ADVCOMP 2024

The Eighteenth International Conference on Advanced Engineering Computing and Applications in Sciences

ISBN: 978-1-68558-184-8

September 29 - October 03, 2024

Venice, Italy

ADVCOMP 2024 Editors

Dominique Briechle, Technische Universität Clausthal, Germany

Biagio Peccerillo, University of Siena, Italy

ADVCOMP 2024

Forward

The Eighteenth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2024), held between September 29th, 2024, to October 3rd, 2024, in Venice, Italy, continued a series of international events meant to bring together researchers from the academia and practitioners from the industry in order to address fundamentals of advanced scientific computing and specific mechanisms and algorithms for particular sciences.

With the advent of high-performance computing environments, virtualization, distributed and parallel computing, as well as the increasing memory, storage and computational power, processing particularly complex scientific applications and voluminous data is more affordable. With the current computing software, hardware and distributed platforms, effective use of advanced computing techniques is more achievable.

The conference provided a forum where researchers were able to present recent research results and new research problems and directions related to them. The conference sought contributions presenting novel research in all aspects of new scientific methods for computing and hybrid methods for computing optimization, as well as advanced algorithms and computational procedures, software and hardware solutions dealing with specific domains of science.

We take here the opportunity to warmly thank all the members of the ADVCOMP 2024 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to ADVCOMP 2024. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the ADVCOMP 2024 organizing committee for their help in handling the logistics of this event.

We hope that ADVCOMP 2024 was a successful international forum for the exchange of ideas and results between academia and industry for the promotion of progress related to advanced engineering computing and applications in sciences.

ADVCOMP 2024 Chairs

ADVCOMP 2024 Steering Committee
Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium, FERIT, Croatia
Juha Röning, University of Oulu, Finland
Hans-Joachim Bungartz, TUM, Germany
Marcin Hojny, AGH University of Science and Technology, Poland
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany
Alice E. Koniges, University of Hawaiʻi at Mānoa, USA

ADVCOMP 2024 Publicity Chairs
Laura Garcia, Universidad Politécnica de Cartagena, Spain
Lorena Parra Boronat, Universidad Politécnica de Madrid, Spain

ADVCOMP 2024
Committee

ADVCOMP 2024 Steering Committee

Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium, FERIT, Croatia
Juha Röning, University of Oulu, Finland
Hans-Joachim Bungartz, TUM, Germany
Marcin Hojny, AGH University of Science and Technology, Poland
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany
Alice E. Koniges, University of Hawaiʻi at Mānoa, USA

ADVCOMP 2024 Publicity Chairs

Laura Garcia, Universidad Politécnica de Cartagena, Spain
Lorena Parra Boronat, Universidad Politécnica de Madrid, Spain

ADVCOMP 2024 Technical Program Committee

Waleed H. Abdulla, University of Auckland, New Zealand
José Abellán, Catholic University of Murcia, Spain
Mohamed Riduan Abid, Alakhawayn University, Morocco
Rashmi Agrawal, Manav Rachna International Institute of Research and Studies, India
Francisco Airton Silva, Federal University of Piauí, Brazil
M. Azeem Akbar, Nanjing University of Aeronautics and Astronautics, China
Haifa Alharthi, Saudi Electronic University, Saudi Arabia
Sónia Maria Almeida da Luz, Polytechnic Institute of Leiria - School of Technology and Management, Portugal
Madyan Alsenwi, Kyung Hee University, Global Campus, South Korea
Mohamed E. Aly, California State Polytechnic University, Pomona, USA
Daniel Andresen, Kansas State University, USA
Anindya Das Antar, University of Michigan, USA
Abhinav Arora, Meta Platforms, USA
Ehsan Atoofian, Lakehead University, Canada
Vadim Azhmyakov, Universidad Central, Bogota, Republic of Colombia
Carlos Becker Westphall, University of Santa Catarina, Brazil
Raoudha Ben Djemaa, ISITCOM | University of Sousse, Tunisia
Peter Bentley, University College London, UK
Alessandro Borri, CNR-IASI Biomathematics Laboratory, Rome, Italy
David Bouck-Standen, Kingsbridge Research Center, UK
Sofiane Bououden, University Abbes Laghrour Khenchela, Algeria
Hans-Joachim Bungartz, TUM, Germany
Xiao-Chuan Cai, University of Colorado Boulder, USA
Patricia Camacho Magriñán, Universidad de Cádiz, Spain
Jadson Castro Gertrudes, Federal University of Ouro Preto, Brazil
Graziana Cavone, Polytechnic of Bari, Italy
Sébastien Cayrols, University of Tennessee Knoxville, USA

Penghui Zhang, Arizona State University, USA
Pengmiao Zhang, University of Southern California, USA
Qian Zhang, Liverpool John Moores University, UK

Table of Contents

MORUS-PRNG: a Hardware Accelerator Based on the MORUS Cipher and the IXIAM Framework

Alessio Medaglini, Mirco Mannino, Biagio Peccerillo, Sandro Bartolini
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
e-mail: {medaglini | mannino | peccerillo | bartolini}@diism.unisi.it

***Abstract*—High-quality Pseudo-Random Number Generator (PRNG) is crucial in many applications that span a variety of fields. A common way to implement PRNGs is by exploiting an underlying secure ciphering algorithm, since its ciphertexts have statistical properties very close to those of a random sequence. Depending on the nature of the application requiring random values and its constraints, the ability of such a PRNG to generate numbers with high throughput and/or limited latency can be paramount. In recent years, programmers and researchers have been relying on hardware accelerators for many computation tasks where performance matter, moving progressively away from classic all-CPU software solutions. Ciphering algorithms and PRNGs have benefited from this tendency as well. In this paper, we propose a PRNG based on the MORUS cipher as an integrated accelerator that can be connected to CPU cores through the IXIAM layer, which allows a fast host-accelerator communication with RISC-V instructions. We measure performance in CPU cycles per number in the gem5 architecture simulator, and compare our implementation against plain software solutions provided by the C++ standard library. We show that our implementation outperforms them, with speedups above 2×.**

Keywords*-*cryptographic accelerators; hardware accelerators; simulation; ciphers; pseudorandom sequences.

I. Introduction

The ability to generate random number sequences has always found many applications in a variety of fields. According to Knuth, these include simulation, sampling, numerical analysis, computer programming, decision making, cryptography, aesthetics, and recreation [1], but many more can be added. A notable example is the generation of large prime numbers, which have a fundamental role in asymmetric key encryption algorithms since the introduction of RSA [2]. A popular method to obtain them involves generating random numbers, applying a primality test to them, and stopping when enough prime numbers are found [3]. Since the amount of random numbers necessary to obtain the required amount of primes cannot be known in advance, the ability to generate long random sequences with high performance is paramount.

However, generating *truly* random numbers may be difficult, and for most applications a number sequence generated deterministically that looks random *enough* is sufficient. Such a sequence is said a *pseudo-random* number sequence, and a module (hardware or software) implementing its generation algorithm is said Pseudo-Random Number Generator (PRNG).

A common way to implement a PRNG is by relying on an underlying cipher. In fact, the ciphertext produced by a cipher considered secure must present randomness properties, as no information about the original message should be obtainable from it: in practice, the ciphertext can be regarded as a sequence of random numbers. For this purpose, both stream ciphers and block ciphers operated in counter mode can be used, with examples pertaining to both categories being available in literature [4], [5].

MORUS [6] was selected as a finalist in the CAESAR competition announced by the National Institute of Standards and Technology (NIST) for authenticated encryption. It is an example of *authenticated cipher*, which outputs both a ciphertext and an authentication tag, providing both confidentiality and integrity. Its attractiveness derives from its potential speed both in HW and in SW, even on platforms not featuring dedicated or widespread ISA-extensions (e.g., AES-NI [7]). Compared to AEGIS, the winner of the aforementioned CAESAR competition, MORUS can be implemented with higher efficiency in hardware (both throughput per area and throughput per energy, as shown in Figure 6 and Figure 7 in [8]). Therefore, MORUS is particularly amenable to be adopted as the *heart* of a hardware accelerator serving as a PRNG. Furthermore, in the number generation task, the calculation of an authentication tag can be avoided, thus gaining further performance.

In this paper, we design *MORUS-PRNG*, a high-performance PRNG modular architecture encompassing an integrated hardware accelerator based on the MORUS cipher and suitable for modern multi-core processor systems. We design it as an IXIAM-ready accelerator [9] so to take advantage of reduced communication latency and a flexible and general interface between cores and accelerator. We simulate the architecture in gem5 simulator [10] and evaluate its performance, using all-CPU software-only PRNGs included in the standard library of the C++ programming language as a baseline.

The main contributions of this paper can be listed as follows:

- We design MORUS-PRNG, an integrated hardware accelerator implementing a MORUS-based PRNG, interfaced via the IXIAM framework to the CPU cores;
- We evaluate its performance, comparing it against software-only PRNGs included in the C++ standard library.

The paper is organized as follows. In the next section, we give some background on both MORUS and IXIAM. In Section III, we present our solution. In Section IV, we evaluate our proposal. Finally, we conclude in Section V.

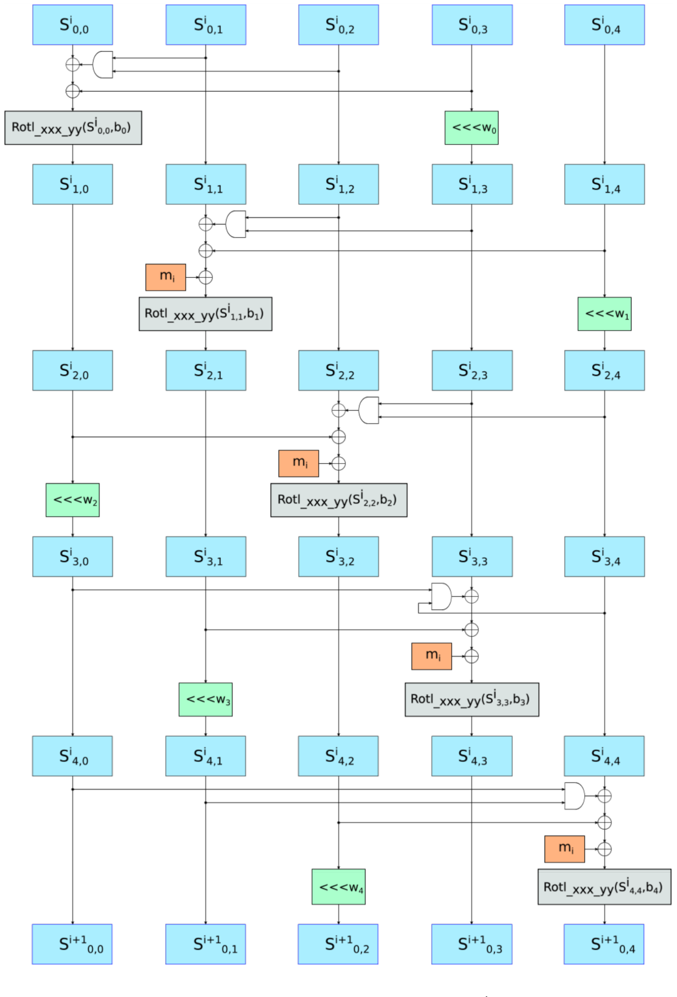


Figure 1. MORUS StateUpdate function, from [6]. $S^i_{n,m}$ is the m-th state block at the beginning of the n-th round, i-th step. w_n and b_n are constants, and m_i is the i-th block of the plain message. Rotl_xxx_yy(S, c) is the operation of dividing an xxx-bit S block in yy-bit words and perform left rotation by c bits.

II. Background

In this section, we give some background on the MORUS Authenticated Cipher and the IXIAM host-accelerator interface, which form the backbone of our proposal.

A. MORUS

MORUS is a family of authenticated ciphers which include three ciphers: MORUS-640-128, MORUS-1280-128, MORUS-1280-256 [6]. They can be described as stream ciphers with an internal state of 5 blocks which can have 128 or 256 bits each and deal with 128- or 256-bit keys. The names can be read as MORUS-x-y, with x being the size of the internal state and y the size of the key. In the following, the generic term MORUS is used to refer to all the three ciphers interchangeably, unless specified otherwise.

MORUS has been designed with speed in mind, with all the phases relying on different combinations of few basic operations (AND, XOR, shift, and rotate), chosen to be easily mapped on SIMD instructions in x86 processors. These design

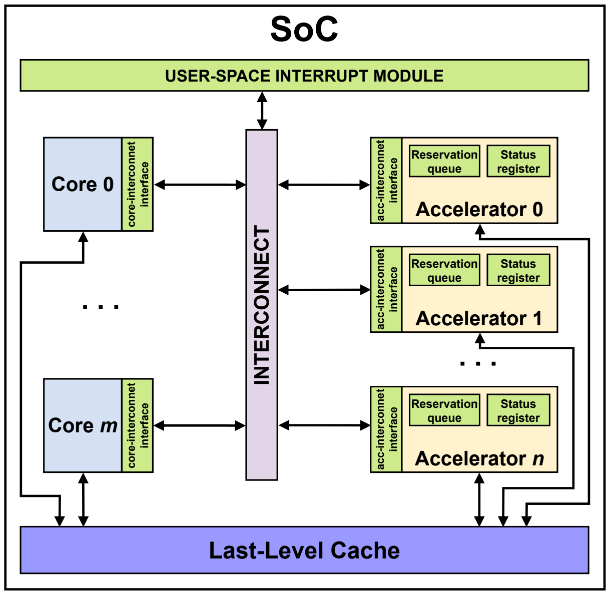


Figure 2. IXIAM hardware interface, highlighted in green on a generic SoC, from [9]. It consists of a core-interconnect interface for each core; an accelerator-interconnect interface, a reservation queue, and a status register for each managed accelerator; and a user-space interrupt module.

choices permit reaching 0.69 cycles per byte (cpb) on Intel Haswell processors [6]. MORUS is "authenticated" because the encryption phase produces, in addition to the ciphertext, also a 128-bit tag that can be used to verify decryption.

The cipher is articulated in four fundamental phases: initialization, encryption, decryption, and finalization. Initialization is performed by loading a key and a 128-bit Initial Value (IV) and running a *StateUpdate* function 16 times, mixing both key and IV into the internal state. Optionally, there is the possibility to use some Additional Data (AD) of any size to further *mix* the internal state and introduce additional non-linearity. Encryption is performed by encrypting a whole message of arbitrary size, processing one 128-bit (MORUS-640-128) or 256-bit (MORUS-1280-128 and MORUS-1280-256) block at a time. Each block is encrypted with 5 basic operations and a StateUpdate call that mixes the plain text into the internal state. Finalization consists of a XOR, a StateUpdate call, and a tag generation (achieved with 4 basic operations). The StateUpdate function, depicted in Figure 1, is used as a fundamental building block in the various phases and consists of 5 rounds in which each block of the internal state is updated with 5 basic operations. Decryption is analogous to the encryption. Message encryption/decryption is performed by initializing the cipher, invoking the encryption/decryption phase, and then finalizing to get the authentication tag.

B. IXIAM

ISA eXtension for Integrated Accelerator Management (IXIAM) is a hardware-software framework for Systems-on-a-Chip (SoCs). It permits controlling integrated accelerators directly from the cores, with specific CPU instructions triggering packet-sending towards the target accelerator and possibly generating a packet response, which is sent back to the core [9]. It is articulated in a limited hardware infrastructure and a RISC-V ISA extension. RISC-V was selected as the target ISA due to its open-source and modular nature, which allow for open-source implementation of the IXIAM framework. With respect to classic driver-based solutions, IXIAM ensures a lower latency in communicating with the accelerators, giving significant performance advantages, especially for small to medium workloads [9].

The hardware infrastructure, depicted in Figure 2, includes a few components on the accelerators, on the cores, and on the SoC, shared between them. For each accelerator, it consists in a FIFO reservation queue to manage requests from different processes and a status register to hold the accelerator status, so to be able to quickly distinguish between “busy”, “free”, and “error”. Both accelerators and cores need to send and receive specific packets through the SoC interconnect, and achieve this through the accelerator-interconnect and core-interconnect interfaces, respectively. Finally, IXIAM provides a light user-space interrupt mechanism which is managed by an ad-hoc module on the SoC.

The ISA extension provides 12 additional instructions for RISC-V ISA: RESERVE to ask for accelerator reservation, CHECK to check the reservation outcome, TGL/TGS to load/store data from/to a specific memory location into/from the accelerator, TL to move data across different memory resources on the accelerator, TRL/TRS to load/store data from/to a CPU register into/from the accelerator, EXEC to trigger an operation execution on the accelerator, ISBUSY to check the accelerator status, RELEASE to release the accelerator, AFENCE to block the CPU pipeline until flying transfer instructions on the accelerator complete, and RUISR to indicate a function to serve user-space interrupts. The majority of instructions are designed as *asynchronous* instructions, in the sense that the execution on the CPU can proceed undisturbed after an instruction commit, with no need to wait for response packets. The only exceptions are: CHECK, TRS, ISBUSY, and AFENCE.

III. Our Proposal

Figure 3 shows our design for an integrated MORUS-based PRNG hardware accelerator based on the IXIAM framework. It includes the following modules:

MORUS PRNG Engine is the processing engine, which is responsible of doing MORUS-based pseudo-random number generation;

Output buffer is the buffer that will contain the array of generated numbers;

Register file includes a register to hold the amount of pseudo-random numbers to be generated and four key registers, each holding one word of the key;

Controller is responsible for reading the IXIAM packets from the interconnect, translate them into accelerator commands, and send response packets to the calling core through the interconnect;

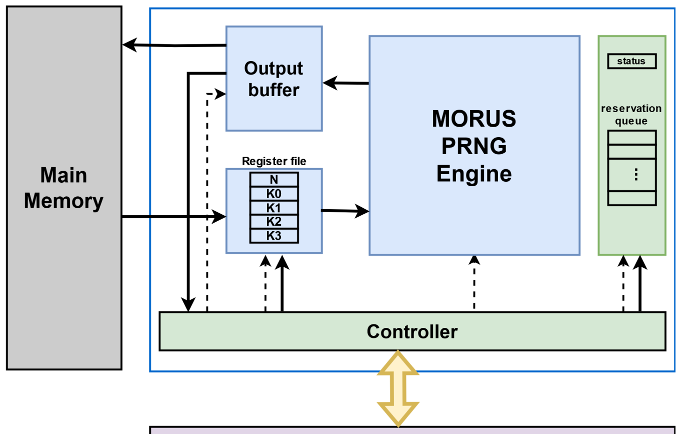


Figure 3. MORUS-PRNG, an integrated hardware accelerator with the necessary components to communicate with the IXIAM framework. Solid lines indicate data exchange, dashed ones indicate control signal exchange. The register file holds the number of pseudo-random numbers to generate and the four words of the key. The output buffer will hold the generated numbers and the MORUS PRNG Engine implements the number generation logic.

IXIAM HW infrastructure includes a status register and a reservation queue.

Without loosing generality, in the following, we consider the underlying cipher as being MORUS-1280-128. Thus, the four words composing the key are 32-bit words.

A. Technical Details

gem5 is a well-established tool for computer architecture researchers. It is a cycle-accurate simulator with a modular nature, in which architectural components (CPU, memory, caches, NoC, accelerators, etc.) are treated as individual objects that communicate with each other through ports. Every operation can be simulated functionally and be described in terms of associated latency. gem5 supports multiple ISAs and is easily extensible as its code is open-source. IXIAM was proposed and evaluated by the means of gem5 components [9].

We design our solution in terms of gem5 modules. We implement their operations functionally and characterize them from a latency standpoint. We acknowledge that other techniques such as VHDL or Verilog description would allow us to achieve higher accuracy. However, we are interested in the performance that such an accelerator could achieve within the system, rather than its precise gate-level performance. In this case, gem5 offers a more appropriate level of abstraction. We leave the implementation of such solutions to future work.

We design the MORUS PRNG Engine as a standalone processing element inside our MORUS-PRNG accelerator. We consider it as being analogous to the ASIC design described by Muehlberghuber and Gürkaynak in [11]. This exposes a variable throughput that grows with the amount of numbers to be generated and varies from 2.54 to 250Gbps. We select an operating frequency of 250MHz.

For the Output buffer, we select a size of 1MiB that can host up to 262’144 numbers. The Register file includes the 5 32-

bit registers specified above. Considering a SRAM technology for these memories, we estimate with CACTI [12] an access latency of 2 and 1 cycles (at 250MHz), respectively.

To model the Controller, we associate a latency to the "decode and execute" of the IXIAM instructions contained in the packets coming from the interconnect. Since it is a lightweight controller with limited responsibilities, we can associate a 1 cycle latency to every instruction, except those that need to access the reservation queue (RESERVE, RELEASE, CHECK), for which we associate 3 cycles. However, these latencies do not include the *operative* part of the instructions: for instance, after 1 cycle of an EXEC instruction, the execution operation begins and ends after a number of cycles that depend on the amount of work to be done. The same happens for the transfer instructions, where we add to the "decode and execute" latency the one calculated in the buffer/register file that models the effective data read/write.

B. Operations

In the rest of this section, we describe the two operations exposed by the accelerator: Initialize and Generate.

1. Initialize

The Initialize operation is responsible for triggering the initialization phase in the underlying MORUS cipher, embodied by the MORUS PRNG Engine. The user is responsible for writing into the K_n registers the four words composing the 128-bit key. The 128-bit IV value necessary for initialization, together with the key, is hard-coded into the MORUS PRNG Engine.

Each of the four K_n registers can be written via a TRL or a TGL instruction. The former reads a value from a CPU register and sends it to the accelerator, while the latter passes a main memory location to the accelerator that reads a value from it.

Initialize is triggered with an EXEC instruction with `op_id` parameter set to 0. When the accelerator receives the corresponding packet, the MORUS PRNG Engine reads the key from the four local registers and the hard-coded IV value and triggers the underlying MORUS initialize. This modifies its internal state and brings the engine in a state that is ready to perform subsequent encryptions, necessary to generate pseudo-random numbers. An internal 128-bit counter is set to 0.

