Simplifying Usability of Simulation Frameworks using a UI for ASTRA-sim

T2000

for the

Bachelor of Science

from the Course of Studies Computer Science

at the Cooperative State University Baden-Württemberg Stuttgart

by

Anouk de Brouwer

08.09.2025

Due Date	14.07.2025 – 08.09.2025
Student ID, Course	8878508, TINF23A
Company	Hewlett Packard Enterprise, Böblingen
Supervisor in the Company	Susanne Helmer
Reviewer	Ph.D Lianjie Cao

Confidentiality Statement

The T2000 on hand

Simplifying Usability of Simulation Frameworks using a UI for ASTRA-sim

contains internal resp. confidential data of Hewlett Packard Enterprise. It is intended solely for inspection by the assigned examiner, the head of the Computer Science department and, if necessary, the Audit Committee at the Cooperative State University Baden-Württemberg Stuttgart. It is strictly forbidden

- to distribute the content of this paper (including data, figures, tables, charts etc.) as a whole or in extracts,
- to make copies or transcripts of this paper or of parts of it,
- to display this paper or make it available in digital, electronic or virtual form.

Exceptional cases may be considered through permission granted in written form by the author and Hewlett Packard Enterprise.

Stuttgart, 08.09.2025

Anouk de Brouwer

Author's declaration

Hereby I solemnly declare:

1. that this T2000, titled *Simplifying Usability of Simulation Frameworks using a UI for ASTRA-sim* is entirely the product of my own scholarly work, unless otherwise indicated in the text or references, or acknowledged below;
2. I have indicated the thoughts adopted directly or indirectly from other sources at the appropriate places within the document;
3. this T2000 has not been submitted either in whole or part, for a degree at this or any other university or institution;
4. I have not published this T2000 in the past;
5. the printed version is equivalent to the submitted electronic one.

I am aware that a dishonest declaration will entail legal consequences.

Stuttgart, 08.09.2025

Anouk de Brouwer

Abstract

this is not the abstract, write one instead

Contents

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
ASTRA-Sim	Accelerator Scaling for TRAining Simulator
CLI	Command-line Interface
CSV	comma-seperated values
DML	Distributed Machine Learning
ENIAC	Electronic Numerical Integrator and Computer
ET	Execution Trace
FIFO	First-In, First-Out
GEMM	General Matrix Multiplication
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HBM	High Bandwidth Memory
HCI	Human-Computer Interaction
HPC	High-Performance Computing
HPE	Hewlett Packard Enterprise
HTSim	High Throughput Simulator
HW	Hardware
IT	Information Technology
JSON	JavaScript Object Notation
LIFO	Last-In, Last-Out
LLM	Large Language Model
LSQ	Logical Scheduling Queues
ML	Machine Learning
NIC	Network Interface Card
NLP	Natural Language Processing
NPU	Neural Processing Unit
NS-3	Network Simulator 3
SGD	Stochastic Gradient Descent
SOU	States of Understanding
SSOU	Simulated States of Understanding
SSP	Stale Synchronous Parallel
SW	Software
TCP	Transmission Control Protocol

UI	User Interface
US	United States of America
UX	User Experience

List of Figures

1. Introduction

1.1. Motivation

Machine Learning is used in many aspects of daily life, business and research. To fit demands of high scalability, distribution of the training is used. Distributed Machine Learning (DML) is dependable of configurations such as parallelization strategy, topologies and communication. Important is to find configurations to minimize used training time and optimize computation and communication distribution. Finding such configurations is difficult, as it depends on an expensive hardware base and a huge variety of available system choices. Optimizing choices uses many resources, such as money, time and power. To reduce these costs, simulation can be used. One simulator for research of DML is Accelerator Scaling for TRAining Simulator (ASTRA-Sim). When companies want to train their own Machine Learning (ML) model, they face similar challenges. Here Hewlett Packard Enterprise (HPE) could provide solutions such as the necessary hardware infrastructure, making those companies their customers. Additionally, they should provide associated system choices to use the system efficiently. To gain the companies as their customers, they should provide realistic insights in how long the training would take. To find and help evaluate such information, a simulator like ASTRA-Sim can be used. As it is a research tool and not designed for a sales use case like this, its usage is challenging. It's multiple versions and parameters, that need prior training and expertise, make usage for a non-expert user group like HPE customers difficult.

1.2. Problem Statement

1.3. Objectives

1.4. Structure

2. Literature and State of the Art

In this chapter, the current state of the art and literature get presented. First, the development of machine learning to distributed machine learning gets sketched. Then, ASTRA-Sim gets presented and compared to alternative DML (simulation) approaches. Lastly, User Interface (UI) and User Experience (UX) best practices, with a focus on evaluation criteria and UI for scientific tools get presented.

2.1. Development of Distributed Machine Learning

DML is a concept based on ML distributed on multiple machines. To deeply understand it, one needs to know basics of ML first.

History of Machine Learning

The concept of machines learning similar to human learning was first proposed in 1950 by Turing [1]. He proposed the first known of theoretical description of the concept that later became known as Artificial Intelligence (AI). In his paper, he introduces basics such as the idea that machines could simulate human intelligence, if they are given the right data and algorithms. He states that learning, similar to children education is central and supports it with evolutionary algorithms. He claims that machines, which can be seen as discrete-state machines, can simulate anything, which enables them to universal computational capabilities useable for ML.

Following this basic idea, the first working ML program was introduced in 1959 by Samuel [2]. Samuel presents a program that is able to play the game checkers better than its programmer, with only 8 – 10 hours of playing and information such as rules of the game. This program is based on self-improvement, it adjusts its strategy based on previous outcomes. Therefor it is the first documented “self learning” algorithm. The learn process has two approaches; one is memorizing each board positions evaluation and the other one to adjust this evaluation function based on experiences. In this version, it uses the delta between expected and actual result to update expected parameters.

At this time early ML begun and these concepts were expanded, as in 1963 Abramson presents further pattern recognition and machine learning approaches [3]. Here, statistical approaches were introduced and data was first viewed as vectors in a multidimensional space. He framed pattern recognition as a classification problem in vector spaces with two subproblems, being the partitioning techniques and choice of measurements. Also, the first idea of a distinction between supervised and unsupervised training was explained. But there were still gaps in research that only had to be discovered in the following years, like the selection of relevant features.

In the following years, many concepts of ML were revised and newly discovered. Such as backpropagation in the 80s [4], that introduced using gradient calculation for loss correction, or the big data field especially in the 2000s [5].

As ML grew and gained importance, problems regarding growing demands arose. One is the growing availability of datasets. With more data available, training once possible in a few ours might take days or even years to finish, if not optimized. The data is also more complex, forming vectors of increasing dimensions, compared to past data. Also, models are increasing in complexity, so do deep networks have trillions of parameters compared to shallow neural networks in 1990 [6]. These problems demand a solution. Parts of these problems could be solved by scaling with improved hardware as an option, as chips could be made more efficient by scaling with smaller transistors [7]. This approach, based on Moore's law [8], is not upholdable anymore, in the 2010s it was declared "dead" [9]. Naturally, the developed solution was distributing the training. DML is a subset of ML that splits the training process onto multiple Neural Processing Units (NPUs) [10]. In 2024, distributed training became the standard for large scale systems and remained the state of the art [11].

DML Characteristics

With DML a solution for scaling was introduced, and new challenges have to be solved. Distributing the machine learning process is not straightforward—it depends on many factors, such as the way of splitting, network latencies and communication strategies.

The basic DML process combines individual computation of different workers with a communication between them. That way they can train one model together by separating tasks. This separation can follow two approaches visualized in figure 2.1 by the official PyTorch website (https://docs.pytorch.org/torchrec/high-level-arch.html) [12]. One, called data parallelism, is splitting the data that the model is supposed to train on. That way, all workers iteratively train a part of the data onto the same model and communicate the models parameters and gradients between computation iterations. While data parallelism has the advantage of being easily implementable and reduce computation times nearly linearly (TODO check if true lol), its communication can create new overheads if workloads are distributed unevenly and a synchronous communication is used. Detailed on that and further advantages and disadvantages get discussed in section 2.1. The other parallelization, called model parallelism, a type being for example pipeline parallelism, is focussing on splitting the models parts onto the workers. Here, the layers of the model are divided and trained separately with according communication. This has the advantage of enabling training to include large models, but requires more complicated communication than data parallelism. Common are also hybrid approaches in which both data and models get split. This combines advantages such as a high scalability and the possibility for an efficient training performance, but new introduced challenges are finding the optimal distribution between data and model parallel strategies as well as for load balancing communication and computation efficiently [13].

