specifically, require computationally intensive numerical schemes to solve the equations that result in a burden Cea and Costabile [2022]. Developments are going on that are decreasing the computational burden of this type of modelling. These advances are in the field of computing science, where cloud computing devices can be utilized to enable parallel computing Teng et al. [2017] as in the field of artificial intelligence and machine learning (AI-ML). In the latter field, the simulations are not performed numerically, but based on techniques such as surrogate modelling. However, given that AI-Ml models require extensive data and training, this type of modelling is expected to be an extension of mechanistic resource models and also requires advances with respect to computational speed Morales-Hernández et al. [2020].

2.1.1 Water System Analysis

Flood risk assessment of regional water systems consists of the hydrological translation of extreme precipitation events to the probability of exceedance of water levels, to test the norms of the national administrative agreement on water (Nationaal Bestuursakkoord Water, NBW). The objective of the analysis is to verify whether the norms for flood inundations are met for the different return periods and, if not, what actions should be performed to comply with the regulation. The focus of the water system analysis is not on extreme flood events, but on the inundation of areas that are not intended to inundate. The largest difference with respect to modelling extreme floods is that in the water system analysis, it is about economic damage and/or nuisance in daily life, with no fatalities. The role of hydrodynamic modelling in water systems analysis is to perform simulations on regional urban and rural areas of interest with input parameters that represent real-life events that could occur. The simulations are performed for parameters like groundwater table, precipitation volume and precipitation patterns, where all possible combinations of the parameters need to be computed, resulting in a large batch of hydrodynamic simulations. The model output of all the simulations is evaluated, and return period maps are obtained. These return period maps could be used as input for a damage assessment to get the associated cost of extreme precipitation events, which could be used for the investment policy of water risks Velner and Spijker [2011].

2.1.2 HPC in hydrodynamic modelling

HPC refers to solving computational intensive problems with parallel processing on large-scale computer clusters such as supercomputers or cloud infrastructure Morales-Hernández et al. [2020]. The computational burden of hydrodynamic model simulations on a local computer or a small-scale computing cluster, should be managed more efficiently with HPC. However, high-performance computing for hydrodynamic modelling requires both developments on the software side and the computational side. In this section, the focus will be on the software side discussing the possibilities and limitations of the used software for model simulations.

The key characteristic of HPC is parallel processing, computing multiple processes simultaneously. Each simulation from a batch will be computed individually, however, rather than performing them sequentially, these simulations are performed simultaneously parallel to each other. The number of simulations that can be computed in parallel is dependent on the infrastructure that is being used, the size of the compute instances and the number of these active instances, as the model characteristics, such as the amount of resources that are required and allocated to each simulation. Thus a suitable infrastructure is required to perform the HPC. However, from the software perspective, this should also be possible. There are different types of parallel computing, in this case, these will be distinguished based on the communication requirement. When communication between parallel processes takes place, called tightly coupled modelling or grained parallelism and where no communication takes place, called loosely coupled modelling or embarrassingly parallelism. The latter type is the easiest to parallelize, since in multi-CPU systems the computational burden shifts from the number of operations to the communication between processors Morales-Hernández et al. [2020].

In the case of tightly coupled models, a single model is partitioned and computed on different instances, however, the partitions are still dependent on each other. After each time step, it is required to exchange boundary conditions, since the cells require the water level and flow velocity from their neighbouring cells for the next time step Deltares [2022]. When performing the computation on a multi-CPU system, the communication of these boundary conditions becomes the new computational burden Morales-Hernández et al. [2020]. However, GPU computing has emerged in the last few years which could cause an acceleration for the parallel architecture. A central processing unit (CPU) is a versatile electronic component that executes a wide range of instructions sequentially, serving as the primary ”brain” of a computer system. In contrast, a graphics processing unit (GPU) is a specialized electronic component specifically optimized for computationally intensive tasks related to graphics and mathematical calculations. However, GPU systems require software adjustments whereas on CPU systems that is not the case. From the software side, there are elements that could be considered to improve the efficient use of the resources, focusing on load balancing and limiting communication.

One of the most important aspects of tightly coupled modelling for hydrodynamic modelling is domain partitioning since this influences the overall computation process significantly. Firstly, when the partitions are computed independently from each other, the computation time per time step is determined by the slowest partition. After every time step, the boundary conditions should be communicated with the neighbouring partitions, and this can only take place when all partitions are finished. Therefore, all partitions should be well-balanced computationally, meaning that the computation times of the partitions are comparable. In hydrodynamic modelling, the traditional handling of wet/dry cells is that dry cells are skipped in the computational cycle, and the compute point is moved onto the next cell. This poses an extra challenge to perform load-balancing Morales-Hernández et al. [2020]. Dry-wet cell modelling is a dilemma for water resources hydrodynamics that can result in load imbalance and computational inefficiencies that can affect scalability Tallent et al. [2010].

Furthermore, there is communication between the partitions that influence the computation time. When researching the performance of three-dimensional hydrodynamic models with Delft3D, communication has been found one of the bottlenecks since this takes a relatively long time in comparison with the overall computation process Mogé et al. [2019]. When the model is partitioned, subgroups of the model are created, and virtual boundaries in the model are applied. However, at these virtual boundaries, boundary conditions should be exchanged between the subgroups since parameters such as water level and flow velocity of the neighbouring partition are still dependent on each other. These boundary conditions that should be exchanged between the partitions after each time step in the numerical simulation are called communication. With domain partitioning, well-balanced partitions and limiting the required communication between the partitions is a relevant topic for high-performance computing in hydrodynamic modelling. A large share of computation time spend on data transfer results in inefficient use of resources that affect scalability Morales-Hernández et al. [2020].

Workloads, where the parallel processes do not communicate and/or are independent of each other, are called loosely coupled or embarrassingly parallel. These types of models are suitable for infrastructure that does not have optimal communication between computation cores since network performance has been seen as one of the main bottlenecks for the adaptation of tightly coupled computations in the cloud De Sensi et al. [2022]. In case of the water system analysis, the batch computation consist of individual hydrodynamic models that are simualted independently from each other, therefore, this can be classified as an embarrassingly parallel workload. From the software perspective, these types of computations are more accessible, since the partitioning of a model and the communication between the sections are not necessary. It involves computing different tasks in parallel, but all these tasks are independent. However, in comparison with the regular computing method, on a local computer or cluster, there are still differences that require attention in comparison with high-performance computing. From the software side, considering purely the concept of embarrassingly parallel computing, no significant adjustments are required. The D-HYDRO 1D2D software package is used for hydrodynamic computations. The standard software package however is not compatible with the computation method of the cloud infrastructure, that is done with Kubernetes. Therefore, a Docker Image of the D-HYDRO software has been used, which is a pre-release (beta version) with version Delft3D FM Suite 2022.04 1D2D. Computing with loosely coupled systems requires attention with respect to the vertical and horizontal scaling strategies that are enabled with the 'limitless' resources that the cloud infrastructure has to offer.

2.2 Cloud Computing

Cloud computing is a term that is broadly used. In essence, it can be defined as physical infrastructure that consists of data centres worldwide that are equipped with computing resources that are linked together and made accessible to the public. There are different services that are being offered. From applications that run in the cloud environment, also called Software as a Service (SaaS) to using the hardware and systems software in the data centres, where the infrastructure is offered as a service (IaaS) Armbrust et al. [2010]. For High-Performance Computing, the focus is on utilizing the computational resources of the cloud infrastructure to run HPC applications in a flexible and cost-effective manner, as a potential alternative to on-premise systems Reed et al. [2022].

The advantage of cloud computing in comparison to on-premise owned or long-term rent systems is the possibility of running an application on the most appropriate computational resources in a cost-effective way without the investment in a self-owned system. These resources consist of a large variety of instances that are characterised in terms of memory, CPUs, accelerators and network bandwidth De Sensi et al. [2022]. The public cloud is accessible to the broad public in a pay-as-you-go manner, paying only for the resources that are required for the computation process, enabling scalability options that can be adjusted to the application's requirements Armbrust et al. [2010]. The flexibility to select computational resources that suit the application or to meet the requirements of the modeller such as computation time, is enabled by the two types of scaling that is offered by the cloud: horizontal scaling and vertical scaling.

Horizontal scaling involves adding more computing resources, such as additional servers or virtual machines, to an existing system to increase its capacity. This method is typically used when there is a need to handle a higher volume of traffic or data without overloading the existing infrastructure. On the other hand, vertical scaling involves increasing the capacity of an existing system by upgrading its hardware resources, such as adding more RAM, CPU, or storage. This method is typically used when there is a need to handle larger individual workloads, such as running complex simulations or data analytics. Cloud computing provides the flexibility to implement both horizontal and vertical scaling techniques to meet the needs of a wide range of applications, allowing optimizing the computing resources based on their specific requirements Armbrust et al. [2010].

For the case of vertical scaling is determining the CPU for an instance. There can be selected different processors, with different numbers of cores, clock frequency and architecture, these processors can are categorized as general-workload and compute-optimized instances. Besides multi-CPU solutions, there is also the option at various providers to select GPUs or TPUs (Tensor Processing Units). However, the application should be qualified to run on GPUs and TPUs, which is a limitation since most of the applications run standard on CPUs. Another relevant configuration setting is the network bandwidth, which is associated with different computation instances De Sensi et al. [2022]. Setting these configurations is called vertical scaling, and also forms the basis of horizontal scaling, since in a cluster the upfront selected node type will be added or removed.

The cloud environment has been considered a good match for embarrassingly parallel workloads since minor differences in the compute performance of an HPC in the cloud and an equivalent on-premise HPC system are expected. However, there is a large difference between the network performance when comparing those systems, making this one of the main bottlenecks for the adaptation of tightly coupled computations in the cloud. This difference in network performance could be justified by the ten times increase in latency that has been found on the AWS and GCP infrastructure. Another benefit of the cloud environment is that in these systems are frequently equipped with new hardware, whereas local compute resources have longer life cycles De Sensi et al. [2022].

The shift from on-premise computing resources to cloud resources induces changes in the economic considerations associated with computing. On-premise resources are typically acquired through capital expenses and offer a certain level of computational power, while cloud resources are acquired through operating expenses, with payments being made for the specific resources and the amount of resources used. In the context of batch analytics, cloud computing exhibits a notable feature referred to as ”cost associativity,” wherein the cost of using 1000 machines for an hour is the same as using a single machine for 1000 hours. Nonetheless, when comparing the costs of on-premise systems with those of the pay-as-you-go approach in the cloud, the concept of elasticity must be considered. This concept facilitates the transfer of risks associated with overprovisioning and underprovisioning from the user to the cloud provider Armbrust et al. [2010].

2.2.1 HPC for hydrodynamic modelling on the cloud

Flood inundation modelling serves the purpose of understanding, assessing and predicting flood events. The application purpose of any modelling is to provide context for the output variables of interest, time and space scales, the required level of accuracy and computational demands Teng et al. [2017]. In the water system analysis, relevant output parameters are the maximum flood extent, the water depth and its return period as input for a damage assessment Velner and Spijker [2011].

In the cloud, the computational resources are offered in a pay-as-you-go manner, which in practice means that the costs for the computing process are expressed explicitly in a monetary value. In cases where the computation time of large computation processes is a limiting factor that influences model choices such as grid size, simulation period and variables that are considered, the computation cost could replace this factor. The computation time when performing batch computations in the cloud will not be a limiting factor anymore, since in theory the computation time can be reduced to the computation time of the longest simulation when all events are run in parallel. A large number of simulations can be performed, with a lot of possible combinations or variables and model decisions, however in the context of a water system analysis, the simulations are performed to obtain information about the maximal water depth per pixel per return period inside the flood prone area. This information can again be used to assess the damage of the extreme precipitation events, thus the gathered information has value when it comes to the application of the results. When the cost of the computation is explicit and the value of the information can be assessed and quantified, how can these be used in an evaluation of different configurations of HPC for flood simulations?

2.3 Information theory

Hydrodynamic simulations generate information about inundations, information that is used as input for a damage assessment and the basis of water system policy to evaluate whether the norms are met and what the economical damage is of extreme precipitation events. When considering the value of information, this can be defined as the expected value of the outcome given a decision using that information minus the expected value of the outcome with a decision based on prior information Weijs [2011]. Modelling decisions influence could influence the quantity of information that is collected, for example, the grid size of the model. A higher model resolution results in a more accurate inundation map, from which a more accurate damage estimation can be made. The finer the model resolution, the larger the computation time and computation cost. Furthermore, the output files are bigger, which corresponds with longer writing and saving time. In this section, the literature on the quantification of information with Shannon's Entropy will be discussed, followed by literature about the damage assessment, that could be used to express the value of the additionally obtained information.

2.3.1 Shannon's Entropy

In 1948 Shannon proposed a measure of information or uncertainty for any discrete probability distribution based on the principle of entropy thermodynamic entropy. To quantify the information produced by a Markoff process and to translate this to the rate at which information is produced Shannon [1948]. The input for determining the entropy is a set of possible events with probabilities of occurrences $p1, p2, ...pn$. These probabilities are everything that is known concerning which event will occur and are translated to a measure of how much information is collected or of how specific the outcome is Shannon [1948]. When applying the theorem of Shannon's Entropy on an event, events with a uniform distribution will have a larger entropy than a sharply peaked one. This corresponds with the intuition that the former distribution represents more uncertainty and unpredictability. Shannon's entropy can be determined with the following equation where the outcome has the units of bits of information:

$$\textit{Shannon's Entropy} \qquad -K\sum_{i=1}^{n} p_i log(p_i) = H \tag{2.6}$$

Shannon's entropy has been applied in the water management and hydraulic engineering field in various ways. Mishra et al. [2009] used the marginal entropy, to investigate the variability associated with monthly, seasonal and annual time series of precipitation data. Where the marginal entropy refers to the entropy of a single random variable in a system consisting of multiple random variables. Haghizadeh et al. [2017] evaluated the flood potential of watersheds when there is a lack of high-quality data with the use of Shannon's Entropy, finding that using this measure, flood-susceptible areas were recognized. Singh [1997] reviewed contributions on entropy applications in hydrology and water resources and stated that entropy-based modelling is versatile, robust and efficient and can be used to evaluate the adequateness of available information.

The events that are simulated with hydrodynamic modelling 'produce' information with every time step and simulation that is being computed. As introduced in the cost-benefit analysis section, the production of this information needs to be quantified in order to consider in the analysis. In the literature, this has not been achieved with the use of Shannon's entropy. However, with the provided insight in the use cases contributes to quantifying information obtained during the hydrodynamical modelling process

2.3.2 The damage assessment

Information is collected when the hydrodynamic simulations are performed. The value of information is the expected value of the outcome given a decision using that information, minus the expected value of the outcome with a decision based on prior information. To determine the values of the outcomes of certain decisions, the information from the hydrodynamic simulations is used to determine the damage. This evaluation is done with the usage of the 'Schade Slachtoffer Module 2017 (SSM-2017), a tool provided by Rijkswaterstaat to assess the damage value as a function of the water depths and land use type. Physical damage is quantified as the sum of the loss of capital or production per affected area or land use type. The relation between damages and inundation characteristics for each land use type has been characterized by damage functions Slager and Wagenaar [2017].