The possibility of including AD of arbitrary size in the initialization phase is not managed by the proposed engine. This choice, together with having a hard-coded IV, limits the degrees of freedom with respect to a classic MORUS cipher. However, for this particular application (PRNG), the sole 128-bit key as the only degree of freedom may be considered sufficient, as it serves the same purpose as a seed in analogous pseudo-random number generation algorithms, which are usually 32-bit integers [13]. In any case, this design can be easily improved by adding six more registers to set before initializing: four for the IV, one for the memory location containing the beginning of AD, and one for its size. This would allow for seeding the PRNG with data of arbitrary size, improving the *quality* of the generated numbers, but increasing the duration of the initialization phase.

2. Generate

The Generate operation triggers the encryption phase in the underlying MORUS cipher, generating pseudo-random numbers as a consequence. The user writes into the N register the amount of numbers they want to generate. Also in this case, by the means of a TRL or TGL instruction.

Generate is triggered with an EXEC instruction with `op_id` parameter set to 1. As a first step, the engine reads the content of the N register and interprets it as the amount of 32-bit pseudo-random numbers to generate. If this exceeds the capacity of the Output buffer, an error code is written in the status register and the operation terminates. Otherwise, the pseudo-random number generation can proceed and is performed by encrypting the content of an internal 128-bit counter. It is initialized to 0 in the Initialize and is incremented after each encryption step. At each step, a 128-bit block of ciphertext, which can be interpreted as four 32-bit numbers, is generated this way. The generation terminates when the counter value minus its initial state (which is saved into an internal register when encryption begins) equals $\lceil N/4 \rceil$, a comparison that can be easily done in hardware by checking whether said difference is greater or equal than N without its two least significant bits.

The generated numbers are written in the Output buffer starting from address 0. From there, they can be retrieved with a TGS instruction as soon as the number generation terminates. TGS is responsible for copying the generated numbers to a main memory location, where the CPU can read them when needed.

At the end of Generate, the internal counter is not reset: only Initialize is responsible for that. This way, the number that will be generated next is completely determined by the internal state of the cipher and the state of the counter, which are, in turn, uniquely determined by the key chosen by the user and the amount of numbers generated so far. This way, initializing the engine with key K and generating m numbers first and n numbers then leads to the same sequence obtained by initializing the engine with K and generating $m + n$ numbers in one go.

No MORUS finalize phase is invoked, as it would be responsible for producing the authentication tag, which is not needed in the pseudo-random number generation task at hand.

IV. EVALUATION

In order to evaluate our proposal, we implement MORUS-PRNG and its interfacing in the gem5 architecture simulator. For this purpose, we take advantage of the infrastructure proposed in [9]. Table I lists the specifications of the simulated system.

The accelerator performance is evaluated in the context of a simple C++ 17 application in which a variable amount of pseudo-random numbers is generated. The CPU-accelerator communication is wrapped in a C++ generator engine class that executes Initialize when constructed and exposes two methods:

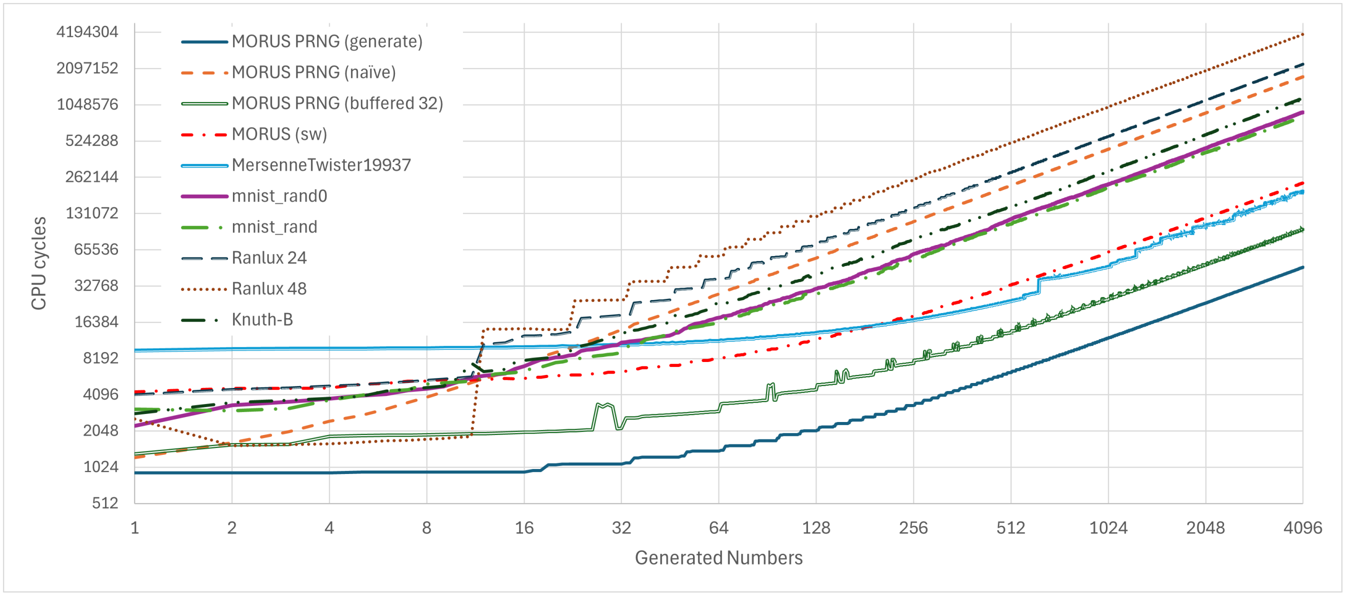


Figure 4. Performance comparison between MORUS-PRNG and other PRNGs included in the C++ standard library. For each PRNG, it is displayed the number of CPU cycles necessary to generate the amount of numbers indicated on the X axis (lower is better).

TABLE I. SPECIFICATION OF THE SIMULATED SYSTEM.

CPU	Quad-core, 3.4GHz, RISC-V, MinorCPU
L1 I/D Cache	32KB, 8-way, write-back, 64B block size, non-blocking, 2-cycles access time, private
L2 Cache	512KB, 8-way, write-back, 64B block size, non-blocking, 10-cycles access time, private
L3 Cache	8MB, 16-way, write-back, 64B block size, non-blocking, 36-cycles access time, shared
Interconnection	ring-based, 16-cycles average latency
Main Memory	DRAM-DDR4, 16GB, 300-cycles access time, classic memory model
MORUS-PRNG	engine throughput 2.54-250Gbps, 250MHz, buffer size 1MiB, buffer latency 2 cycles, register latency 1 cycle

operator() outputs a single generated number;
generate fills an array with **n** generated numbers.

The first method is compliant with the C++ specification, so an instance of this MORUS-based generator class can be passed to a distribution object, according to the modern syntax introduced in C++11 [13]. An evident limitation of this design is that at most 1 number is generated per method call, so filling an array of **n** elements requires **n** method invocations. The **generate** method, conversely, adopts a more efficient design, as it permits of generating the needed amount of numbers in the minimum number of steps. This is determined by the capacity of the Output buffer, which can host at most 256 32-bit numbers (see Table I).

We design two variants of **operator()** solutions:

naïve triggers the generation of 1 number on the accelerator, reads it from there, and returns it to the caller;
buffered triggers the generation of a few numbers on the accelerator, reads them in a local buffer, returns 1 buffered number at each method call until all of them have been consumed, and generates another amount at that point.

Without loosing generality, we tune the local buffer of the *buffered* version to a capacity of 32 elements.

To evaluate our solution, we compare it with other PRNGs defined in the **random** C++ header. The classes included there allow generating pseudo-random numbers using a combination of *generator* and *distribution* objects. The former generates uniformly distributed numbers, while the latter transforms number sequences generated by a generator into number sequences that follow a specific random variable distribution (e.g., uniform, Normal, or Binomial). The generator can be instantiated with a seed, then passed as an input parameter to the **operator()** of the distribution object, to generate one pseudo-random number per method invocation.

Several PRNGs are included in the **random** C++ header:

- **linear congruential engine (mnist_rand, mnist_rand0)**: they are the simplest engines in the STL library that generate pseudo-random unsigned integer numbers by using $x = (ax + c) \bmod m$, where x is the current state and a, c, and m are different parameters.
- **MersenneTwister19937**: it is a random number engine based on the Mersenne Twister algorithm [14]. It produces high quality unsigned integer random numbers in the interval [0, (2^w)-1], where w is the word size (i.e., number of bits of each word in the state sequence).
- **Ranlux24, Ranlux48**: they are 24-bit and 48-bit RANLUX generators by Martin Lüscher and Fred James [15], based on the subtract with carry algorithm.
- **Knuth-B**: It is a shuffle_order_engine adaptor that returns shuffled sequences generated with the simple pseudo-random number generator engine *minstd_rand0*.

For completeness, we add a further PRNG which encompasses an all-software implementation of the MORUS cipher: *MORUS-sw*. We compile our application using g++14 from the RISC-V GNU toolchain [16].

Figure 4 shows the performance achieved by MORUS-PRNG (*generate*, *naïve*, and *buffered* versions) in generating a variable amount of pseudo-random numbers, in comparison with other PRNGs included in the standard library of the C++ programming language. Performance is measured in CPU cycles (so, lower is better). In the following, we refer to the amount of numbers generated as the "workload size".

What emerges from the performance comparison is that MORUS-PRNG in its *generate* version outperforms the other PRNGs, and the performance gap grows with the workload size: with respect to the second best, from 2.46× (mnist_rand0) with 1 element, up to 4.26× (MersenneTwister19937) with 4096 elements. As specified before, the different interface between MORUS-PRNG *generate* and the other PRNGs has a non-negligible impact, as *generate* is an optimal design which minimizes the method invocations, with consequent minimization of CPU-accelerator communication.

MORUS-PRNG *naïve* and *buffered 32* have the same interface as the other PRNGs. As expected, *generate* performs better than both of them. *naïve* evidently pays the communication latency between CPU and accelerator, which happens at every method invocation. While this is negligible with few elements, its performance are surpassed by other PRNGs (MersenneTwister19937, mnist_rand0, mnist_rand, Knuth-B) for workload sizes above 24 elements.

MORUS-PRNG *buffered* proves to be a more reasonable design, with the advantage of being compliant with the C++ generators syntax and being able of outperforming all the library-provided PRNGs for workload sizes greater than 11 elements (when Ranlux48 generation time suddenly increases). Speedup with respect to MersenneTwister19937, which is the best C++-provided PRNG when 36 or more numbers are needed, tends to 2.07× as the workload size increases.

We investigated the sudden performance worsening of Ranlux48 at the 12th number generation, which is clearly visible in the figure. Looking at Ranlux48 source code, we noticed that it is implemented as a *discard block engine* with two template integer parameters: block-size and used-block, which are set to 389 and 11, respectively. These parameter regulate its functioning: every 11 (used-block) elements generated, 389 - 11 (block-size minus used-block) are generated and discarded, causing a spike in the elapsed time every 11 elements.

Interestingly, the MORUS-sw proves as a valid alternative with respect to the other PRNGs included in the C++ standard library, proving its value per-se, even with no hardware acceleration involved. In fact, it is faster than all the other C++-provided PRNGs for workload sizes between 12 and 196, and is outperformed by MersenneTwister19937 when the workload size surpasses 195 numbers, with the speedup between the two tending to 1.18× in favour of MersenneTwister19937.

In conclusion, MORUS-PRNG provides a valid solution to generate pseudo-random numbers, also in a version that maintains compliance with the C++ syntax, as long as buffering techniques are adopted in its implementation in order to reduce the CPU-accelerator communication costs.

V. Conclusion and Future Work

In this paper, we proposed MORUS-PRNG, an integrated accelerator for pseudo-random number generation based on the MORUS cipher. We designed it to communicate with the CPU through the IXIAM framework, which allows users to control it directly with CPU instructions. We evaluated it in a simulated environment in the gem5 architectural simulator, comparing its performance against PRNGs included in the C++ standard library. We showed that it is able to outperform them.

As future work, we plan to implement our solution in hardware and conduct a more accurate evaluation. Also, we plan to evaluate it against other accelerators aimed at pseudo-random number generation.

References

[1] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997, pp. 1–2, ISBN: 0-201-89684-2.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978, ISSN: 0001-0782. DOI: 10.1145/359340.359342.

[3] Igumnov, "Generation of the large random prime numbers," in *2004 International Siberian Workshop on Electron Devices and Materials*, 2004, pp. 117–118. DOI: 10.1109/PESC.2004.241202.

[4] B. Peccerillo, S. Bartolini, and Ç. K. Koç, "Parallel bitsliced AES through PHAST: a single-source high-performance library for multi-cores and GPUs," *Journal of Cryptographic Engineering*, vol. 9, no. 2, pp. 159–171, Jun. 2019, ISSN: 2190-8516. DOI: 10.1007/s13389-017-0175-4.

[5] T. Tuncer and E. Avaroğlu, "Random number generation with LFSR based stream cipher algorithms," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 171–175. DOI: 10.23919/MIPRO.2017.7973412.

[6] H. Wu and T. Huang, *The Authenticated Cipher MORUS (v2)*, Submission to CAESAR: Competition for Authenticated Encryption. Security, Applicability, and Robustness (Round 3 and Finalist), Sep. 2016.

[7] J. K. Rott, *Intel Advanced Encryption Standard Instructions (AES-NI)*, Retrieved: 15-09-2024, Feb. 2012.

[8] S. Kumar, J. Haj-Yahya, M. Khairallah, M. A. Elmohr, and A. Chattopadhyay, *A comprehensive performance analysis of hardware implementations of CAESAR candidates*, Cryptology ePrint Archive, Paper 2017/1261, Retrieved: 15-09-2024, 2017.

[9] B. Peccerillo, E. Cheshmikhani, M. Mannino, A. Mondelli, and S. Bartolini, "IXIAM: ISA EXtension for Integrated Accelerator Management," *IEEE Access*, vol. 11, pp. 33 768–33 791, 2023. DOI: 10.1109/ACCESS.2023.3264265.

[10] N. Binkert *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[11] M. Muehlberghuber and F. K. Gürkaynak, *Towards Evaluating High-Speed ASIC Implementations of CAESAR Candidates for Data at Rest and Data in Motion*, en, Other Conference Item, Workshop on Directions in Authenticated Ciphers (DIAC); September 28-29, Singapore, 2015.

[12] S. J. E. Wilton and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model.," *IEEE Journal of Solid State Circuits*, vol. 31, no. 5, pp. 677–688, 1996. DOI: 10.1109/4.509850.

[13] ISO, *ISO/IEC 14882:2011 Information technology — Programming languages — C++*. Geneva, Switzerland: International Organization for Standardization, Feb. 2012, 1338 (est.) Retrieved: 15-09-2024.

[14] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, Jan. 1998, ISSN: 1049-3301. DOI: 10.1145/272991.272995.

[15] M. Lüscher, "A portable high-quality random number generator for lattice field theory simulations," *Computer Physics Communications*, vol. 79, no. 1, pp. 100–110, Feb. 1994, ISSN: 0010-4655. DOI: 10.1016/0010-4655(94)90232-1.

[16] *RISC-V GNU Compiler Toolchain*, Retrieved: 15-09-2024, 2024.

Accelerating Differential Privacy-Based Federated Learning Systems

Mirco Mannino, Alessio Medaglini, Biagio Peccerillo, Sandro Bartolini
Department of Information Engineering and Mathematics
University of Siena
Siena, Italy
e-mail: {mannino | medaglini | peccerillo | bartolini}@diism.unisi.it

***Abstract*—The number of mobile, wearable, and Internet of Things (IoT) devices we are using is increasingly growing, especially those implementing machine learning applications on-the-edge. Relying on a centralized server for processing and storing of this ever-increasing amount of data might not be the optimal solution, from both performance and privacy points of view. *Federated Learning* is a good solution to avoid sending user's data to a central server to train machine learning models. In order to guarantee privacy in a *Federated Learning* system, it is possible to leverage several techniques. *Differential Privacy* is one of the most popular, since it provides robust privacy protection. In this paper, we target mobile devices, proposing ideas on how to speed up training in *Differential Privacy*-based *Federated Learning systems* through a dedicated hardware accelerator.**

Keywords-differential privacy; federated learning; hardware accelerator.

I. Introduction and Background

Traditional Machine Learning (ML) approaches expect to collect data into a central computational system (e.g., server) and train models using such data. Nowadays, with the unstoppable diffusion of mobile devices (e.g., smartphones and wearable ones), more and more data is being collected locally, with a growing need to keep it private and unshared. Federated Learning (FL) was introduced by Google in 2017 [1] as a solution to implement a distributed training approach where individual model replicas are trained locally on different user's devices, and then global aggregated training strategy is orchestrated. In a FL system, there are two kinds of players: 1) a centralized server and 2) a group of N clients. Each client has a local database built with data collected locally that should not be shared with others. The training process can be summarized in the following steps:

- The server shares an untrained model among clients;
- Each client performs a local training procedure using its own data;
- Clients send trained models to the central server;
- The server aggregates them into an updated model;
- The server shares the updated model among clients.

The training process is iterative, continuing until it converges on the optimal model. During training, it is also possible that clients exchange parameters among themselves. Figure 1 shows an overview of a FL system.

One of the key aspects of FL is ensuring the privacy of data collected locally. Among all the techniques proposed to ensure privacy, Differential Privacy (DP) is one of the most promising [2]. Although DP can be achieved in different ways, the key-idea is to add noise to guarantee privacy. There are several research proposals that leverage DP [3]: 1) in *Local DP* techniques (e.g., [4]), the clients alter local data and send them to the server for centralized aggregation, protecting both the clients and the server from potential private information leaks; 2) DP based distributed Stochastic Gradient Descent (e.g., [5]) techniques aim to perturb the gradient during the training phase on the client devices; 3) DP meta learning [6] techniques aim to learn a model that can quickly adapt to new tasks using a few data points.

Another aspect concerning the world of FL, which is not usually taken into account in *traditional* ML applications, is the possibility of doing training at the edge. Indeed, usual ML models are trained on high-end platforms equipped with several types of hardware accelerators, e.g., Tensor Processing Unit (TPU). With the FL approach, client devices need to be readapted in order to guarantee efficiency and performance during the training phase. Indeed, training, compared to inference, involves the repetition of several steps: feed-forward, backpropagation, and weight gradient [7], [8]. For this reason, companies are encouraged to produce systems allowing training on the edge, especially introducing more storage and specialized hardware. In terms of specialized hardware, it is possible to distinguish three main categories [9]: Graphics Processing Unit (GPU), Field Programmable Gate Array (FPGA), and Application Specific Integrated Circuit (ASIC). GPU-based acceleration is the more flexible in terms of programmability, but it leads to a higher power consumption compared to the other two categories. On the other hand, FPGA- and ASIC-based accelerators allow to reach higher

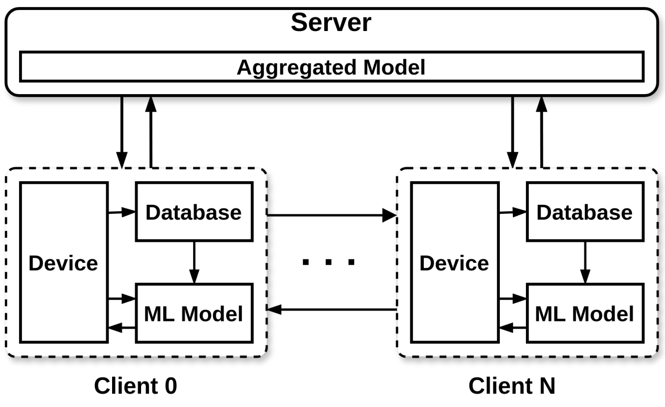


Figure 1. Overview of a Federated Learning system.

performance and lower power consumption. ASICs suffer from a lower flexibility in terms of design and development, compared to FPGAs.

Nevertheless, it is worth mentioning that, in a FL system, there is a huge degree of heterogeneity and it is not possible to assume that all the client nodes incorporate the same hardware resources. Finding solutions that can be optimal for all the categories is the only way to achieve an efficient training phase from a system perspective.

The remainder of the paper is organized as follows. In Section II, we summarise the key points of differential privacy-based FL acceleration from our point of view, and in Section III we give an overview of a possible hardware accelerator design. Finally, in section IV we conclude.

II. Acceleration of Differential Privacy Training

It is clear that FL brings new challenges to the client devices, both from an algorithmic and architectural points of view. Differential Privacy adoption needs the addition of *noise* to the local data before sending them to the server, and the training process performed on-the-edge needs to be as much performant and efficient as possible.

One of the main challenges is to leverage, in an efficient way, the heterogeneity of client platforms and their interaction with the server node. We believe that a hardware/software (HW/SW) co-design is the key to find an optimal solution to the problem. In particular, there is the need to design and implement solutions that can be adopted by all the client platforms. The key points can be summarized as follows:

- Robust framework allowing the orchestration of all the players in the system.
- Algorithmic improvement for DP both on client and server side.
- Introduction of specialized hardware in heterogeneous architectures to accelerate common operations in FL systems, ensuring energy efficiency.

The increasing interest in FL led to the creation of several open-source frameworks, such as Tensorflow Federated [10] and FATE [11]. The open-source nature of these frameworks provides engineers with robust tools that are continuously developed and improved. For this reason, we focus our attention on the other two points of the previous list.

Algorithmic improvements and design of specialized hardware can be developed together as a HW/SW co-design process. Since we are targeting DP-based FL, the algorithmic aspects include both DP and deep learning training operations.

Our proposal is to design and evaluate a dedicated circuit, called *Federated Learning Processing Unit* (FLPU). It should be integrated in the current architectures, providing highly specialization in DP operations (e.g., noise addition, encryption) and deep learning operations carried out during training (e.g., backpropagation). The novel module should be included in any client platform: as an autonomous module on GPUs, an IP-block for Systems-on-a-Chip or FPGAs, adapting to the compatibility needs of each of them. This way, the programming side would also benefit.