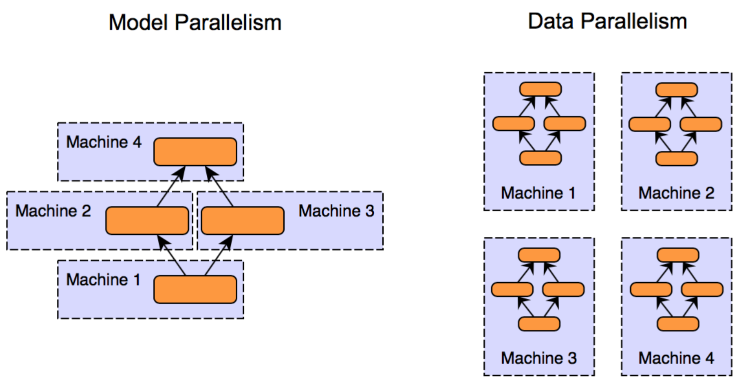

Figure 2.1.: Parallelism Strategies

Hardware Details

The workers used in the DML process, generally called NPUs, are usually Graphics Processing Units (GPUs). They can appear in high amounts, from 8 GPUs for research purposes to 1000s for Large Language Models (LLMs) like GPT-5 [14].

A set of NPUs structured together is called cluster, and each cluster is able to take different amounts of them. How they are connected in each cluster is described by a physical network topology. Every cluster in one machine, or called node, respectively, communicates via intra-node-communication. NPUs communicating between different nodes is possible over Network Interface Cards (NICs), which are connected in an inter-node-network. While intra-node communication has low latencies and high bandwidths, due to connections via NVLinks or PCIe switches, inter-node-communication has rater high latencies and low bandwidths beacause of used network interfaces like InfiniBand [15]. This would make a distributed training with few machines attractive, but realistic distribution onto many machines is much more scalable. That is due to the limited amount of GPUs in one server. Therefore, large distributed systems use and depend on both, intra and inter-node-communication.

The topologies of distributed systems can be based on different architectures. Verbreaken presents that there are four types of topologies that can exist for distributed systems [16]. Generally they can be divided into centralized, decentralized and fully distributed topologies, based on the degree of the distribution. A structural overview can be found in 2.2.

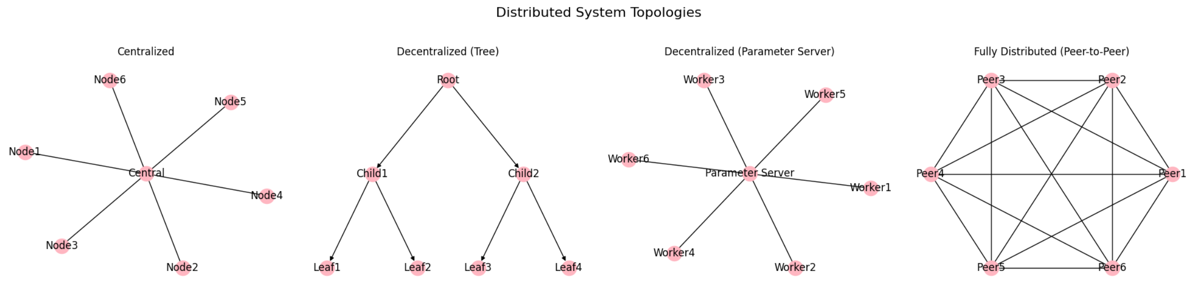

Figure 2.2.: Network Topologies

The centralized topology is characterized by a strict hierarchical structure in which all nodes are connected with one central server that orchestrates them and performs the necessary steps for the combination of the machine learning. It serves a simple coordination but is limited in scalability due to having a single point of failure.

The tree topology is a decentralized topology in which the nodes are structured hierarchical. The communication takes place between the nodes in specific directions. Information is communicated upwards and distributed downwards. This is scalable as every node only communicates with children or parents.

The parameter server topology is a second decentralized topology that combines centralized parameter servers to store and retrieve gradients with decentralized ML nodes. These servers have a shared memory, so the parameters can be synchronized between multiple parts. The disadvantage of this is that all communication happens at the parameter servers, creating bottlenecks if many workers are combined with few servers.

A fully distributed topology works like a peer-to-peer network, meaning each node holds its own copy of the models parameters and is able to communicate with all other nodes. Communication is possible in every direction and at any time. This is very scalable, as new workers can be added without a high centralized overload, but it has the disadvantage that the communication can create a high overhead if every worker broadcasts its information.

Topologies can be asymmetric or symmetric, depending on if the connections are bidirectional or unidirectional. For fully distributed systems both could be possible, while all other topologies depend on bidirectional connections [17].

Communication Strategies

While physical topologies are showing different ways of organizing and centralizing the connections of machines, logical topologies can be used to specify how the actual communication is practiced. For this two main communication strategies a distinction between two new types of parallelization has to be made. They can either be synchronous, meaning all machines communicate at the same time collaboratively, or asynchronous, where each worker communicates as soon as it is finished with its own computation. Both are not applicable to every physical topology. Asynchronous communication is best suited for decentralized topologies with parameter servers. In that case workers push and pull parameters anytime necessary without the need for waiting for other workers to finish. Possible are also more varying approaches such as a trade-off version, called local Stochastic Gradient Descent (SGD), that allows workers to communicate asynchronous for a set amount of time until needing to synchronize [16] or Stale Synchronous Parallel (SSP), which additionally allows for cached parameters to be used for synchronization for the set amount of time [18].

Depending on the parallelization strategies and split of the training process varying types of communication have to be performed. While data parallelism relies on frequent exchange of calculated gradients, for example tensor parallelism, a type of model parallelism, needs to communicate between steps as early as in the forward pass [19].

Those types of communications can be achieved by using for example communication collectives. That are a set of synchronized communication patterns, which can be used for sharing and combining information across multiple machines in distributed systems, for instance *All-Reduce*, *All-to-All* or *Reduce-Scatter*. These collectives are most common for data parallelism, as they help to synchronize and combine gradients, and share parameters. They are blocking, meaning, nodes have to wait for each other's computation to finish, and they have to be orchestrated by a centralized controller. Generally, collectives can be separated into redistributive operations, that share data and consolidative operations, that aggregate data. The most common collective used for data parallelism is *All-Reduce*. In the following the approach of collectives is explained, based on *All-Reduce* as an example. For guidance, figure 2.3 shows a visualization of the collective. The source for this explanation is the 25th course lecture of UC Berkeley's course *CS 168* in spring 2025 [20].

This collective can be used for p nodes with each having a p-sized vector of values. This example uses $p = 4$, four nodes with each having four parameters that the model gets trained on. Every node needs to know the number of p and which index they have, meaning which parameter they are going to be responsible for. Also, this example uses a ring topology for easy visualization. The actual implementation can differ, in the following the theoretical background gets explained, more on that afterward.

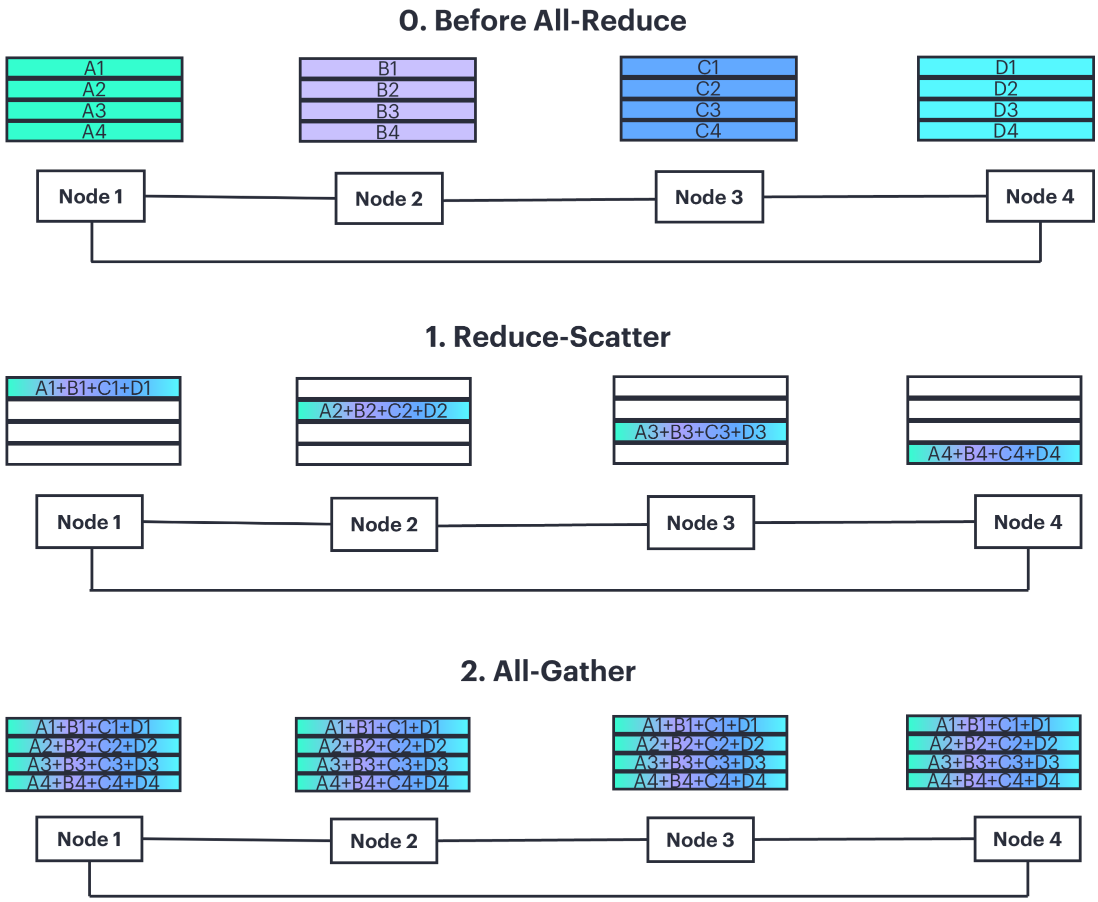

Figure 2.3.: All-Reduce

In every iteration, the communication starts when all workers, here called nodes, are finished with their computation. Every Node has their four parameters stored. The first step of the *All-Reduce* is a *Reduce-Scatter*, which itself involves two steps, a *Scatter* and a *Reduce*. *Scatter* is the redistributive operation that shares every i-th parameter to the i-th node. With this, the first node receives all first parameters, the second all second ones and so on. *Reduce* is the consolidative operation, that combines the received set of parameters to one single aggregation. Together, they make every i-th node store one value in their i-th vector-place, combining the previous i-th values in every node's vector.