The damage assessment provides critical information regarding potential inundation damage and harm to essential and susceptible infrastructure. This information is particularly important for municipalities, water boards, and provinces that require it to devise built environment policies Slager and Wagenaar [2017]. Utilizing a more detailed model when evaluating regional inundations can provide a higher level of precision in assessing damage, thereby resulting in a more precise estimation of the damage caused. This, in turn, is used as input for water safety policies. The value of information, defined as the difference between the expected value of an outcome following a decision with higher accuracy versus a decision with lower accuracy, is crucial in determining the feasibility of an investment aimed at preventing or reducing damage. For example, when comparing two hydrodynamic simulations - one more detailed than the other - the outcome provides a more precise or less precise damage assessment. If the more accurate model predicts a lower damage assessment, it could result in a decrease in public spending for required investments. Conversely, if the more accurate model predicts a higher damage assessment, then the required investment may be larger, thereby resulting in fewer damage costs.

3 Methodology

3.1 Case Study Vlietpolder

The Vlietpolder is a polder located in the municipality of Naalwijk, the Netherlands (51.990688, 4.227530) and falls under the waterboard of the Hoogheemraadschap van Delfland. For this area, a water system analysis will be performed, where the return period of the inundation area and depth are mapped for different precipitation events. The hydrodynamical modelling of this study consists of the simulation of precipitation events with a range of rainfall volumes and patterns. This is done with a 1D2D model, where the waterways are modelled in 1D and the surrounding land in 2D, with the D-HDYRO 1D2D Suite from Deltares. The model is built up from a flexible mesh grid, with changing grid cell dimensions that are larger at the land areas and become smaller closer to the waterways. In the modelling suite, the standard component D-flow is used for the hydrodynamic modelling itself, in combination with the Rainfall-Runoff module to simulate precipitation events and the Real-Time Control module, for water infrastructural elements such as weirs and pumps.

The Vlietpolder is characterized by the large number of greenhouses which influence the rainfall-runoff in this area since the infiltration of precipitation into the soil is limited and needs to be managed with the channels. The greenhouses are equipped with a basin for the temporary storage of the precipitation for compensating the lack of infiltration in the soil. This measure should store the water during and shortly after the precipitation event to gradually release the water during a later stage as a measure to prevent inundation.

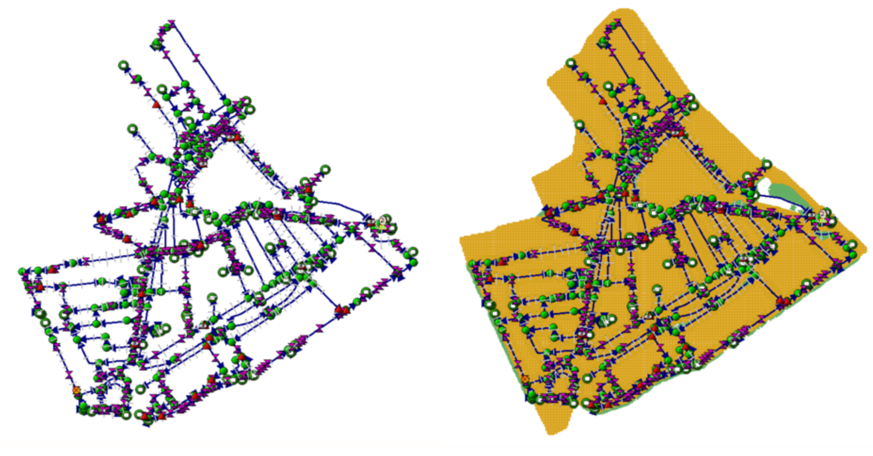


Figure 3.1: D-HYDRO 1D2D model representing the channels in the Vlietpolder with the accompanying objects that are included in the Real-Time-Control. Including and excluding the Flexible Mesh.

To perform the water system analysis, the stochastic method (stochastenmethode according to Velner and Spijker [2011]) is used to identify the relevant system parameters that could lead to extreme water levels. For the Vlietpolder, the precipitation volume and pattern are selected as the key parameters, although other factors, such as the roughness coefficient of waterways and the filling rate of retention basins at greenhouses, may also be relevant. Typically, the groundwater level is included as a parameter, as it affects infiltration and precipitation runoff. However, due to the large proportion of hard surfaces in the Vlietpolder, infiltration is limited, and the groundwater level is not expected to impact the hydrodynamic modelling significantly.

The distribution for the precipitation volume is discretized into bins, and the accompanied probability of occurrence is determined, following the approach outlined in Velner and Spijker [2011]. For the Vlietpolder, a rainfall event with a duration of 4 hours is chosen as decisive, although longer rainfall events are typically relevant in rural areas and shorter events in urban areas. The Vlietpolder is classified as a rural area, but due to its substantial land-covered area, the 4-hour precipitation event is selected. The precipitation data is obtained from the KNMI that provides per precipitation duration, the relation between the precipitation volume and the return period Beersma et al. [2019]. This relation has been added in figure 3.2. This distribution is discretized to precipitation volumes that will be used for the events of the batch computation. For the precipitation volume, 19 values ranging from 10 to 100 mm with steps of 5 mm are computed. An overview of the volumes with the corresponding probability of occurrence is added to the appendix 1.

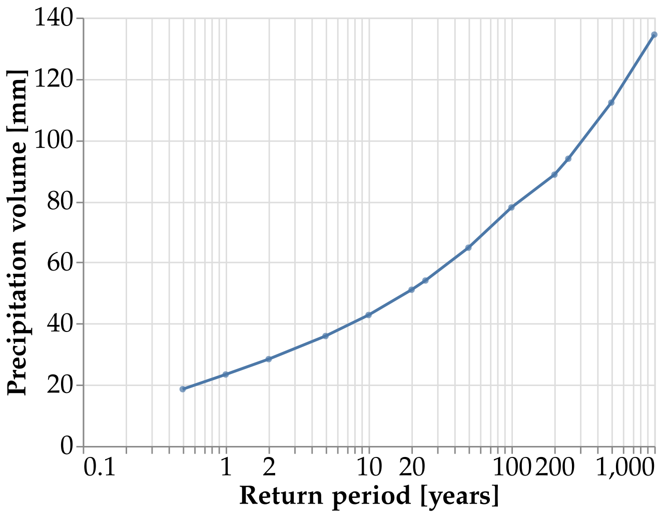


Figure 3.2: The precipitation volume distribution for 4-hourly precipitation events.

The 7 different patterns show the distribution of the precipitation volumes during the 4-hour simulation. The accompanied probabilities are added to the appendix in table 2. The precipitation patterns are also stated in the technical report of the STOWA Beersma et al. [2019] and are shown in figure 3.3. A pattern shows the distribution of the precipitation volume during the simulation. The two most distinguishable patterns are the one that is the most uniform over time, representing a constant precipitation event, and the one with a large peak, indicating a short but intense event. The other 5 events are combinations of these two extremes.

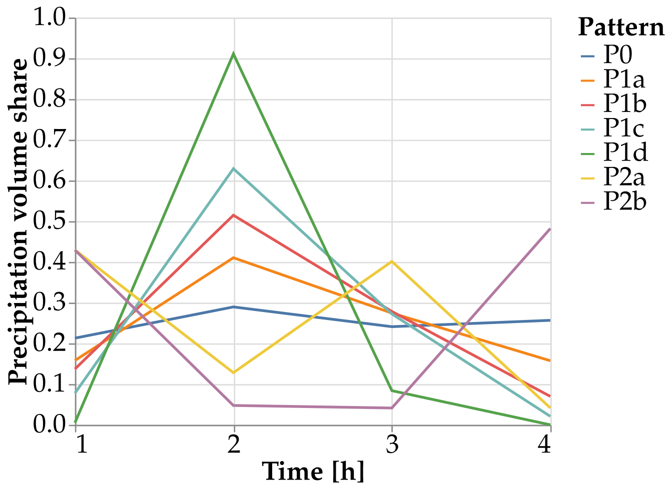


Figure 3.3: The precipitation patterns of the 4-hour event, indicating the share of the precipitation volume per hour.

For each discretized precipitation volume, the seven precipitation patterns will be simulated. This results in a batch of 133 hydrodynamic 1D2D simulations that must be computed. In this case study, two parameters have been selected for the stochastic method, the precipitation pattern and volume. However, for other water system analyses, at least two additional parameters are not unusual. This results in significantly more computations needed in the hydrodynamic batch computation, since each parameter will be discretized and all possible combinations of parameters are computed when no subselection is made. This makes this case study a relatively small batch computation. This batch computation is a representative start to compute and analyze the results and scale up to even more compute-intensive tasks for this research that focuses on utilising cloud infrastructure for High-Performance Computing purposes. These hydrodynamic models also provide organized output that can be quantified with information theory to analyze and review its applicability and apply this in the context of a cost-benefit analysis.

3.2 High-performance computing on cloud infrastructure

Numerous cloud computing platforms offer similar utility computing services on a pay-as-you-go basis. Prominent examples of such platforms include Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure (Azure). These platforms provide comparable service levels, and the Google Cloud Platform has been selected to be used as the infrastructure for this research. While the high-level structure of these cloud platforms is similar, there are differences in the names and designations of their individual elements. However, at a lower level, there are divergences in the working methods of these platforms. Therefore, it is essential to carefully consider the specifics of each platform's working methodology to ensure that it aligns with the desired outcomes and objectives of the project.

For each simulation, an individual pod is created, which in the context of calculation processes can be executed embarrassingly parallel, comprising a large batch of computations. These pods are considered independent and self-contained environments, wherein input parameters are provided for a computation to perform the required software. Once a computation completes and its output results are saved externally, the environment is subsequently deleted. In figure 3.4, the following paragraphs will discuss an architectural overview of the cloud infrastructure and all the components.

To execute the computations, Argo Workflow is an engine that allows container-based workflows, facilitating the building, deploying, and managing of complex workflows within a Kubernetes environment. With this workflow orchestrator, each computation is set up in a pod, which loads the required software and input parameters from external sources. Additionally, the required and allocatable resources, such as CPU and Memory, can be defined for each pod in the configuration step. In appendix 6, an example of a workflow that consists of a YAML file has been added.

After designing the workflow, it can be submitted to a Kubernetes environment, such as the Google Kubernetes Engine (GKE) in Google Cloud Platform (GCP), which automates the deployment, scaling, and management of containerised applications. The steps defined in the workflow are executed within GKE, which retrieves the Docker image from the registry on GCP, loads the input from Google Storage, and runs the computation on the Compute Engine. A Docker image is a lightweight, standalone, executable package that includes everything needed to run an application, including the code, libraries, runtime, environment variables, and system tools. The Docker Image from the D-HYDRO 1D2D Suite has been used for the computations.

In the Compute Engine, a node represents a single computing unit on which the pods will be scheduled. The node can have different types and sizes, making it the basis of the vertical scaling strategy. Single nodes can be combined in a larger network that works together to perform a specific task or provide a particular service. This network is called a cluster. In the cluster, the number of nodes needs to be configured. GKE also offers auto-scaling options, where Kubernetes manages the scaling of the number of nodes. In these tests, a fixed number of nodes per cluster are set to be able to analyse the horizontal scaling strategy. On this cluster, the GKE will schedule the pods to be computed. Following the successful completion of all pods, the environment is deleted, retaining only the output files of the D-HYDRO 1D2D simulation and deleting the computation logging.

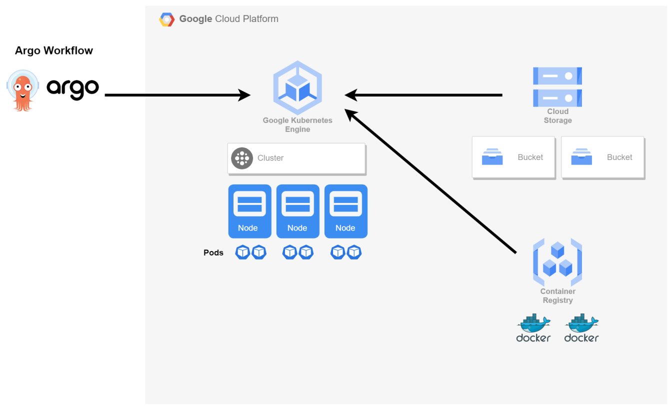


Figure 3.4: The overview of the Cloud Infrastructure shows the utilised resources' architecture.

3.2.1 Resource Allocation

To execute the computations, resources must be allocated to the pods to define the resources that can be used for that computation. These resources include vCPU, memory, and ephemeral storage. vCPU represents a portion of the physical CPU assigned to a virtual machine and is responsible for executing logical and mathematical operations. The number of allocated vCPU to a pod is expected to influence the computation time, as it determines the number of operations that can be executed simultaneously. Allocatable RAM determines the maximum size of hydrodynamic numerical simulations that can be computed. The numerical models used in hydrodynamic simulations often require large matrices, which can consume a significant share of the available RAM. Insufficient available memory can result in computation failure. Thus, the influence of vCPU on the computation time and the memory limit will be tested.

The allocation of these resources is crucial for the performance of computations and the efficient use of cloud infrastructure. To investigate the influence of vCPU and memory on the computation process, this study will test different settings with events from the case study.

The case study consists of 133 batch simulations with varying precipitation volumes and patterns. The simulation with the largest precipitation volume and pattern resulting in the largest inundated area will be used as a benchmark to compare different simulations on a specific configuration. Two grid sizes have been defined, a resolution of 10 by 10 meters and one of 5 by 5 meters, with the former grid used for the benchmark. The latter resolution is assumed to be the most desirable model and also assumed to be the most accurate. However, these are assumptions as these have not been verified with the model outputs and their implications on the objective of the model use.

To test the influence of the vCPU of the pod, the benchmark model is run for different pod vCPU, 0.1, 0.25, 0.4, 0.5, 0.6, 0.75, 0.8, 0.9, 1, 1.1, 1.2, 1.5, 2.0, with a constant memory of 4GB. Ten simulations were run for each pod setting to obtain the average value of the computation times and to analyse the variability of computation times with different pod settings.

To investigate other computational aspects that may influence computation time, the model is also tested with a lighter simulation and a smaller grid size, as well as the same simulation with finer grid size. These four simulations tested with different pod settings are expected to provide more insight into the effect of allocatable vCPU on the computation. An overview of the tested combinations is given in 3.1.

Table 3.1: Overview of the configurations for testing the resource allocation of vCPU.

Event	**Grid size**	**Compute Instace**
V100P2a	10x10	General Workload
V100P2a	10x10	Compute Optimized
V100P2a	5x5	General Workload
V10P0	10x10	General Workload

In addition to vCPU, memory allocation is also expected to have an impact on computation. The same tests for memory allocation will be performed as for the vCPU. In that case, the vCPU will be kept constant at 1.5vCPU and the memory will be discretized in equal steps from 0.5GB to 6GB.

3.2.2 Cluster Configuration

One of the advantages of cloud computing is its ability to perform vertical and horizontal scaling, which offers a broad spectrum of configuration options. This section will focus on testing these options. Vertical scaling refers to defining the characteristics of the computing instance, which includes vCPU, memory, and network bandwidth associated with a node, as well as the hardware itself, such as general workload or compute-intensive instances. Whereas, horizontal scaling involves adjusting the number of nodes in a cluster based on the workload demands. In this case, the nodes initially selected for the cluster are used for scaling. Therefore, the type and size of the node are crucial when setting up a cluster, as it serves as the starting point for horizontal scaling. Due to the various available options, different configurations will be employed for running the batch computation of the case study. Subsequently, parameters from the cloud environment, such as the utilization rate of a node, CPU usage per node and pod, and memory usage per node and pod, will be downloaded and analyzed to evaluate the computation process on the defined cluster.