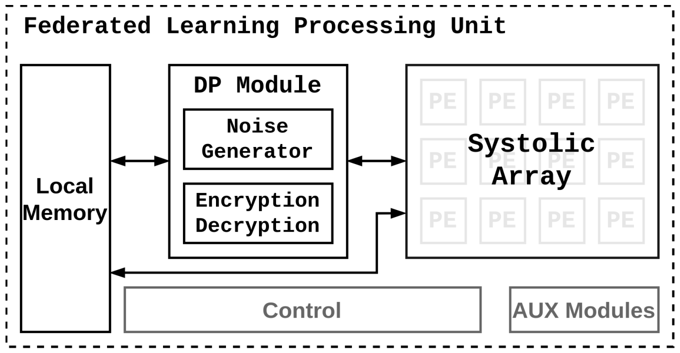


Figure 2. Overview of the Federated Learning Processing Unit (FLPU).

Under DP conditions, the training process performed among several clients may need encryption/decryption operations, and the generation of noise according to a certain distribution. For this reason, the FLPU should be equipped with an engine capable of efficiently carrying out these operations. At the same time, provisions should be made for deep learning-related tasks, including specialized hardware to accelerate deep learning processes (e.g., systolic arrays) and dedicated memory for storing weights during the backpropagation phase. The adoption of novel and existing algorithmic optimizations (e.g., reduction of memory consumption during backpropagation [8]) should be evaluated in order to reach an optimal design.

III. Federated Learning Processing Unit

The FLPU is in charge of accelerating the training phase on the heterogeneous client devices under DP conditions. Figure 2 shows an overview of its architecture. The systolic array in the architecture is used to accelerate deep learning computations, e.g., matrix multiplication. The systolic array can be implemented using different dataflow strategies. For example, input-, weight-, and output-stationary dataflows can be utilized [12]. A more detailed workload analysis is essential to determine the optimal design choice. Moreover, a series of auxiliary modules are included in the architecture. In particular, these modules are useful to accelerate common operations in deep learning algorithms, such as activation function and quantization.

The FLPU is equipped with an on-chip local memory. It is used to store input data, weights, and output data. The specific design of the memory will be established after an accurate assessment of the workloads involved. Among the potential solutions, one option is to use a single memory unit to store all data types, or alternatively, multiple local memories, each dedicated to storing specific types of data.

One of the key components is the *DP module*, responsible for ensuring the implementation of differential privacy mechanisms. The main role of DP module within the FLPU is the possibility to add noise and encrypt data within the accelerator itself. Indeed, this design introduces an additional layer of security that a conventional accelerator (e.g., TPU) would not have. The DP module is mainly composed of two components: *noise generator* and *encryption/decryption* *module*. The noise generator exploits some random physical signals that can be read from the device (e.g., temperature). The encryption/decryption module can be implemented by exploiting cryptographic accelerator designs [13].

IV. Conclusion and Future Work

Federated Learning is a promising approach to exploit computation on-the-edge and preserving users' privacy. In this paper, we focus on *Differential Privacy*, discussing how it can be implemented in FL systems. Moreover, we point out the key points needed to obtain a performant and energy efficient heterogeneous FL system. In the future, we will explore these aspects further, starting with the design and implementation of ad-hoc modules, either as novel chips or integrated into existing architectures.

Another area of focus will be the HW/SW co-design required to efficiently implement both DP and deep learning training operations. In this scenario, a deeper investigation of both DP and deep learning training algorithms is needed to jointly understand and optimize them.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, PMLR, 2017.

[2] A. El Ouadrhiri and A. Abdelhadi, "Differential privacy for deep and federated learning: A survey," *IEEE access*, vol. 10, pp. 22 359–22 380, 2022.

[3] K. Wei *et al.*, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE transactions on information forensics and security*, vol. 15, pp. 3454–3469, 2020.

[4] S. Wang *et al.*, "Local differential private data aggregation for discrete distribution estimation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, 2019.

[5] N. Wu, F. Farokhi, D. Smith, and M. A. Kaafar, "The value of collaboration in convex machine learning with differential privacy," in *2020 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2020, pp. 304–317.

[6] J. Li, M. Khodak, S. Caldas, and A. Talwalkar, "Differentially private meta-learning," *arXiv preprint arXiv:1909.05830*, 2019.

[7] Y. Wang *et al.*, "Trainer: An energy-efficient edge-device training processor supporting dynamic weight pruning," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 10, 2022.

[8] N. Kukreja *et al.*, "Training on the Edge: The why and the how," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2019, pp. 899–903.

[9] H. G. Abreha, M. Hayajneh, and M. A. Serhani, "Federated learning in edge computing: A systematic survey," *Sensors*, vol. 22, no. 2, p. 450, 2022.

[10] "TensorFlow Federated: Machine Learning on Decentralized Data," [Online]. Available: https://www.tensorflow.org/federated (visited on 09/18/2024).

[11] "An Industrial Grade Federated Learning Framework," [Online]. Available: https://fate.fedai.org/ (visited on 09/18/2024).

[12] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[13] N. Samardzic *et al.*, "F1: A fast and programmable accelerator for fully homomorphic encryption," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 238–252.

Application of a Maneuver-Based Decision Making Approach for an Autonomous System Using a Learning Approach

Xin Xing, Sebastian Ohl
Faculty of Electrical Engineering
Ostfalia University of Applied Sciences
Wolfenbuettel, Germany
e-mail: {xi.xing | s.ohl}@ostfalia.de

***Abstract*—Autonomous driving technology has progressed significantly, necessitating advanced maneuver-based decision-making systems for complex driving environments. Traditional methods often fail in unpredictable real-world scenarios, leading to the adoption of learning-based approaches, such as Deep Learning (DL) and Reinforcement Learning (RL). This paper explores safety-critical car-following models and traffic management, focusing on Adaptive Cruise Control (ACC) and Automatic Emergency Braking (AEB). Traditional mathematical models are limited under extreme conditions, thus this study leverages machine learning to enhance vehicle responsiveness. Specifically, we apply RL to train car-following models. We emphasize policy-based RL methods, including Policy Gradient (PG) and Proximal Policy Optimization (PPO), within a simulated environment. The results demonstrate that PPO converges faster and exhibits fewer errors compared to PG. This study confirms that RL can effectively automate maneuver-based decision-making, highlighting the need for further research in diverse traffic conditions.**

Keywords-Autonomous Driving; Decision-making; Reinforcement Learning; Car-following models; Adaptive Cruise Control; Automatic Emergency Braking; Proximal Policy Optimization; Policy Gradient.

I. Introduction

Autonomous driving technology has made significant strides in recent years, driven by the imperative need for a decision-making system that can navigate complex and evolving driving environments. Traditionally, decision-making methods in autonomous driving have relied on robust, yet often rigid, frameworks that struggle to accommodate the unpredictable nature of real-world scenarios [1]. This limitation has led to the growing adoption of learning-based approaches, especially utilizing Deep Learning (DL) and Reinforcement Learning (RL), aimed at enhancing the adaptability and accuracy of Advanced Driver Assistance Systems (ADAS) [2][3][4]. Similarly, the *ExerShuttle* project is focused on the development of autonomous campus shuttle services that can transport passengers to desired locations within the campus, such as a library or cafeteria. This initiative aims to leverage intelligent driving technologies to enhance accessibility and convenience in campus environments.

A critical aspect of intelligent driving systems is car-following [5], which involves high-precision models crucial for ensuring driving safety, alleviating urban traffic congestion, and reducing the driver's workload. In this context, systems, such as Adaptive Cruise Control (ACC) [6] and Automatic Emergency Braking (AEB) [7] play pivotal roles. ACC adjusts the vehicle's velocity to maintain a safe distance from the car ahead, thereby easing the driver's burden. Conversely, AEB systems engage automatically to mitigate or prevent collisions by applying brakes when a potential risk is detected. The operational efficacy of both ACC and AEB is contingent upon accurate vehicle tracking models that respond promptly and reliably in varied driving conditions [8].

Traditional car-following models have largely been mathematical and, while useful, occasionally fall short under extreme conditions, thereby compromising safety. To overcome these limitations, large amounts of trajectory data are utilized and machine learning techniques are applied to reveal underlying patterns. These models, which include traditional Machine Learning, Deep Learning, and Deep Reinforcement Learning approaches, potentially enhance the responsiveness of vehicles to diverse driving scenarios, thus improving system accuracy and generalizability [9][10][11].

The paper particularly focuses on the use of Deep Reinforcement Learning (DRL) to train car-following models. RL is a policy-oriented decision-making method that aims to maximize rewards through trial-and-error behaviors, such as Policy Gradient (PG) [12]. DRL algorithms that integrate deep neural networks with RL principles, such as Deep Q-Networks (DQNs), have been demonstrated to effectively manage the complexity of ADAS algorithms and significantly enhance the system's ability to respond effectively to hazardous situations [2]. In [8], a novel RL-based longitudinal control and collision avoidance algorithm is developed that effectively takes into account the behavior of both the front and rear vehicles using the Deep Deterministic Policy Gradient (DDPG) model. The algorithm is shown to be capable of preventing potential serial collisions.

DQN and DDPG are mainly value-based methods, while PG and PPO [13] are direct policy optimization methods. DQN and DDPG usually require more samples for training, so they may encounter policy instability and convergence problems in actual training. In comparison, PG and PPO are more easily adapted to environments that have specific requirements on the form of the policy, such as scenarios that require the policy to output specific probabilistic information or continuous action [2]. Furthermore, PG and PPO demonstrate superior adaptability in environments where the policy must respond to dynamic or uncertain factors. This makes them well-suited for training car-following models, offering more robust and flexible solutions. This paper therefore emphasizes policy-based Reinforcement Learning methods, including PG and PPO algorithms, for training sophisticated car-following models.

The remainder of the paper is organized as follows: Section II discusses the problem formulation. Section III provides theoretical background. Section IV details the methodology. Section V evaluates the results, and Section VI concludes the paper and outlines future work.

II. Problem Formulation

In this paper, we explore the ACC and AEB systems within autonomous car-following models. The Autonomous Vehicle (AV) maintains a safe following distance from a Leading Vehicle (LV) or brakes urgently to avoid obstacles, such as a yellow duck used in simulations.

RL typically employs a Markov Decision Processes (MDP) to represent the interactions between the vehicle and its environment, taking actions based on the state of the environment and then receiving new states in response. For situations if states are not fully observable, a Partially Observable Markov Decision Process (POMDP) is employed to provide a more realistic representation of the state space. In order to design advanced policy using RL techniques, we formulate the switching between ACC and AEB as a POMDP.

A. States and Observations

To simplify the training model, the state and observation parameters are the AV's velocity, the LV's velocity, the gap between the AV and the LV or obstacle ahead, and the current action: $s_t = [V_{AV}, V_{LV}, G, A]$. The termination state is defined as when the AV collides with the LV or obstacle ahead, or if the LV's velocity is 0 and the gap between the AV and the LV or obstacle in front is less than the safe distance, which is calculated from the Time To Collision (TTC).

B. Actions

The action space consists of two elements, ACC and AEB: $A = [ACC, AEB]$. ACC adjusts the AV's velocity based on the Intelligent Driver Model (IDM) [14] to maintain a safe distance from the LV or an obstacle ahead. In contrast, AEB maximizes the AV's negative acceleration in order to halt the vehicle quickly when necessary.

C. Reward

Rewards depend on maintaining or breaching a safe distance between the AV and the LV or an obstacle. Safety enhances rewards, while penalties are assigned for risky proximities, balancing safety with comfort. The simulation terminates upon collision, adding penalties to prevent future rewards.

III. Background

RL consists of three components: the actor, the environment, and the reward function. The policy inside the actor determines the actor's actions, i.e., given an input, it outputs the action that the actor should now perform. All we have to do is to adjust the policy inside the actor so that the actor gets the maximum reward. The formulas presented below have been derived from [12] and [13].

A means of optimizing a policy π to solve the problem is provided by RL

$$\theta^* = \arg\max_{\theta} R(\pi_\theta), \tag{1}$$

where π_θ denotes a policy with parameters θ and $R(\pi_\theta)$ denotes the expected finite-horizon undiscounted return of the policy, often as a neural network.

A. Policy Gradient

PG method is a common method in RL. It uses gradient ascent to maximise the expected reward

$$\nabla R(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [\Sigma_{t=0}^{T} \nabla \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)], \tag{2}$$

where τ is a trajectory and A^{π_θ} is the expected sum of rewards for the current policy. The policy parameter is updated via stochastic gradient ascent

$$\theta_{k+1} = \theta_k + \eta \nabla R(\pi_\theta), \tag{3}$$

where η is learning rate of neural network.

B. Proximal Policy Optimization

PPO is a state-of-the-art RL algorithm that belongs to the type of actor-critic algorithm. The actor is responsible for deciding which actions to take, while the critic is responsible for evaluating the actions taken by the actor. It is an on-policy algorithm, which means that it learns from the actions taken within the current policy, rather than from a separate set of data.

PPO is an improvement on the Trust Domain Policy Optimization (TRPO) [15] algorithm, which uses trust domain constraints to ensure that the new policy does not deviate too far from the old policy, thus providing stability. PPO-Clip builds on this idea by using a clipping function to limit policy changes. This allows PPO to make major policy updates while still maintaining stability.

The loss for the actor network is called Conservative Policy Iteration (CPI), which is the ratio between the policy under old parameters to the policy under new parameters multiplied by the advantage value

$$L_t^{CPI}(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, \tag{4}$$

where π_{θ_k} is old policy parameter and $\hat{A}_t$ is the advantage reward value.

PPO-Clip adds an additional parameter ϵ. With the help of ϵ, the actor loss will be calculated by taking the minimum value between the cropped and uncropped values and multiplying it by the dominance:

$$L_t^{PPO}(\theta) = \mathbb{E}_t \Bigg[\min \Bigg(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, \\ \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \Bigg) \Bigg]. \tag{5}$$

The critic loss is calculated as the Mean Square Error (MSE) between the predicted value estimate and the true value estimate. In other words, the critic loss is the MSE between the predicted value function and the true value function:

$$L_{\text{critical}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{V}(s_i) - V(s_i) \right)^2 , \tag{6}$$

where $\hat{V}(s_i)$ is the predicted value function and $V(s_i)$ is the true value function for state s_i, and N is the number of samples.

The learning process in the context of PPO-Clip model is visualised in Figure 1. This model employs an actor-critic framework where two distinct networks are utilized: the Actor Network and the Critic Network. The Actor Network proposes actions given the current state of the environment, which are evaluated both by the environment and the Critic Network. The Critic Network estimates the value function of a given state, helping in the calculation of the Advantage Function, which measures how much better an action is compared to the average. The CPI ensures that the updates of the policy are kept within a certain range, preventing large policy updates that might destabilize learning.

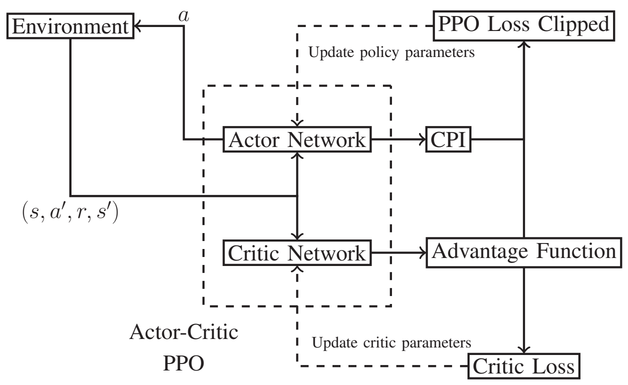


Figure 1. Reinforcement Learning PPO-Clip model

IV. Methodology

A. Simulation Environment

A simulated test environment constructed from the main test field of the *ExerShuttle* project is shown in Figure 2. The road network is mainly a closed road with two lanes. It has a maximum allowable velocity of 30 km/h and is connected to the 50 km/h road at the bottom of the figure. Since the test field is an university campus, the roadway will be relatively complex. There are private vehicles, buses, motorcycles, bicycles, and pedestrians on the road. There are no traffic lights at the 4-way stops road, so special attention must be paid to suddenly appearing vehicles and pedestrians. To reduce the reset time of the training environment, the simulated environment depicted in Figure 3. is used for training the model.

During training and evaluation, the IDM is used as the velocity control model for ACC. During the training period, the desired velocity of the vehicle is 30 km/h. The safe time headway is set to 1.5 s and the minimum distance is set to 7 m. The absolute values of both acceleration and negative acceleration are set to 1.5 m/s^2.

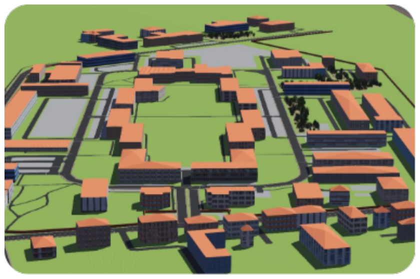


Figure 2. Simulation Environment of *ExerShuttle* Project

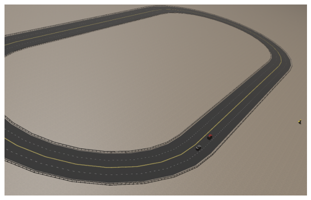


Figure 3. Simple Simulation Environment

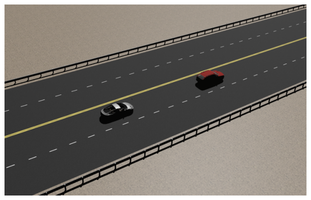


Figure 4. AV follows the LV if there is no obstacle

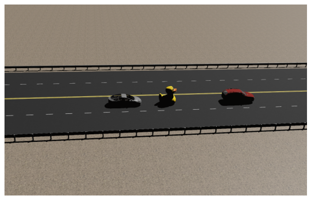


Figure 5. A yellow duck in front of the AV

Figure 6. Collision between the yellow duck and the AV

The training environment is a sequential sequence, as illustrated in Figures 4 to 6. During the training period, the AV will initially follow the LV, which is traveling at a speed of 20 km/h. After the AV has traveled for 4 s, an obstacle, such as a yellow duck, will randomly appear in front of the AV. If the car does not brake in time, the car will collide with the duck. In the absence of an obstacle, the LV will cease movement after 25 s. The AV must therefore be able to make the appropriate decision in a variety of circumstances.

B. Action Space

As stated in subsection II-B, the action space A includes only two actions: ACC and AEB. The AEB action is selected only if there is a sudden close-by obstacle in front of the AV, or if the LV applies an emergency brake. In the car-following model, the driving behavior of the AV is a comfortable driving behavior similar to human driving by setting the appropriate parameters of IDM. It is inadvisable to select the AEB action in inappropriate situations. For instance, the AV should have followed the LV using ACC, or the AV should have braked slowly and decelerated to a stop but emergency braking is selected instead. This also results in a significant decrease in passenger comfort. Mostly the AV uses ACC to follow the LV.

C. State Space

The gap between the AV and the LV and their velocities are included in the state space. The velocity of the AV is to be determined via the Global Positioning System (GPS) sensor, while that of the LV is to be gauged by the Radar. The distance of the obstacle in front of the AV will be determined by the distance sensor. The maximum range of the distance sensor is set to $50\,\mathrm{m}$. Obstacles or vehicles in front of the vehicle will be ignored when the distance is greater than $50\,\mathrm{m}$. This allows the vehicle to be driven at the maximum allowed velocity.

The selection of actions as part of the state can be described as history-dependent [16]. In partially observable environments, state information may not be sufficient to fully characterize the current state of the environment. By combining previous actions and states, an augmented state representation can be formed, which allows the policy to better capture the dynamics of the environment.

D. Reward Function

In an automated driving system, the reward functions of ACC and AEB are designed to ensure that the vehicle's behavior is safe and comfortable. Therefore, it is necessary to design the reward functions of ACC and AEB separately. The goal of ACC is to maintain a safe distance between the vehicles and the appropriate velocity. The AV should maintain a safe distance from the LV or from an obstacle, too close will be penalized:

$$R_{\mathrm{acc_{dist}}} = \begin{cases} 10, & \text{if } D_{\text{actual}} > D_{\text{safe}} \\ -10, & \text{otherwise} \end{cases}. \tag{7}$$

where D_{actual} is the current distance between the AV and the LV and D_{safe} is Distance to Collision (DTC), calculated from the TTC. The AV maintains a consistent speed with the LV, whenever possible. Excessive velocity difference will be penalized:

$$R_{\mathrm{acc_{velocity}}} = -k_1 \times V_{\text{diff}}. \tag{8}$$

The primary objective of AEB is to prevent collisions and provide safe braking in emergency situations. The occurrence of a collision is subject to significant penalties:

$$R_{\mathrm{aeb_{collision}}} = \begin{cases} -100, & \text{if collision} \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

In the event that the distance between vehicles is too close, the vehicle slows down quickly to avoid a collision:

$$R_{\mathrm{aeb_{dist}}} = \begin{cases} -10, & \text{if } D_{\text{actual}} > D_{\text{safe}} \\ 10, & \text{otherwise} \end{cases}. \tag{10}$$

In addition, AVs should avoid using the emergency brake while following. Thus, the decision to take longer to follow a vehicle in the same following situation will be penalized more severely:

$$R_{\mathrm{T}} = -k_2 \times T. \tag{11}$$

Equation (12) is the total reward function. The k_1, k_2 in (8) and (11) are weight coefficients.

$$R_{\text{totel}} = R_{\mathrm{acc_{dist}}} + R_{\mathrm{acc_{velocity}}} + R_{\mathrm{aeb_{collision}}} + R_{\mathrm{aeb_{dist}}} + R_{\mathrm{T}} \tag{12}$$

E. Training Architecture

The RL Environment is constructed using the OpenAI Gym [17] and the Webots simulator [18] in Python. Webots is an open-source application for simulating robots. It provides a Driver controller for controlling the vehicle and a Supervisor for modifying the parameters of the simulation environment. For instance, the Driver is capable of acquiring and controlling the vehicle's velocity and steering. The Supervisor is more powerful and interacts with the environment to obtain and set state variables, such as the position of an obstacle. The Driver and Supervisor sample state variables and select actions at a rate of $5\,\mathrm{Hz}$, which corresponds to a time step of $0.2\,\mathrm{s}$ in the simulation. In the event of a collision or timeout, the simulator is reset. A timeout is initiated after $100\,\mathrm{s}$ of simulation. The architectural framework is depicted in Fig 7. As presented in Subsection IV-A, Driver 1 and Driver 2 control the motion of the AV and the LV, respectively. The Supervisor obtains and transmits information about the vehicles, such as velocity or sensor data, by interacting with the parameters of the two Drivers. Furthermore, the Supervisor also controls the translation of obstacles in the environment. A gym-based simulation environment is used to train the models for RL.