The second step is an *All-Gather*, which is an advanced version of *Gather*. That is an operation that is the reverse of *Scatter*, as it combines every i-th nodes i-th value in one vector. *All-Gather* expands this by a *Broadcast*, which is the reverse of *Reduce* and distributes the resulting vector to all nodes. Afterward, the *All-Reduce* is finished, and every node has a p-sized vector of the combined and shared values.

The used operations can be implemented on top of various topologies, which influences how efficient the operation performs. In figure 2.4, common logical topologies are visualized. They can be for example a *Mesh*, *Tree*, or *Ring*.

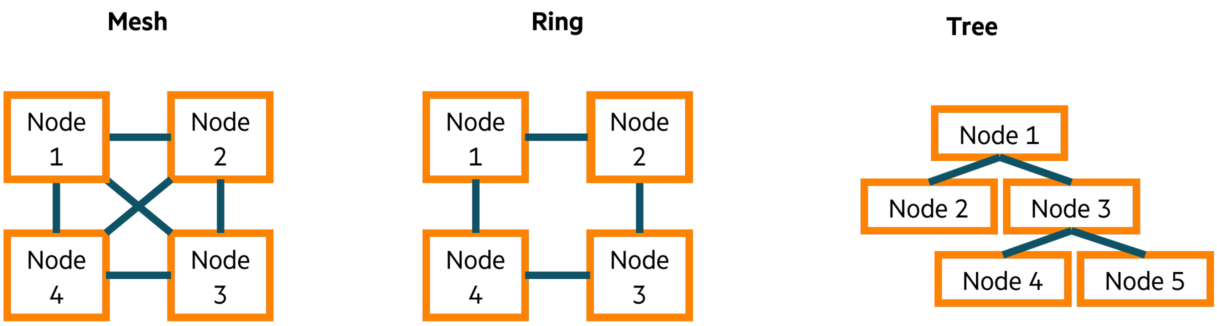

Figure 2.4.: Logical Topologies

Depending on the underlying physical topology, different logical topologies can be useful for the different collectives, as different complexities of bandwidths are relevant. Also, overlay topologies can be used to create virtual links to support exchanging data between unconnected nodes. Furthermore, topologies can be defined on multiple dimensions. Examples are a two-dimensional ring or a three-dimensional torus.

DML Challenges

With many factors to consider, the DML process allows a high range of configurations. The first challenge is the communication overhead created when synchronizing shared parameters with workers that have different efficiency in their computations. Especially in centralized topologies with synchronized communication, workers waiting for others to finish should be avoided [21]. Additionally, when increasing computation allowance between communication times, a decrease of accuracy is likely to happen, as the model is trained less cohesively [22].

Yang et al. claim that nowadays configurations of large DML systems base on decentralized architectures with asynchronous communication [22]. Challenges like those get solved by research and evaluation tests of many configurations. Such adn apporach needs to consider model accuracy, energy consumption, throughput and latency, and cost[23]. To save resources and optimize configuration finding simulators can be used.

2.2. ASTRA-Sim and Related Simulation Tools

The most extensive publicly available DML simulator with a focus on Software (SW)/Hardware (HW)-Co-Design is ASTRA-Sim [24]. Its goal is to enable researchers to model and analyze configurations of DML. It has a focus on hierarchical systems and communication for both intra- and inter-node-communication. While there are many different versions of ASTRA-Sim the following explanation focusses on ASTRA-Sim 1.0 with an analytical backend. Explanations and comparison to other versions are provided in the subsection 2.2.

ASTRA-Sim Approach and Implementation

Generally ASTRA-Sim consists of layers which enable users to configure and simulate specific input fields. The core of ASTRA-Sim features two parts. A workload layer that is used to specify the to be trained model, with expected layers of communication, and a system layer that can enable configure used collectives and scheduling policies. ASTRA-Sim is a network centered simulator, naturally an additional feature layer lets the user configure physical dependencies like topologies and bandwidths. Initially, ASTRA-Sim featured the existing Garnet network simulator [25].

The workload layer uses an external computation model, like Scale Sim [26] to calculate each layers used General Matrix Multiplications (GEMMs). Its inputs are based on the parallelization strategy *(Data, Model, Hybrid)*, size of communication and structure of layers.

The system layer implements collective operations dependent on logical topologies *(All-Reduce, All-Gather, ...)*. Other configurable dependencies are the scheduling policy *(Last-In, Last-Out (LIFO), First-In, First-Out (FIFO))* and specifics such as the amount of splits of the dataset or Logical Scheduling Queuess (LSQs), which are the active chunks processed per dimension. This layer is able to simulate different phases in the communication, like multistage collectives and real-time differences between multiple dimensions.

The network layer simulates the hardware dependencies. The analytical version is able to consider configurations for multiple dimensions too. For each dimension a topology *(Ring, Fully Connected, Switch)* with its amount of NPUs can be specified. The simulator includes latencies for links, routers and NICs and bandwidths for links. The number of links is also important. Lastly the network layer can also simulate High Bandwidth Memory (HBM), with its latency, bandwidth and memory scale.

Figure 2.5 shows the original architecture with the Garnet network simulator. The different network backends are interchangeable as they all implement the same AstraNetworkAPI.

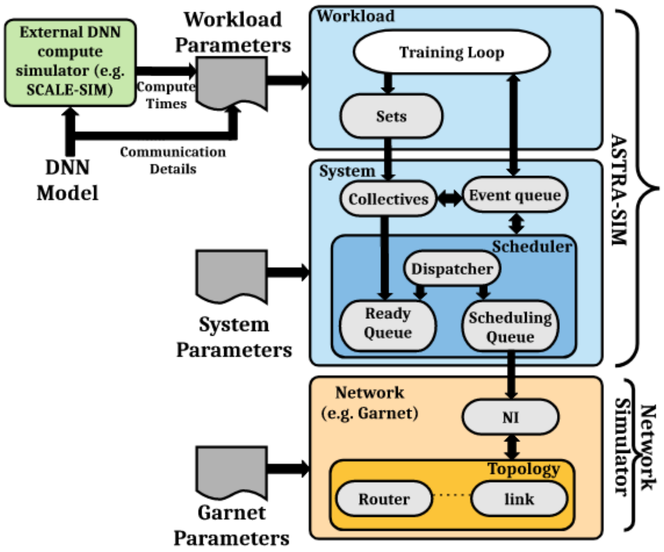

Figure 2.5.: ASTRA-sim Architecture Overview, Official ASTRA-Sim 1.0 release [24]

Each simulation iteration start with the initialization, where the user can specify their configurations. In the implementation this is separated by layer to a `network.json` file for the network, a `system.txt` file for the system and a `workload.txt` file for the workload. In the second step the workload gets produced. The loop produces sets of communication which again get separated into chunks for the processing. In the next step the chunks get sorted into LSQs and dispatched. This stream gets supervises and restarted if necessary. Next the final network simulation happens. In the analytical network the backend network gets simulated purely mathematically. Lastly, ASTRA-Sim exports its results into comma-seperated values (CSV) files. These show among other aspects the time of communication per layer, the distribution of computation vs communication and bottlenecks.

Versions and Modifications of ASTRA-Sim

Its initial release in 2020 includes a Garnet network. It was expanded later by an analytical network simulation. While Garnet is cycle accurate, so it simulates clock and packets in detail, the analytical network is based on mathematical approximate models. Therefore, the analytical model is slightly inaccurate, as it is for example not congestion aware, but it is more scalable than the garnet network.

In 2023 a second version of ASTRA-Sim was released [27]. It addresses the need for a higher scalable simulator, as deep learning models grew.