3.2.3 Node size

This experimentation is aimed at investigating the effects of both horizontal and vertical scaling on the batch computation of 133 simulations from the coarse model resolutions across the three clusters. Specifically, parameters such as computation time, duration of batch computation, and node utilization will be analysed to determine the influence of node size on the computational process. The number of simultaneous simulations conducted on the thin nodes and the fat nodes will differ, with two simulations being computed on the thin nodes, and 21 pods on the fat nodes.

In order to investigate the impact of cluster configuration choices, an experiment involving the deployment of three clusters with varying node sizes and numbers of nodes has been conducted. Specifically, the three clusters will consist of thin, medium, and fat nodes, with the thin nodes being allocated 4 vCPU and 16GB of memory, the fat nodes having 32 vCPU and 128 GB of memory, and the medium nodes 8 vCPU and 32GB of memory. The total amount of vCPU and memory in each cluster remains consistent by adjusting the number of nodes, despite the variation in node sizes and numbers. The thin and fat nodes will be represented by 16 and 2 nodes, respectively, as shown in table 3.2.

Table 3.2: The cluster configuration overview for tests with thin, medium and fat nodes.

	Cluster		
	Light	**Medium**	**Heavy**
vCPU per node	4	8	32
Memory per node	16	32	128
No. of nodes	16	8	2

Furthermore, it is essential to consider the pricing schemes of the clusters since they are composed of nodes with varying sizes and numbers. The cost of running the fat nodes is higher compared to that of the thin nodes. The cluster management fee is $0.10 per hour that the cluster is active. The total cost of computation for each cluster will be analyzed using the cost output from Google Cloud Platform.

3.2.4 Virtual machine type

The virtual machine type is another fundamental aspect of vertical scaling, allowing for the adjustment of a node's computational capacity to meet application requirements. The virtual machine type refers to a predefined configuration or specification of a virtual machine instance offered by a cloud service provider. The terms Compute Engine instance, virtual machine instance, VM instance, and VM are synonymous. On the Google Cloud platform, six standard computational instances are available, spanning from general workload instances to memory- and compute-optimized instances. General-purpose machines exhibit the most favourable price-performance ratio and the most flexible vCPU-to-memory ratio, for standard and cloud-native workloads. Since general-purpose VMs provide a well-balanced combination of CPU, memory, and storage resources, which allows for efficient utilization of resources, making them cost-effective for many applications. Compute-optimized instances, offer a high performance and frequency per core, making them well-suited for performance-intensive workloads like batches of hydrodynamic simulations. However, compute-optimized instances are more expensive than general-purpose machines, presenting a trade-off between computation speed and cost. It should be noted that there are more VM types available in Google Cloud Storage, these are suitable for purposes other than High-Performance Computing, such as web services, database management and virtual desktops, making these not relevant for further analysis.

The configuration of the cluster with regard to the node type is expected to impact computation speed and cost. Therefore, a series of experiments have been conducted to provide more insight into these distinctions. Initially, batch computation will be performed for the coarse model on a light and medium compute-intense cluster. Although it would be ideal to perform the computation on the cluster with fat nodes, it is not feasible to obtain precisely the same size fat nodes for general-workload machines as for compute-intensive machines. The computation times per pod for this batch can be compared to those obtained from general-workload node types. Pod computation times are derived from the time the pod has a vCPU, which includes loading the input and uploading the output from and to the cloud storage bucket. The finer model is anticipated to require more time for computation than for loading and uploading data in comparison to the coarser model. To investigate these differences, the batch computations are computed with the fine model on a light cluster for both general workloads and compute intense machines. Table 3.3 presents an overview of all tests performed to compare the influence of node type on the computational process.

Table 3.3: The cluster configuration overview for testing general workload and compute-intensive instances with different model resolutions and node sizes.

Grid Size	Cluster	Node Type	VM Machine
10x10	Light	General workload	E2
10x10	Medium	General workload	E2
10x10	Light	Compute intensive	C2
10x10	Medium	Compute intensive	C2
5x5	Light	General workload	E2
5x5	Light	Compute intensive	C2

This experiment investigates the impact of various computing instance types on hydrodynamic batch computations. The analysis of the computation process from these experiments provides insights into the influence of these instance types on the computation time and cost. The inclusion of two model resolutions enables the evaluation of the effect of node types on both small and large-scale computations.

3.3 Information quantification of the model results

One of the objectives of this study is to measure the amount of information obtained from hydrodynamic modelling by quantifying the output parameters. The D-HYDRO 1D2D Suite generates various parameters, including 1D and 2D flow velocity and water depth in the channel, which is used to determine the inundation area and depth for certain return periods and for assessing the damage caused by extreme precipitation scenarios. While including flow velocity is relevant for fluvial and coastal floodings, for pluvial inundations the 2D inundation depth of the land area is the crucial parameter. Therefore, only the 2D inundation depth is used to quantify the collected information.

To quantify the amount of information of the hydrodynamic model output, Shannon's Entropy is used as a measure, with its discrete form presented in equation 2.6 in the literature review. To determine the spatial variability of the inundation depth over the land area, the model output is represented as a 2D grid of the land area with inundation depths. The probability of occurrence of water levels is calculated by representing the distribution of the numerical data with a histogram, with the size of the bins determined using Sturges' Rule.

$$\textit{Sturges' rule} \qquad 1 + 3.322 \times log(N) = K \tag{3.1}$$

Where K is the number of class intervals and N is the number of observations in the set. The grid cells of the model will be used as the number of observations. The course and fine model of the Vlietpolder consist of 82614 and 311455 grid cells, resulting in 39 and 43 bins. The lower boundary of the distribution will be set at 0, which represents the dry cells, the minimal value of inundation. The upper boundary of the distribution will be set at the largest water level that has been computed for any cell during the simulation, for both the course and fine model specifically. The y-axis of the histogram presents the number of grid cells of the model in a respective bin. The equation of Shannon's Entropy requires a probability as input, therefore the count per bin is divided by the total number of grid cells.

The total number of bins influences the maximum value of Shannon's Entropy. As different bin sizes will be compared later on, normalization of this value is required. In the discrete form of Shannon's Entropy, the maximum value of entropy is $ln(n)$, which occurs when all the bins have the same probability of occurrence. The smallest possible value for entropy is zero, which corresponds with the definition of Shannon's Entropy. The normalization of Shannon's Entropy, as proposed in Kumar et al. [1986], is done by dividing Shannon's Entropy by the maximum value, $ln(n)$, which results in an entropy between 0 and 1.

$$\textit{Normalized entropy} \qquad -K \times \frac{1}{\ln(n)} \sum_{i=1}^{n} p_i \ln(p_i) = H \tag{3.2}$$

To determine the normalized entropy of a single inundation map, the inundation depths are discretized into bins, and the count per bin is divided by the total number of grid cells. The total entropy per simulation of an event is obtained by continuously writing the inundation map results during the simulation, with a reporting time interval of 15,000 seconds. The total entropy per event is then computed as the cumulative entropy of the inundation maps with a reporting interval of 15,000 seconds. The course and fine grid models have an average calculation time interval of 22 and 13 seconds, respectively. Writing the inundation map output at each calculating interval would result in very large output files, and hence a reporting time interval of 15,000 seconds is chosen as a trade-off between quantifying the collected information and data manageability.

The methodology introduces a new method of quantifying the hydrodynamic model output, which was not previously used. The proposed quantification method has the potential to provide valuable insight into the information associated with individual events and batch computations. This method will be used to quantify the information from a coarse-resolution model and a fine-resolution model.

3.4 Cost-Benefit Analysis

The computational cost for the hydrodynamical simulations on the cloud infrastructure is explicitly defined and can be obtained from the GCP BigQuery billing export. For operational purposes, this means that on a project basis, the computational costs can be considered. This could also result in the focus on the computation cost rather than on the computation time, which could be a limiting factor for standard computing tasks. This shift of focus initiated the idea of reflecting on a hydrodynamic simulation in the cloud from a cost-benefit perspective.

A cost-benefit analysis is a systematic approach to evaluating the potential benefits and costs of a proposed decision. The purpose of conducting a cost-benefit analysis is to determine whether the benefits of a proposed course of action outweigh the costs, and if so, to what degree. It allows decision-makers to compare the expected costs and benefits of different alternatives and choose the option that is most likely to result in a positive net benefit. By taking a comprehensive view of the expected costs and benefits of cloud configuration and modelling decisions, modellers can make more informed decisions, allocate resources more effectively, and improve the overall efficiency of their operations.

The information that is obtained from the hydrodynamic modelling will be quantified with Shannon's entropy, as discussed in the previous section, 3.3. In the cost-benefit analysis, the information that is collected, and the quantity of this, is assumed to be the benefit. The cost in the CBA can be expressed in various ways, depending on the scenario's objective. An objective could be, for example, to obtain the largest amount of information for the minimum HPC cost or to obtain information the fastest with respect to HPC time. In these cases, the cost in the CBA is the HPC costs and the HPC duration, respectively. The cost could be dependent on the objective of a modelling scenario. Therefore, in the CBA, not just a single parameter for the cost should be used, but rather various factors should be included.

3.4.1 Multi-Criteria Analysis

In a Multi-Criteria Analysis, multiple criteria or factors are included to evaluate and compare alternatives. The cost in a CBA is dependent on the objective of a scenario and, therefore, could be represented by multiple factors. The benefit for each scenario will be the estimated amount of information that is dependent on the model resolution, and the cost will depend on the selected factor. For each scenario, a decisive parameter is proposed that could be used to select the most suitable model configuration that suits the scenario's objective.

- The **largest** amount of information: when the hydrodynamic model output is evaluated with Shannon's entropy, a preference could be to collect the largest amount of information.
- The **cheapest** scenario, collects the largest amount of information for minimal resources. For this scenario, the indicator of the *entropy/euro* will be used, since this provides a comparison between the different configurations and model resolutions. With this indicator, the largest value will be most effective, since it provides the largest entropy per euro.
- The **fastest**, the duration of the computation considers the computation time of the modelling process, including the parallelization of the cluster. Therefore, the indicator for the fastest scenario would be the *entropy/duration*, since this expresses the collected information over the computation time, with a larger value resulting in more information being generated in a shorter period of time.
- The **most efficient on the system**, could be expressed with the indicator of *entropy/ computation time*. The duration of the process includes the parallelization of the processes on the cluster set-up. However, when looking at the computation time, strictly time spent in computation is considered the efficiency of the configuration is included.

These scenarios will be used to evaluate the configuration settings of the cloud infrastructure and the modelling decisions that could be taken. During the analysis of the configuration settings of the cloud infrastructure that will be done in section 2.1.2, the computational process and associated costs will be evaluated. The data obtained in this process, in combination with the quantification of the information in section 3.3, will be used to calculate the indicators that are decisive for selecting a certain scenario.

3.4.2 The Marginal Cost-Benefit Analysis

Besides evaluating the different configurations for various scenarios with a Multi-Criteria Analysis, two HPC configuration settings will be evaluated more thoroughly with a marginal cost-benefit analysis. The marginal analysis is applied to examine the marginal changes in costs and benefits associated with a particular configuration. The objective of this analysis is to evaluate the marginal benefits and costs of the two configurations to propose the most suitable configuration.

The two configurations will be the light cluster with compute-intensive and general-workload machines. On both HPC configurations, the fine and the coarse model will be simulated since it is expected these will have a different amount of information. The amount of information is assumed to be the benefit for this analysis. The cost will be expressed as the total computation time of D-HYDRO. For both models on the two HPC configurations, this variable is expected to be different. To compare the marginal cost-benefits of the two configurations, for each set-up the marginal benefit is determined by subtracting the amount of information of the coarse model from the fine model. This represents the information gain. The marginal cost is computed by subtracting the computation time from the coarse model from the time of the fine model, as this represents the cost of the additionally gained information. These marginal costs and benefits are compared for each HPC configuration.

3.4.3 Evaluating the damage cost

Hydrodynamic simulations are employed to develop inundation maps for different return periods in the context of water system analysis, including the present case study. These maps serve as inputs for damage assessments resulting from inundation events affecting the built environment. The level of detail of the assessment depends on the resolution of the model input, with finer models providing higher levels of detail compared to coarser models. Consequently, damage assessments based on finer models are expected to be more accurate than those based on coarser models. In cases where damage assessments based on the finer model are lower than those based on the coarser model, the investments needed to meet policy requirements could be reduced. Conversely, a more precise assessment resulting from a higher level of detail of the inundation models can lead to less damage if the appropriate safety measures are taken during events. Overall, a higher level of detail in inundation models can have a positive cost-benefit for damage assessment, but the balance between the benefits and costs needs to be carefully evaluated.

The damage assessment in this study was conducted using the Schade Slachtoffermodule (SSM) of Rijkswaterstaat Rijkswaterstaat. The minimum input required for this module is an inundation map of the area of interest, which can be complemented with additional data such as water level increase speed and flow velocities. However, the latter data are more relevant for large-scale floods and less so for regional inundation caused by precipitation events Slager and Wagenaar [2017]. The maximum water depth of an event was used as model input, and the output was the damage assessment in terms of economic damage per land use type. To enable a comparison of the assessment for different grid sizes, all events were evaluated for models with two grid sizes.

For each event, the probability of occurrence was determined using statistics from KNMI regarding the precipitation pattern and volume. By multiplying the difference in damage cost with the probability of occurrence of the event, the annual cost of the damage difference was estimated.

In short, the steps were performed to get the flood risk of the precipitation events:

1. Get the maximum inundation depths for all events for both grid resolutions.
2. Run the 'Schade slachtoffermodule' (SSM) for all events.
3. Get the probability of occurrence per event based on the precipitation pattern and volume.
4. Evaluate the annual cost per year by multiplying the probability of occurrence of the event with the damage assessment

This damage assessment will show the impact of the modelling decision of different grid sizes on the cost estimation of precipitation events. In the result and discussion section, these will be considered scenarios for the modeller.

4 Results

4.1 Introduction

Tests were conducted using various compute instances, different node sizes, and varying numbers of nodes in a cluster, in order to identify the most efficient use of cloud infrastructure for high-performance computing. The batch computation for the water system analysis of Vlietpolder, consisting of 133 events, was computed using several cloud infrastructure configurations, and the computational process was analyzed and compared along with its associated costs. Moreover, information theory concepts, particularly Shannon's Entropy, were applied to evaluate the model results of the hydrodynamic models and quantify the information obtained. This measure of uncertainty or randomness was applied to the inundation maps of the model output, and the entropy of the system was evaluated. Finally, a cost-benefit analysis was conducted to assess the value of the generated information from the hydrodynamic simulations and the cost of using cloud infrastructure. This analysis involved weighing the cost of obtaining more detailed information against its impact on the accuracy of the damage estimation, which is linked to the inundation maps.

4.2 HPC on cloud infrastructure

4.2.1 Pod size Allocation

In this study, two rainfall events were tested on two grid sizes (5X5 and 10X10 $[m^2]$) and two compute instances in a cloud environment with various pod configurations to investigate the impact of allocatable resources, such as CPU and memory, on the computation time. One of the most extreme rainfall events was also tested with both grids on a general-purpose machine, with a fixed memory and vCPU ranging from 0.1 CPU to 2 CPU. Each model configuration was run 10 times, and the average computation time, as well as the minimum and maximum values, were recorded and visualized.

The results indicate that the computation times for the coarse grid are significantly smaller compared to the fine grid. Moreover, a larger vCPU than 1.0 does not result in a further decrease of the computation time, while both configurations show an increase in computation time beneath 1.0 CPU. The spreading of the minimal and maximal values exhibits two significant changes. The coarse model has a larger spreading of the values with decreasing allocatable vCPU, and the spreading of the values is larger for the fine model compared to the coarse model.