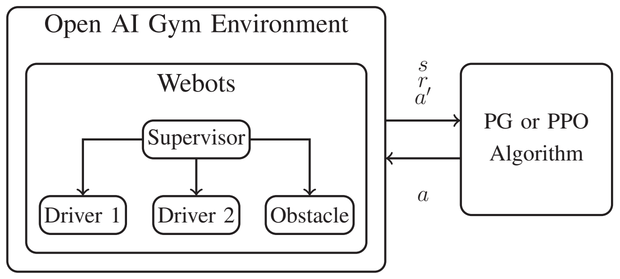


Figure 7. Training Architecture of Reinforcement Learning

The agents are trained using the online policy algorithms PG and PPO. A broader range of algorithms has not been evaluated, as our focus is on exploring the feasibility of using RL models for state switching in car-following models. In order to accommodate the training environment, the gym-based environment is rebuilt in Webots. The agent is trained using PPO-Clip for 1000 episodes on an Intel i9-8950HK and a NVIDIA Quadro P2000.

The neural network utilized during training comprises two hidden layers with a width of 256, as shown in Figure 8. The widths of the input and output layers correspond to the number of items in the state and action sets, respectively. Furthermore, the neural network is employed with ReLU and Softmax activation functions to streamline the computation and circumvent the gradient vanishing issue. The output is transformed into a probability distribution, which is also suitable for classification

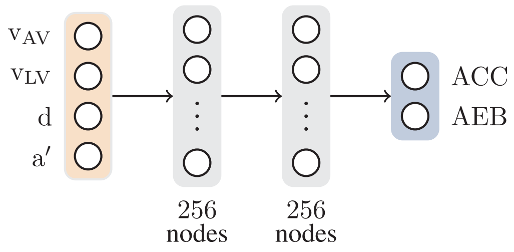


Figure 8. Simple Actor Network for PPO-Clip. Orange layer: inputs, blue layer: outputs, grey layers: hidden nodes.

tasks. Furthermore, the PPO experience pool is employed in the training process. At the conclusion of each iteration, the previous data set is discarded and a new round of data collection and training commences. The objective of this process is to ensure that when the policy is updated, the data collected based on the latest policy is utilized. The values of the key parameters of the PG and PPO algorithms are presented in Tables I and II, respectively.

TABLE I. AGENT PARAMETERS FOR PG

Parameters	*Value*
Learning rate	0.003
Discount factor	0.8

TABLE II. AGENT PARAMETERS FOR PPO-CLIP

Parameters	*Value*
Learning rate	0.0003
Number of steps	200
Number of epochs	10
Batch Size	2048
λ of GAE[a]	0.95
Clipping range	0.2
Discount factor	0.99

Note: [a]Generalized Advantage Estimation

V. EVALUATION

A. Training

The model successfully converges using both the PG and PPO-Clip algorithms. However, by adjusting the parameters of the training models, it is found that the PG algorithm is more likely to converge successfully than the PPO algorithm. For both implementations of the algorithm, the reward values begin at approximately -1800, as shown in Figure 9. The mean reward during training of the PG algprithm shwon in royal blue while the episode reward shown in light royal blue. The mean reward during training of the PPO algprithm shwon in orange while the episode reward shown in light orange. However, if the PG algorithm is employed, the reward value stabilizes at approximately 500 after approximately 150 episodes. In contrast, if the PPO-Clip algorithm is utilized, the reward value stabilizes at approximately 500 after about 50 episodes. It can be observed that the PPO model converges at a faster rate than the traditional PG model. This is primarily due to the fact that the PPO model limits the magnitude of change in the policy update step and avoids the introduction of excessive policy changes. PPO employs multi-step data sampling to optimize the policy and enhance the efficiency of data utilization.

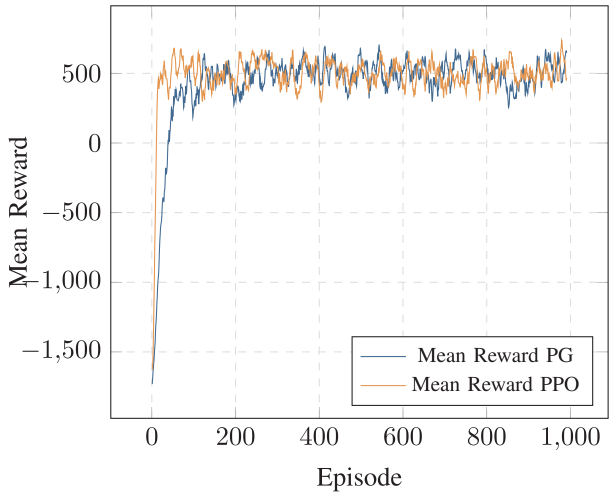


Figure 9. Training results of the PG and the PPO model

Furthermore, PPO employs additional optimizations when addressing rewards. These include the use of GAE to balance the variance and bias, and the estimation of the policy gradient with greater accuracy.

The action selection of ACC and AEB is relatively chaotic in the early stages of training, with an average of 100 to 150 episodes required for the reward value to remain stable. Once this occurs, the correctness of the action selection rate is greatly improved. Furthermore, the accuracy of the action selection also increased significantly and is maintained until the conclusion of the training period.

B. Results

The generated models has been subjected to evaluation in a simulation environment. For each evaluation, 1000 car-following tests are conducted and a random obstacle is placed in front of the AV. The results of the car-following tests are determined in two main ways: whether a collision occurred or not, and the selection of inappropriate behaviors. This includes instances where the AV braked inappropriately and collided with the obstacle and instances where the driving behavior suddenly chose the emergency braking action when it should follow the LV. If the AV is able to brake safely in front of an obstacle via the ACC system, it is also considered to be driving correctly. The results are presented in Table III.

TABLE III. COMPARISON OF DRIVING BEHAVIOR UNDER TWO ALGORITHMS

Algorithm	*Wrong behavior or Collision / %*	*AEB Selection / %*
PG	1.5	24.85
PPO	0.3	1.0

The trained models based on the PPO algorithm demonstrate superior performance overall. The mean number of erroneous behavioral choices or collisions across 1000 tests is 3. However, the mean number of instances in which the trained model based on the PG algorithm exhibited an error is 15.

When encountering obstacles, the PPO model rarely triggers the AEB, activating it less often than the PG model, which uses the AEB 24.85% of the time. This suggests that the PPO model relies primarily on the ACC system for emergency interventions. This difference can be attributed to the PG model's extensive exploration of both policy options during training, which helps it learn different emergency braking scenarios. In contrast, PPO's conservative update approach, characterized by clipped probability ratios and a targeted objective function, limits its exploration of certain actions, such as AEB. This conservative strategy may cause the PPO model to underutilize AEB in unforeseen scenarios during validation, resulting in less frequent use of emergency braking. However, this does not compromise the vehicle's ability to stop effectively, as it can still use either ACC or AEB to avoid collisions.

VI. Conclusion and Future Work

This study examines a maneuver-based decision-making approach in a simulation framework. The objective is to implement and test the selection of ACC and AEB in a car-following model using traditional PG and PPO algorithms. The results include:

- Both PG and PPO models are able to effectively select the ACC and AEB systems to follow the vehicle or emergency obstacle avoidance.
- The PPO algorithm converges faster, stabilizing at a reward value of 500 after about 50 episodes, compared to 150 episodes for the PG algorithm.
- Over multiple 1000 follow-up tests, the PPO-trained model have an average error rate of 0.3% for misbehavior or collisions, while the PG-trained model had an error rate of 1.5%.
- The simulations provide valuable insights showing that RL can automate maneuver-based decision making in driving is feasible.

The results thus far remain constrained by a number of limitations: The vehicle is capable of autonomously selecting between the ACC and AEB systems, utilizing a car-following model trained through RL. Nevertheless, the current simulation is trained in a relatively simple traffic environment. Consequently, future work should consider more diverse traffic conditions and improve the generality of the results by optimizing the training algorithm and adjusting the parameters. In addition to the ACC and AEB systems in the car-following model, it is also possible to consider the integration of systems for reasonable overtaking into the overall training environment.

The sensors utilized in vehicle simulations are still relatively simple in design. Consequently, if the results of these simulations are to be utilized in real-world environments in the future, it is imperative that the sensors employed in vehicle of the future be given greater consideration. In addition, the trained models will be validated and optimized in the *ExerShuttle* project in real world traffic. Further research should also concentrate on integrating a wider range of driving behaviours into the training models, followed by rigorous testing and validation in real-world conditions.

References

[1] F. Leon and M. Gavrilescu, "A review of tracking, prediction and decision making methods for autonomous driving", *arXiv*, 2019.

[2] Z. Zhu and H. Zhao, "A survey of deep rl and il for autonomous driving policy learning", *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 14 043–14 065, 2021.

[3] Q. Liu, X. Li, S. Yuan, and Z. Li, "Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook", in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 30–37.

[4] Y. Ye, X. Zhang, and J. Sun, "Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment", *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 155–170, 2019, ISSN: 0968-090X.

[5] M. Brackstone and M. McDonald, "Car-following: A historical review", *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999, ISSN: 1369-8478.

[6] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control", *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 143–153, 2003.

[7] L. Yang *et al.*, "A systematic review of autonomous emergency braking system: Impact factor, technology, and performance evaluation", *Journal of Advanced Transportation*, vol. 2022, F. Galante, Ed., pp. 1–13, Apr. 2022, ISSN: 0197-6729.

[8] D. Chen, Y. Gong, and X. T. Yang, "Deep reinforcement learning for advanced longitudinal control and collision avoidance in high-risk driving scenarios", *ArXiv*, 2024.

[9] P. Qin, H. Li, Z. Li, W. Guan, and Y. He, "A cnn-lstm car-following model considering generalization ability", *Sensors*, vol. 23, no. 2, p. 660, 2023, ISSN: 1424-8220.

[10] T. Li and R. Stern, "Car-following-response-based vehicle classification via deep learning", *ACM Journal on Autonomous Transportation Systems*, vol. 1, no. 1, p. 23, Mar. 2024.

[11] X. Yang, Y. Zou, H. Zhang, X. Qu, and L. Chen, "Improved deep reinforcement learning for car-following decision-making", *Physica A: Statistical Mechanics and its Applications*, vol. 624, p. 128 912, 2023, ISSN: 0378-4371.

[12] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation", in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press, 1999, pp. 1057–1063.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv*, 2017.

[14] S. Albeaik *et al.*, "Limitations and improvements of the intelligent driver model (idm)", *SIAM Journal on Applied Dynamical Systems*, vol. 21, no. 3, pp. 1862–1892, 2022.

[15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization", *arXiv*, 2015.

[16] G. Tennenholtz, N. Merlis, L. Shani, M. Mladenov, and C. Boutilier, "Reinforcement learning with history dependent dynamic contexts", in *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, Honolulu, Hawaii, 2023.

[17] G. Brockman *et al.*, "Openai gym", *arXiv*, 2016.

[18] O. Michel, "Webots: Professional mobile robot simulation", *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

You've Got a Plan? A Domain Modelling Approach for Collaborative Product Disassembly Planning with PDDL

Dominique Briechle
Institute for Software and Systems Engineering
University of Technology Clausthal
Clausthal-Zellerfeld, Germany
dominique.fabio.briechle@tu-clausthal.de

Andreas Rausch
Institute for Software and Systems Engineering
University of Technology Clausthal
Clausthal-Zellerfeld, Germany
andreas.rausch@tu-clausthal.de

***Abstract*—Product disassembly has become more and more relevant to leverage repair, refurbish and remanufature (3Rs) operations while simultaneously enabling access to spare parts from products whichs lifecycle cannot be extended. Such processes can help tackling global ecological production impact as well as overall resource shortage. Nowadays, those operations are still expensive, time-consuming and error-prone because of the high variety in overall product composition, the cost of manual labour and the limitations of disassembly systems in terms of adaption. Artificial intelligence (AI)-based planning could hereby act as a suitable solution to enable automated systems dealing with decomposition tasks. The Planning Domain Definition Language (PDDL) offers a domain-independent canvas, which is suited to deal with a broad level of compositional variety. However, the lack of a suitable systematic methods to describe hierarchical compositions in PDDL limits its application for adaptive disassembly task planning. This work, therefore, aims to overcome these limitations by proposing a methodology to describe compositions and disassembly systems. We introduce, in the scope of the paper, a formal domain meta-model, capable of depicting such hierarchical structures and therefore enabling a precise disassembly of product compositions. Finally, we conduct two disassembly planning tasks and show the applicability of our method to handle hierarchical compositions and product variety.**

***Keywords*—Collaboration, Disassembly, AI-based planning, PDDL, Circular Economy.**

I. Introduction

With the rise of global demand for products and items, especially electronic and electric products, the number of products per person is at an all-time high.

Simultaneously, product lifecycles are falling short of their actual lifespan because of lack of repair, leading to shorter product lifetime and amplifying the accumulation of electronic and electric waste [1]. The consequences are, therefore, drastically decreasing environmental quality and an increase in harmful emissions [2], which affect the quality of life around the globe.

In order to tackle the generation of new waste, product life cycles must be extended and circular economy operations like repairing, refurbishing and remanufacturing (3Rs) must be leveraged [3]. However, still a huge amount of products currently in use are not treated in a Circular Economy compatible way, which drastically limits their lifetime and at the same time reduces the possibility of repairing, refurbishing and remanufacturing products in order to reuse them [4]. The main barriers are hereby diverse and span from a lack of skilled professionals to economic factors and product-related issues like technical obsolesce and inability to upgrade [5].

In terms of cost reduction and substitution workforce, smart automated systems can compensate to a certain degree those shortcomings with the corresponding soft- and hardware tool sets [6]. In addition, automated systems offer the integration of a huge variety of tools in order to conduct adaptable operations, especially in the field of disassembly of products to prepare them for repairing, refurbishing and remanufacturing, which further elevates the economic feasibility in the long term [7].

However, based on the huge variety of products and their composition, disassembly planning of products is still a difficult process, requiring a high level of domain knowledge and technical skill, especially because of the difficulties in modeling hardware-based, hierarchical products [8]. This must be reflected as well by the automated system and requires therefore a semantically understanding of the product composition to enable adaptable disassembly planning. Additionally, different sub-systems, like multiple robots with different tools, must collaborate with one another to ensure the proper conduction of such operations, which has to be considered in the planning system as well [6].

To ensure such collaborative and adaptable approaches, AI-based planning relies on descriptive tools, like the PDDL, which can support system operators with a sufficient canvas for the generation of a sequenced disassembly plan for specific products. Although PDDL presents the opportunity to describe domain-unrelated problems, it lacks a comprehensive framework suitable for the description of hierarchical structures, which we find in physical products. Products are, therefore, composed of several sub-assemblies, relying on one another and having interconnections in order to form the product. This kind of semantic understanding is however crucial to derive an adaptive plan with atomic disassembly actions based on the individual features of a product, since different products feature a manifold of different part and connector types, let alone assemblies and modifications.

The paper, therefore, aims to propose a domain meta-model capable of describing hierarchical and complex structures, which allows the adaptive definition of product models. Additionally, the model incorporates a system for illustrating the high variety of given product compositions by extending its core entities with sub-types for the description of physical parts and connectors. Finally, an according domain is formalized in PDDL, mirroring the domain meta-model and therefore enabling the automated generation of disassembly sequence planning for different products.

The paper is structured as follows. Section II presents the related work, consisting of the background of the automated disassembly domain, followed by example use-cases as motivation, in Section III. The overall system's concept is illustrated in Section IV. The PDDL domain description and methodological background are described in Section V. Section VI contains the implementation of the domain meta-model in PDDL and the application of the system on the use-cases described in Section III. The paper closes with a discussion of the findings (Section VII) as well as a conclusion of the paper (Section VIII).

II. Related Work

The currently investigated methodologies and technologies for disassembly planning are manifold. Chang et al. [9] are listing a variety of different approaches, which can be used for this matter. This list includes classical approaches like Graph- and Petri-based planning systems as well as more autonomous methods like intelligent planning tools and algorithms. In general, Lambert et al. [10] differs between two major groups after Heemskerk et al. [11]: disassembly planning and disassembly scheduling. These groups consist of the planning of the detailed level (for sub-compositions) and the sequences necessary to disassemble them and scheduling, defining the planning of the tasks required for the process [10].

Especially for the alignment of different levels of planning into one system the PDDL is a suitable methodology, merging benefits from both the Action Description Language (ADL), developed by Pednault [12] and the Universal Method-Composition Planner (UMCP), proposed by Erol et al. [13][14].

The usage of PDDL for decomposition tasks of assemblies has already been topic in several scientific research works. Hoebert et al. [15], for example, used PDDL for the planning of an unscrewing operation conducted by robots, integrating additionally re-planning to tackle uncertainties in the setup. A similar use-case with emphasis on decision explainability of robotic disassembly was investigated by Zhang et al. [16]. PDDL is further used as adaptive planning foundation for human-robot collaboration cases, which bears resemblance to the use-case described in section four with its multi-tool collaborative aspect [17]. However, these applications are investigating on their behalf different application scenarios and key aspects.

III. Illustrating Examples as Problem Motivation

As already stated, our core motivation is to provide suitable disassembly planning for a variety of different products. As examples, we selected two distinct products, a power tool battery and a smoke detector, which consist both of a hierarchical component setup, while simultaneously consisting of just a few components, making both ideal for the small demonstration use-cases in the scope of our paper.

As can be seen in Fig. 1, both the products feature an external housing, which encloses the inner life of the product. Because of that, the logical conclusion is that those housings have to be removed in order to reach the inner components of the products. To disassemble the product, we, therefore, need certain steps, which are required to disconnect parts and connections in order. In contrast to human operators, automated systems are way less intuitive and therefore require a clear structure of operations. To disassemble the products in a similar way to Fig. 2 we, therefore, need an accurate plan, consisting of different steps, hence action sequences.

Fig. 1. Images of the power tool battery and smoke detector.

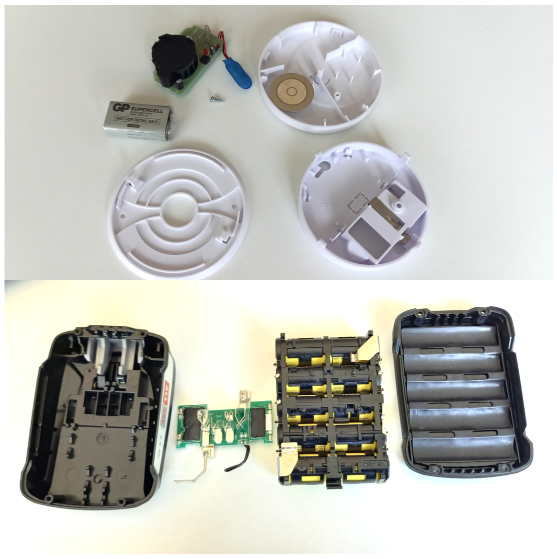


Fig. 2. Image of the disassembled products.

The key is, therefore, the derivation of a common understanding, hence a semantic, by automated entities, which allows the proper disassembly. Therefore, our domain meta-model approach must not only consider the hierarchical setup of those products but as well the adaptability of the system which is required by the overall product variety. To evaluate our proposed model, we will, therefore, model both products and generate an corresponding disassembly plan.

IV. Overall Concept

In order to reach our goal of an automated, adaptable disassembly system, we derived a concept from our general idea consisting of several sub-systems steps, as shown in Fig. 3.

The disassembly system consists of three overall sections, covering different sub-systems which are responsible for interacting with one another. The system's physical component is the **Disassembly Line**, consisting of the **Scanning device** and the **Disassembly Tools**. The idea of the concept is to capture the external features a product by recording it with an optical device, resulting in a digitized model of the product. This happens through different perceptive sensor units, like 3D cameras, capable of recording a realistic digital depiction of our product.

The product will then be identified by the so-called **Product identifier**, who is responsible for matching the products' system ID with the components of the **Knowledge Base**. This section consists of two subbranches, the **Product Assembly Description Library (PADL)** and the **Disassembly Action Library (DAL)**. The matching of the **Product identifier** is made against the product models contained in the **PADL**. These models consist of a textual description of the product setup, usable as problem definition for the systems' **Planner** and containing the information required for the generation of a sequence plan such as composition of products and sub-assemblies, type of parts and connectors and the relation between these entities. The **DAL** on the other hand contains the counterpart of the planning system to the **PADL** and consists of the set of actions and their descriptions necessary to enable the disassembly planning. They, therefore, have requirements which have to be fulfilled by the systems environment in order to apply them, such as connection types or state conditions. It is, therefore, the component of the system that ensures the adaptability in terms of disassembly plan generation-based on the different incoming models of the **PADL**. Based on the matching of the product model, the **Planner** is using the corresponding set of actions from **DAL** and will generate a sequenced plan containing atomic disassembly actions.

This plan will be taken in by the **Executor**, which is responsible of mapping the action-based plan into command, which can be processed by the **Disassembly Tools'** controllers. Via the respective controllers, the **Disassembly Tools** are able to disassemble the incoming products in a collaborative manner corresponding to the generated plan.