The loop is similar, with a few changes and extensions. First, the workload layer process. This workload layer was replaced with a chakra traced based workload layer. That is a graph based workload documentation [28]. So, it replaces the previous static trainings loop with a dynamic one based on Execution Traces (ETs). An ET is a graph that reproduces a real ML training process. Each of its nodes represents either a GEMM, an access of memory or a collective communication. They can for example be extracted form frameworks such as PyTorch and converted to Chakra ETs. This workload layer supports more specific parallelization strategies than the 1.0 version *(Pipeline, Hybrid-Parallel, ...)*. This represents a more realistic model of parallelization, as the NPUs are capable of executing varying operations. After the ET is read and distributed on configures NPUs, each gets its own graph to simulate. Time spend on the computation, communication and usage of storage get collected. These use other layers information.

The system layer still manages the collective operations, depending on topologies and scheduling strategies. A relevant difference to 1.0 is input got reduced to the only necessary parameters, ones reflected by chakra traces were crossed out. As a result it only features a topology, number of NPUs, internal bandwidth and internal latency. The idea is to completely abandon the inputs and retrieve them from chakra ET too.

For network simulation the network layer was used, and the two existing implementations were revised and additional backends added. The analytical backend is still implemented as a scalable and fast approximate network simulator, but the Garnet backend was abandoned. Now instead the Network Simulator 3 (NS-3) [29] and the High Throughput Simulator (HTSim) [30] backend can be used for the network simulation. The NS-3 is a well established network simulator, that includes realistic network conditions with congestion and routing, which makes it suitable for simulation that needs to be very accurate and realistic. HTSim rather focusses on the integration of Ultra Ethernet, an alternative to the standard Transmission Control Protocol (TCP). It is more lightweight and an alternative to the NS-3 that includes view of network traffic as flows rather than individual packets, which is especially relevant for DML setups. Both are more accurate than the analytical model.

The 2.0 version also includes a new layer, called memory layer. It is connected via an own memory Application Programming Interface (API). This models the HBM previously managed by the system layer. It supports disaggregated memory architectures and realistic pipelines transfers with their according bandwidths, latencies and topologies. That way ASTRA-Sim is able to compare these different known memory architectures depending on underlying DML configurations.

In general, ASTRA-Sim 2.0 is more flexible that 1.0, by allowing many more configurations and realistic workloads. Its disadvantage is, that it is still being developed and changes are introduced. For a stable simulation 1.0 is the better choice, but in the future and for current research 2.0 will be more relevant.

Comparison to Further Simulation Tools

A specialty of ASTRA-Sim is that it supports hierarchical topologies, so it can model asymmetrical bandwidths. Also, it allows a differentiation between logical and physical topologies. This is especially relevant for research contexts. Additionally, it is easily extendable as layers can be exchanged and new factors such as new topologies or strategies can easily be added.

2.3. User Interface Design for Scientific Tools

Research tools in computer science are mostly based on Command-line Interfaces (CLIs). Including Graphical User Interfaces (GUIs) can increase use cases and amount of usage [31]. While UI refers to the visual and interactive design of a digital product, UX refers to the entire experience users go through when using digital tool. Both areas are highly interconnected and central for the users' satisfaction with a digital product [32]. The study of UI and UX is Human-Computer Interaction (HCI), which in return gives it a scientific base. In the following, design principles of UIs in general and in this projects specific context get explained and the current state of the art presented.

Human-Computer Interaction Foundations

HCI is an interdisciplinary area that deals with the design, evaluation and implementation of interactive computer systems [33]. Its goal is to improve these systems in their functionality and usability. Specific goals also include methods for designing and implementing user-friendly UIs as well as criteria for evaluating and comparing interfaces. HCI also focusses on the development of new interaction techniques and models to human machine interactions.

This discipline was relevant from the moment on the first electronic computer in 1946 was used, called the Electronic Numerical Integrator and Computer (ENIAC) [34], which was used for computation of military firing tables. Later, with the development of GUIs, the internet and a wide base of useable tools, HCI became an important field, also relevant for website design.

HCI claims that machines are worthless until they are properly useable. Two main aspects are the functionality and usability of machines. Functionality refers to the set of services a program provides. This value becomes only value if it is useable efficiency. Usability is the extent of which the system helps the users to activate their goals when using the machine. It depends on the needs of the target user group. Generally, in HCI a system is only considered qualitative, if functionality and usability are balanced.

One important sub category of HCI is cognitive load. Kosch et al. [35] present its characteristics and measurements in the following way. Defining cognitive workload is a challenge, it describes the amount of complexity the human brain gets presented with when executing tasks. In the field of website GUI design this can be transferred as the complexity of brain processes a user goes through to navigate the website. Naturally, they should be minimized to ensure the least possible complexity when using the website. Because digital systems are getting increasingly more complex, measurements of cognitive workloads gets increasingly more difficult. They present metrics such as questionnaires, which are very subjective, physiological sensors which are very objective or presented behavior, like the users mouse activity. With those evaluations different layouts or navigation structures can be tested and compared.

Another important sub-field of HCI are feedback loops. Feedback is the direct response of a system to a users action and is central for users, especially when they interact with computers in the wrong way. In GUI design for example when entering values in a form or uploading files that are not meeting specific criteria, it needs to be made sure that users understand what went wrong and to give them feedback to reduce the possibility of this problem occurring again. No feedback, delayed feedback and wrong feedback can make the user feel frustrated and unable to take full advantage of functionalities. Pérez-Quiñones and Sibert present a foundational study on the feedback model based on the linguistic theory of States of Understanding (SOU) [36]. That is a model explaining which state humans think they are in during a conversation, based on expectations they have when communicating with each other. Instead of seeing communication as ping-pong iterations between two participants, here it is viewed as collaborative process based on Clark et Al. [37]. Basically, it includes four states a listener can be in after a speaker send a message. This original concept from the linguistic area was applied onto the HCI area. The resulting five Simulated States of Understanding (SSOU) represent states needed in communication between humans and computers to match the humans expectations. The stages are:

1. The system is ready to receive inputs,
2. The user entered an input and the system processes that,
3. The result gets reported by the system,
4. The system ignores the input,
5. The system stores the inputs for a later processing.

For every stage the computer needs to provide specific feedback, so the user feels informed, and unnecessary interactions can be reduced.

UI/UX Best Practices and Evaluation Metrics

One aspect that needs to be considered is the difference that target user groups bring with them. One technique to characterize user groups is by using personas [38]. Using personas is supposed to help developers understand expectations, goals and contexts of users, to optimize the end product for the user. That way the user is central for the design process and can specifically be targeted. How efficient personas are depends on the quality of data it is based on and the depth of them. There are many possible strategies of building personas based on existing datasets. If there is no dataset available, an approach including expert interviews, existing knowledge and assumptions have to be used [39]. Each personas specification should include information such as daily routines, goals, fears, attitudes, technology behaviour, demographics, and existing knowledge. Personas can be used to design specific usability tests or specifications of necessary functionality. Combined with HCI it can be used to evaluate the necessity of certain aspects such as the extent of necessary feedback.

One classification that could be made for users is whether they are experts or non-experts in the field of the website. This is especially important for scientific tools, as they deal with specific areas that are not relevant in daily life of most people. An expert in the field of usability and HCI is Nielsen, that states that intuitiveness is central for website usage for novice users [40]. Even though his post was publicized in 2000, it still holds relevance today, but has to be regarded with conciseness. He states that the interface has to be differentiated between expert and novice users. While non-experts should be getting training-wheel designs to get guidance of usage, experts should get interfaces that provide a powerful tool set. Also, the development from novices to experts can be supported but need to be observed with usability tests.

A recent approach of methodically revising and restructuring UI recommendations was done by Diehl et Al in 2022[41]. The method included expert interviews and a content analysis with a revalidation of an expert group. The resulting 69 general recommendations can be divided into 12 categories. These are feedback, recognition, flexibility, customization, consistency, error handling, help, accessibility, navigation, privacy, visual design and emotional design. The entire list can be found in the appendix A. The recommendations are closely connected to presented HCI approaches. Mentionable recommendations for instance also include that systems should enable users to easily recognize user functions (recognition, rule 3), it should ensure consistency in appearance (consistency, rule 1), it may concentrate the information mainly in the center of the screen (visual, rule 2), and it should provide the options and information in a logical sequence (navigation, rule 1). Further relevant rules are presented in chapter 3.

3. Design

ASTRA-Sims versions allow differentiated research, depending on goals and configurations. The to be developed UI should make one suitable version compatible with the usage of novice users such as HPE presales. Best practices of UI/UX design should be considered while using an existing version of ASTRA-Sim. To optimize inputs and outputs of it, knowledge of DML needs to be known and referenced.

3.1. Requirements Analysis

To understand the broad description of *HPE Presales* for the High-Performance Computing (HPC) infrastructure direction, a simple persona was created. As there was no data about this target group available, is the persona based on 1. expert knowledge gained by interviews with employees with more background knowledge and 2. assumptions about non-experts in the DML field. The website will be designed based on this persona and with user tests later on adjusted.