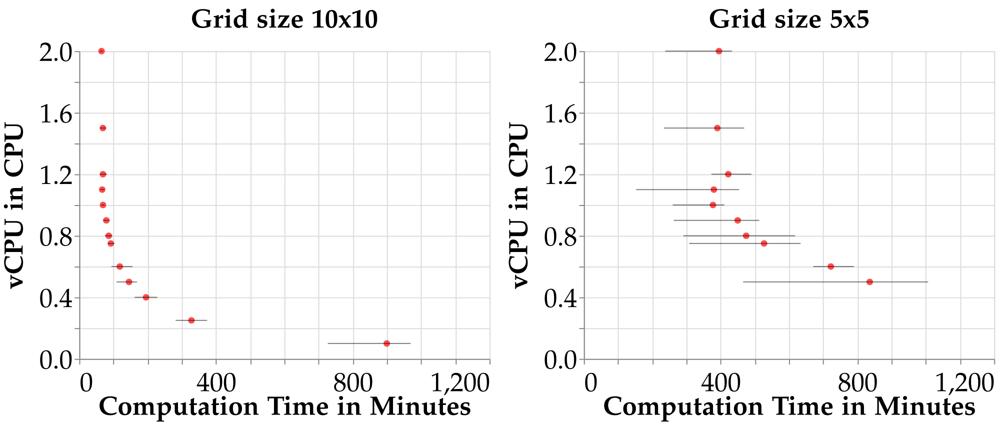


Figure 4.1: The influence of the allocated vCPU on the computation time of the event V100P2a for model resolutions 10x10 and 5x5. The dot represents the average out of 10 simulations, and the error bar is the spreading of the pod computation time with the minimum and maximum value out of the 10 simulations with the same configuration settings.

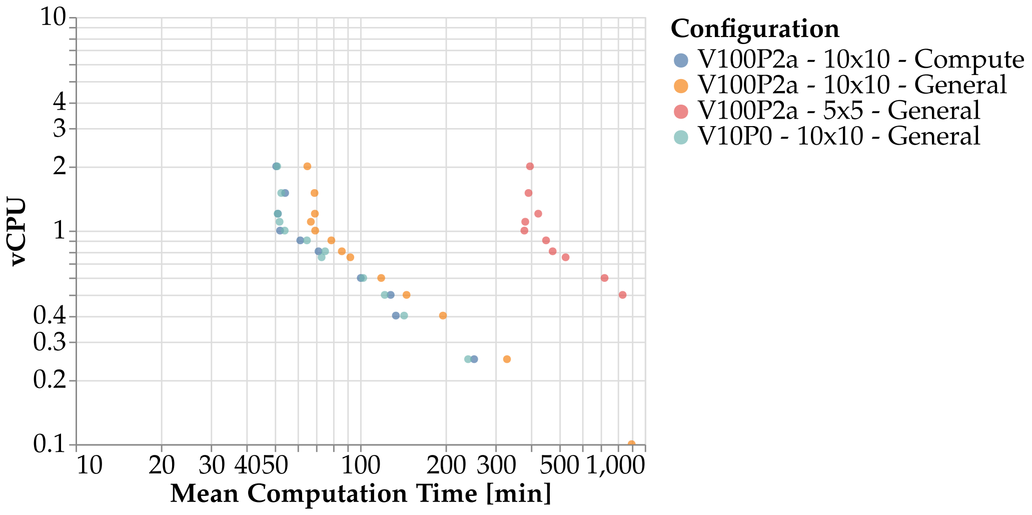


Figure 4.2: The influence of the allocated vCPU on the computation time of different event types, model resolutions and compute instances. The compute-intensive machine and the general-workload machine as the two different types of compute instances that have been tested. The dot represents the average out of 10 simulations of the same configuration.

The batch of computations consists of various events with different rainfall patterns, ranging from a uniform precipitation pattern during the simulation to an event with a high density at one specific moment during the simulation. Figure 4.1 considers one of the most intensive events. Another test was performed with one of the least intensive events to see whether there were differences. Furthermore, the tests have been performed on general-purpose machines, also the compute-intensive machines will be considered and thus included in the tests. These tests with another precipitation event and on a compute-intensive machine, are performed in the same way as the previous test with the grid size. The results have been visualized all together in figure 4.2, plotted on both log axis to show all the results in one figure. The simulations on the coarse grid, but with variations in the precipitation events and compute instance, show similar behaviour but with a shift in computation time. Furthermore, the model that has been performed on the fine grid shows a less fluent pattern in comparison with the 3 coarse models that have been tested. The vertical asymptote at a vCPU greater than 1.0 for the 5x5 model in 4.2 is not as straight as for the 10x10 models. Moreover, the linear line of the increasing computation time with a decreasing vCPU shows more fluctuations for the fine model compared to the coarse model.

4.2.2 Cluster configuration

The cluster is configured with two types of nodes, general-workload and compute intensive, and varying node sizes, namely thin, medium, and fat. The goal is to identify the impact of node sizes and types on the computational process. To this end, the influence of node sizes in the cluster has been evaluated, followed by the node types.

Table 3.2 presents an overview of the configuration settings and batch simulation results for various node sizes. The simulations were performed on three clusters with the same total capacity for CPU and memory, but a different number and sizes of nodes. To analyze the impact of node size on computational performance, the computation times of the 133 events were plotted in a histogram for the light and heavy cluster configurations, which is illustrated in Figure 4.3. The outcomes reveal that thin nodes have the fastest computation times, followed by medium nodes, while fat nodes result in the longest computation time. In Table 4.1, the duration of the computation process, which comprises the scheduling of pods on different cluster configurations, and the computation time, which considers the overall computed minutes of all 133 pods, are presented. The computation process duration varied by a few tens of minutes, whereas the total computation time differed by 2300 minutes between the light and heavy clusters.

Table 4.1: The computation results for the batch simulation with the coarse model resolution on the three different clusters; light, medium and heavy.

Cluster	Grid Size	Node Type	Duration [min]	Computation time [min]	Utilization
Light	10x10	E2	259	7102	0.43
Medium	10x10	E2	242	8247	0.53
Heavy	10x10	E2	265	9438	0.56

Further analysis of the results shows that the batch computation on a cluster with thin nodes does not behave in the same manner as the same batch calculated with the same total amount of resources on fat nodes. Specifically, the total computation time is 33% longer on the cluster with fat nodes compared to the cluster of thin nodes. However, when considering the duration of the computation, the process only takes 2% longer on the cluster with fat nodes, since multiple processes take place in parallel. The medium cluster results in a total computation time that is 16% larger with a computation time that is 6% smaller than the duration. The utilization rate refers to the percentage of time that a virtual CPU is actively executing tasks or processing instructions compared to its total available capacity. in 4.1, the fat nodes show a larger utilization rate compared to the light nodes. This indicates that on the large cluster, the vCPU is more effectively being used to perform computational workloads compared to the medium and light cluster.

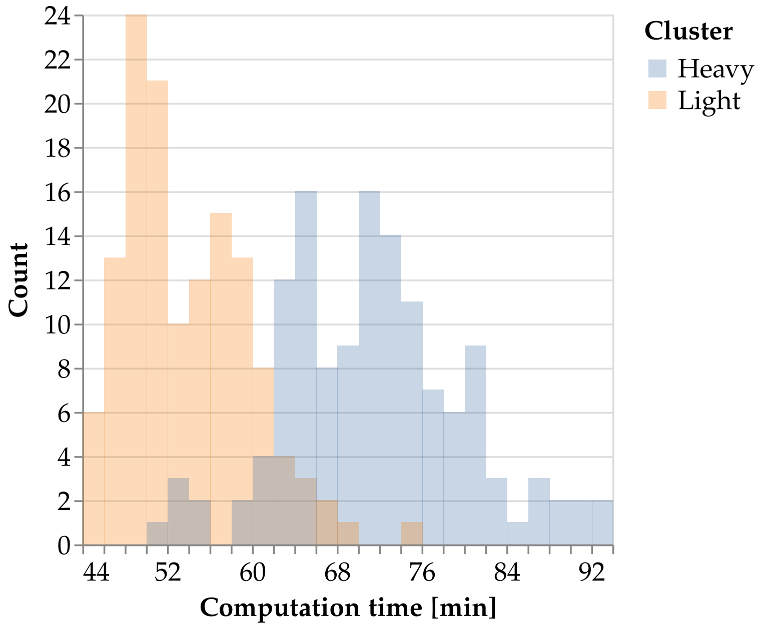


Figure 4.3: Distributions of batch computation times on light and heavy cluster for the model resolution of 10x10.

In addition, the influence of node types on the computational process has also been evaluated. The tests were performed with compute intensive nodes, which have more suitable hardware for computational processes. The results indicate a significant decrease in the computation time. Specifically, the coarse grid has been tested on both medium and light clusters, which show a decrease in the computation time in both cases of 25%. The fine grid has been tested only on the light cluster, showing a decrease of 39%. Figure 4.4 illustrates the distribution of the computation times of the pods on different node instances for the light cluster using the coarse and fine resolution models.

Table 4.2: The computation results for the batch simulation with both model resolution, two different cluster node configurations and node types. The speedup of the compute-intensive nodes is expressed in percentages compared to the equivalent settings with general workload nodes.

Cluster	Grid Size	Node Type	Duration [min]		Computation time [min]	
Light	5x5	E2	1825		77410	
Light	5x5	C2	1520	-16.7%	47055	-39.2%
Light	10x10	E2	259		7102	
Light	10x10	C2	196	-24.3%	5342	-24.8%
Medium	10x10	E2	242		8247	
Medium	10x10	C2	194	-19.8%	6181	-25.1%

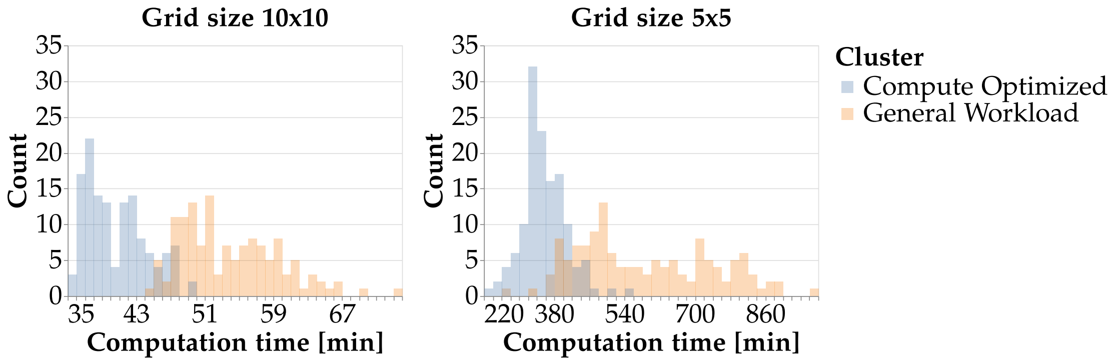


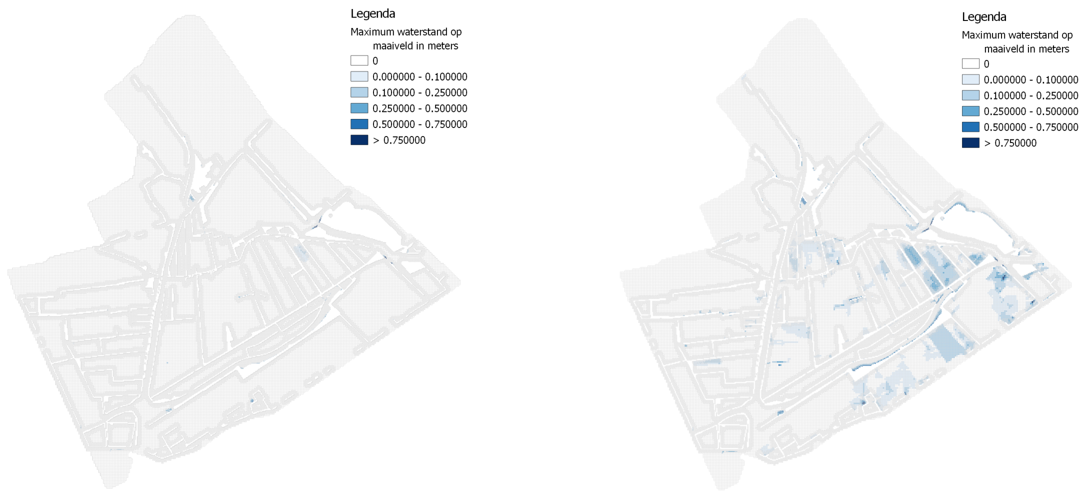
Figure 4.4: Distribution of computation time pods for light nodes on different compute instances, presenting the coarse and fine model resolution.

The computation time range varies as the fine model requires significantly longer computation time than the coarse model. The bin sizes for both distributions are the same, enabling a comparison of both graphs. For both grid sizes, the dispersion of computation times on the compute-intensive nodes is smaller when compared to the general workload nodes. Additionally, the distributions of the computation times on the compute-intensive cluster are shifted towards the lower computation times for both models, similar to the computation times presented in table 4.2. The increase in computation time when the number of parallel processes increases, as observed at the three cluster configurations on the general workload nodes also account for the compute intensive cluster. Where the computation time of the batch on the light compute intensive cluster is 5342 min, this is 6181 min on the medium cluster. For both node types, the computation time increases 16% comparing the light cluster with the medium cluster. In conclusion, the results demonstrate that compute-intensive nodes yield faster computation times.

Overall, the results of this study suggest that the cluster configuration, specifically node sizes and types, significantly impacts the computational process. The findings can inform the optimization of cluster configuration for hydrodynamical modelling, taking into account both computational performance. In section, 2.3, the costs for the set-ups have been analysed based on the computational performance will be showed.

4.3 Information Quantification of Model Results

In this study, Shannon's Entropy was utilized to measure the amount of information of the output model for the 133 events in the batch computation of two grid sizes, using the method outlined in Section 3.3. The temporal distribution of water level inundation maps during the simulation was quantified using Shannon's Entropy. The model generated results 105 times during the simulation with a writing interval of 15000 seconds, and entropy was computed at each time interval. Two inundation maps, representing medium and extreme precipitation events, were compared in Figure 4.5, revealing differences in the extent and depth of inundation.



(a) Inundation map V50P2a (b) Inundation map V100p2a

Figure 4.5: Inundation map comparison at timestep 84 of the model simulation between precipitation event V50P2a and V100P2a for the model resolution of 10x10.

The inundation depth per grid cell per time step was stored in a table, which was used to calculate the entropy of the inundation maps. The output data was binned and represented as a histogram, as shown in Figure 4.6. Analysis of the histograms revealed that the first bin, representing no to a small amount of inundation, had the largest count in both cases. However, for the medium precipitation event, there were relatively small counts for larger inundation depths, while the extreme event had larger counts and more bins with cells showing inundation at greater depths. The resulting comparison of the histograms demonstrated that the extreme precipitation event had a larger spread, more variability, and unpredictability, in line with the definition of Shannon's Entropy.