In the scope of this paper, we will focus on the part of our Adaptive Product Disassembly System responsible for the planning of our action sequences, hence our approach consisting of a domain meta-model and, derived from that, our PDDL domain.

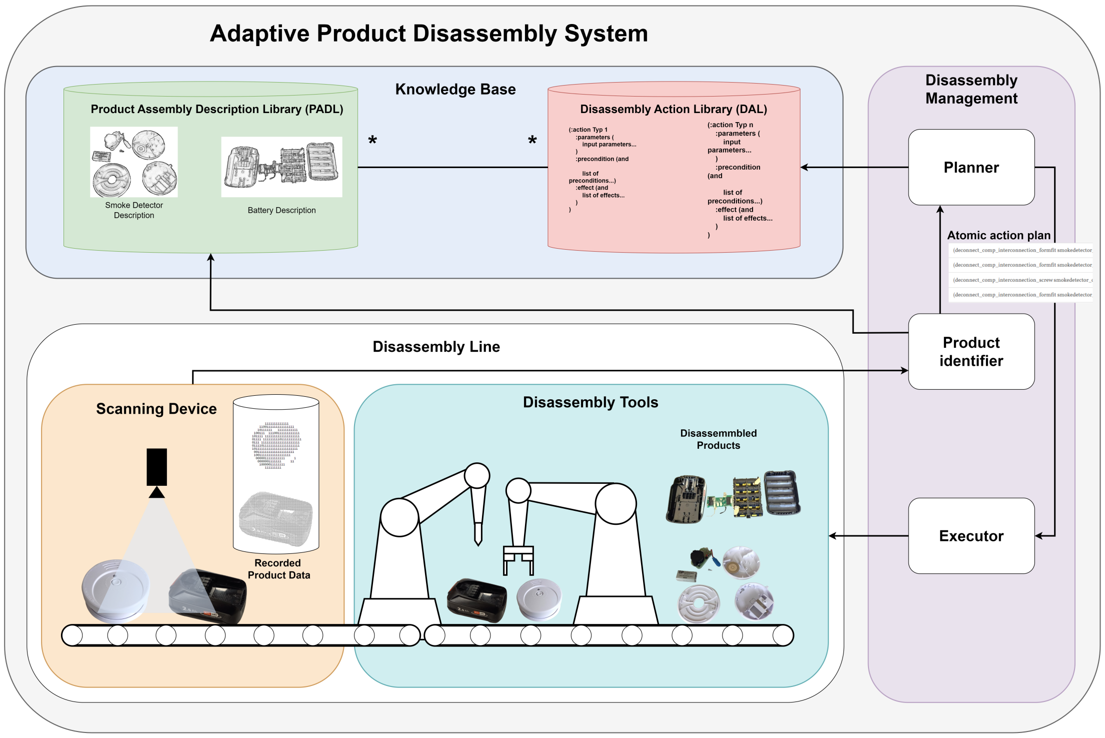
V. Domain Meta-Model

The domain meta-model, shown in Fig. 4, builds the systematic foundation of the disassembly planning system. It provides the structure required to describe the components and the setup of our product (described in the green block of the figure) and the corresponding actions necessary to dissolve a given link between the model's entities (red block). The green-marked section of the illustration can therefore be viewed as the product assembly concerning part of the model, the **PADL**, while the red part deals with the representation of the external tools and their effect on the model's structure, hence the actions reflected by the **DAL**.

The general idea of this kind of model description is derived from the block-based software architecture domain [18][19], containing therefore similar elements adjusted to our hardware-based domain. The product model therefore incorporates the four major entities parts, connections, compositions and connectionports, which reflect our domain and enable the construction of product models and their assembly groups on the different levels of the product's assembly hierarchy [18][19].

The model's entities are hereby similar to the once [18] used to describe components for architectural conceptions of software systems. The detailed functionalities of the system's entities and links are described in detail in the following section.
The first sections contains the entities regarding the **PADL**:

- *Composition*: The *Composition* provides the descriptional context of a specific product and the adjunct sub-assemblies. Therefore every hierarchy level and assembly group has an associated *Composition* which has a systemic link to the adjunct *Parts*, *Connections* and *ConnectorPorts*. The *Composition* therefore bridges the different hierarchical levels in a product's structure and enables the domain to dissect the *Connections* from the *Parts*.
- *Part*: The *Part* reflects one of the two physical elements in our domain meta-model. *Parts* are, therefore, the "hardware" of a product and are crucial for the disassembly of the same since they are one of two component types. Based on the type of product, the disassembly planning has to take different compositional hierarchies of specific components into account. *Parts* can therefore have a superior and subordinate *Composition*, which defines the product's structure and the *Part's* setup in the disassembly model. In case of a subordinate *Composition*, the *Part* has a sub-composition which consists of subordinate *Parts* and *Connections*, whereas if the *Part* is part of a superior *Composition*, it is a piece of a governing assembly.
- *ConnectorPort*: *ConnectorPorts* are the instances of the connector-intakes of the assembly's *Connections*. Each *Part* has therefore a minimum of one *ConnectorPort* to



```
(:action Typ 1
    :parameters (
        input parameters...
    )
    :precondition (and

        list of
preconditions...)
    :effect (and
        list of effects...
    )
)
```

```
(:action Typ n
    :parameters (
        input
parameters...
    )
    :precondition
(and

        list of
preconditions...)
    :effect (and
        list of effects...
    )
)
```

```
(deconnect_comp_interconnection_formfit smokedetector_
(deconnect_comp_interconnection_formfit smokedetector_
(deconnect_comp_interconnection_screw smokedetector_c
(deconnect_comp_interconnection_formfit smokedetector_
```

Fig. 3. Illustration of the overall concept.

be connected to an adjunct *Connection* in order to place it into a larger assembly, hence a *Composition*. *ConnectorPorts* can intake on their side multiple *Connections* to be able to design compositions with different setups and *Connection* relations.

- *Connection*: The *Connection* link builds the entities in the meta-model which are responsible for the inter-linkage between the *Parts*. The *Connection* features the attribute ”hierarchy”, which defines the *Connection* to be one of two possible types: *Interconnections*, which mirror physical entities internally in a specific sub-composition and *Transconnections*. These *Transconnections* are the type of *Connections* which enable our meta-model to describe hierarchical *Compositions*. They therefore reflect physical *Connections* which tie different *Compositions* in an assembly to one another. In a physical *Composition*, the *Connections* are reflecting components like screws, solder joints or nails, which are establishing the connection between the *Parts* of an assembly. The *Connections* can be extended, as illustrated in Fig. 4, by different subtypes to match the setups of specific products.

The portions of the meta-model concerning the disassembly actions are contained, as already mentioned, in the **DAL**:

- *DisassemblyAction*: The *DisassemblyAction* is a representation of the actions required to loosen a certain *Connection*. It is therefore directly linked with the corresponding operation to the *Connection*. As the name implies, the action is not an entity of the hierarchical product model and is therefore presented in another color (red). Rather than presenting a component of a product, the *DisassemblyAction* is a representation of the functionality of the disassembly tools required to dismantle certain components. To highlight this aspect, the for the different *Connections* represented extensions are mirrored here by the action extensions.

The different entities of the model are connected to one another via systemic links. These links describe the relation of the entities to one another on a meta-level and are the foundation for the conception of the PDDL actions which we use in order to design our PDDL domain. The following section contains a brief description of the different connectors:

- *has_Comp*: *has_Comp* links a *Part* to a subordinate *Composition*. It therefore describes a relation of a higher level *Part* connected to a lower *Composition*, describing therefore the *Composition* of the sub-assembly of a superior structure.
- *has_Con*: *has_Con* is the interlink between a superior *Composition* and a subordinate *Connection* in the assembly. It is therefore one of two kinds of systematic links to hardware components.
- *has_Part*: The opposite of the aforementioned *has_Con* is the *has_Part* link. It connects in our model a superior *Composition* to a subordinate *Part*.
- *Con_has_CP*: The link *Con_has_CP* describes the relation between a *Connection* and a *ConnectorPort*. Therefore each *Connection* is linked to at least two different *ConnectorPorts* in order to reflect the inlays of a specific *Part*.

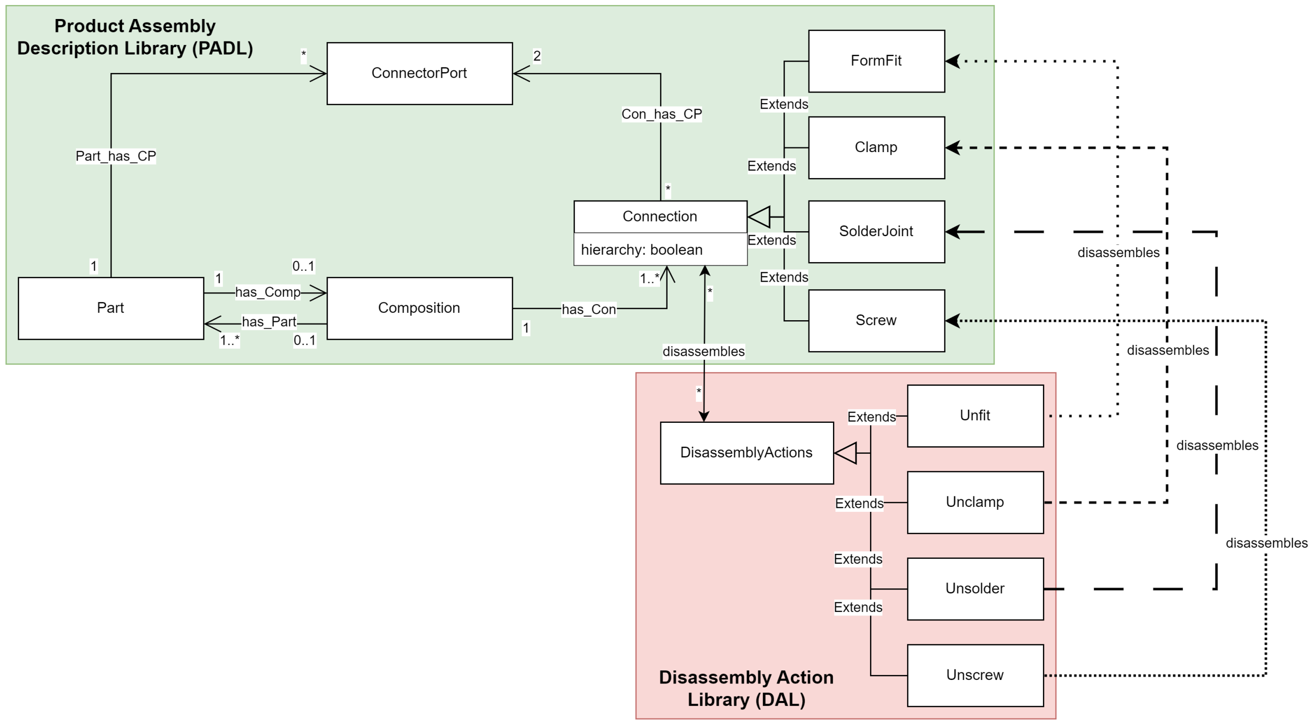


Fig. 4. Domain meta-model for the conception of product models.

- *Part_has_CP*: As the counterpart to the before-mentioned link, the *Part_has_CP* links the *Part* to the specific *ConnectorPorts*. A *Part* must therefore have at least one connected *ConnectorPort* in order to be considerated by an assembly, respectively being disassembled.

VI. Implementation in PDDL

This section describes the foundational idea of PDDL as well as the implementation of the suggested domain model in the definition language.

A. *Planning Domain Definition Language*

The technical foundation for the used planning system is, as already mentioned, the Planning Domain Definition Language. The Planning Domain Definition Language allows the description of a specific domain. It, therefore, enables the definition of certain constraints and effects, which need to be fulfilled in order to alter the ”world”, meaning the environment of the model, in a certain way. The planning system consists of three distinct elements: the domain, the problem and the planner. The domain and problem are formulated with the help of the PDDL and describe the world's boundaries in types and conditions as well as its instantiation for a specific scenario. The domain hereby contains the formulation of the world's restrictions and conditions. These global conditions are represented in the domain as **(:predicates)**, which are at some point in a plan either true or false and are therefore the core aspects, which are defining the interactions in the world.

The abstract entities in the system are described in the PDDL in a hierarchical matter in the form of **(:types)**. These **(:types)** define the inheritance between different entities allowing, therefore, systems with sub-types.

The central element of the domain are however the **(:actions)**. Those **(:actions)** are the ”moving” parts of the system, allowing the planner to create a sequenced row of steps with their corresponding input **(:parameters)**. They therefore have certain **(:preconditions)** defined in order to be ”activated” as well as certain **(:effects)** which will result by conducting an **(:actions)**. As a result of a successfully conducted **(:action)**, the certain **(:effects)** will be triggered, which is an alteration of the world based on the formulated outcome.

While the domain describes the modeling constraints and rules of our world, the problem relates to a concrete instantiation of a certain model. Therefore, the problem contains in our case the description of our product. The components of the product are hereby defined in **(:objects)**, which reflect the instances of the before defined **(:types)** of the domain. The **(:objects)** section is, therefore, composed of all of the entities of an instantiated environment. The core component of the problem is, however, the **(:init)**. This section of the problem contains the initial state of our product, hence the world instantiation. The initialisation of the world's stage is in PDDL conducted by initializing the formulated **(:predicates)** by setting them as true or false. Finally, the goal stage of the resulting plan is formulated in the problem. This goal is formulated as well out of a certain instantiated **(:predicates)** which have to be either true or not at the end of a planning cycle [14].

B. PADL Meta-Model implementation

With the help of our **PADL** section in the meta-model, the foundation for the PDDL specific **(:types)** and **(:predicates)** is set. The usage of **(:types)** requires in PDDL the setup of a specific requirement, namely the requirement **(:typing)**. As can be seen in the following code Lst. 1, the **(:types)** are directly aligned to the entities in the meta-model. For the base model, the sole attribute is the one for the *Connections* in order to describe the hierarchical correlation between the different layers. The extensions depicted in Fig. 4 are, therefore, the same as in PDDL and will be used in the PDDL domain as well as in the problem of our two example models.

```
(:requirements :typing)
(:types
    part - object
    connectorport - object
    connection - object
    composition - object
    interconnection - connection
    Transconnection - connection

)
```

Listing 1. Typing of the PDDL domain.

The different core elements, typed as objects, can be extended in order to outfit them with fitting subordinates. This can be e.g. specific types of connectors like screws and bolts or types of parts like housings, cells and motors, depending on the targeted product.

The foundation for the later defined **(:actions)** of our PDDL domain are the **(:predicates)**. These **(:predicates)** are directly derived from our meta-model and describe, as already mentioned in the Related Work section, the state functions of our world and can therefore be either true or false for the given objects and their instances in the PDDL problem. As it is the case in our meta-model, the **(:predicates)** are a representation of the links between the system's entities. The formulated **(:predicates)** are, therefore, establishing the link to the respective elements in our PDDL domain, as can be seen in Lst. 2.

```
(:predicates
(comp_has_cp     ?part - part
                 ?connectorport - connectorport)
(has_comp        ?part - part
                 ?composition - composition)
(has_part        ?composition - composition
                 ?part - part)
(has_con         ?composition - composition
                 ?connection - connection )
(con_has_cp      ?connection - connection
                 ?connectorport - connectorport)
)
```

Listing 2. Domain predicates.

C. DAL Meta-Model implementation

As shown above, the **(:actions)** are contained in the **DAL** section of the model. The **(:actions)** are formulated in PDDL on the basis of the **(:predicates)**. Here, the **(:parameters)**, **(:preconditions)** and resulting **(:effects)** are defined, which will be the outcome of a conducted **(:action)**. The **(:parameters)** are, therefore, taking the specified input variables into account, on which the **(:action)** acts on. The PDDL domain, based on the meta-model, consists of four basic **(:actions)**:

- *disconnect_composition-interconnection*: This **(:action)** is designed to cut the systematic link between a superior *Composition* and its subordinated *Interconnections* (Lst. 3). It therefore ensures, that all links, which are part of a certain *Composition* are decoupled, before the disassembly process continues.

```
(:action
disconnect_composition-interconnection
    :parameters (
        ?comp - composition
        ?i1 - interconnection
        ?p1 - part
        ?p2 - part
        ?c1 - connectorport
        ?c2 - connectorport
    )
    :precondition (and
        (has_con ?comp ?i1)

        (forall (?deleg - transconnection)
            (not(has_con ?comp ?deleg))
        )
        (forall (?parts - part)
            (not(has_comp ?parts ?comp))
        )
        (part_has_cp ?p1 ?c1)
        (part_has_cp ?p2 ?c2)
        (has_part ?comp ?p2)
        (has_part ?comp ?p1)

        (con_has_cp ?i1 ?c1)
        (con_has_cp ?i1 ?c2)
        (not(= ?c1 ?c2))
        (not(= ?p1 ?p2))
    )
    :effect (and
        (not(has_con ?comp ?i1))
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i1 ?c2))
    )
)
```

Listing 3. Action: disconnect_composition-interconnection.

- *disconnect_composition-Transconnection*: The **(:action)** is functioning in the same manor as the before mentioned *disconnect_composition-interconnection* action, but deals with the sub-type *Transconnection* (Lst. 4). It, therefore, decouples all of the *Transconnections* of a given *Composition* and must be carried out before the *Interconnections* of the *Composition* are disconnected. It is, therefore, responsible for releasing the hierarchy spanning links.

```
(:action
disconnect_composition-transconnection
    :parameters (
        ?comp - composition
        ?t1 - transconnection
        ?p1 - part
        ?p2 - part
        ?c1 - connectorport
        ?c2 - connectorport
    )
    :precondition (and
        (has_con ?comp ?t1)
        (forall (?allcomp - composition)
            (not(has_part ?allcomp ?p1))
        )
        (has_comp ?p1 ?comp)
        (has_part ?comp ?p2)
        (part_has_cp ?p1 ?c1)
        (part_has_cp ?p2 ?c2)
        (not(con_has_cp ?t1 ?c1))
        (not(con_has_cp ?t1 ?c2))
        (not(= ?c1 ?c2))
        (not(= ?p1 ?p2))
    )
    :effect (and
        (not(has_con ?comp ?t1))
        (not(con_has_cp ?t1 ?c1))
        (not(con_has_cp ?t1 ?c2))
    )
)
```

Listing 4. Action: disconnect_composition-transconnection.

- *disconnect_part-composition*: To continue with the disassembly process, the connection between a *Part* and its subordinate *Composition* must be dissolved (Lst. 5). The action, therefore, checks as **(:preconditions)**, that the specific *Part* isn't bound anymore to a superior *Composition*.

```
(:action
disconnect_part-composition
    :parameters (
        ?part - part
        ?comp - composition
        ?c1 - connectorport
        ?c2 - connectorport
        ?i1 - interconnection
        )
    :precondition (and
        (forall (?over - composition)
        (not(has_part ?over ?part))
        )
        (has_comp ?part ?comp)
        (part_has_cp ?part ?c1)
        (part_has_cp ?part ?c2)
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i1 ?c2))
        (not(= ?c1 ?c2))
     )
    :effect (and
        (not(has_comp ?part ?comp))
    )
)
```

Listing 5. Action: disconnect_part-composition.

- *disconnect_composition-part*: This **(:action)** is carried out in order to dissolve the systematic connection between a superior *Composition* and its subordinate *Part* (Lst. 6). The **(:precondition)** is, therefore, that the *Part* does not have any *Connection* to a *Part* of a superior *Composition*.

```
(:action
disconnect_composition-part
    :parameters (
        ?comp - composition
        ?part - part
        ?c1 - connectorport
        ?c2 - connectorport
        ?i1 - connection
        ?i2 - connection
    )
    :precondition (and
        (part_has_cp ?part ?c1)
        (part_has_cp ?part ?c2)
        (has_part ?comp ?part)
        (forall (?links - connection)
            (not(has_con ?comp ?links)
        )
        )
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i2 ?c2))
        (not(= ?c1 ?c2))
        (not(= ?i1 ?i2))
    )
    :effect (and
        (not(has_part ?comp ?part))
    )
)
```

Listing 6. Action: disconnect_composition-part.

D. Demonstrator: Power tool battery & Smoke detector

Considering the upper presented domain meta-model and general PDDL domain, we can now define a specified PDDL-domain and problem in order to formulate our disassembly scenario and depict collaborative disassembly approaches for our power tool battery and smoke detector. We therefore have to use the proper extension to describe the parts, connections and tools in a more detailed and domain-specific matter, as shown exemplary in the composition structure diagram Fig. 5. This is necessary in order to assign the different **(:actions)** to the specific tools, which is required to disassemble a specific sub-type of *Connection*, e.g. screw with a screwdriver, and therefore, to enable disassembly in a collaborative matter by dividing tasks between the different tools.

We extend the types *Interconnection* and *Transconnection* with the four different sub-types solder-, screw-, clamp- and formfit- trans/interconnection. The extension of the hierarchy offers the ability to reduce the potential scope of an **(:action)** to a specific sub-type. The **(:actions)** are, therefore, building the framework necessary for our collaborative and automated disassembly environment. It enables our system to plan and assign tasks based on different requirements, e.g. **(:preconditions)**, to different tools and, therefore, using the unique features of the specific collaborator.