The designed persona is "Angelina B. Smith". She is a 34 year old employee in the HPC department. More detailed information can be found in the appendix B, here the most relevant information gets presented. She got a bachelors degree in business informatics and has been in her role as a "Presales Specialist for AI HPC Infrastructure" for 2 years. Her goals are to convince customers with infrastructure proposals that suit their expectations, communicate the complex technical concepts clearly and build trust with the customers. At the same time she has the fear to overwhelm customers and not understanding new tools. Tools she does use currently are Excel and PowerPoint, and she likes working remotely and in the office, with her laptop but also smartphone. She has an example knowledge of Information Technology (IT) infrastructure and network topologies, but no deep knowledge of simulation tools or DML.

Based on the persona perspective the primary identified aspects of inputs are:

- *Understandable*: The user needs to understand easily what the input means that they are entering

- *Easily Reproducible*: A user should be able to reapply known experiments and modify them for easy comparison. This is important for the use case and a feature the CLI based version of ASTRA-Sim includes.
- *Accessible*: The inputs should be easily identifiable as clickable and the values should be easily inputable. Ideally would be if the user gets an implicit guidance through the inputs even if they have not much expertise.

And the outputs should be:

- *Understandable*: The user should be able to understand the results at the first view, without detailed prior knowledge on the topic and on the visualizations. This is especially important as visualizations might be used for discussions with customer, which might not have a background in the DML field.
- *Easily Exportable*: Similarly to inputs should outputs be able to be exportable to include them in presentations or process them manually in the current workflow with Excel files.

3.2. Technologies

The to be developed UI will be part of a full stack web application. The choices of technology are based in two ways. One, to fulfill UI best practices and two, be suitable for the extent of the project.

The interior design of the website was made with Figma. That is a tool that supports early detailed wireframes, which are necessary for early user feedback and evaluation. The frontend implementation includes a TypeScript React framework based on Vite and node.js. This setup was chosen as it supports component based development with grommet and type checking. For a consistent experience within the product but also next to existing HPE websites, the components are based on Grommet. That provides pre-configured components all reflecting the branding, and look and feel of HPE webpages.

The backend is a Flask server based on Python. That allowed a lightweight API, which purposes it is to be the pipeline between the UI and ASTRA-Sim. Pythons right library availability also allowed for plots of CSV outputs to be created and send to the frontend.

To complete the setup, the frontend and backend are organized via docker containers. This allows including different versions of ASTRA-Sim in the future with minimal effort.

3.3. UI/UX Design Process

There were two main approaches to design the general UI. The first was to use a multistep form, called a wizard. In it the user gets guided through the necessary input parameters and gets support for understanding what they mean. This has the advantage that user exactly knows what is expected from them. They cannot forget to enter specific information and the inputs are easily reproducible. Also, compared to a single page form, the user gets less overwhelmed by the amount and diversity of available parameters. The alternative idea was a natural language interface in which the user can enter questions about specific DML configurations and gets general results. This idea was inspired by AskCricinfo, a Natural Language Processing (NLP) based web-based information retrieval system for cricket games[1]. That brings the advantage of giving the user a high level of accessibility and an easy way of learning.

Ultimately, it was decided against the NLP based idea as it does not give the user enough guidance for the usage of the tool and is not suitable for the requirement of not overloading a users' cognitive load, as it requires a high amount of self thinking and knowledge to use such a tool in a complex area like DML. A wizard was chosen with the potential opening to an additional NLP input field in the future.

The design process was developed iteratively. First, a general draft for the page navigation was proposed and evaluated. Then more specific wireframes have been designed and were iteratively improved in two iterations. The feedback in both rounds was collected in a free format, which allowed feedback givers not to feel limited to certain aspects but of course is not scientifically holdable. The first round of feedback was provided by employees in the HPE Labs who can be considered familiar or experts with ASTRA-Sim and know how to use it. The second round was done in the environment of an intern fair in which United States of America (US) HPE employees were able to get informed about current projects and are able to give early feedback.

The first draft of the general website can be found in figure 3.1. It shows the navigation next to how one page of the wizard could have been structured.

This page features roughly five components that have to be mentioned. A header, a headline, the input fields, a file upload and a user guide. The header that is displayed holds three tabs, being the simulation configuration, the results and an additional training page for getting more background information. This reflects the two main purposes the page serves to simplify usage of ASTRA-Sim, with simplifying inputs, and outputs and additionally providing direct access dor guidance for non-experts.

[1] https://www.espncricinfo.com/ask

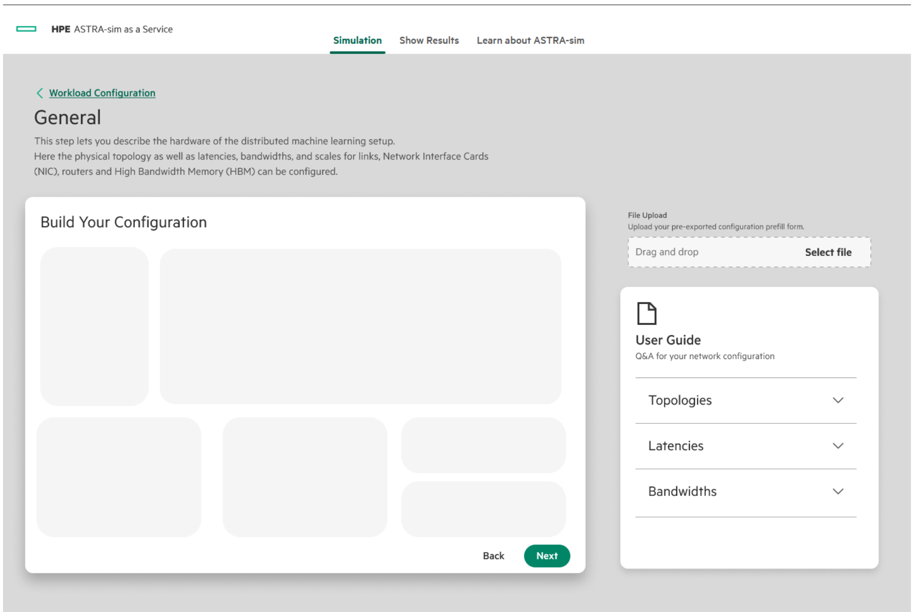

Figure 3.1.: General Wireframe v1.0

The headline on top shows which step of the wizard the user currently is at. It also leaves room for a short explanation and a button to the previous page, because users might expect that in the top left corner. The input fields are individual by what page the user is at and which inputs they gave before. In this wireframe version this page is located on the left, implicitly getting attention before the last two components but not getting the full focus. The file input was decided to put on every wizard page to allow uploads for every layer directly which makes it easy to directly visualize and modify inputs.
The user guide, located on the right down corner was supposed to contain descriptions of the input fields for the user to click if necessary.

For a more specific view the network page of this base is displayed in figure 3.2. It includes parameters for the number of dimensions, possible topologies for each dimension and configurations for links like their number, bandwidth and latency. That way it displays ASTRA-Sims' inputs for the network layer excluding HBM details.

This setup includes help for the user for example by providing pictures of the available topologies and a dynamic way of adding in dimensions. Additionally, it includes an information button for every input subcategory, that opens a descriptive popup. That aims at increasing flexibility between experts and novice users by subtilely supporting novices learn process.

Problems with that setup can also be seen at first glance. Fitting all inputs of one layer of ASTRA-Sim is not possible with the space available in the chosen wizard layout. Also, the connection between the headline and descriptive text to the wizard content as well as the connection between the wizard and the user guide is not intuitively noticeable. One further critic point is that the link inputs are located to the right of the topology inputs, making them

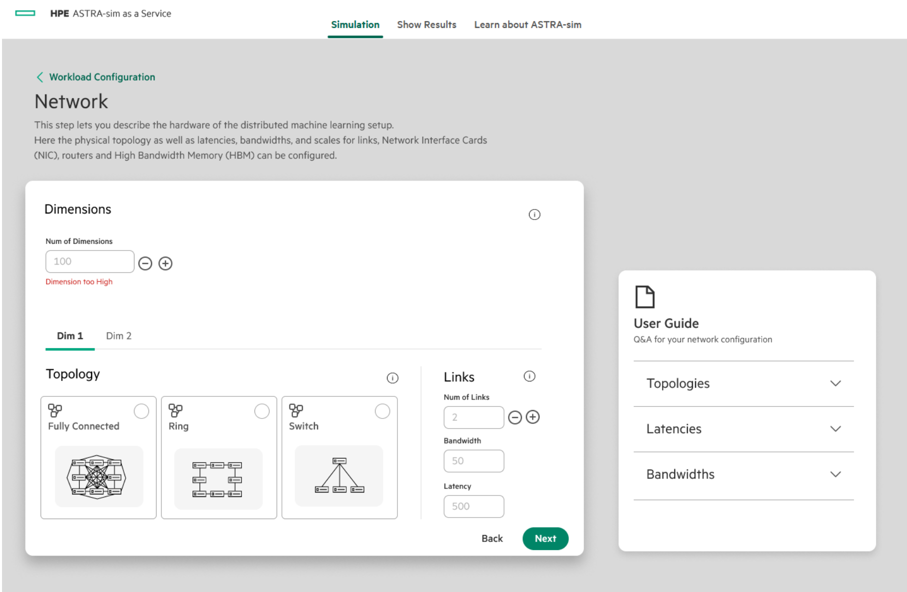

Figure 3.2.: Specific Wireframe Network v1.0

easily overlooked, because they are not below the topologies where the user would expect the to be.