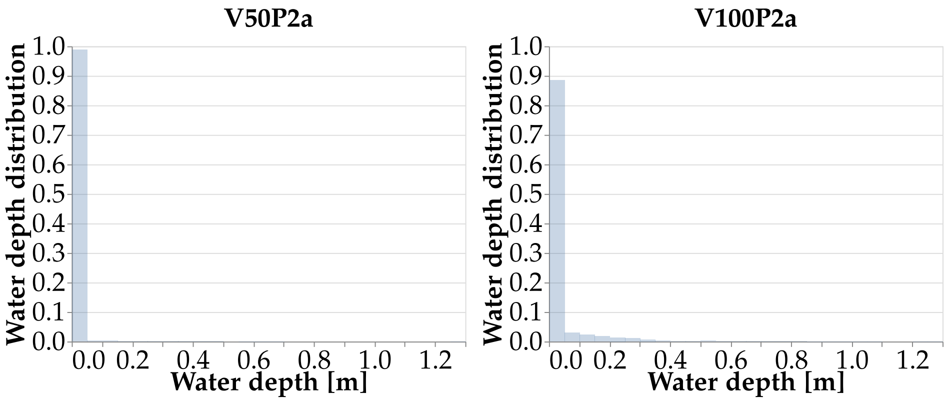


Figure 4.6: Histogram comparison of the water depths in the grid cells at timestep 84 of the model simulation between precipitation event V50P2a and V100P2a for the model resolution of 10x10.

Finally, the development of entropy during the simulation was plotted, resulting in the graphs of figure 4.7, where each point represents the cumulative entropy of the time steps. The entropy development commenced at around time step 70, corresponding to the model setup where the first half of the simulation was dedicated to wetting and filling the system and reaching steady state initial condition before the actual precipitation event started. The intensive precipitation event exhibited a steeper incline of entropy development compared to the medium event. Both graphs showed a similar linear trend in the entropy development after time stamp 350 hours, at which point the precipitation event was concluded and the system slowly returned to its original state. This linearity could be attributed to the system slowly flushing out the water, resulting in similar entropies since the inundated area remained constant.

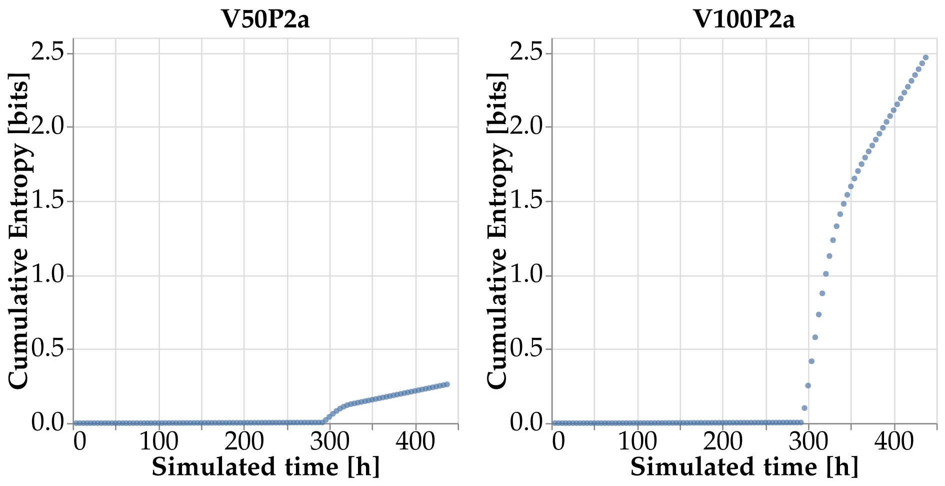


Figure 4.7: Entropy development during the model simulation of event V50P2a and V100p2a for the model resolution of 10x10. The cumulative entropy development of the simulation is shown, where per time step the entropy is determined.

The cumulative entropy per event has been determined as the sum of all entropy estimations per time step. The cumulative entropy for each of the 133 events of both grid sizes from the batches have been sorted and plotted in figure 4.8. To compare both systems, in the computation of the entropy a normalization has been done with respect to the number of grid cells. The result is a distribution of the entropy of all events between the 0 and 3 bits. The first 45 ranked events have similar entropies, the inundation areas and depths are relatively small for these scenarios, what results in a limited difference for the inundation depth distribution and thus entropy. Between 45 and 73 events, the entropy for the fine grid is larger, followed by 27 events where the coarse grid has a larger entropy. From event 90 to the final one, the entropy for the finer grid is greater than the coarser grid again. A larger entropy represents a larger variability of the input distribution. From an entropy of around 1.3 bits, the fine grid has a greater entropy. In figure 4.8, also a histogram of the entropy per event is shown. For both models there are a significant number of events up to 0.8 bits, with an especially high bin count for the smallest entropy. After the 0.8 bits, the bins show a relatively uniform distribution, where the fine grid counts at the largest amount of entropy.

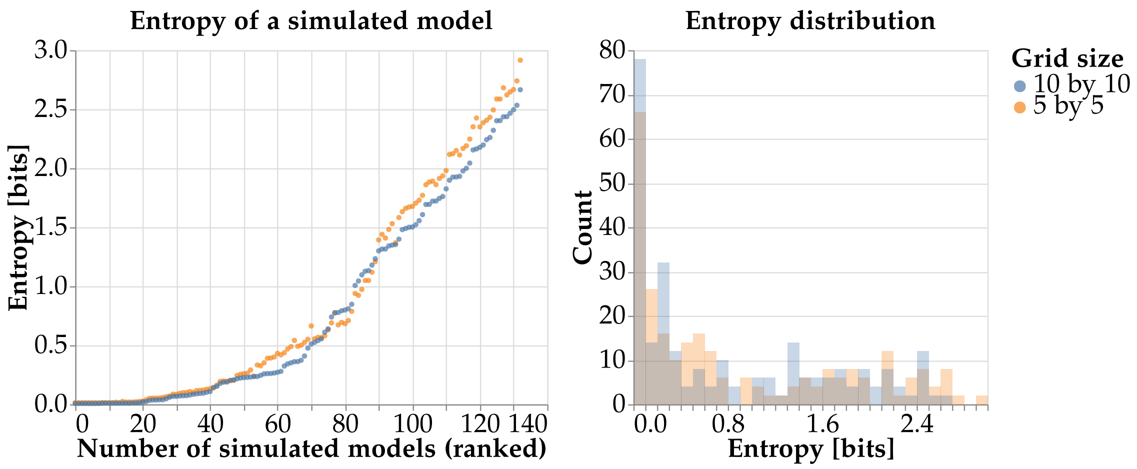


Figure 4.8: The development of the entropy with the ranked simulated models and the histogram distribution of the collected entropy per event.

When looking at the rate information is produced, the rolling window average entropy development out of the simulations divided by the computation time of the simulations that have been derived from the D-Hydro model output, two main observations can be made. The entropy development for the coarse model is higher than the fine model, which indicates that the additional generated information from the fine model is generated at a lower rate than the information produced of the coarse model. The events are ranked on entropy on the x-axis. The other trend is that the entropy development rate is larger for model that contain more entropy, for both model resolutions. Despite the lower computation time of the events with a lower entropy, the entropy development for events with a larger entropy is still higher.

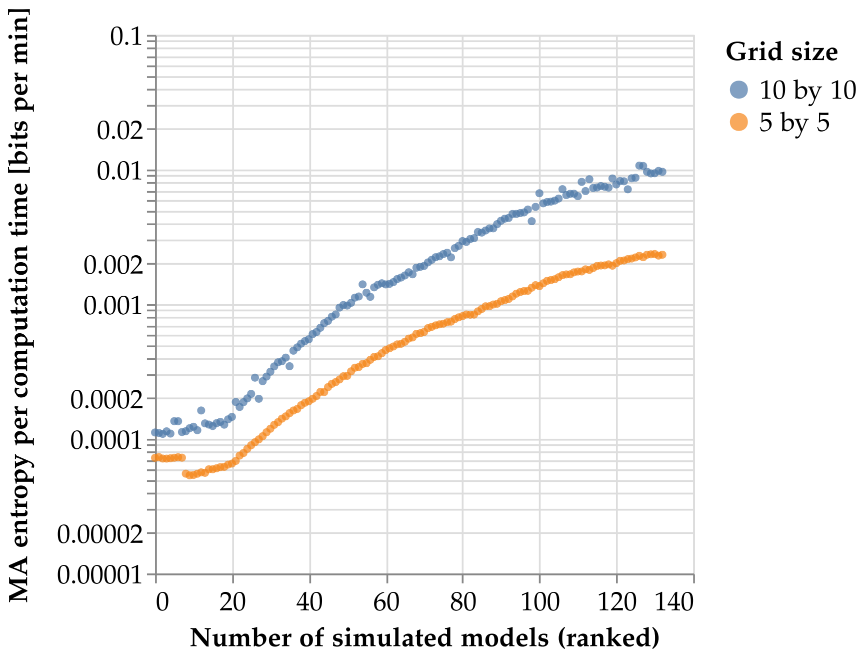


Figure 4.9: The entropy development rate of the batch computation ranked on entropy. Computed by dividing the rolling window over the individual computation time of the pods for both model resolutions

4.4 Cost-Benefit Analysis

This study presents various computational parameters for specific configurations that could assist in making an appropriate decision for a given scenario. Specifically, this section presents the costs associated with different model simulations on various cloud configurations, followed by the results of damage assessments with different model resolutions to demonstrate the economic impact of more accurate data. Finally, the study concludes by outlining criteria that can be employed to determine the most suitable configuration for specific scenarios.

4.4.1 Cloud computing costs

In table 4.3 the configurations of the computation process and model resolutions are compared with respect to the computing duration and associated costs. The model with the fine resolution shows a larger computation time and associated costs with respect to the coarse model resolution. The different set-ups that have been tested with the coarse resolution show that the compute duration of the batch is smaller on compute-intensive clusters, however, there are additional costs associated with this speed-up. In this table, the costs are presented of model and computation choices. These are linked to a benefit, whether it is a speed-up of the computation time, higher model precision or lower costs. Firstly, a more detailed overview will be provided of the build-up of the cloud computing costs, followed by the damage assessment.

Table 4.3: Duration and cost from computational process on different cloud configurations.

Cluster	Grid Size	Node Type	Duration [min]	Cost [€]
Light	5x5	E2	1825	€ 103.10
Light	5x5	C2	1520	€ 113.02
Light	10x10	E2	259	€ 12.61
Light	10x10	C2	196	€ 14.26
Medium	10x10	E2	242	€ 12.00
Medium	10x10	C2	194	€ 13.56
Heavy	10x10	E2	265	€ 12.54

The initial experiment was performed with different node sizes in the cluster, comprising thin, medium, and fat nodes, using the coarse grid. Table 4.3 illustrates that the cost for the three cluster setups on the general workload nodes does not exhibit a substantial difference. Furthermore, Figure 4.10 provides an overview of the costs associated with different categories employed in the billing of the Google Cloud Platform, i.e., Compute Engine, Kubernetes Engine, Cloud storage, and Networking. The computation costs for the same batch calculation on the three clusters are relatively alike, varying from 12 euros for the medium cluster to 12.61 euros for the light cluster. Compute Engine, which encompasses the infrastructure and hardware exclusively used for computing, has the most substantial share of the costs. Cloud Storage follows it, covering the costs for uploading and downloading data from storage to nodes, and Kubernetes Engine, both having an equal share, with a minor fraction of costs allotted to networking.

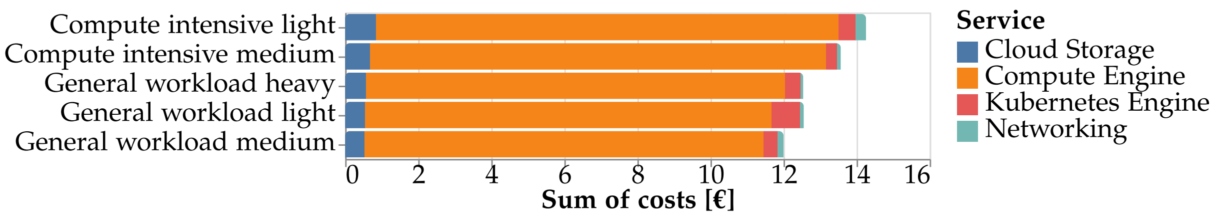


Figure 4.10: Detailed cloud cost overview of computation model resolution 10x10 for different configuration with respect to node size and type.

When considering the cost overview of the simulation of the high resolution models, the compute engine has the largest share of the total costs. The share for Cloud computing has a similar amount as for the fine model. It should be noted that the axis have a different scale, indicating larger absolute cloud computing costs. Which could be justified by the fact that the high model resolution has larger output files that are transferred from the compute engine to the cloud storage.

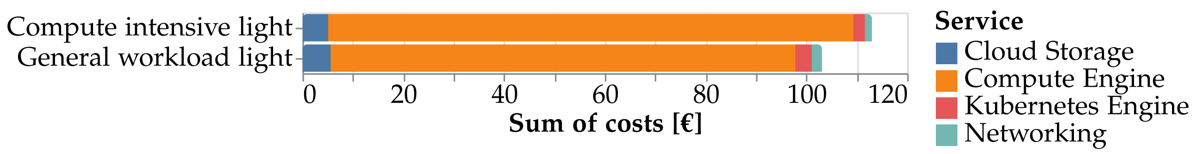


Figure 4.11: Detailed cloud cost overview of computation model resolution 5x5 for different node types.

Upon comparing the costs between the general workload and compute-intensive nodes for computing the coarse model, an increase of approximately 13% is observed. The detailed cost overview indicates that the difference in cost is primarily due to the rise in the Compute Engine. This aligns with the anticipated outcome since optimized hardware is utilized, and the Cloud Storage, Networking, and Kubernetes Engine components should not exhibit a substantial variance.

Two batch computations have been performed for the same model, with two different grid sizes. These consist of a coarse model and a fine model, with larger computation times and costs for the fine model. In figure 4.12, the development of entropy is shown with the associated costs of HPC on the cloud infrastructure. Comparing both models, the coarse batch computation cost is significantly lower than for the fine model. The coarse model reaches a certain amount of entropy with fewer resources. However, the fine model has a larger total entropy.

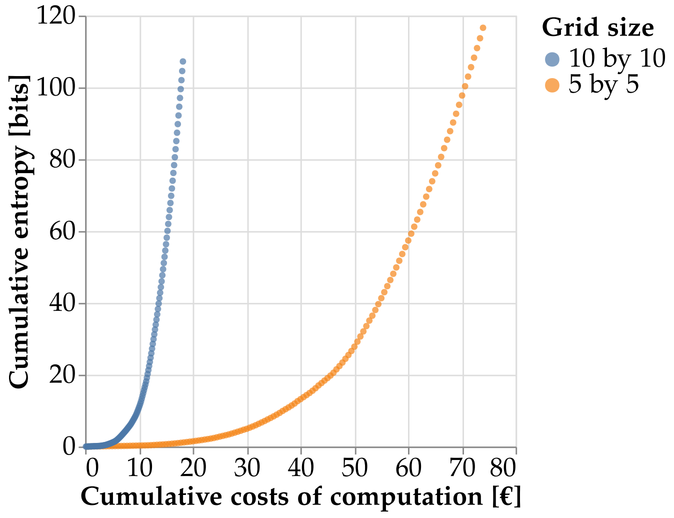


Figure 4.12: Cumulative entropy development versus the associated cumulative cloud computing costs for the coarse and fine resolution model.