The PDDL problem, as mentioned before, reflects the instantiated model and contains all of the components of the product. This is of course highly individual and dependent on the product type and sometimes even the individual product. The **(:objects)** are, therefore, containing all the described components of the product, while the **(:init)** contains their initial condition (instantiated **(:predicates)**, hence the product setup (illustrated in Fig. 5 for one of our example cases)). The **(:goal)** of this demonstrations is formulated as a dissection of the battery's cells and the smoke detectors connection between the battery and the microcontroller, which is respectively placed on the lowest hierarchy level of both of the product's composition. This results in two plans, as shown in Fig. 6 and Fig. 7, consisting of 25 actions for the power tool battery and 15 actions for the smoke detector. For solving the plan,

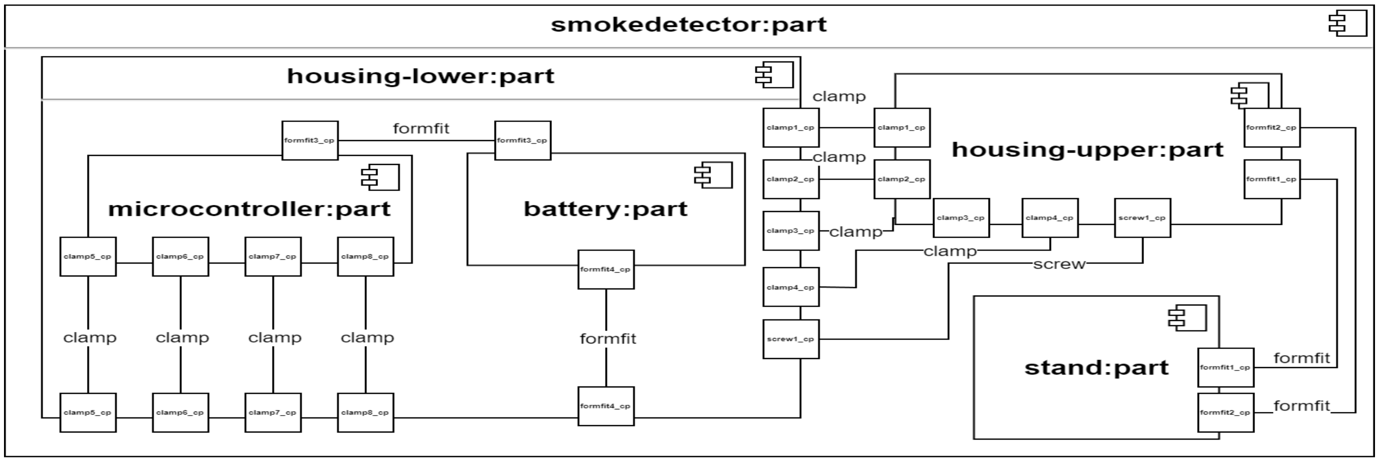


Fig. 5. Composition Structure Diagram of our modeled smoke detector.

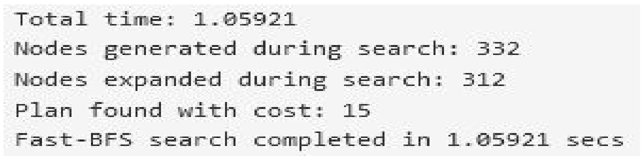

```
Total time: 1.05921
Nodes generated during search: 332
Nodes expanded during search: 312
Plan found with cost: 15
Fast-BFS search completed in 1.05921 secs
```

Fig. 6. Planner Output of the smoke detector disassembly resulting in 15 actions until reaching the goal state.

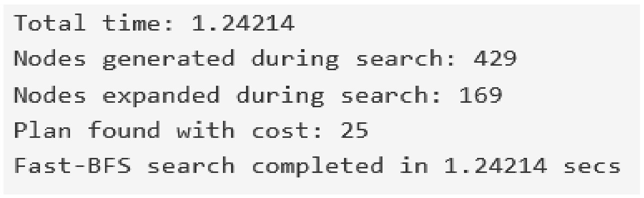

```
Total time: 1.24214
Nodes generated during search: 429
Nodes expanded during search: 169
Plan found with cost: 25
Fast-BFS search completed in 1.24214 secs
```

Fig. 7. Planner Output of the power tool battery disassembly resulting in 25 actions until reaching the goal state.

a Best First Width Search (BFWS) solver was used together with a Fast Forward (ff) parser, which proved to be the best performant solver for both our problems.

The PDDL domain proves to be applicable for both our defined use-cases and shows, how domain meta-modeling can be used as a foundation to derive a suitable domain definition for hierarchical decomposition planning. The extensions of the sub-types have been used in both use-cases, enabling the coupling of certain operations not only to a specific **(:action)**, but also enabling further specification of a task through the used parts and connectors. This results in a more universal PDDL domain, applicable for different disassembly planning tasks.

VII. Discussion

The proposed meta-model shows how a unified structure on the one hand supports disassembly planning by providing a common ground for specific disassembly problems, while on the other showing the applicability of the Planning Domain Definition Language for describing component-based assembly structures. Additionally, it provides an easy-to-use foundation, which allows the integration of different tools based on systematic links of the model's entities.

A. Theoretical Contribution

The theoretical contribution of the paper consists of the domain meta-model and its framework for the domain extensions of the entities based on the instantiation for a specific product. The meta-model provides the theoretical foundation for the formulation of component-based assembly structures. The hierarchical concept depicted by the *Connections* and its two base attributes allows the formulation of more complex *Compositions* to be solved. The model can, therefore, be used as a starting point for further research regarding product disassembly planning via PDDL.

B. Practical Use-Case

The practical implementation shows the application of the formerly described domain meta-model in the context of actual product use-cases in order to demonstrate its effectiveness in disassembly planning scenarios. Especially the extensions of the entities with adjunct sub-types opens up the possibility to design planning models for collaborative scenarios. The applied extensions are, therefore, shown in two different use-cases which feature both the basic expandability of the presented meta-model. The representation of different physical connectors formulated as sub-types mimics the actual composition of a power tool battery and a smoke detector, as shown in Fig. 5. However, the current disassembly tasks have only been simulated on a planning level. The system's overall applicability has yet to be determined in real-world, hardware-driven scenarios with actual automated tools.

C. Model Limitations and Future Research

The current model is suitable to describe and, therefore, plan hierarchical composition scenarios. However, the model has in its current state a major limitation, which affects its application for real-world applications, since it does not take the state of products into account. The state of the physical components affects the decomposition planning drastically, hence given uncertainties for planning. Defect screws can lead for example to the inability to use a screwdriver as a primary tool and, therefore, leverages a huge impact on an adjunct robotic tool. Currently, the depiction of the state is not fully integrated into the base model, which limits its application to only ”factory new” composition disassembly plans and the states of ”present/not present” for certain components. However, it is planned by the authors to extend the model to allow the proper depiction of the state of different components and to design a methodology to take the same into account during planning.

VIII. Conclusion

The paper describes a system concept, which is capable of disassembling products based on the items individual setup. The main contribution is hereby the meta-model with its instantiated PDDL domain, which enables the generation of highly adaptable disassembly action plans based on the defined product model.

It provides a common ground for depicting disassembly problem modeling and offers an extendable core, usable for a variety of problems. This extendable core further allows the representation of different disassembly tools and enables the automated and adaptive conduction of disassembly steps by the hardware components.

The model, therefore, offers insight into the general mechanisms of the cross-entity interaction and enables the application for a large variety of use-cases. Future research should focus on the aforementioned limitation, especially on the extension of the parameters to depict disassembly problems on a more detailed level and the capability of the system to deal with uncertainties and derivations regarding the product state. Furthermore, the base model can be used in different contexts in order to display its generality for cross-domain applications.

Acknowledgement

This work was conducted in the scope of the project ”Life_TWIN” by the Federal Ministry for Economic Affairs and Climate Action (Research Grant: 03EI5014A).

References

[1] J. Bachér, Y. Dams, T. Duhoux, Y. Deng, T. Teittinen, and L. F. Mortensen, “Electronic products and obsolescence in a circular economy,” *Eionet Report - ETC/WMGE*, no. 3, 2020.

[2] Y. Sitaramaiah and M. K. Kumari, “Impact of electronic waste leading to environmental pollution,” *Journal of Chemical and Pharmaceutical Sciences*, no. JCHPS Special Issue 3, 2014.

[3] M. A. Khan, S. Mittal, S. West, and T. Wuest, “Review on upgradability - A product lifetime extension strategy in the context of product service systems,” *Journal of Cleaner Production*, no. 204, pp. 1154–1168, 2018.

[4] X. Zhang, L. Zhang, K. Y. Fung, B. R. Bakshi, and K. M. Ng, “Sustainable product design: A life-cycle approach,” *Elsevier Chemical Engineering Science*, no. 217, 2020.

[5] N. Roskladka, A. Jaegler, and G. Miragliotta, “From “right to repair” to “willingness to repair”: Exploring consumer’s perspective to product lifecycle extension,” *Journal of Cleaner Production*, no. 432, 2023.

[6] G. Foo, S. Kara, and M. Pagnucco, “Challenges of robotic disassembly in practice,” *29th CIRP Life Cycle Engineering Conference*, no. 29, pp. 513–518, 2022.

[7] K. Wegener, W. H. Chen, F. Dietrich, K. Dröder, and S. Kara, “Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries,” *The 22nd CIRP conference on Life Cycle Engineering*, no. 22, pp. 716–721, 2015.

[8] S. L. Soh, S. K. Ong, and A. Y. C. Nee, “Design for Disassembly for Remanufacturing: Methodology and Technology,” *Procedia CIRP 15*, no. 15, pp. 407–412, 2014.

[9] M. M. L. Chang, S. K. Ong, and A. Y. C. Nee, “Approaches and Challenges in Product Disassembly Planning for Sustainability,” *The 27th CIRP Design*, no. 27, pp. 506–511, 2017.

[10] A. J. D. Lambert, “Disassembly sequencing: a survey,” *International Journal of Production Research*, no. 41, pp. 3721–3759, 2003.

[11] C. Heemskerk, L. Reijers, and H. Kals, “A Concept for Computer-Aided Process Planning of Flexible Assembly,” *CIRP Annals*, vol. 39, no. 1, pp. 25–28, 1990, ISSN: 0007-8506.

[12] E. Pednault, “ADL: Exploring the middle ground between STRIPS and the situation calculus,” *International Conference on Principles of Knowledge Representation and Reasoning*, no. 1, pp. 324–332, 1989.

[13] K. Erol, J. A. Hendler, and D. Nau, “UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning,” *Conference on Artificial Intelligence Planning Systems (AIPS)*, no. 2, pp. 249–254, 1994.

[14] D. McDermott, M. Ghallab, A. Howe, *et al.*, “PDDL-The Planning Domain Definition Language,” *Tech Report CVC TR-98-003/DCS TR-1165*, 1998.

[15] T. Hoebert, D. Neubauer, M. Merdan, W. Lepuschitz, Thalhammer, and M. Vincze, “ROS-driven Disassembly Planning Framework incorporating Screw Detection,” *International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, no. 20, 2023.

[16] Y. Zhang, H. Zhang, W. Zhigang, S. Zhang, H. Li, and M. Chen, “Development of an Autonomous, Explainable, Robust Robotic System for Electric Vehicle Battery Disassembly,” *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 409–414, 2023.

[17] S. U. Lee, A. Hofmann, and B. Williams, “A Model-Based Human Activity Recognition for Human–Robot Collaboration,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 736–743, 2019.

[18] C. Deiters, “Description and consistent composition of building blocks for the architectural design of software systems,” Ph.D. dissertation, University of Technology Clausthal, Clausthal-Zellerfeld, GER, 2015.

[19] S. Herold, “Architectural Compliance in Component-Based Systems - Foundations, Specification, and Checking of Architectural Rules,” Ph.D. dissertation, University of Technology Clausthal, Clausthal-Zellerfeld, GER, 2011.

Automating Benchmarking Process for Multimodal Large Language Models (MLLMs) in the Context of Waste Disposal

Sundus Hammoud
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
e-mail: sundus.hammoud@tu-clausthal.de

Robert Werner
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
e-mail: robert.werner@tu-clausthal.de

***Abstract*—Multimodal Large Language Models (MLLMs) are systems that can handle both text and non-text input by the user. They can also be prompted to follow certain instructions that can influence their behavior. These capabilities make them an excellent candidate for waste disposal classification. However, these models are trained on general knowledge, and they fail to answer simple questions about recycling because local recycling rules vary across regions. In addition, language models tend to respond in long and detailed text, which makes it very daunting for a human to go through thousands of lines of text while benchmarking such models to evaluate their answers. We propose an approach to automate the benchmarking process in the context of waste disposal and minimize human intervention by introducing a Large Language Model (LLM) to evaluate the answers of another LLM. We also leverage the prompting strategies to achieve this and to resolve the region-based recycling rules problem. We achieved promising results and sped up the benchmarking process significantly by saving researchers from hours of manual evaluation.**

Keywords-Large Language Model; Multimodal Large Language Model; Benchmarking, LLM-as-a-judge.

I. Introduction

Germany has always been one of the leading countries in the field of sustainability. Having a successful recycling system allows us to push the circular economy forward and decrease the dependency on raw materials, saving us from exploiting some of the non-renewable materials like plastic. Recycling also plays a significant role in reducing the waste landfill sizes and therefore protecting the environment from the emissions of toxic greenhouse gases [1].

However, Germany faces a common issue: people often dispose of garbage in the wrong bins. This is often due to confusing recycling rules and limited access to reliable information. A survey in Germany [2] shows that 60% of German citizens lack detailed information on the correct disposal and separation of packaging and household waste. Wrongly disposed waste cannot be used for recycling or even hinders the recycling of materials in the same bin. While many disposal companies offer guidelines and sometimes even human support for waste separation, a low-barrier support bot would be more accessible to users and more affordable.

Businesses have been searching for ways to tackle this situation with the rise of Large Language Models (LLMs) that can interact with only text-based input from a user, and Multimodal Large Language Models (MLLMs) that can handle both text and non-text input. This means that they can interact with images and answer questions about them. This has influenced many businesses to introduce chatbots based on such models to interact with users where they can ask recycling questions and upload images.

Large foundational models currently available are trained on general knowledge regarding waste disposal like Qwen [3] or Llama [4] herds of models. Unfortunately, this is not enough for a model to be useful in Germany, because every region has its own recycling rules, for example, organic waste must be disposed of in the black bin in the city of Goslar but in the green bin in the city of Wolfenbüttel [5]. So, the model must be provided with regional-specific disposal information, according to the region, in which its being used.

Benchmarking a chatbot-based assistant is challenging. Evaluating text answers manually can be a very time-consuming task, because the generated answers are often long and detailed, which makes it extremely difficult to go through thousands of text paragraphs.

Lastly, the research process usually contains many iterations to enhance the algorithm or tweak the model's parameters. So, it is imperative to have a quick evaluation to speed up the process and let the researcher focus more effectively on other aspects of the research.

For these reasons, an automatic benchmarking system is necessary. However, benchmarking a chatbot that has no machine-readable interface but is rather designed to use human language is not possible to achieve algorithmically. Instead, natural language needs to be evaluated with all of its nuances and context.

The rest of the paper is organized as follows: In Section II, the paper proceeds with related work that is similar to the proposed approach and similar projects. Section III focuses on the proposed approach, including data preparation, classification, automated evaluation, and benchmarking evaluators against a human. In Section IV, the results are presented, and then insights about the results are discussed in Section V. Lastly, the paper closes with a conclusion and outlook in Section VI.

II. Related Work

Along with the variety and improvement of LLMs came the problem of comparing them reliably. Recently, there has been a growing interest in investigating the ability of LLMs to evaluate texts and even ones generated by other language models. The most important advantage of using LLMs as a judge is that they provide scalability, meaning that they allow for benchmarking many models without the need for a human to spend hours evaluating the performance of a certain model. This also means that comparing the performance of multiple models will take significantly less time [6]. One of the first applications, BERTScore, evaluates text generation and assigns a score that is meant to align with human evaluation [7]. Similarly, LLMs have previously been utilized to generate training data [8] or compare the performance of different LLMs [9], e.g., by generating grading categories and scores [10].

Automating a benchmarking system to evaluate LLMs that are instructed to follow certain tasks is also discussed in [11]. The authors selected text classifiers to do the evaluation task. These classifiers output a score between 0 and 1. When a score is higher than 0.5, it is considered to belong to the category the LLM is evaluating for.

In [12], an LLM is used to evaluate the output of another LLM using a scale of 5 points. Then, the authors compare the LLM evaluation with a human evaluation. The evaluated task is the ability to generate a story based on a given prompt. They evaluate two groups of models, namely, open source models like Llama:7b and Llama2:7b/13b and commercial models like GPT-4-turbo and GPT-3.5-turbo.

Unlike approaches with scoring methods, and evaluating models for general purposes, we propose a benchmarking system that adapts to regional variations. We significantly reduce the time and effort of manual evaluation and introduce a new insight into handling recycling complexities.

III. Proposed Approach

This chapter covers our approach to benchmarking the support bot based on an MLLM. Firstly, we explain how data for the benchmark was collected and prepared. Then, the automated benchmarking system will be introduced. It consists of two main phases, the classification phase and the evaluation phase. The classification phase will be responsible for prompting the Multimodal Large Language Model (MLLM) used for classification with the recycling rules and collecting answers from the model. In the evaluation phase, these answers will be judged by another model and the system finally outputs the results. Lastly, we also compare a few LLMs that claim to have high linguistic capabilities to select the best judge model for our benchmarking system.

A. Data Preparation

Benchmarking a recycling system requires a well-designed dataset that spans the general recycling bins and represents objects in a realistic environment. There are 6 categories represented in this dataset, first of all, we have the main four bin categories, which are yellow, blue, green, and black. Two more containers are added which are clothes containers and glass containers to dispose of cloths and glass objects respectively. We rely in this work on the recycling rules in Wolfenbüttel, which were published in an official document on their website that contains lists of objects and the corresponding bin. We have collected a total of 207 images. Figure 1 shows an example of some images from the dataset. For each object inside the list, 3 photos are collected manually using Google's image search or kleinanzeigen.de website, which is a website for people who want to sell an item they have. People can post about the product and upload an image of it.

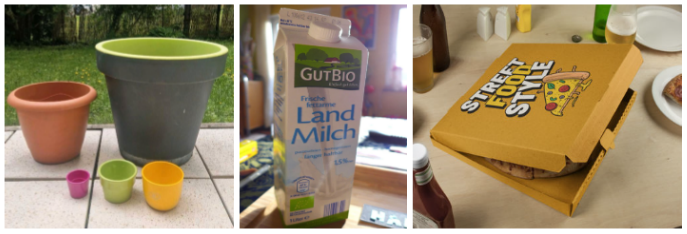


Figure 1. Examples of the images collected for the dataset.

The images should contain objects that are in a realistic environment, with no white background, the reason for choosing a colorful background is that white backgrounds make it very easy to identify an object in an image because the item would be pretty isolated. However, we want the images to be as realistic as possible and test how well the MLLM can identify the object correctly even with colorful and noisy backgrounds.

B. The Classification Phase

The first step of the benchmarking system is the classification phase. In this phase, the collected dataset of images will be used and the system will interact with an MLLM and ask it where to dispose of the object in each image and obtain the answers. The system initializes the process by setting the local recycling rules, which the model should follow when being asked to classify the object into one of the recycling bins to be disposed of. The system then reads the images as an input, iterates through them, and sends a request to the MLLM's API, to ask it where to dispose of the item in the image. The language model will generate an answer as text and all answers will be stored to enter the next phase. Figure 2 shows the architecture of the classification system.

The critical element is to provide the MLLM with instructions on how to behave and data to properly assist with sorting waste. This data is region-specific and in order to provide a scalable solution, is provided via a system prompt. A prompt is a text through which a human being can interact with the language model, it is written in natural language and its purpose is to give instructions or information to the model so that when it answers, it follows the user's wish that has been declared in the prompt [13]. An example of a prompt:

```
"Write a short story about people who
figure a way to travel back in time and
change certain events in their lives to
```

```
help them create a better future in no
more than 1000 words."
```

This prompt specifies the task that it is required to perform for the system. The prompt must be detailed and clear so that the model fully understands the instructions.

There are two types of prompts which the model can interact with. Both are important and will be used in this project:

- **System Prompt:** is a prompt that influences the entire model's behavior, it could be a set of rules to follow or some information related to a context that the system must take into consideration before answering. This prompt is given to the model only once during its initialization and before any user interaction [13].
- **User Prompt:** is when a prompt is given to the system while expecting an answer, through which the user usually interacts with the model [13].

This makes the system prompt a very good solution for the regional recycling rules problem, because the model can be prompted with the recycling rules before a user can interact with it, and then, all the upcoming answers that a model will generate will be following those rules.

There are also a few prompting strategies that have been discussed in the literature, the *persona strategy* is applied in this work This strategy gives the model a personality with perspective and knowledge on how to act if a user asks it a question, just like the type of help a human being would get asking someone in real life with that role [14]. The following system prompt is used in the first phase to instruct the system with the recycling rules, giving it the role of a recycling assistant with a set of rules to follow:

```
"You are an assistant. Here are the
local recycling rules:
1. If an item is made of glass, then
it must be disposed of in the glass
containers.
2. If an item is clothing--such as jeans,
a shirt, t-shirt, dress, shorts, socks,
hoodie, pullover, pajamas, or skirt--it
must be disposed of at this address:
'Recyclinghof Klein Elbe, 38274 Elbe.'
3. If an item is made of plastic or is a
food container, aluminum foil, beverage
carton (such as a milk or juice carton),
toothpaste tube, bottle of shampoo or
soap, plant pot, cutlery, CD or DVD cover,
bucket, kids' toys, clothes hanger, pan,
bowl, or toothbrush, it must be disposed
of in the yellow bin.
..."
```

The prompt contains several parts, at first we define for the model what the purpose of its existence is and what role it plays. Then we define the context that this model is being used for which is waste disposal, then we give it a set of rules on how to judge in which bin the item belongs. The rules include example objects, which are taken from the Wolfenbüttel document mentioned earlier. We can also see that for the clothes bin, an address of the disposal place has been provided, as mentioned in the documentation provided by the city. For each picture that has been collected, the model will be asked where to dispose of the item that is visible in it.