While this initial draft has the strength of supporting users' necessity of help and gives them direct access to many features it lacks one specific but very important criteria, being to not overload the users' cognitive load. For example with adding the file upload field on the same page as the inputs, the user could already be confused and challenged by the system. It might not be easy to understand that the file upload is optional, and the user might be confused by the structural logic of needing to click through all pages to upload specific files, a task that is repetitive and not to the point.
Therefore, if evaluated by the 69 rules of UI design, as presented in chapter 2, it actively disregards the rules 1, 3, 4 of the recognition category, rule 2 of flexibility, rule 2 of consistency, and rule 1 of navigation.

To improve this design two steps were taken. First, all inputs were revised and newly categorized and second, the general page structure was restructured. The first step was necessary, because the full set of 25 input parameters is rather too much for the user and therefore only the most influential and target group specific parameters should be asked for. The second step naturally follows from presented challenges to reduce navigation complexity and provide a more accessible navigation.

The full classification of inputs can be found in the appendix C. The 22 inputs of ASTRA-Sim plus additional three parameters that are included within the `workload.txt` file were collected and prioritized. This priority was estimated based on how huge the influence of certain parameters is compared to others.

Therefore, parameters such as *topologies per dimension* were considered important, because physical topologies can make huge differences in DML setups and are important for the user to configure, while inputs such as *preferred dataset splits* are considered rather unimportant as it is only small influential factor, that can also be easily estimated by default factors of other parameters such as the provided workload.
Next, it was identified which inputs could be made more accessible for the user with only little prior knowledge. These include the workload parameters, which are usually depending on which specific model should be trained. If we assume the target user does not know specifics such as the workload, they could for example choose one prior configured model type whose parameters can be estimated.

Next, the identified parameters were newly categoriesed, as the ASTRA-Sims mapping is in parts misleading.
For example is HBM misleading to include in the network section, as it functions as an accelerator, not a primary part of the network.
Three new categories were identified. First, the network, in itself separated into general network settings and link specifics; workloads and the associated collectives and additional accelerators. For a logical sequence that feels natural for the user the sequence in figure 3.3 gets proposed.

Figure 3.3.: Wizard Page Flow

The user gets to choose general information such as the workload model or a configuration of the history of their previously simulated runs. The network category is the first thematically page as the users main knowledge lays in that field. This aims at not overwhelming the user as they feel familiar with inputs. That makes the first rule of the recognition category apply. Next following is the category of links, as that is a subcategory of the network and therefore connects seamlessly.
Next, the workload gets specified. That choice was made, because this page mainly focusses on specifying logical topologies. Therefore, it makes sense to put it as closely as possible to the physical topology to make the navigation back to the physical topology to check the difference as easily as possible.
Then, the accelerator is added to give it the feel of configuring additional hardware to make the DML training process faster. Lastly, a wizard page that sums up all inputs was added

too. That is a wizard best practice that provides an easy input confirmation to reduce user caused simulation miscalculations.

Figure 3.4 shows the final wireframe of a wizard page on the same example, which is the network category.

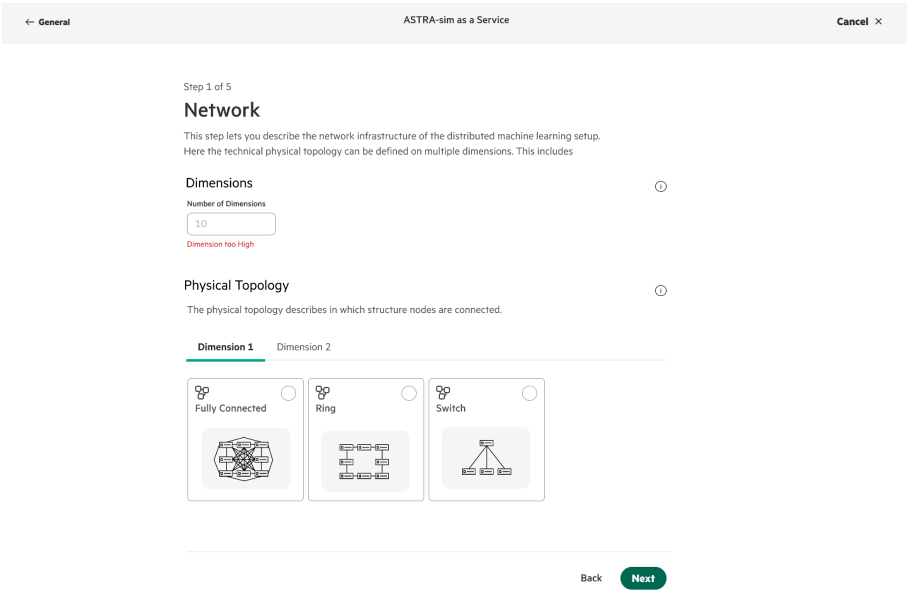

Figure 3.4.: Specific Wireframe Network v2.0

The first improvement made was to change the subpage wizard to a full-page wizard. Not only does this comply with HPE internal branding guidelines, but it also makes the user focus all their attention to the most important steps. That solves the issue of rule 4 of the recognition being violated before.

Other details were changed, like a specific feedback if users break rules set by the UI. That way, for every parameter, a set of allowed parameters was configured to guide the user and make it hard for them to input unuseful configurations. For instance, the upper limit for configurable dimensions was 5, as dimensions up to 3 are expected, but the user should get a little upper flexibility for more complex setups. Higher dimensions do not make sense for a realistic simulation and can potentially overwhelm the user, as with every added dimension, a linear increasing amount of additional input fields get added.

Another improvement for this setup is, that all wizards inputs are arranged under each other, never next to each other, except for choices of one input. That ensures predictability of the wizard and an understanding for the user, which is important for the rules of the category navigation.

4. Implementation

4.1. Optional: Overall Architecture

Figure 4.1 shows the applications' infrastructure. The backend is able to include multiple ASTRA-Sim versions by building them from inside the container. The API gets a copy of the built version and can use it from that point on. By exchanging the to be cloned repository from ASTRA-Sim 1.0 to another, the build version gets exchanged. That way the application stays modular which is important for future updates to new versions because of new use cases or more wanted accuracy.

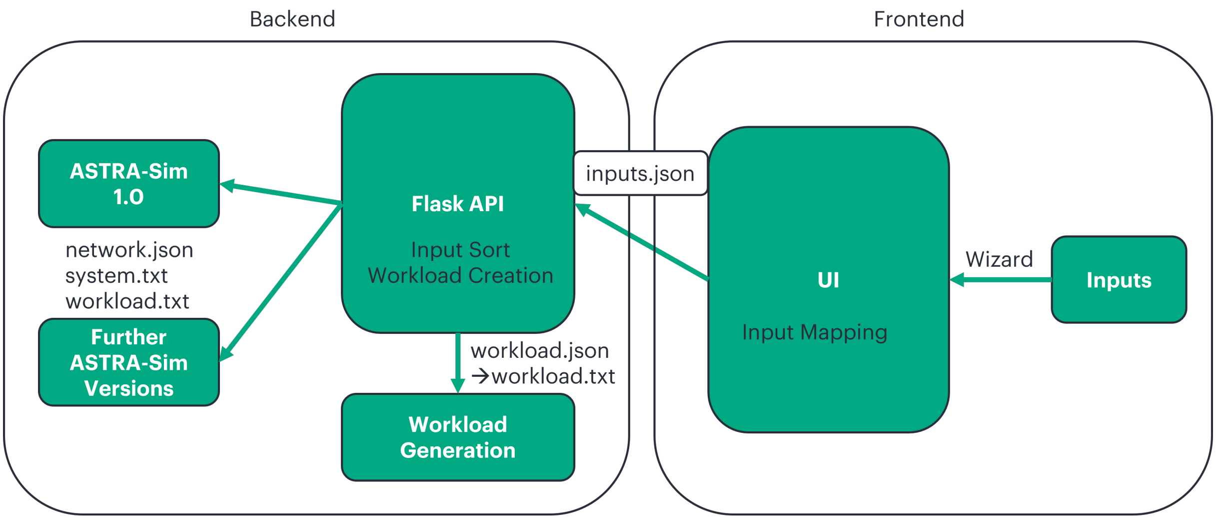

Figure 4.1.: General Structure Overview

The API takes an input of one big JavaScript Object Notation (JSON) file, that specifies all of ASTRA-Sims' parameters. That allows retrieving the parameters form the frontend in a summed up way. Based on configurations depending on the ASTRA-Sim version, different parameters for the three (or five) input files can be specified. The API also takes the responsibility of generating the final wanted files. Because the different versions depend on varying input fields this can be adjusted in the baked by changing the algorithm of generating the files. Naturally, the frontend can support any structure of inputs and is not fixed to ASTRA-Sim inputs with this data pipeline. The frontend takes inputs from the user, here in form of a wizard and maps them to necessary ASTRA-Sim inputs.