The previous test outcomes revealed that the fine models exhibited significantly larger computation times, and consequently higher computation costs. Specifically, on the standard light and the compute light cluster, the computation cost for the fine model was seven times higher than that of the coarse model. Furthermore, in all categories of the cost definition, the costs escalated, with a significant proportion of the costs attributable to the compute engine. The compute-intensive clusters, which demonstrated speed-ups of the computation process by 25 percent, were 13 percent more expensive compared to the standard nodes. The compute engine, responsible for the faster but more expensive nodes, exhibited the most substantial increase in the total cost of the cluster. The fine models are more expensive to compute, and from the previous section about Shannon's Entropy, is was observed that the higher resolution models resulted in a larger entropy. However, the value of this information, or the benefit that is achieved by computing a finer resolution model will be shown with the means of a damage assessment.

4.4.2 The Damage Assessment

To express the value of the additionally obtained information with the fine model, the inundation output maps of the simulations were used to make a damage assessment. Per grid cell, the land-use type was defined. This is linked to a function that assesses the damage at a certain water level, and with the inundation map of the Vlietpolder area, the total damage per event has been determined. In figure 4.13, the cumulative flood risk of the batch computation is shown for both grid sizes. The output of the damage assessment is the damage cost per event, however, since the events have a different probability of occurrence, these cannot be compared by simply summing them up. More extreme events, that result in a larger inundation area and damage have a smaller probability of occurrence. To include both the damage cost and the probability of the event, it was decided to present the flood risk, the product of these two has been considered. The cumulative flood risk for the fine grid is lower than that for the coarse grid. This implies that when this assessment is performed with the coarse grid, the damage is overestimated compared to the fine grid, which could impact decisions made in the water policy.

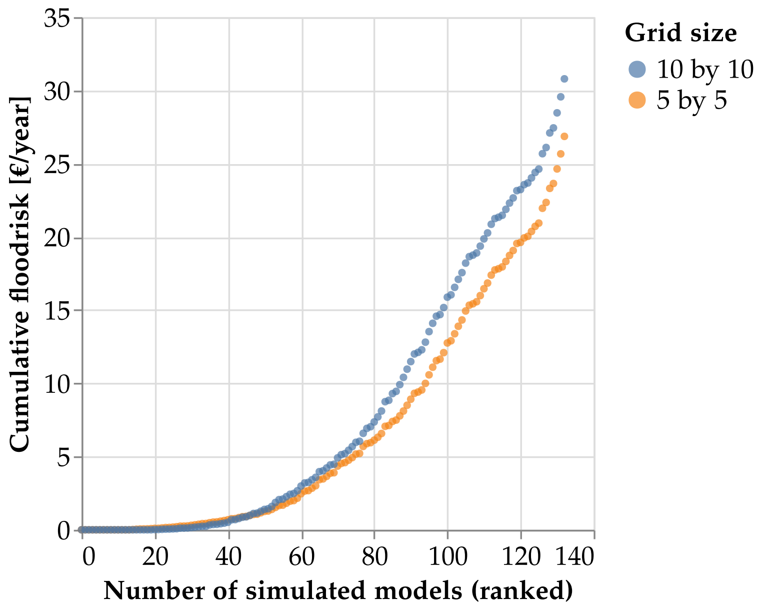


Figure 4.13: Cumulative flood risk the simulated events versus the number of simulated models that are ranked on entropy, both the coarse and fine resolution model.

4.4.3 The Scenarios of the Multi-Criteria Analyis

The batch computations with the different resolutions have been performed on different cloud computing configurations, as presented in 4.4. There have been added two configurations based on estimations and/or simplifications for the matter of comparison, a benchmark of performing the computation on a local computer in series and in theory, the most optimal solution to compute everything in parallel. For these configurations, four columns have been added that present a value or ratio that is relevant for different scenarios. These indicators are:

- **Total Entropy**, indicating the total amount of entropy that has been collected during the simulation, normalized for the number of grid cells that varies with the different model resolutions. This indicator could be used to compare the model setup.
- **Bits/€**, the bits over the computation cost show the rate at which information is produced per euro, also known as the benefit-cost ratio. When the most economical solution wants to be selected, this is the relevant indicator.
- **Bits/duration**, the bits over the duration of the computation, indicate the rate at which the entropy increases with respect to the duration of the computation. With the duration, the computation time of the processes, including the parallelization, is considered. When the objective is to collect the largest amount of information in the shortest period of time, these parameters should be considered.
- **Bits/Comp. time**, the bits over the computation time indicate the raw computation time of the process, excluding the parallel processes and the utilization rate of the configuration. This indicator represents the efficiency of the entropy generation.

With these values and ratios presented, for different scenarios, a suitable decision can be made. In the discussion, these results will be discussed in more detail.

Table 4.4: Overview computational process of various configuration settings and model resolutions with indicators for deciding the most suitable option for specific scenarios. The values in this table are for the computation of the complete data set of 133 accumulated models.

Cluster	Grid size	Node Type	HPC Time [min]	D-Hyd. Time [min]	Cost [€]	Entropy [bits]	Bits/€	Bits/ D-Hydro time	Bits/HPC time
Light	5x5	E2	1825	77410	€ 103.10	116.61	1.13	0.0015	0.0639
Light	5x5	C2	1520	47055	€ 113.02	116.61	1.03	0.0025	0.0767
Light	10x10	E2	259	7102	€ 12.61	107.24	8.50	0.0151	0.4141
Light	10x10	C2	196	5342	€ 14.26	107.24	7.52	0.0201	0.5471
Medium	10x10	E2	242	8247	€ 12.00	107.24	8.94	0.0130	0.4431
Medium	10x10	C2	194	6181	€ 13.56	107.24	7.91	0.0173	0.5528
Heavy	10x10	E2	265	9438	€ 12.54	107.24	8.55	0.0114	0.4047
Local Series*	10x10	N/A	7102	7102	€ 3.50	107.24	30.64	0.0151	0.0151
Parallel*	10x10	E2	74	7102	€ 12.33	107.24	8.70	0.0151	1.4492

*Estimated

Table 4.4, gives an overview of the indicators that could be used to select the most suitable configuration for the given scenarios. However, another approach is the comparison of the configurations considering both the D-HYDRO computation time and the HPC cost, a combination of two scenarios as mentioned in the MCA. This has been visualized in 4.14, where for each configuration, the entropy rate of the batch computation is computed. The entropy rate is defined as the amount of information divided by the HPC cost times the D-HYDRO computation, representing a trade-off between the computation cost and computation time. In the graph, there is a large difference between the entropy rates of the fine model and the coarse model. Prior results showed that the difference in the amount of information between the coarse and the fine model is limited. However, the computation time and cost increase significantly. The entropy rates for the coarse model are more similar, with the entropy rate for the light cluster with compute-optimized nodes being the largest.

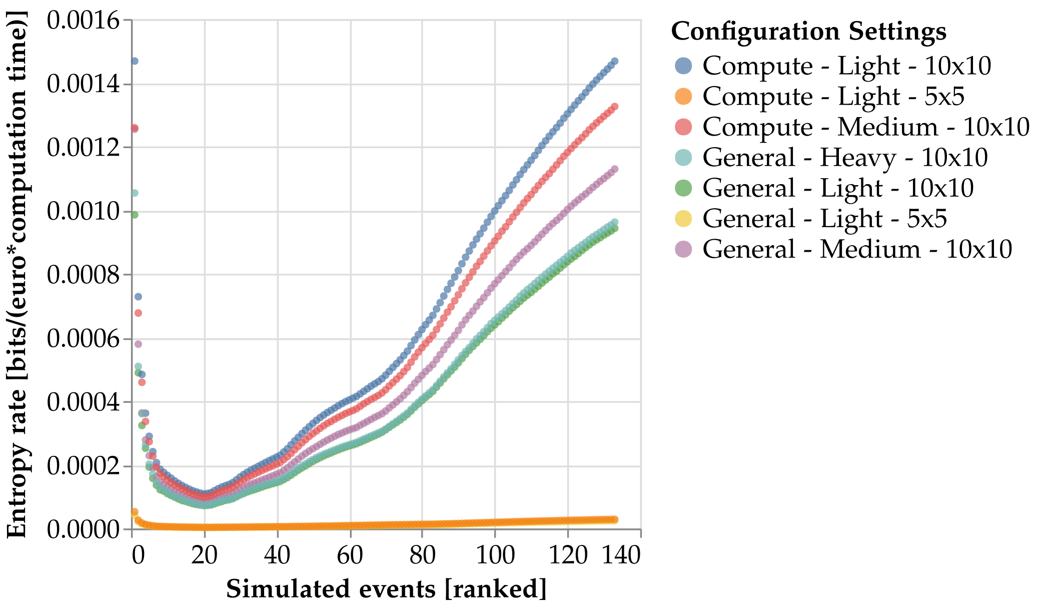


Figure 4.14: The entropy rate per configuration setting during the batch computation. The compute-optimized and general-workload nodes, the light, medium and heavy cluster and the coarse and fine model resolutions are included. The entropy rate is expressed as the cumulative amount of information divided by the cumulative HPC cost and cumulative D-HYDRO computation time.

4.4.4 The Marginal Cost-Benefit Analysis

The marginal cost-benefit analysis compares two configurations, the light cluster consisting of compute-optimized and general-workload nodes. Where in the previous sections, the benefit of the CBA was defined as the amount of information that was collected during the simulations, is the marginal benefit defined as the additionally gained amount of information when running a more detailed (fine) model. The marginal cost is the additionally required

D-HYDRO time to run the more detailed model, whereas the cost was beforehand defined as the total HPC time, the total D-HYDRO time and the HPC cost.

In 4.15, the marginal costs and benefits are shown for the two configuration settings where that were used to simulate the coarse and the fine models. These two models with different resolutions have been computed on different configurations, resulting in similar model outputs. This results in almost equal marginal benefits for both curves in the graph as the cumulative entropy difference is the same. However, the marginal costs that are expressed with the cumulative time difference of the D-HYDRO computation times differ significantly. In the graph, it can be observed that the marginal cost for the cluster with the compute-optimized nodes is smaller compared to the general-workload nodes.

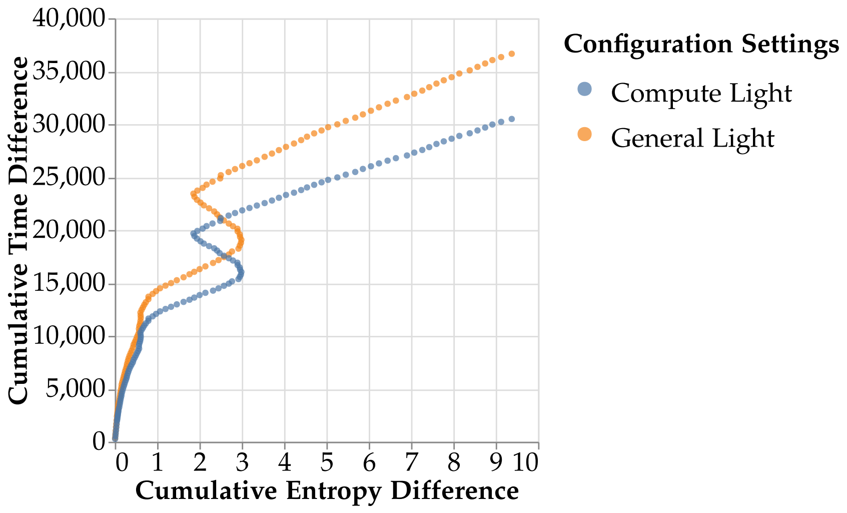


Figure 4.15: Comparison of the marginal costs and benefits when computing the coarse and fine resolution model on the light cluster with compute-optimized and general-workload nodes.

5 Discussion

5.1 The configuration of the cloud infrastructure

Hydrodynamic modelling is a computationally intensive task that requires significant computational resources. Cloud computing offers a flexible and scalable solution to meet these demands. To optimize the performance and cost-effectiveness of cloud-based high-performance computing (HPC) resources for hydrodynamic modelling, it is essential to configure the infrastructure appropriately. In this regard, infrastructure analysis and comparison are conducted at different levels of the cloud computing environment, ranging from the lowest level, which involves pod allocation, to higher levels considering node sizes and types. The tests that were performed with the water system analysis of the Vlietpolder enabled a comparison of different cluster set-ups and resource allocations that provided insight into its influence on the computational process. In this section, the results will be discussed, with an analysis of the processes taking place and their influence on application purposes.

5.1.1 Pod allocation

The efficiency of resource utilization depends on the appropriate allocation of resources for each pod, which is at the lowest level of the cloud infrastructure. Overallocation of resources for a pod results in underutilization, where computational space is reserved but not used, leading to increased cost. On the contrary, there is under allocation of resources to the pod. When this is the case, the implication on the computational process differs for both the vCPU and the memory. In section 4.2.1, tests were performed to visualize the influence of different allocated resources on the computation. In the case of vCPU, the computation slowed down after a certain level but continued, as can be observed in 4.3 and 4.4. For the memory, there is a limit of minimal allocation the computation needs, and when this is not met, it fails. For the model with the coarse resolution, this limit is 0.75Gb and for the fine model 2.0Gb. The memory requirement is dependent on the model characteristics. If the container exceeds the memory request of the pod and the node that the pod is running runs out of memory overall, it is likely the pod will be evicted The Kubernetes Authors [2023].

Figure 4.1 shows that the bandwidth of computation times increases as vCPU decreases for the coarse grid. The D-HYDRO 1D2D Suite is a single-core program that uses a maximum of one vCPU, even when more resources are allocated. When less than one vCPU is allocated, the computation continues, but CPU throttling occurs, which could affect computational stability. This phenomenon becomes more severe as allocated resources decrease, possibly justifying the larger bandwidth of smaller allocated vCPU on the computation time. However, the broad bandwidth of computation times for the fine model is not justified by this phenomenon. The spreading of the computation times for the fine model cannot be specifically identified. However, multiple factors could have an influence. It could be the load-balancing techniques that are used by the cloud platform that distributes the computation across multiple servers that could introduce additional variability for larger computations Shafiq et al. [2022]. Another factor that could influence the variability of the computations is when the virtualized hardware is shared with another user that makes extensive use of the hardware components that influence the performance of our virtual machine Schad et al. [2010].

The insight into the resource allocation could be used operationally when performing computations with this single-core D-HYDRO 1D2D Suite image since the vCPU setting will be applicable to all of those models. The required memory for the model simulation is model dependent. For the coarse grid, 4Gb has been used and for the fine model 5Gb. For hydrodynamic modelling, the required resources are dependent on the size of the matrix that needs to be solved.

5.1.2 Node size

At the node level of the cloud infrastructure, there is the configuration of the cluster that considers the nodes' size and type. Firstly, the size of the nodes in a cluster is discussed with the implication of the findings on optimizing the use of the cloud instances. Tests have been performed with the batch computation of the hydrodynamic models of the Vlietpolder on three different clusters. These clusters have the same total capacity with respect to vCPU and memory, however, were comprised of different node sizes, a lot of thin nodes or a few fat nodes. The HPC time of the batch process was similar on these three different cluster configurations. However, the cumulative D-HYDRO time of the pods on the three clusters significantly differed and increased for the larger node sizes.

The increase in computation time on larger nodes can be explained by the fact that on larger nodes, more processes are taking place simultaneously. In this setup, there were two pods running on an individual thin node and twenty pods on a fat node. The memory and CPU allocation of the individual pod does not change. However, for other parameters, such as internal node processes or external network devices, resource contention can be relevant. These pods pull the image, read the input data in the Google Cloud Storage, write the output first on the node and once the computation is finished in the GCS. These processes result in increased network traffic and communication overhead, which could also result in delays. An HPC performance analysis of the computational processes could provide more insight into which of the bottlenecks, or a combination, results in the increased computation time. These insights result in practical insight when setting up a cluster and whether these conditions apply to specific types of batch computations.