C. The Evaluation Phase

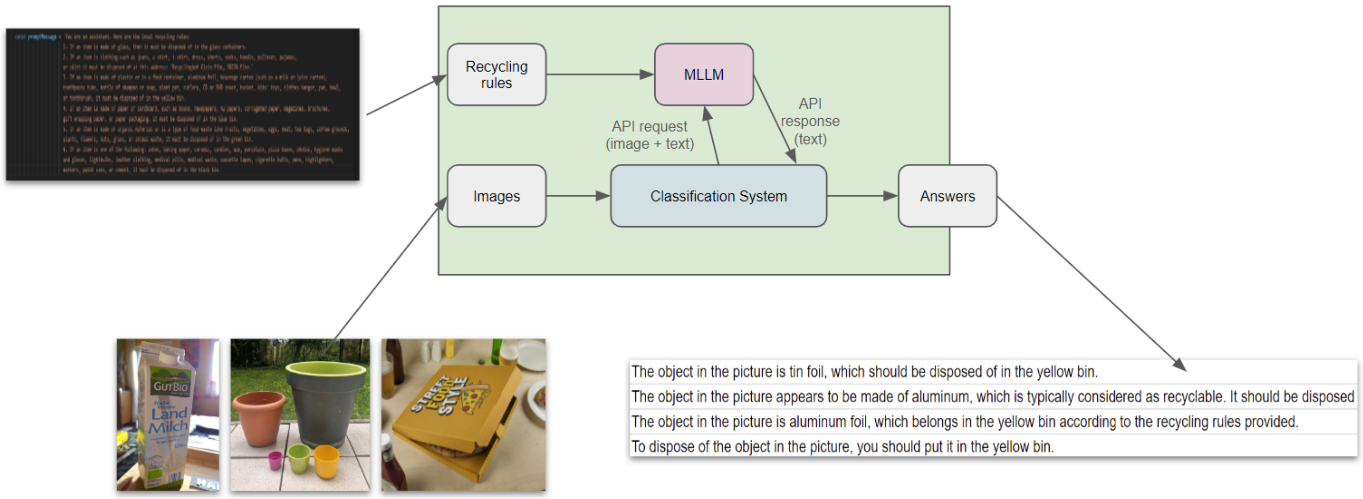
In this phase, the system's task is to evaluate the answers from the previous phase. Figure 3 shows that the system will have two inputs, the first one is the output of the previous phase which are the answers generated by the MLLM, and the second is the source of truth file, which contains the image ID, the object inside it, and the correct bin in which it should be disposed of. It should be noted here that the information about the object in the object column is not passed to the model neither during the classification phase nor during the evaluation phase. It is only used for referencing purposes.

The evaluation system will then iterate through the source of truth file and send a request to another text-based large language model to compare the output from the previous step with the bin mentioned in the source of truth and evaluate if the answers are semantically equivalent. If so, the system with output *correct*; otherwise, it will output *incorrect*, and the output will be saved into a file.

1) LLMs Evaluating LLMs: According to [6] there are three types of LLM-as-a-judge. In this project, we will apply the approach *Reference-guided grading*, which means the system is provided with a reference answer, which is the correct answer to the question, and another language model's answer, and the model must compare if they match. The reason for applying this approach is the model can not rely on the general knowledge that it was trained with to judge, because the classification system makes decisions according to the regional-based rules in the prompt, and it is tailored for a certain city, which is Wolfenbüttel in our work. So, since the evaluation language model does not know about the recycling rules in Wolfenbüttel, we store the correct answers in the source of truth file, where they will serve as reference answers.

2) Prompting The Evaluation Model: In this phase, we want to prompt the model that plays the role of the judge in the evaluation phase. In the evaluation phase, we received the answers from the previous phase, we also have the source of truth which contains the actual answers in the *Bin* column as shown in Figure 3. So, according to the *Persona* prompt strategy [14], we want to design the system prompt for the evaluation model so that it would act like a judge and compare two texts semantically, and output the word *correct* if they semantically match or output *incorrect* if they don't. The following prompt is the system prompt of the evaluation model:

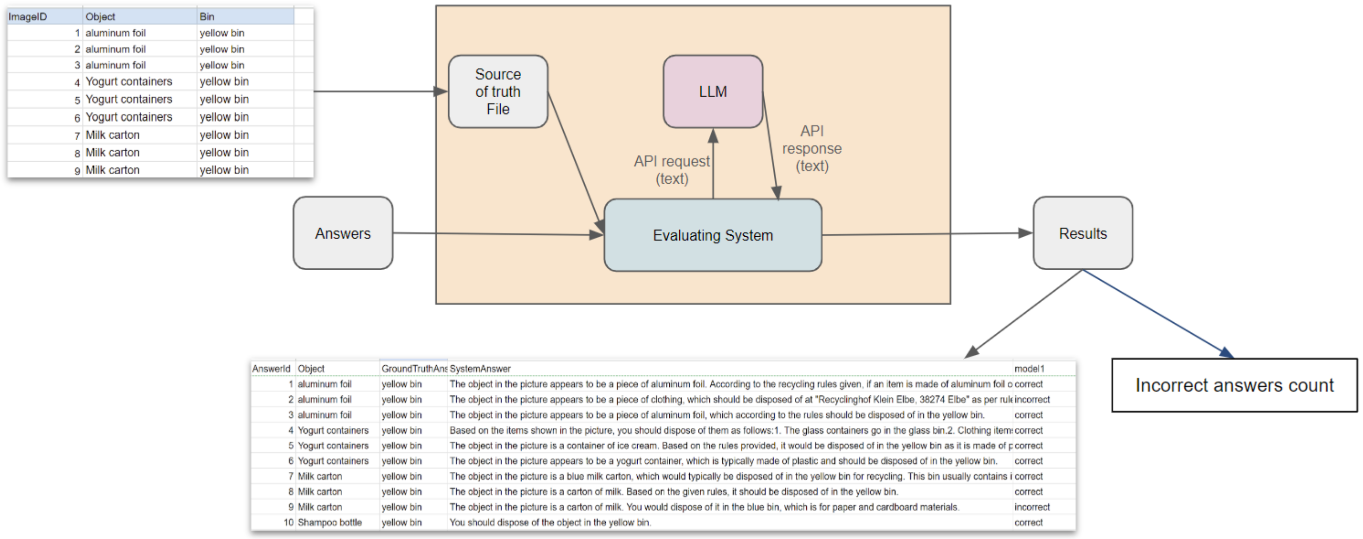
```
"You are an evaluation assistant.
Your task is to compare two texts: the
first text contains the source of truth,
and the second text contains system
answers. Determine if the bin mentioned
in the source of truth matches the bin
mentioned in the system answer. Respond
```



```
const promptMessage = `You are an assistant. Here are the local recycling rules:
1. If an item is made of glass, then it must be disposed of in the glass containers.
2. If an item is clothing such as jeans, a shirt, t-shirt, dress, shorts, socks, hoodie, pullover, pajamas,
or skirt: it must be disposed of at this address: "Recyclinghof Klein Elbe, 38274 Elbe."
3. If an item is made of plastic or is a food container, aluminum foil, beverage carton (such as a milk or juice carton),
toothpaste tube, bottle of shampoo or soap, plant pot, cutlery, CD or DVD cover, bucket, kids' toys, clothes hanger, pan, bowl,
or toothbrush. It must be disposed of in the yellow bin.
4. If an item is made of paper or cardboard, such as books, newspapers, A4 papers, corrugated paper, magazines, brochures,
gift wrapping paper, or paper packaging. It must be disposed of in the blue bin.
5. If an item is made of organic material or is a type of food waste like fruits, vegetables, eggs, meat, tea bags, coffee grounds,
plants, flowers, nuts, grass, or animal waste, it must be disposed of in the green bin.
6. If an item is one of the following: ashes, baking paper, ceramic, candles, wax, porcelain, pizza boxes, photos, hygiene masks
and gloves, lightbulbs, leather clothing, medical pills, medical waste, cassette tapes, cigarette butts, pens, highlighters,
markers, paint cans, or cement, it must be disposed of in the black bin.`
```

The object in the picture is tin foil, which should be disposed of in the yellow bin.
The object in the picture appears to be made of aluminum, which is typically considered as recyclable. It should be disposed
The object in the picture is aluminum foil, which belongs in the yellow bin according to the recycling rules provided.
To dispose of the object in the picture, you should put it in the yellow bin.

Figure 2. The classification phase of the benchmarking system.



ImageID	Object	Bin
1	aluminum foil	yellow bin
2	aluminum foil	yellow bin
3	aluminum foil	yellow bin
4	Yogurt containers	yellow bin
5	Yogurt containers	yellow bin
6	Yogurt containers	yellow bin
7	Milk carton	yellow bin
8	Milk carton	yellow bin
9	Milk carton	yellow bin

AnswerId	Object	GroundTruthAns	SystemAnswer	model1
1	aluminum foil	yellow bin	The object in the picture appears to be a piece of aluminum foil. According to the recycling rules given, if an item is made of aluminum foil o	correct
2	aluminum foil	yellow bin	The object in the picture appears to be a piece of clothing, which should be disposed of at "Recyclinghof Klein Elbe, 38274 Elbe" as per rule	incorrect
3	aluminum foil	yellow bin	The object in the picture appears to be a piece of aluminum foil, which according to the rules should be disposed of in the yellow bin.	correct
4	Yogurt containers	yellow bin	Based on the items shown in the picture, you should dispose of them as follows:1. The glass containers go in the glass bin.2. Clothing items	correct
5	Yogurt containers	yellow bin	The object in the picture is a container of ice cream. Based on the rules provided, it would be disposed of in the yellow bin as it is made of p	correct
6	Yogurt containers	yellow bin	The object in the picture appears to be a yogurt container, which is typically made of plastic and should be disposed of in the yellow bin.	correct
7	Milk carton	yellow bin	The object in the picture is a blue milk carton, which would typically be disposed of in the yellow bin for recycling. This bin usually contains i	correct
8	Milk carton	yellow bin	The object in the picture is a carton of milk. Based on the given rules, it should be disposed of in the yellow bin.	correct
9	Milk carton	yellow bin	The object in the picture is a carton of milk. You would dispose of it in the blue bin, which is for paper and cardboard materials.	incorrect
10	Shampoo bottle	yellow bin	You should dispose of the object in the yellow bin.	correct

Figure 3. The evaluation phase of the benchmarking system.

`with one word: 'correct' if they match, and 'incorrect' if they don't."`

Now that we set the system prompt, we can start looping over the answers and ask the model if they match the bin mentioned in the ground truth.

D. Benchmarking Evaluation Models Against Human Evaluation

After the setup of the evaluation phase, the last step is to select a large language model that will play the role of the judge. This step is very important because a benchmarking system will only be good if the evaluation model is good. However, there are many large language models that have recently gained the attention of the community and we select the following set of models because they are open-source models, they are also under constant updates, and it is worth mentioning that Qwen2 series of models [15] and Llama3 models [4] both being released around 2 months ago. We will compare the performance of each of them to select the best one that is capable of reasoning about long texts and deciding if they match the source of truth. Here are the selected models:

- Qwen:7b [3]
- Qwen:32b
- Llama3:8b [4]
- Llama3:70b
- Qwen2:7b [15]
- Qwen2:72b
- Llama3.1:8b
- Llama3.1:70b

It should be pointed out that the model name Qwen refers to version 1.5, while Qwen2 refers to version 2. The number followed by the letter *b* in the name stands for the number of parameters used to fine-tune the model. For example, the model Llama3.1:8b has 8 billion parameters. Each model is equipped with a certain number of parameters that can be used, for example, Llama3.1 provides 8b, 70b, or 450b. We chose to work with a parameter count that is less than 80b because the higher the parameter count is the slower an LLM responds to answers. And also we were limited by the server size the models were installed on. To compare which model performs best, we let all the models evaluate the same answers obtained from the classification phase, using the same system prompt for evaluation, the final output is assembled in a CSV file, and all output models are present in a separate column. After obtaining this output, we need a human evaluation to compare these models against, so we can determine which one is the better judge. So, a new column is added where each answer from the classification phase has been notated by a human as *correct* or *incorrect*, then a confusion matrix has been calculated for each of the 8 models.

We see in Figure 4 that the first three columns (answerId, object, and GroundTruthAnswer) represent the values from the source of truth table. The *SystemAnswer* column holds the answers that were output from the classification phase, the *HumanEvaluation* is the notation added by a human to evaluate the answers in the previous column if they match the ground of truth, and the rest of the columns from model1 to model8 contain the evaluation of the models respectively.

IV. Results

In this section, we present the results of benchmarking different LLMs as a judge and compare them against the human evaluation.

For this, we calculate the confusion matrix based on the human evaluation as well as the output from the other models. However, since LLMs produce non-deterministic responses, this means that we may receive different results when we interact with them, to solve this problem, we run the classification phase once to obtain the answers and then label them by a human. Then for that output, we run the evaluation models three times, calculate the accuracy, precision, recall, and f1 score for each one, and take the average. Table I shows that the model Qwen:32b outperforms all the others, and performs better than higher version models.

TABLE I. Results of calculating accuracy, precision, recall and f1 score after 3 runs.

Model	accuracy	precision	recall	F1 score
Qwen:7b	0.72	0.62	0.95	0.75
Qwen:32b	0.91	0.87	0.95	**0.91**
Llama3:8b	0.86	0.86	0.82	0.84
Llama3:70b	0.91	0.87	0.94	0.9
Qwen2:7b	0.89	0.85	0.9	0.88
Qwen2:72b	0.9	0.83	0.98	0.9
Llama3.1:8b	0.86	0.87	0.82	0.84
Llama3.1:70b	0.91	0.94	0.86	0.9

We also notice that it has a relatively low parameter count with only 32b, compared to Llama3:70b for example.

V. Discussion

In general, models with a higher parameter count perform better than those with a lower parameter count within the same version, for example, Llama3.1:70b performs better than Llama3.1:8b. However, one should also know that even though a high number of parameters may ensure better results, it is slower in performance and requires more memory and computing energy, so there's always a trade-off between how fast or how accurate you want your application to be. For example, using Llama3.1:8b achieves an f1 score of 0.84, and using Llama3.1:70b would make the system a few seconds slower with only 0.06 improvement because it achieves an f1 score of 0.9.

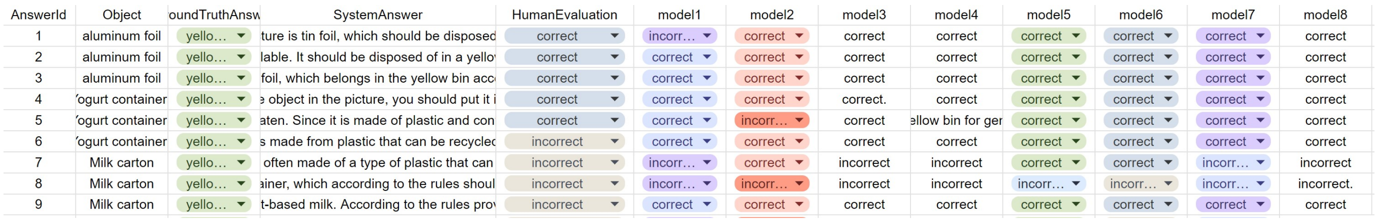
Since we now know that the model Qwen:32b generates the most accurate answers, we want to judge if the classification system is reliable or not, the model used for classification is *Llava-1.6-mistral:latest*. As shown in Figure 3, the evaluation phase will also output the number of incorrect answers, the incorrect answers count using the Qwen:32b is 108/207 while the incorrect answers annotated by a human are 116/207. We see that the numbers are very close, and the benchmarking system results show that more than half of the answers from the classification phase are wrong, so this result tells the researcher that the classification model is not reliable and needs to be either enhanced or replaced.

The cost for the benchmarking system is relatively low since it only handles text input and output Figure 3. The output in particular is very short with a token length of only a single token. The input on the other hand is much longer since it includes the entire output of the classification phase as well as the ground truth answer. On average, the input token length amounts to about 35 tokens [16]. According to current pricing on the Alibaba Cloud with 0.002$/1000 input tokens and 0.009$/1000 output tokens. This makes the evaluation of 1000 answers cost about 0.073$. This means that our benchmarking system is very affordable even for large amounts of data.

VI. Conclusion and Future Work

In this work, we investigated the possibility of using large language models to automate the benchmarking process and evaluate other models in the context of waste disposal. We compared different models available that can play the role of a judge in benchmarking systems and we saw from the results that Qwen:32b achieved the best performance. However, there are always ways to further improve the benchmarking system results, the following are some suggestions for future work.

- **Keeping up-to-date:** The evaluation model still makes a few mistakes while judging the answers, but the good news is, these models are under constant maintenance, and even new models are always under development. Researchers must stay updated on the model landscape to ensure the best-performing system.



AnswerId	Object	oundTruthAnsw	SystemAnswer	HumanEvaluation	model1	model2	model3	model4	model5	model6	model7	model8
1	aluminum foil	yello…	ture is tin foil, which should be disposed	correct	incorr…	correct	correct	correct	correct	correct	correct	correct
2	aluminum foil	yello…	lable. It should be disposed of in a yellov	correct	correct	correct	correct	correct	correct	correct	correct	correct
3	aluminum foil	yello…	foil, which belongs in the yellow bin acc	correct	correct	correct	correct	correct	correct	correct	correct	correct
4	'ogurt container	yello…	e object in the picture, you should put it i	correct	correct	correct	correct.	correct	correct	correct	correct	correct
5	'ogurt container	yello…	aten. Since it is made of plastic and con	correct	correct	incorr…	correct	ellow bin for ger	correct	correct	correct	correct
6	'ogurt container	yello…	s made from plastic that can be recyclec	incorrect	correct	correct	correct	correct	correct	correct	correct	correct
7	Milk carton	yello…	often made of a type of plastic that can	incorrect	incorr…	correct	incorrect	incorrect	correct	correct	incorr…	incorrect
8	Milk carton	yello…	ainer, which according to the rules shoul	incorrect	incorr…	incorr…	incorrect	incorrect	incorr…	incorr…	incorr…	incorrect.
9	Milk carton	yello…	t-based milk. According to the rules prov	incorrect	correct	correct	correct	correct	correct	correct	correct	correct

Figure 4. A snippet of the assembled output of all the models in addition to the classification system's answers and the human evaluation.

- **Prompt Engineering:** There are a set of techniques for structuring the best prompt, in this project, some of them have been applied, but it would boost the performance once this topic can be applied with more depth.
- **Categories-oriented evaluation:** An evaluation system like this gives only an overall view of the classification system. However, this view is not detailed and it does not show the researcher the areas in which the system excels or behaves poorly. For example, in which category the model used for classification *Llava-1.6-mistral:latest* performs the worst? Is it by the yellow bin or the glass container? This would help the researcher to decide where the models need to be improved.
- **Separate models for separate tasks:** In the classification phase, only one model was used to perform both tasks of identifying the object in the image and guessing the correct bin for it. However, there could be models that perform better at image recognition and others that perform better regarding reasoning about the rules in the prompt, so to enhance the classification phase, we propose using two separate models, one for the image recognition task and one for reasoning about the recycling rules.
- **Benchmarking framework:** This project is only a console project, but it can grow into a benchmarking framework, where the users can interact with an interface to upload the source of truth files, upload the dataset, select a classification model, enter the recycling rules and compare the results for different models, datasets and prompts.

Overall this work has achieved good results, and we believe it is ready to be applied in different domains. With the aforementioned future work, we believe that the benchmarking system can be further enhanced to provide even more reliable results and better insights.

References

[1] G. Maitlo *et al.*, "Plastic waste recycling, applications, and future prospects for a sustainable environment," *Sustainability*, vol. 14, no. 18, 2022, ISSN: 2071-1050. DOI: 10.3390/su141811637.

[2] A. Subklew, "Verbraucherumfrage zur mülltrennung," Nov. 2020, [Online]. Available: https://www.muelltrennung-wirkt.de/fileadmin/user_upload/Presse/Factsheets_und_Studien/Duale-Systeme_Factsheet_BUS_Umfrage.pdf.

[3] J. Bai *et al.*, *Qwen technical report*, 2023. arXiv: 2309.16609 [cs.CL].

[4] A. Dubey *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. arXiv: 2407.21783 [cs.AI].

[5] W. L. W. A. (ALW), "Abfallfibel 2024," Oct. 2023, [Online]. Available: https://www.alw-wf.de/index.php/component/phocadownload/file/57-abfallfibel-2024.

[6] L. Zheng *et al.*, *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*, A. Oh *et al.*, Eds. Curran Associates, Inc., 2023, vol. 36, pp. 46 595–46 623.

[7] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," *arXiv preprint arXiv:1904.09675*, Apr. 2020. arXiv: 1904.09675 [cs.CL].

[8] B. Peng, C. Li, P. He, M. Galley, and J. Gao, "Instruction tuning with gpt-4," *arXiv preprint arXiv:2304.03277*, Apr. 2023. arXiv: 2304.03277 [cs.CL].

[9] W.-L. Chiang *et al.*, "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, vol. 2, no. 3, p. 6, Mar. 2023.

[10] W. Xie, J. Niu, C. J. Xue, and N. Guan, *Grade like a human: Rethinking automated assessment with large language models*, Mar. 2024. arXiv: 2405.19694 [cs.AI].

[11] Y. Chen, B. Xu, Q. Wang, Y. Liu, and Z. Mao, "Benchmarking large language models on controllable generation under diversified instructions," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, pp. 17 808–17 816, Mar. 2024. DOI: 10.1609/aaai.v38i16.29734.

[12] C.-H. Chiang and H.-y. Lee, "Can large language models be an alternative to human evaluations?" *arXiv preprint arXiv:2305.01937*, 2023. arXiv: 2305.01937 [cs.CL].

[13] M. Weber and M. Reichardt, "Evaluation is all you need. Prompting generative large language models for annotation tasks in the social sciences. A primer using open models," *arXiv preprint arXiv:2401.00284*, Dec. 2023. DOI: 10.48550/arXiv.2401.00284.

[14] J. White *et al.*, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, Feb. 2023. DOI: 10.48550/arXiv.2302.11382.

[15] A. Yang *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024. arXiv: 2407.10671 [cs.CL].