4.2. Frontend Design

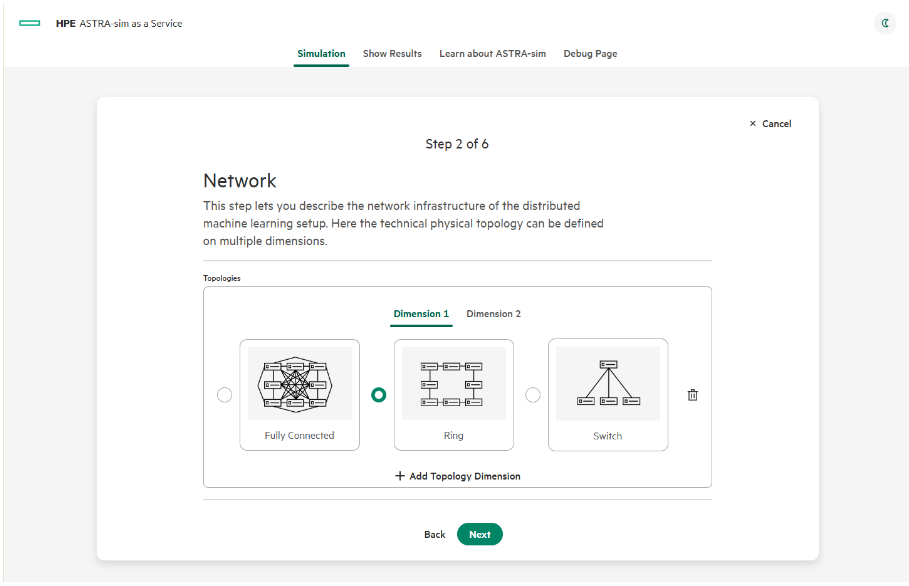

Figure 4.2.: Final Implementation Network Page

4.3. Backend Integration

5. Evaluation

5.1. Feedback Integration

5.2. Challenges

5.3. Solutions

6. Conclusion and Future Work

6.1. Summary of Findings

6.2. Limitations

6.3. Next Steps

6.4. Future Work

Bibliography

[1] A. M. Turing. “Computing machinery and intelligence (1950).” In: *Perspectives on the computer revolution*. USA: Ablex Publishing Corp., June 1989, pp. 85–107. ISBN: 978-0-89391-369-4. (Visited on 08/16/2025).

[2] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers.” In: *IBM Journal of Research and Development* 3.3 (July 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: https://ieeexplore.ieee.org/document/5392560 (visited on 08/16/2025).

[3] N. Abramson, D. Braverman, and G. Sebestyen. “Pattern recognition and machine learning.” en. In: *IEEE Transactions on Information Theory* 9.4 (Oct. 1963), pp. 257–261. ISSN: 0018-9448. DOI: 10.1109/TIT.1963.1057854. URL: http://ieeexplore.ieee.org/document/1057854/ (visited on 08/16/2025).

[4] Raúl Rojas. “The Backpropagation Algorithm.” en. In: *Neural Networks: A Systematic Introduction*. Ed. by Raúl Rojas. Berlin, Heidelberg: Springer, 1996, pp. 149–182. ISBN: 978-3-642-61068-4. DOI: 10.1007/978-3-642-61068-4_7. URL: https://doi.org/10.1007/978-3-642-61068-4_7 (visited on 08/17/2025).

[5] Ignacio Perez Karich and Simon Joss. “Emergence and Evolution of ‘Big Data’ Research: A 30-Year Scientometric Analysis of the Knowledge Field.” en. In: *Metrics* 2.3 (Sept. 2025). Publisher: Multidisciplinary Digital Publishing Institute, p. 15. ISSN: 3042-5042. DOI: 10.3390/metrics2030015. URL: https://www.mdpi.com/3042-5042/2/3/15 (visited on 08/17/2025).

[6] Věra Kůrková. “Limitations of Shallow Networks.” en. In: *Recent Trends in Learning From Data*. Ed. by Luca Oneto et al. Vol. 896. Series Title: Studies in Computational Intelligence. Cham: Springer International Publishing, 2020, pp. 129–154. ISBN: 978-3-030-43882-1 978-3-030-43883-8. DOI: 10.1007/978-3-030-43883-8_6. URL: http://link.springer.com/10.1007/978-3-030-43883-8_6 (visited on 08/18/2025).

[7] Jaime Sevilla et al. “Compute Trends Across Three Eras of Machine Learning.” en. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. arXiv:2202.05924 [cs]. July 2022, pp. 1–8. DOI: 10.1109/IJCNN55064.2022.9891914. URL: http://arxiv.org/abs/2202.05924 (visited on 08/18/2025).

[8] Gordon Moore. “Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.” In: *Solid-State Circuits Newsletter, IEEE* 11 (Oct. 2006), pp. 33–35. DOI: 10.1109/N-SSC.2006.4785860.

[9] Thomas N. Theis and H.-S. Philip Wong. “The End of Moore’s Law: A New Beginning for Information Technology.” In: *Computing in Science & Engineering* 19.2 (Mar. 2017), pp. 41–50. ISSN: 1558-366X. DOI: 10.1109/MCSE.2017.29. URL: https://ieeexplore.ieee.org/abstract/document/7878935 (visited on 08/18/2025).

[10] R. Bekkerman, M. Bilenko, and J. Langford. *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011. ISBN: 978-1-139-50190-3. URL: https://books.google.com/books?id=9u0gAwAAQBAJ.

[11] Wenxue Li et al. “Understanding Communication Characteristics of Distributed Training.” en. In: *Proceedings of the 8th Asia-Pacific Workshop on Networking*. Sydney Australia: ACM, Aug. 2024, pp. 1–8. ISBN: 979-8-4007-1758-1. DOI: 10.1145/3663408.3663409. URL: https://dl.acm.org/doi/10.1145/3663408.3663409 (visited on 08/18/2025).

[12] *TorchRec High Level Architecture — TorchRec 1.0.0 documentation*. URL: https://docs.pytorch.org/torchrec/high-level-arch.html (visited on 08/24/2025).

[13] Matthias Boehm et al. “Hybrid parallelization strategies for large-scale machine learning in SystemML.” In: *Proc. VLDB Endow.* 7.7 (Mar. 2014), pp. 553–564. ISSN: 2150-8097. DOI: 10.14778/2732286.2732292. URL: https://doi.org/10.14778/2732286.2732292 (visited on 08/16/2025).

[14] Spheron Network. *How Much GPU Memory is Required to Run a Large Language Model?* en. Sept. 2024. URL: https://blog.spheron.network/how-much-gpu-memory-is-required-to-run-a-large-language-model-find-out-here (visited on 08/24/2025).

[15] Adithya Gangidi et al. “RDMA over Ethernet for Distributed Training at Meta Scale.” en. In: *Proceedings of the ACM SIGCOMM 2024 Conference*. Sydney NSW Australia: ACM, Aug. 2024, pp. 57–70. ISBN: 979-8-4007-0614-1. DOI: 10.1145/3651890.3672233. URL: https://dl.acm.org/doi/10.1145/3651890.3672233 (visited on 08/24/2025).

[16] Joost Verbraeken et al. “A Survey on Distributed Machine Learning.” In: *ACM Comput. Surv.* 53.2 (Mar. 2020), 30:1–30:33. ISSN: 0360-0300. DOI: 10.1145/3377454. URL: https://dl.acm.org/doi/10.1145/3377454 (visited on 08/16/2025).

[17] William Won et al. “TACOS: Topology-Aware Collective Algorithm Synthesizer for Distributed Machine Learning.” In: *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. ISSN: 2379-3155. Nov. 2024, pp. 856–870. DOI: 10.

1109/MICRO61859.2024.00068. URL: https://ieeexplore.ieee.org/abstract/document/10764470 (visited on 08/16/2025).

[18] Zhenheng Tang et al. *Communication-Efficient Distributed Deep Learning: A Comprehensive Survey*. arXiv:2003.06307 [cs]. Sept. 2023. DOI: 10.48550/arXiv.2003.06307. URL: http://arxiv.org/abs/2003.06307 (visited on 08/16/2025).

[19] Harry Dong et al. *Towards Low-bit Communication for Tensor Parallel LLM Inference*. arXiv:2411.07942 [cs]. Nov. 2024. DOI: 10.48550/arXiv.2411.07942. URL: http://arxiv.org/abs/2411.07942 (visited on 08/21/2025).

[20] : Sylvia Ratnasamy, Rob Shakir, and Kao Peyrin. *Collective Operations*. en. Lecture. Berkeley, CA, 2025. URL: https://textbook.cs168.io/beyond-client-server/collective-operations.html (visited on 08/21/2025).