However, the larger D-HYDRO time of the pods on the fat nodes does not influence the total HPC time, which has to do with the utilization of the nodes. This is larger for the fat nodes than the thin nodes, which the scheduling of the pods could justify. Namely, all pods that have been scheduled on the three clusters had the same allocated resources. These pods are scheduled on the nodes until the node does not have enough capacity for another one. The space that is left on the node is wasted. A cluster consisting of 16 thin nodes, has 16 times some wasted space on the node. Whereas a cluster with two fat nodes, only has twice this wasted space, enabling a larger utilization. The advantage of the larger utilisation weighs in this specific scenario up to the disadvantage of the slower computation times. The obtained insight into the resource allocation to the pods could be applied when configuring the workflow to schedule pods on the nodes to increase utilization.

5.1.3 Node type

Thus far, only general-workload instances have been discussed as a vertical scaling strategy. However, the node type can also be adjusted as part of vertical scaling, as compute-optimized nodes have been used in the experiments. As expected, the results indicate that utilizing compute-intensive nodes reduces computation time, resulting in faster simulations than general workload nodes. Specifically, when computing on a compute-intensive cluster rather than a general workload on the medium and light clusters, the computation time speedup for the coarse grid model was approximately 25%, whereas that for the fine grid was 39.2%. This difference in speed-up could be explained by the fact that the computation time for the fine model is larger than that for the coarse model. The compute-optimized machines possess more computation power and facilitate the acceleration of the computation time, while the reading and writing processes remain constant. The net I/O time, the time required to read the input files and write the output files, is expected to be greater for the fine grid than the coarse grid, primarily due to the size of the model output. However, the relative I/O time for the fine grid is expected to be shorter, given that the model simulation requires more time. The speedup of the computation is attributed to the faster model simulation and will not affect the I/O time.

Furthermore, the distribution of the computation times for both model resolutions on the compute-optimized and general workload light clusters are significantly different. On the compute-optimized nodes, computations not only occur faster, but the spreading of the computation times is smaller, as can be observed in the histogram of figure 4.4. This indicates a more stable computation. This narrower range of computation times applies to both grid sizes. Since the clusters have identical configurations, with the same number of nodes, node sizes, and allocated resources to the pod, the discrepancy must be linked to the type of nodes utilized. However, it could also be attributed to the underlying infrastructure since the nodes are located within a larger cluster and data centre, which influences the performance of the reserved hardware.

Allocation of resources to pods is also a critical configuration parameter that significantly impacts the efficient utilization of computation, thereby increasing the utilization of nodes. The size of the nodes, in turn, affects the performance of all pods scheduled on a particular node, indicating a negative effect on overall node performance when the number of parallel processes on a node rises, both applying for general workload and compute-intensive instances. Meanwhile, the node type shows a reduced computation time and greater stability in computational performance. These observations are based on the impact of cloud configuration on the computational process, and the associated costs of these choices will be discussed later, as they also play a significant role in the configuration settings.

5.2 Quantification of Information with Shannon's Entropy

The use of high-performance computing on cloud infrastructure has made it possible to perform many simulations in parallel, thereby eliminating the limiting factor of computation time. However, another factor could be added as the computation costs of the cloud infrastructure can be explicitly defined in monetary terms. To compare the computation time and cost of the computation process, the obtained information from this process must be quantified. In this discussion section, we explore the use of Shannon's Entropy for quantifying the output of hydrodynamic simulations of precipitation events.

Shannon's entropy is a measure of the amount of uncertainty or randomness in a system. The differences in information arise from the variability in the input data, the complexity of the hydrodynamic model, and the assumptions and simplifications made in the model. In the context of hydrodynamic simulations of precipitation events, Shannon's entropy can be used to assess the amount of information contained in the inundation maps generated by the simulations. However, the entropy value does not provide information on the physical meaning or quality of the produced output variable, and it should be used in combination with other metrics and analysis techniques to gain a comprehensive understanding of the hydrodynamic model.

During a single model simulation, the model output was reported at 105 intervals. The cumulative entropy was computed for both individual events and batch computations. Shannon's Entropy was normalized to account for the varying resolutions of different systems. The quantification of entropy at each reported interval was performed by considering only the inundation map, which translates the inundation area and depths into a quantity of information. It should be noted, however, that the inundation map is not the only output generated by the model. The model also provides information on the flow velocity in the 2D area, as well as water depth and flow velocities in the 1D channels. In the context of water system analysis, where the primary focus is on the return periods of specific inundation depths, the inundation map serves as the most relevant parameter. Nevertheless, depending on the intended use of the model output, other parameters may also be of interest and should therefore be taken into account in the computation of entropy.

The entropy value of a single inundation map represents the amount of variation in the water depths. A minimum entropy value of 0 occurs when all grid cells have the same water depth, which can align with the modelling purpose when there is no inundation in the area. However, when the entire area is inundated with the same water depth, the entropy value will still be zero because there is no variability. In that case, the definition of information does not align with the use of the information in the context of water management. On the other hand, a maximum entropy value occurs when the distribution of the histogram is uniform, indicating that there is an equal probability for all water depths to occur with a significant amount of variation in the distribution of water depths. While Shannon's entropy measure is based on the variability of a system, hydrodynamic modelling requires information on the return period of certain inundation levels resulting from a batch computation. During this batch computation, numerous simulations need to be performed where each event will have a unique entropy.

Combining the total entropies of all the events in a figure, such as in 5.1, an entropy increase can be seen. For the first 60 models, the entropy are within a small range, resulting in a high density of models with a similar entropy. However, after model 90, the spreading of the entropy increases, and gaps begin to appear. In this case, the gaps between the entropies of the simulated events are relatively small, indicating a good selection of combinations of events that result in a well-spread distribution. From the perspective of model entropy, however, the density of the models with low entropies is high and a the several events with low values, give the same low amount of information. Only considering the quantity of information and not the quality of the information, events in this range could be discarded from the analysis since the same amount of information is provided.

Remarkably, the grouping of models with similar entropy values can be observed for the coarse grid after 90 simulations and for the fine grid after 100 simulations, where the common parameter is precipitation volume. Figure 5.1 shows that the events are grouped by precipitation volume, and the entropy varies within these groups due to different precipitation patterns.

The gaps between the groups are caused by the discretization of the precipitation volume, which is one of the key steps when initializing the relevant system parameters and setting up the 'stochastic method'. In this case study, the gaps between the groups are limited due to a sufficient number of steps in discretization. However, when the step size is increased, the gaps become larger, and simulation events are missed. When performing a batch computation, an overview of the entropy distribution could support the modeller in this situation, whether enough events have been simulated or how much more and what type of events should be added. Further research to link the entropy coverage to an end product, such as the return period for inundation levels in this specific case, could provide more insight into the influence of the entropy distribution on the end product.

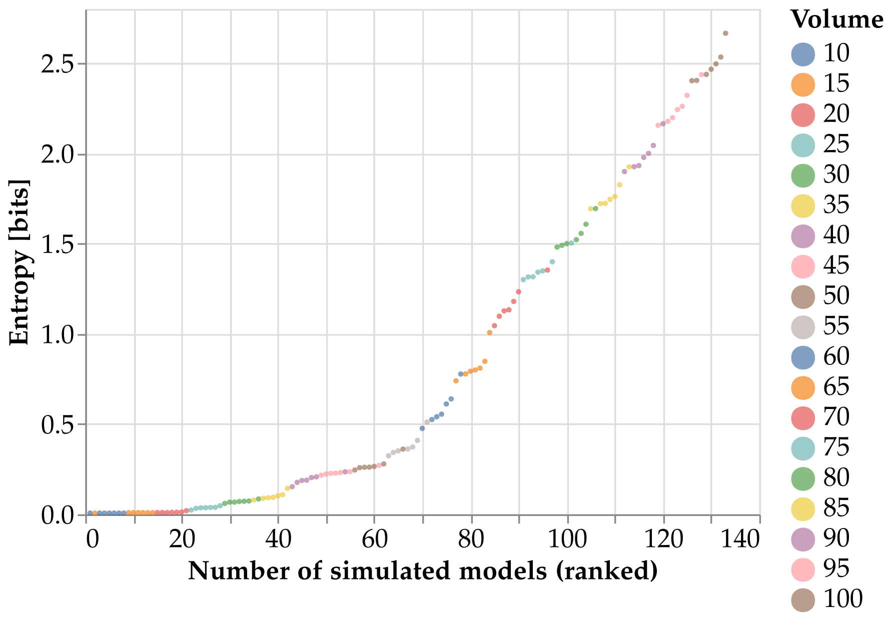


Figure 5.1: The entropy distribution versus the number of simulated event ranked on entropy for the model resolution of 10x10, grouped by the precipitation volume.

The temporal evolution of entropy during a hydrodynamic model simulation via the cumulative entropy of intervals is illustrated in figure 4.7. The entropy begins to develop at time step 70, which coincides with the occurrence of a precipitation event in the model. Before this time step, the model was initialized by simulating a light rainfall event and a period of no further external forcing, allowing the system to reach a state of equilibrium as the initial state. The entropy increment starts with an initial phase exhibiting an exponential function, followed by a linear increment phase, which follows after the termination of the precipitation event, while the system remains partially inundated. During these last intervals, the inundation map exhibits a high degree of similarity, thereby resulting in comparable entropies which determine the angle of inclination of the linear increment. The cumulative entropy obtained from the summation of intervals yields the total entropy of the simulation. However, after the precipitation event has stopped, the entropy still increases due to the presence of inundated areas. However, the benefit of this supplementary information for the modelling purpose is limited. In combination with optimizing the HPC configurations, it could also be relevant to consider the efficiency of the simulation of a single model. Inefficiencies in a single model that is going to be used for a large number of simulations make the overall batch that is computed on HPC infrastructure also less efficient. The wetting and filling of the system and the simulation period after the external forcing for the model of the Vlietpolder have a large share in the total computation time. Optimizing this part has a large influence on the total computation as well.

A comparison was made between the entropy development of coarse and fine models, as depicted in Figure 4.7. The entropy values were normalized for the purpose of this comparison. The results reveal that the differences in entropy between the modelled simulations are relatively minor. This can be attributed to the terrain and grid sizes used in the models. Specifically, one cell of the coarse grid consists of four cells of the fine grid, resulting in a higher level of detail. The fine grid showcases the variation that is reduced to a single level in the coarse grid. However, the underlying terrain is relatively flat, mainly comprising agricultural land and greenhouses, resulting in comparable inundation depths spread across the area, with minimal variations. Since the variation of the inundation depth is the input for Shannon's Entropy, these differences are slight. When examining the 10 by 10 and 5 by five grids, the higher level of detail did not yield significant variation due to the slight differences in the order of magnitude of the grid cells. These modelling resolutions are already highly detailed, and more significant differences are expected when comparing higher resolutions, such as 100 by 100 and 10 by 10.

In this case study, the model output has been evaluated with Shannon's Entropy, which allowed for an evaluation of the information obtained from the simulations and supported in determining whether an adequate number of models had been computed. A potential area of interest lies in the application of this method to model inputs and the analysis of entropy transfer from input to output. Further exploration of this application may support modellers in selecting representative events or specific variable combinations that would reduce the number of simulations required. However, the complexity of the hydrodynamic model, as well as the assumptions and simplifications made, may impact model results, requiring the consideration of the relationship between input and output entropy. Further research on the entropy of model input and entropy transfer, would require a suitable case study, one where the variables are represented as distributions rather than deterministic values.

This discussion covers an exploration of the use of Shannon's entropy as a measure for quantifying the output of hydrodynamic simulations of precipitation events. While Shannon's entropy is a valuable metric for characterizing the variability of inundation maps produced by simulations, it is recommended to combine it with a measure to oversee the quality of the information, as now only the quantity of the information has been considered. By comparing the total entropies of individual events, information distribution can be obtained that provides insight into possible missing data, and the temporal evolution of entropy during a hydrodynamic model simulation can provide valuable insights into the system's behaviour. Shannon's entropy offers a mean for quantifying the information obtained by hydrodynamic simulations. This parameter can be used as the benefit in the CBA as this enables the comparison of different configuration settings.

5.3 The cost-benefit analysis of HPC on the cloud

In this study, the cost of cloud usage was analyzed and divided into four categories, with the most significant cost category being the Compute Engine. In the cloud, the cost associativity concept applies to this category, which implies that the cost of running one machine for 1000 hours is equivalent to running 1000 machines for one hour. This implies that embarrassingly parallel experiments can be accelerated to the extent allowed by the available cloud resources until all simulations are executed in parallel. The remaining three cost categories are linked to the transfer of data and I/O, which is model-dependent, and the cluster management fee, which is charged per hour the cluster is active. To reduce the cluster management fee, which is charged at a rate of $0.10 per hour per cluster, charged in 1-second increments, it is cheaper to run more processes in parallel to reduce the active cluster time. The cost of data transfer and I/O is associated with the model being computed and is charged independently of the cloud configuration.

As demonstrated in the damage assessment, a more accurate model with a larger entropy results in a larger total cost assessment. However, the difference in costs of the damage assessment between the two model resolutions is smaller than the differences in entropy between the two resolutions. The differences in entropy seem to be dampened out in the damage assessment. This indicates that calculating cheaper and faster coarse model resolutions will be better compared to fine-resolution models.

Selecting the appropriate node type, which tests the general workload and the compute-intensive, also has a financial impact. It was observed that the computation duration decreased by 20% to 32%, with an increase in the cost of 9% to 13%. However, a uniform recommendation for using a specific node type cannot be made since it depends on the preference of the modeller and the scenario under which the simulations are being performed. For the four scenarios that were proposed in the methodology and the values for the indicators that were shown in the results, the following configuration settings are the most suitable for each scenario:

- The **largest** amount of information: The largest amount of information, expressed as entropy in bits, is obtained with the grid size of 5x5. Two configurations have the same amount of entropy. Therefore, the modeller can choose the preferred option, the cheaper or the faster one.

- The **cheapest** scenario: This indicator that belongs to this scenario is the amount of information per euro, which is the fully parallel option performed on a local computer with the 10x10 grid size. However, this is using no HPC and thus takes a long time. From the HPC configurations, the medium cluster with general-workload nodes is the most suitable with the coarse grid resolution.

- The **fastest**: The fastest scenario is selected by choosing the largest value of the indicator that expresses the amount of collected information per unit of time. From the HPC configurations, the medium cluster with compute-optimized nodes is the most suitable with the coarse grid resolution.

- The **most efficient on the system**, is expressed with the indicator of *entropy/ computation time*. For which the light cluster with the coarse grid on compute-optimized nodes is the most suitable configuration.

For these scenarios the most suitable configuration has been selected based on the minimal or maximal value of an indicator that has been computed for each configuration. However, in figure 4.14, it was shown that a trade-off between the D-HYDRO computation time and the computational HPC cost show different results. In that case, the entropy rate for the configuration with the light cluster with compute-optimized nodes and the coarse model is the most suitable set-up.

Furthermore, there is the marginal CBA, where the marginal costs and benefits of the computation of a finer model are evaluated on two configurations. This evaluation only consists of two configurations since the fine model has just been computed on the light cluster with compute-optimized and general-workload nodes. In figure 4.15, the compute-optimized nodes show a smaller marginal cost for the computation of additional information. It should be noted that the marginal cost is expressed as the extra computational time that is required to perform a more detailed model. This is again dependent on the objective of the modeller and the given scenario. In the comparison of these marginal costs and benefits, the configuration with the compute-optimized nodes would be the preferred option.