[21] Vivienne Sze et al. "Hardware for machine learning: Challenges and opportunities." In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*. ISSN: 2152-3630. Apr. 2017, pp. 1–8. DOI: 10.1109/CICC.2017.7993626. URL: https://ieeexplore.ieee.org/abstract/document/7993626 (visited on 08/16/2025).

[22] Aiqiang Yang et al. "Balancing communication overhead and accuracy in compression integration: a survey." en. In: *The Journal of Supercomputing* 81.8 (June 2025), p. 964. ISSN: 1573-0484. DOI: 10.1007/s11227-025-07451-z. URL: https://doi.org/10.1007/s11227-025-07451-z (visited on 08/23/2025).

[23] Seunghak Lee et al. "On Model Parallelization and Scheduling Strategies for Distributed Machine Learning." In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/hash/186b3d044a8c9898679d98dbd0d9b860-Abstract.html (visited on 08/16/2025).

[24] Saeed Rashidi et al. "ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms." In: *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Aug. 2020, pp. 81–92. DOI: 10.1109/ISPASS48437.2020.00018. URL: https://ieeexplore.ieee.org/document/9238637 (visited on 08/23/2025).

[25] Niket Agarwal et al. "GARNET: A detailed on-chip network model inside a full-system simulator." In: *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 33–42.

[26] Ananda Samajdar et al. *SCALE-Sim: Systolic CNN Accelerator Simulator*. arXiv:1811.02883 [cs]. Feb. 2019. DOI: 10.48550/arXiv.1811.02883. URL: http://arxiv.org/abs/1811.02883 (visited on 08/23/2025).

[27] William Won et al. “ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-model Training at Scale.” In: *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. arXiv:2303.14006 [cs]. Apr. 2023, pp. 283–294. DOI: 10.1109/ISPASS57527.2023.00035. URL: http://arxiv.org/abs/2303.14006 (visited on 08/23/2025).

[28] Srinivas Sridharan et al. *Chakra: Advancing Performance Benchmarking and Co-design using Standardized Execution Traces*. arXiv:2305.14516 [cs]. May 2023. DOI: 10.48550/arXiv.2305.14516. URL: http://arxiv.org/abs/2305.14516 (visited on 08/23/2025).

[29] George F. Riley and Thomas R. Henderson. “The ns-3 Network Simulator.” en. In: *Modeling and Tools for Network Simulation*. Ed. by Klaus Wehrle, Mesut Güneş, and James Gross. Berlin, Heidelberg: Springer, 2010, pp. 15–34. ISBN: 978-3-642-12331-3. DOI: 10.1007/978-3-642-12331-3_2. URL: https://doi.org/10.1007/978-3-642-12331-3_2 (visited on 08/23/2025).

[30] Maciej Besta et al. *Towards Million-Server Network Simulations on Just a Laptop*. arXiv:2105.12663 [cs] version: 1. May 2021. DOI: 10.48550/arXiv.2105.12663. URL: http://arxiv.org/abs/2105.12663 (visited on 08/23/2025).

[31] Harini Sampath, Alice Merrick, and Andrew Macvean. “Accessibility of Command Line Interfaces.” In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21. New York, NY, USA: Association for Computing Machinery, May 2021, pp. 1–10. ISBN: 978-1-4503-8096-6. DOI: 10.1145/3411764.3445544. URL: https://dl.acm.org/doi/10.1145/3411764.3445544 (visited on 08/23/2025).

[32] Nesi Hamidli. “Introduction to UI/UX Design: Key Concepts and Principles.” In: (Mar. 2023). URL: https://www.academia.edu/98036432/Introduction_to_UI_UX_Design_Key_Concepts_and_Principles (visited on 08/24/2025).

[33] Gaurav Sinha, Rahul Shahi, and Mani Shankar. “Human Computer Interaction.” In: *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. ISSN: 2157-0485. Nov. 2010, pp. 1–4. DOI: 10.1109/ICETET.2010.85. URL: https://ieeexplore.ieee.org/abstract/document/5698279 (visited on 08/23/2025).

[34] H.H. Goldstine and A. Goldstine. “The Electronic Numerical Integrator and Computer (ENIAC).” In: *IEEE Annals of the History of Computing* 18.1 (1996), pp. 10–16. ISSN: 1934-1547. DOI: 10.1109/85.476557. URL: https://ieeexplore.ieee.org/document/476557 (visited on 08/23/2025).

[35] Thomas Kosch et al. "A Survey on Measuring Cognitive Workload in Human-Computer Interaction." en. In: *ACM Computing Surveys* 55.13s (Dec. 2023), pp. 1–39. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3582272. URL: https://dl.acm.org/doi/10.1145/3582272 (visited on 08/23/2025).

[36] Manuel A. Pérez-Quiñones and John L. Sibert. "A collaborative model of feedback in human-computer interaction." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '96. New York, NY, USA: Association for Computing Machinery, Apr. 1996, pp. 316–323. ISBN: 978-0-89791-777-3. DOI: 10.1145/238386.238535. URL: https://dl.acm.org/doi/10.1145/238386.238535 (visited on 08/23/2025).

[37] Herbert Clark. *Grounding in communication.* 1991. URL: https://awspntest.apa.org/record/1991-98452-006 (visited on 08/24/2025).

[38] Jane Billestrup et al. "Persona Usage in Software Development: Advantages and Obstacles." en. In: (2014).

[39] John Pruitt and Jonathan Grudin. "Personas: practice and theory." en. In: *Proceedings of the 2003 conference on Designing for user experiences*. San Francisco California: ACM, June 2003, pp. 1–15. ISBN: 978-1-58113-728-6. DOI: 10.1145/997078.997089. URL: https://dl.acm.org/doi/10.1145/997078.997089 (visited on 08/24/2025).

[40] Jakob Nielsen. *Differences Between Novice and Expert Users*. 2006. URL: https://www.nngroup.com/articles/novice-vs-expert-users/.

[41] Ceci Diehl et al. "Defining Recommendations to Guide User Interface Design: Multimethod Approach." en. In: *JMIR Human Factors* 9.3 (Sept. 2022), e37894. ISSN: 2292-9495. DOI: 10.2196/37894. URL: https://humanfactors.jmir.org/2022/3/e37894 (visited on 08/16/2025).

List of Listings

A. Full List of Modern UI Requirements

B. Persona Details

Demographics

- Angelina B Smith
- 34 years old
- American citizen by birth
- Speaks English
- Located in San Jose

Job

- Presales Specialist for AI & HPC Infrastructure
- HPE
- Bachelor's degree in Business Informatics

Daily Routine

- 08:00 Check emails, prioritize customer inquiries
- 09:00 Coordinate with sales and technical architects
- 10:30 Prepare solution proposals for customer projects
- 12:00 Lunch break, often with colleagues
- 13:00 Customer meetings, presentations, technical demos
- 15:00 Research new technologies and internal tools
- 17:00 Documentation, proposal preparation, follow-ups
- 18:00 End of workday, occasionally networking events or webinars

Goals

- Convince customers with suitable infrastructure proposals
- Communicate complex technical concepts clearly
- Build technical trust with customers

Fears

- Overwhelming customers with overly technical language
- Lack of transparency in performance predictions
- Not understanding new tools or technologies quickly enough

Attitudes

- Enthusiastic about technology but pragmatic
- Customer-oriented and solution-focused
- Sees herself as a bridge between technology and sales

Technology Behaviour

- Uses tools like Teams, Excel, PowerPoint daily
- Often works remotely: laptop, smartphone, VPN access

Knowledge

- Solid understanding of IT infrastructure and network topologies
- No deep knowledge of simulation tools or distributed training
- Experience with customers from various industries

C. Input Analysis

System	ASTRA-Sim Category	UI Category	Importance
scheduling-policy	System	Advanced-Workload	0
endpoint-delay	System	Advanced-Network	0
active-chunks-per-dimension	System	Advanced-Workload	0
preferred-dataset-splits	System	Advanced-Workload	0
all-reduce-implementation	System	Workload	1
reduce-scatter-implementation	System	Workload	1
all-gather-implementation	System	Workload	1
all-to-all-implementation	System	Workload	1
collective-optimization	System	Advanced-Workload	0
topology-name	Network	Network	0
dimensions-count	Network	Network	1
topologies-per-dim	Network	Network	1
dimension-type	Network	Advanced-Network	0
units-count	Network	Accelerator	1
links-count	Network	Links	1
link-latency	Network	Links	1
link-bandwidth	Network	Links	1
nic-latency	Network	Advanced-Network	0
router-latency	Network	Advanced-Network	0
hbm-latency	Network	Accelerator	1
hbm-bandwidth	Network	Accelerator	1
hbm-scale	Network	Accelerator	1
parallelization-type	Workload	Workload	1
dnn-layers	Workload	Workload	1
workload	Workload	Workload	1

Table C.1.: System UI Configuration Overview