6 Conclusion and Recommendation

This research focused on utilizing the high-performance computing on cloud infrastructure together with information theory in water system assessments.

The first sub-question to be answered is *What are the possibilities and limitations of HPC on cloud infrastructure for hydrodynamic modelling?*

The scalability, cost-effectiveness, and flexibility of cloud infrastructure provide significant benefits for hydrodynamic modelling. The computational performance of the cloud infrastructure reaches similar levels as the performance of on-premise systems. However, there are limitations, such as network performance and technical expertise required to operate and manage the infrastructure effectively. For embarrassingly parallel processes that do not require communication, these can be accelerated as fast as available cloud resources will allow. Overall, cloud-based HPC resources offer a promising solution to the computational challenges associated with hydrodynamic modelling.

In this research, the following was found about the second sub-question that states, *How to quantify the information obtained with hydrodynamic modelling*?

The use of high-performance computing on cloud infrastructure has enabled large-scale hydrodynamic simulations of precipitation events to be performed in a reasonable amount of time. Shannon's entropy is a useful tool for quantifying the information contained in the inundation maps generated by the simulations. By combining the total entropies of all the events, an overview of the entropy distribution can be obtained, providing insights into the simulation events' coverage. However, it is important to note that the inundation map is not the only output generated by the model and that the use of other parameters should be taken into account in the computation of entropy, depending on the intended use of the model output. Furthermore, it should be noted that the quantity of information should not be compared to the quality of the information.

The third sub-question, *How to optimize the configuration of the cloud infrastructure for HPC with hydrodynamic modelling?* , could be answered as follow:

This study conducted an infrastructure analysis and comparison at different levels of the cloud computing environment to provide insight into the influence of different cluster set-ups and resource allocations on the computational process. The results indicate that the efficiency of resource utilization depends on appropriate allocation at the pod level, where overallocation and under allocation of resources result in underutilization and computational failure, respectively. At a node level, the size and type of nodes in a cluster also play a critical role in the overall performance of the computational process. The larger the nodes, the larger the number of parallel processes that slow down all the processes. When all the simulations are performed in parallel, the cost associativity applies, and it is recommended to use thin nodes to perform the computation the fastest and cheapest as possible. Utilizing compute-intensive nodes reduces computation time, thereby resulting in faster simulations compared to general workload nodes. The obtained insights into resource allocation can be applied when configuring the workflow to schedule pods on the nodes and increase utilization. These findings can be used to optimize computational resources and achieve cost-effective and efficient simulations.

About the fourth sub-question, *How could a cost-benefit analysis be performed of HPC on cloud infrastructure for hydrodynamic modelling?* , the following can be concluded:

A general recommendation for node selection could not be made as it depended on the preference of the modeller and the scenario under which the simulations were performed. The parameters as computation time, duration and cost of the cloud infrastructure, in combination with the quantification of information with Shannon's Entropy, resulted in different indicators that could be used to decide what configuration would be the most suitable for specific scenarios. Since the objective of different scenarios varies, the indicators could be used to make weighted decisions. Shannon's Entropy which is used to quantify the amount of gathered information is an essential concept in the CBA as it represents the benefit of this method, which enables the comparison of configurations settings.

Finally, the main research question will be answered, *How could high-performance computing on cloud infrastructure and information theory enhance hydrodynamic modelling for batch computations, such as in water system analysis?*

High-performance computing on cloud infrastructure and information theory can enhance hydrodynamic modelling for batch computations in water system analysis. Cloud infrastructure provides significant benefits such as scalability, cost-effectiveness, and flexibility. Cloud-based HPC resources offer a promising solution to the computational challenges associated with hydrodynamic modelling. The use of Shannon's entropy is a valuable tool for evaluating hydrodynamic simulations of precipitation events and can enhance the modelling process's efficiency and accuracy. To optimize the performance and cost-effectiveness of cloud-based HPC resources for hydrodynamic modelling, it is essential to configure the infrastructure appropriately, considering the allocation of resources at the pod level and the size and type of nodes in a cluster. The cost of cloud usage is a significant consideration, and appropriate node selection and resolution should be taken into account to achieve cost-effective and efficient simulations.

Recommendations for follow-up research would be:

- **High-Performance Computing on the Cloud with partitioned hydrodynamic model simulations**. The network performance of the cloud infrastructure poses a challenge for tightly coupled systems that require significant communication. Model partitioning that balances the computational weight with dry-wet modelling and minimizing the communication could make this bottleneck less significant.
- **Performance analysis of hydrodynamic simulations on the cloud infrastructure**. In this research, the computational process has been analysed based on the computation time on the pod and cluster levels. A performance analysis would provide insight into the processes that are causing the deviations, rather than, as in this research, only measuring their effect.
- **The application of Shannon's Entropy on the model input and entropy transfer.** Researching the input entropy and entropy transfer in hydrodynamic simulations could gather insight into the effect of making a smart sampling of input variables to minimize the number of events that need to be simulated. This could result in computation time, duration and cost.

Bibliography

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4): 50–58, 2010.

J. Beersma, H. Hakvoort, R. Jilderda, A. Overeem, and R. Versteeg. *Neerslagstatistiek en - reeksen voor het waterbeheer 2019*. 2019.

L. Cea and P. Costabile. Flood risk in urban areas: modelling, management and adaptation to climate change. a review. *Hydrology*, 9(3):50, 2022.

D. De Sensi, T. De Matteis, K. Taranov, S. Di Girolamo, T. Rahn, and T. Hoefler. Noise in the clouds: Influence of network performance variability on application scalability. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(3):1–27, 2022.

Deltares. *D-Flow Flexible Mesh User Manual*. 2022.

A. Fox. The potential of cloud computing: Opportunities and challenges, 2008. URL https://www.naefrontiers.org/25261/Paper.

A. Haghizadeh, S. Siahkamari, A. H. Haghiabi, and O. Rahmati. Forecasting flood-prone areas using shannon's entropy model. *Journal of Earth System Science*, 126:1–11, 2017.

U. Kumar, V. Kumar, and J. N. Kapur. Normalized measures of entropy. *International Journal of General Systems*, 12(1):55–69, 1986. doi: 10.1080/03081078608934927. URL https://doi.org/10.1080/03081078608934927.

S. Matsuoka, J. Domke, M. Wahib, A. Drozd, and T. Hoefler. Myths and legends in high-performance computing. *arXiv preprint arXiv:2301.02432*, 2023.

A. K. Mishra, M. Özger, and V. P. Singh. An entropy-based investigation into the variability of precipitation. *Journal of Hydrology*, 370(1-4):139–154, 2009.

M. Mogé, M. J. Russcher, A. Emerson, and M. Genseberger. Scalable delft3d flexible mesh for efficient modelling of shallow water and transport processes, jul 2019. URL https://doi.org/10.5281/zenodo.3527661.

M. Morales-Hernández, M. B. Sharif, S. Gangrade, T. T. Dullo, S.-C. Kao, A. Kalyanapu, S. Ghafoor, K. Evans, E. Madadi-Kandjani, and B. R. Hodges. High-performance computing in water resources hydrodynamics. *Journal of Hydroinformatics*, 22(5):1217–1235, 2020.

D. Reed, D. Gannon, and J. Dongarra. Reinventing high performance computing: Challenges and opportunities. *arXiv preprint arXiv:2203.02544*, 2022.

Rijkswaterstaat. Schade en slachtoffer module. URL https://www.helpdeskwater.nl/onderwerpen/applicaties-modellen/applicaties-per/aanleg-onderhoud/aanleg-onderhoud/schade-slachtoffer/.

J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471, 2010.

D. A. Shafiq, N. Jhanjhi, and A. Abdullah. Load balancing techniques in cloud computing environment: A review. *Journal of King Saud University - Computer and Information Sciences*, 34 (7):3910–3933, 2022. ISSN 1319-1578. doi: https://doi.org/10.1016/j.jksuci.2021.02.007. URL https://www.sciencedirect.com/science/article/pii/S131915782100046X.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27 (3):379–423, 1948.

V. Singh. The use of entropy in hydrology and water resources. *Hydrological processes*, 11(6): 587–626, 1997.

K. Slager and D. Wagenaar. *Standaardmethode 2017 Schade en slachtoffers als gevolg van overstromingen*. 2017.

N. R. Tallent, L. Adhianto, and J. M. Mellor-Crummey. Scalable identification of load imbalance in parallel executions using call path profiles. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2010.

J. Teng, A. J. Jakeman, J. Vaze, B. F. Croke, D. Dutta, and S. Kim. Flood inundation modelling: A review of methods, recent advances and uncertainty analysis. *Environmental modelling & software*, 90:201–216, 2017.

The Kubernetes Authors. Resource management for pods and containers, Mar 2023. URL https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#how-pods-with-resource-limits-are-run.

R. Velner and M. Spijker. *Standaard werkwijze voor de toetsing van watersystemen aan de normen voor Regionale Wateroverlast*. 2011.

S. V. Weijs. Information theory for risk-based water system operation. 2011.

A Precipitation volume and patterns

Table 1: Precipitation values for batch computation.

Precipitation Volume [mm]	**Probability of Occurance**
10	0.999515
15	0.000135
20	9.74E-05
25	7.03E-05
30	5.07E-05
35	3.66E-05
40	2.64E-05
45	1.9E-05
50	1.37E-05
55	9.89E-06
60	7.14E-06
65	5.15E-06
70	3.71E-06
75	2.68E-06
80	1.93E-06
85	1.39E-06
90	1.01E-06
95	7.25E-07
100	1.88E-06

Table 2: Precipitation patterns batch computation

Pattern	**Probability**
P0	0.1930
P1a	0.1768
P1b	0.1768
P1c	0.1768
P1d	0.1768
P2a	0.0590
P2b	0.0410

B YAML-file workflow

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: gcs-delfland10bij10-test1.0-
spec:
  entrypoint: scenario-workflow
  #ttlStrategy:
    #secondsAfterCompletion: 300 # Time to live after workflow is completed, replaces ttlSecondsAfterFinished
    #secondsAfterSuccess: 5     # Time to live after workflow is successful
    #secondsAfterFailure: 300     # Time to live after workflow fails
  imagePullSecrets:
    - name: gcr-json-key

  templates:
  - name: only-running
    steps:
    - - name: delft3dfm
        template: delft3dfm
        arguments:
          parameters:
          - name: model
            value: DIMR10bij10naar5

  - name: define-subdirs
    steps:
    - - name: define-subdirs
        template: read-members

  - name: scenario-workflow
    steps:
    - - name: define-subdirs
        template: read-members
    - - name: run-scenario
        template: run-scenario
        arguments:
          parameters:
          - name: model
            value: DIMR10bij10naar5
          - name: member
            value: "{{item}}"
        withParam: "{{steps.define-subdirs.outputs.result}}"

  - name: read-members
    inputs:
      parameters:
      - name: args
    container:
      image: eu.gcr.io/striking-theme-361911/readmembers:latest
      command: [echo]
      args: {{inputs.parameters.args}}

  - name: run-scenario
    inputs:
      parameters:
        - name: model
        - name: member
      artifacts:
      - name: my-art
        path: /my-artifact
        gcs:
          bucket: delfland_cloud
          key: "models/{{inputs.parameters.model}}.tar.gz"
          # serviceAccountKeySecret is a secret selector.
          # It references the k8s secret named 'my-gcs-credentials
            '.
          # This secret is expected to have have the key '
            serviceAccountKey',
          # containing the base64 encoded credentials
          # to the bucket.
          #
          # If it's running on GKE and Workload Identity is used,
          # serviceAccountKeySecret is not needed.
          serviceAccountKeySecret:
            name: my-gcs-credentials
            key: gcssecretaccess
        archive:
          tar: {}

      - name: bui
        path: "/my-artifact/rr/default.bui"
        gcs:
          bucket: delfland_cloud
          key: "buien/{{inputs.parameters.member}}/DEFAULT.BUI"
          # serviceAccountKeySecret is a secret selector.
          # It references the k8s secret named 'my-gcs-credentials
            '.
          # This secret is expected to have have the key 'serviceAccountKey',
          # containing the base64 encoded credentials
          # to the bucket.
          #
          # If it's running on GKE and Workload Identity is used,
          # serviceAccountKeySecret is not needed.
          serviceAccountKeySecret:
            name: my-gcs-credentials
            key: gcssecretaccess
        archive:
          tar: {}

    outputs:
      artifacts:
      - name: model-output
        gcs:
          bucket: 'delfland_cloud'
          key: "models-output/{{inputs.parameters.model}}_{{inputs.parameters.member}}.tar.gz"
          # serviceAccountKeySecret is a secret selector.
          # It references the k8s secret named 'my-gcs-credentials'.
          # This secret is expected to have have the key 'serviceAccountKey',
          # containing the base64 encoded credentials
          # to the bucket.
          #
          # If it's running on GKE and Workload Identity is used,
          # serviceAccountKeySecret is not needed.
          serviceAccountKeySecret:
            name: my-gcs-credentials
            key: gcssecretaccess
        archive:
          tar: {}
      # generate hello-art artifact from /tmp/hello_world.txt
      # artifacts can be directories as well as files
        path: /my-artifact
    container:
      image: hkvdeveloper/dhydro_test:1.0
      command: ["bash"]
      args: ["-c","cd /my-artifact/ && ./run_docker.sh"]
      #command: [sh, -c]
      #args: ["ls -l /my-artifact"]
      resources:
        requests:
          memory: "4Gi"
          cpu: "1"
        limits:
          memory: "4Gi"
          cpu: "1"

  - name: delft3dfm
    inputs:
      parameters:
      - name: model
      artifacts:
      - name: my-art
        path: /my-artifact
        gcs:
          bucket: delfland_cloud
          key: "models/{{inputs.parameters.model}}.tar.gz"
          # serviceAccountKeySecret is a secret selector.
          # It references the k8s secret named 'my-gcs-credentials'.
          # This secret is expected to have have the key 'serviceAccountKey',
          # containing the base64 encoded credentials
          # to the bucket.
          #
          # If it's running on GKE and Workload Identity is used,
          # serviceAccountKeySecret is not needed.
          serviceAccountKeySecret:
            name: my-gcs-credentials
            key: gcssecretaccess
        archive:
          tar: {}
    outputs:
      artifacts:
      - name: model-output
        gcs:
          bucket: 'delfland_cloud'
          key: "models-output/{{inputs.parameters.model}}.tar.gz"
          # serviceAccountKeySecret is a secret selector.
          # It references the k8s secret named 'my-gcs-credentials'.
          # This secret is expected to have have the key 'serviceAccountKey',
          # containing the base64 encoded credentials
          # to the bucket.
          #
          # If it's running on GKE and Workload Identity is used,
          # serviceAccountKeySecret is not needed.
          serviceAccountKeySecret:
            name: my-gcs-credentials
            key: gcssecretaccess
        archive:
          tar: {}
      # generate hello-art artifact from /tmp/hello_world.txt
      # artifacts can be directories as well as files
        path: /my-artifact

    container:
      image: eu.gcr.io/striking-theme-361911/hkvdeveloper/
         dhydro_test:test
      command: ["bash"]
      args: ["-c","cd /my-artifact/ && ./run_docker.sh"]
      resources:
        requests:
          memory: "4Gi"
          cpu: "1"
        limits:
          memory: "4Gi"
          cpu: "1"
```