[16] AlibabaCloud, "Open source qwen llms: Billing," Sep. 2024, [Online]. Available: https://www.alibabacloud.com/help/en/model-studio/developer-reference/billing-for-tongyi-qianwen-7b-14b-72b.

Integrative Development and Evaluation of V2X Communication Architectures to Support Autonomous Driving Systems in 5G Campus Networks

Florian Pramme, Bastian Teßin, Gert Bikker
Institute for Distributed Systems
Ostfalia University of Applied Sciences
Wolfenbuettel, Germany
{florian.pramme, ba.tessin, g.bikker}@ostfalia.de

Tamás Kurczveil
Institute for Traffic Management
Ostfalia University of Applied Sciences
Salzgitter, Germany
t.kurczveil@ostfalia.de

***Abstract*—This paper provides an overview of our upcoming research project, which focuses on the development and application of a dynamic driving task with autonomous vehicles based on Cellular Vehicle-to-Everything (C-V2X) functionalities as well as the evaluation of values added with the application through 5G communication within a campus network. Particular interest lies on low-latency and high-throughput 5G communication and Vehicle-to-Everything (V2X) Day-2-Use-Cases as well as the analysis and implementation of corresponding requirements. This paper outlines the pursued concept and lists planned scenarios.**

***Keywords* — V2X, C-V2X, 5G, autonomous driving.**

I. Introduction

Highly automated driving will have a significant impact on future mobility. The interaction of already available Advanced Driving Assistance System (ADAS) allows much shorter reaction times in unpredictable traffic situations than humans are capable of. Current available ADASs essentially rely on sensor data from an ego vehicle. Some Original Equipment Manufaturers (OEMs) have started to roll out Day-1 V2X use cases that allow the first possibilities of perceiving external sensor data. The sensor data recorded by sensors of other road users and the corresponding sensor fusion offers great potential for autonomous driving functions and for increasing traffic safety in general.

V2X in the 5G network offers considerable potential, especially for so-called *Day-2-Use-Cases*, which represent more advanced vehicle communication scenarios. These use cases go beyond basic functions such as emergency brake warnings and use the extended capabilities of 5G to enable more complex and safety-critical applications [1].

Network slicing allows the 5G network to create specific virtual networks for different use cases so that safety-critical applications such as teleoperation or emergency communication are prioritized and executed with maximum reliability [2].

Vehicles can perform complex maneuvers such as automated overtaking on multi-lane roads in close coordination with other road users. In emergencies or difficult situations, control of a vehicle could be handed over to a remote control center operating via a stable 5G connection.

The rest of the paper is structured as follows. In Section II, we give a short overview of our planned concept. Section III presents several scenarios to be examined within the project. Finally, we conclude the work in Section IV.

II. Concept

An important part of the project is equipping a reference track on the Wolfenbuettel campus of the Ostfalia University of Applied Sciences with infrastructure components for vehicle-to-infrastructure communication (V2I) with so-called Road Side Units (RSUs). These RSUs are used to provide data and traffic information on the one hand and to provide information for other road users (e.g., information about detected pedestrians, cyclists and public transport users) via established WiFi standards on the other. The RSUs will also collect data from connected infrastructure sensors (e.g., cameras, laser scanners, radar or Light Detection and Ranging (LiDAR) sensors). If required, this data can be transmitted to a traffic management center. Autonomous vehicles rely on a combination of four primary sensor types: cameras, ultrasonic sensors, radar, and LiDAR. However, each of these sensors has its own limitations. Camera-based sensors are not able to detect objects in foggy areas, in the rain or at night. Radar uses radio waves to detect vehicles/objects and is accurate in all visibility conditions, but cannot distinguish the type of objects without a human driver due to its longer wavelength. A classic 4G connection is sufficient for use cases of extended route guidance using V2X [3], [4]. For future cooperative use cases of autonomous driving, however, an architecture is required that bundles both local and remote information and can process and send it in real time. 5G New Radio (5G NR) improves the latency, reliability and throughput of mobile networks and offers new opportunities to support advanced V2X applications for connected and autonomous driving [5], [6]. Furthermore, 5G NR also improves the accuracy of positioning by real-time kinematics of the satellite navigation system in the centimeter range. These opportunities also arise from the introduction of Mobile Edge Computing (MEC) in 5G. The high communication effort and the enormous amount of data within the development period speak in favor of setting up a dedicated Radio Access Network (RAN) - a campus network based on 5G [7]. This dedicated network enables latencies in the order of 5 ms, which offers a remarkable increase in performance for researching future traffic scenarios, V2X applications and autonomous driving functions.

The objective is to establish an autonomous shuttle service that operates using conventional sensor technology, supplemented by V2X communication and a 3rd Generation Partnership Project (3GPP) Release 16 campus network, which will be deployed near the Ostfalia University campus in Wolfenbuettel, Germany. The autonomous shuttle service drives around a circular route on our campus, characterized by several intercections with poor visibility. An RSU shall be positioned at a 3-way-intersection, equipped with a radar sensor to detect objects in the intersection area to share information using Collective Percepion Message (CPM) messages via C-V2X. This autonomous shuttle service and the campus network will serve as a research platform for the following research questions: 1. To what extent do network slicing and a 5G Standalone Network (5G SA) improve the quality of communication between the vehicle and a teleoperator compared to existing technologies? 2. How helpful are V2X Day-2-UseCases especially in complex intersection scenarios? 3. What is the impact of 5G SA and Ultra Reliable Low Latency Communication (URLLC) to the process of teleoperation and how realistic are low latencies of 1 ms specified by 5G-SA and URLLC in a real environment? 4. How can the application of MEC support participants in the decision making and increase traffic safety by handling computaionally intense operations.

III. Scenario

Consider a scenario in which an autonomous vehicle is operating in an urban environment, embedded in a 5G data network that enables it to communicate in real time via C-V2X (Cellular Vehicle-to-Everything). The vehicle approaches a complex intersection where visibility is severely limited by urban development and sharp angles, requiring increased communication requirements between vehicles and the surrounding infrastructure.

In this scenario, the autonomous vehicle continuously receives high-resolution sensor data from 5G-enabled infrastructure elements strategically positioned at the intersection. These sensors provide detailed information about the flow of traffic, the position and speed of other road users, as well as the movements of pedestrians and the current status of traffic light signals. This information is transmitted to the vehicle via the 5G-SA network with minimal latency, enabling precise and dynamic decision-making.

In parallel, the vehicle sends its own telemetric data, including its position, speed and planned route, to the surrounding traffic infrastructure as well as to other C-V2X-enabled vehicles. This creates a cooperative traffic control system based on a continuous, bidirectional flow of information to ensure adaptive and safe navigation. In this specific situation, the vehicle uses the received data to detect the presence of an oncoming vehicle approaching the intersection from a hidden position and adjusts its driving strategy accordingly to minimize a potential collision risk.

In addition, the autonomous vehicle is confronted with an unforeseen challenge: A sudden obstacle blocks the lane and the applicable traffic rules, such as a solid line, prevent safe overtaking. The autonomous system analyzes the situation in real time and uses 5G communication to request assistance from a remote traffic control center or to evaluate and implement alternative route and driving strategies. These decision-making processes are carried out within milliseconds, without disrupting the flow of traffic or compromising the safety of other road users.

IV. Conclusion

With the introduction of 5G, V2X communication will not only be improved, but fundamentally transformed. The promisingly low latency times enable vehicles to react to environmental information in near real time, which is particularly crucial for safety-critical applications such as collision avoidance. The higher data rates of 5G allow the exchange of large amounts of information between vehicles and the infrastructure, which leads to a more precise perception of the traffic situation. Vehicles can access not only their own sensors, but also information from other vehicles, traffic lights or cameras, which significantly increases safety and efficiency in road traffic.

In addition, the higher connection capacity of 5G enables thousands of vehicles and sensors to communicate simultaneously in urban areas. This creates the basis for a fully networked traffic system in which all road users interact in real time. The reliability of 5G is particularly important for applications that cannot tolerate delays or data loss. Network slicing, a key technology of 5G, ensures that such applications can be operated with the necessary stability.

Overall, 5G lays the technological foundation for more advanced forms of autonomous driving and the integration of vehicles into the smart city infrastructure, which can lead to more efficient traffic and lower emissions. In the long term, this will lead to safer, more efficient and smarter transportation systems and fundamentally change our mobility.

References

[1] Car 2 Car Consortium, "Guidance for day 2 and beyond roadmap," 09.07.2021.

[2] *Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Rerlease 16*, 3GPP, 12 2023, v16.17.0.

[3] S. Chen, J. Hu, Y. Shi, and L. Zhao, "Lte-v: A td-lte-based v2x solution for future vehicular network," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 997–1005, 2016.

[4] J. M. Wille, F. Saust, and M. Maurer, "Stadtpilot: Driving autonomously on braunschweig's inner ring road," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 506–511.

[5] A. Jouini, H. Dalila, and C. Adnen, "Adas and 5g/6g-v2x implementation for sustainable and intelligent transport," in *2023 22nd Mediterranean Microwave Symposium (MMS)*, 2023, pp. 1–6.

[6] S. An and K. Chang, "Enhancing reliability in 5g nr v2v communications through priority-based groupcasting and ir-harq," *IEEE Access*, vol. 11, pp. 72 717–72 731, 2023.

[7] M. C. Lucas-Estañet et al., "Analysis of 5g ran configuration to support advanced v2x services," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–5.

Mixing Flows in Dynamic Fluid Transport Simulations

Mehrnaz Anvari
Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Sankt Augustin, Germany
email: Mehrnaz.Anvari@scai.fraunhofer.de

Anton Baldin
PLEdoc GmbH and Fraunhofer Institute SCAI
Sankt Augustin, Germany
email: Anton.Baldin@scai.fraunhofer.de

Tanja Clees
University of Applied Sciences Bonn-Rhein-Sieg and Fraunhofer Institute SCAI
Sankt Augustin, Germany
email: Tanja.Clees@scai.fraunhofer.de

Bernhard Klaassen
Fraunhofer Research Institution for Energy Infrastructures IEG
Bochum, Germany
email: Bernhard.Klaassen@ieg.fraunhofer.de

Igor Nikitin
Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Sankt Augustin, Germany
email: Igor.Nikitin@scai.fraunhofer.de

Lialia Nikitina
Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Sankt Augustin, Germany
email: Lialia.Nikitina@scai.fraunhofer.de

***Abstract*—This paper presents a new numerically efficient implementation of flow mixing algorithms in dynamic simulation of pipeline fluid transport. Mixed characteristics include molar mass, heat value, chemical composition and the temperature of the transported fluids. In the absence of chemical reactions, the modeling is based on the universal conservation laws for molar flows and total energy. The modeling formulates a sequence of linear systems, solved by a sparse linear solver, typically in one iteration per integration step. The functionality and stability of the developed simulation methods have been tested on a number of realistic network scenarios. The main output of the paper is a functioning and stable implementation of flow mixing algorithms for dynamic simulation of fluid transport networks.**

Keywords-simulation and modeling; mathematical and numerical algorithms and methods; mixing flows; pipeline fluid transport.

I. INTRODUCTION

This paper continues a series of our works on modeling of fluid transport networks. Previous works presented stationary [1] and dynamic [2] modeling of fluid transport networks limited to a single chemical composition and constant temperature. In addition, some aspects of stationary modeling of mixing fluids of different compositions and/or temperatures were considered in [3]. In this paper, flow mixing modeling will be considered in more detail, with special emphasis on the thermodynamic layer of the model. In particular, dynamic mixing equations and algorithms for their solving will be presented. The developed approach is implemented in our Multi-phYsics Network Simulator (MYNTS) [4], which is used to solve actual transport scenarios for natural gas [5], hydrogen [6], carbon dioxide [7], water [8] and other fluids.

Fluid transport modeling is based on the conservation of mass flows in the form of dynamic Kirchhoff equations; Darcy-Weisbach pipeline pressure drop formula, with empirical friction term by Nikuradse [9] and Hofer [10]; equation of state computation by simplified analytical models by Papay [11], Peng-Robinson [12] and Soave-Redlich-Kwong [13] or more complex ISO-norm models AGA8-DC92 [14] and GERG2008 [15]–[17].

In state-of-the-art, a number of previous studies [18]–[24] considered modeling of pipeline fluid transport, both at the universal mathematical level [18], and in various application scenarios. Such scenarios include transport of natural gas [19] [21], steam transport in oil refineries [20], carbon dioxide transport [22]–[24]. All these works are characterized by the presentation of transport equations as laws of conservation of mass, momentum and energy. In the presence of various substances, conservation of molar flows is added, while the general relations of thermodynamics of open systems [25] regulate the relations of energy and temperature.

A common drawback of existing solutions is the closed nature of modeling within blackbox systems. If it is necessary to change the modeling, modify or introduce new equations and variables, the system must be reprogrammed. In addition, existing systems experience difficulties in solving large realistic network problems in the presence of numerical instabilities. The novelty of our approach consists in transparent modeling, where the user can freely change the equations and experiment with different forms of representing physical processes in fluid transport networks. We also pay special attention to the stability and performance of solution algorithms, which is especially important for realistic scenarios with a large number of elements. The purpose of this work is to extend transparent and numerically stable modeling to mixing flows present in realistic fluid transport scenarios.

In this work, Section II presents the modeling of mixing flows incorporating molar and temperature relationships. Section III describes the numerical experiments performed using the developed methods. Section IV summarizes the main results and conclusions of the work.

II. MODELING OF MIXING FLOWS

This section describes the details of modeling of mixing flows, consisting of modeling fluid molar composition and temperature distribution.

A. Molar fluid composition

A fluid transport network is described by a directed graph consisting of nodes and edges connecting them. The graph is described by an incidence matrix I_{ne}, in which each edge e has nonzero entries for the nodes n that this edge connects; -1 for the node that edge comes from, $+1$ for the node that edge enters. Mixing fluid flows are described by following equations

$$V_n\partial\rho_n/\partial t = \textstyle\sum_e I_{ne}m_e, \tag{1}$$
$$V_n\partial(\rho_n\mu_n^{-1})/\partial t = \textstyle\sum_e I_{ne}m_e\mu_e^{-1}, \tag{2}$$
$$V_n\partial(\rho_n\mu_n^{-1}x_n)/\partial t = \textstyle\sum_e I_{ne}m_e\mu_e^{-1}x_e, \tag{3}$$

where V_n is the volume assigned to the node; ρ_n represents the mass density at the node; t denotes time; the sum applies to all edges adjacent to the node; m_e is the mass flow in an edge, considered positive if the direction of flow coincides with the direction of the edge, and negative otherwise; $\mu_{n/e}$ is the molar mass assigned to both the node and the edge; $x_{n/e}$ are the mole fractions of the components that make up the fluid.

Physically, the above equations describe various conservation laws. In particular, (1) is the dynamic Kirchhoff equation and describes the conservation of mass. Here, $V_n\rho_n$ on the left side, with V_n representing a time-independent volume, describes the mass of fluid in the node. The sum on the right side accounts for the mass flow into the node, minus the flow out. Equation (2) describes the conservation of the total molar amount of a fluid, where $V_n\rho_n\mu_n^{-1}$ represents the number of moles in a node, and the sum on the right side is the total molar flow in the node. Finally, (3) describes the molar conservation for each component, $V_n\rho_n\mu_n^{-1}x_n$ represents the number of moles of a given component in a node, and the sum is the molar flow of that component. Equations (1) and (2) are valid in the absence of chemical reactions between the components of the fluid.

The x-vector may also include other quantities to which linear molar mixing applies, such as the molar heat value H_m, and linear approximations (T_c, P_c) used in certain equations of state for critical temperature and critical pressure, among others. Alternatively, such quantities can be calculated in post-processing as a linear combination over the molar composition. Explicit inclusion in the mixing equation allows these quantities to be calculated even when the determination of molar composition is disabled.

The conservation equations of type (1)–(3) are standard, can be found in a textbook, e.g., eq.(4.1) in [25]. Now we will rewrite them in a more convenient form, resolved with respect to derivatives:

$$V_n\rho_n\partial\mu_n^{-1}/\partial t = \textstyle\sum_e' I_{ne}m_e(\mu_e^{-1}-\mu_n^{-1}), \tag{4}$$
$$V_n\rho_n\mu_n^{-1}\partial x_n/\partial t = \textstyle\sum_e' I_{ne}m_e\mu_e^{-1}(x_e-x_n), \tag{5}$$
$$\textstyle\sum_e' = \sum_{e,I_{ne}m_e>0}, \tag{6}$$

where the sum is taken over the flows incoming to the node. To prove it, it is necessary to perform the differentiation in (2) and take into account (1), which will result in (4), in which the sums are taken over all flows, incoming and outgoing. Further, if one takes into account that μ_e^{-1} for an outgoing flow is equal to μ_n^{-1} at a node, the sum can be reduced to the incoming flows. The proof for (5) is similar. The condition of equality of mixed quantities in the node and in the outgoing flow can also be used to reduce the total number of variables. Namely, one can completely eliminate the variables in the edge e, replacing them with the values in the upstream node n', $\mu_e^{-1}\to\mu_{n'}^{-1}$, $x_e\to x_{n'}$. When time derivatives are set to zero, these equations are reduced to stationary formula (see eq.(13) in [3]).

Boundary conditions: $\mu=\mu_{set}$, $x=x_{set}$ are fixed to the specified values in the network entry nodes. The system of (4)–(6) and boundary conditions is closed. Its stationary part on the right side of the equations is non-degenerate if all nodes are connected to at least one entry node in the upstream direction. A complete dynamical system can be non-degenerate even if this rule is violated, for example if all flows are zero. In this case, the dynamic term ensures the preservation of the transported quantities, keeping them at the starting values.

Startup algorithm: at entry nodes, the transported values are initialized to set values to satisfy the boundary conditions. In all other nodes, values are initialized to default values, which are either specified by the user or averaged over all set values. As a part of the general procedure [2], the initial flows are set to zero and all fluid composition-dependent quantities, such as density ρ, are calculated from the appropriate equations of state. This procedure provides a smooth startup, with all equations initially satisfied. Then, fluid starts to propagate from entries to the neighbor nodes with growing massflow, replacing default values with current ones.

Scaling factors: according to the general procedure [2], all equations are scaled to cover the range of 100 units when variables are changed in their physical domain. Such factors can influence convergence of the solver and should be selected carefully.

V_n-definition: in accordance with the discretization scheme formulated in [2], each pipe contributes half of its volume to the end nodes, and all other elements contribute a nominally specified volume V_0.

Linearity of the system: with known m-flows, the μ^{-1}-subsystem (4) is linear; also, for known m and μ^{-1}, the x-subsystem (5) is linear. This property is convenient for controlling convergence, since each linear subsystem in the non-degenerate case is solvable in one iteration. The following algorithm is used to integrate the equations.

Algorithm (simulation workflow):

```
init;
repeat{ mumix; xmix; Tmix; PM; t+=dt; }
```

Here, `init` represents the initialization of all variables according to the startup algorithm described above. `mumix` is the solution of the μ^{-1}-subsystem, `xmix` is the solution of the x-subsystem, `Tmix` is the solution of the temperature subsystem formulated below, and `PM` is the solution of the pressure-massflow subsystem as formulated in [2]. As in [2], our primary goal is to determine stationary solutions of the system by integrating with as large steps as possible until stationarity is achieved. The most stable method suitable for this purpose is time discretization of the implicit Euler type: $\partial v/\partial t \to (v - v_{prev})/dt$, for all dynamic variables v, where v_{prev} is the value from the previous step, dt is the integration step. For a detailed study of dynamic processes, more sophisticated finite-difference schemes [26] [27] can be used.

B. Temperature modeling

The starting point is the law of conservation of energy for open systems (see, for example, eq.(4.14) in [25]):

$$V_n \partial(\rho_n \mu_n^{-1} U_n)/\partial t = \textstyle\sum_e I_{ne} m_e \mu_e^{-1} H_e, \tag{7}$$

where U is the molar internal energy, $H = U + P\mu/\rho$ is the molar enthalpy, and P is the pressure. The equation is similar to the conditions of molar mixing in (3). The difference is that the derivative of the nodal internal energy is on the left side, and the total enthalpy flow in the node is on the right side. Physically, with each flow, internal energy is introduced into the node, as well as the work of the fluid against the pressure in the node. This work can be combined with internal energy, giving enthalpy on the right side of the equation. On the left side, under the derivative, there is still nodal internal energy. In general case, other terms can be present in the conservation law, vanishing for simple mixing in the node. In particular, no additional work is performed in the node, and due to the assumed absolute thermal insulation of the node, heat transfer becomes zero. Possible processes with additional work and heat transfer are assigned to special edge elements and are described below.

We rewrite the equation (7) as follows:

$$\begin{aligned} &V_n \rho_n \mu_n^{-1} \partial H_n/\partial t - V_n \partial P_n/\partial t = \\ &= \textstyle\sum_e' I_{ne} m_e \mu_e^{-1} (H_e - H_n), \end{aligned} \tag{8}$$

the derivation is similar to (5), also here the nodal internal energy is re-expressed in terms of enthalpy and pressure in the node.

Boundary conditions: $H = H_{set}$, enthalpy is fixed to the specified value in entry nodes. Alternatively, one can use the condition $T = T_{set}$, which fixes the temperature at the entry nodes.

In addition, according to eq.(4.14) in [25], gravitational and kinetic terms can be added to the internal energy and enthalpy: $H \to H + \mu g h + \mu v^2/2$, where g is the acceleration of free fall, h is the height, and v is the speed of translational motion of the fluid. To calculate the kinetic term, one needs to know the diameter, which is not available for all types of elements. For example, a compressor is a very complex structure to be described by a single diameter. Also, at nodes where many edges join, complex internal motion occurs, which does not coincide with the simple translational motion described by a kinetic term with a single diameter. On the other hand, for the transport of gases, the kinetic term is usually significantly less than the internal energy, for translational velocities significantly lower than the speed of sound. In our simulation, we made it possible to optionally turn off the kinetic term in the temperature equations.

In (8), H_n represents the nodal value, and H_e represents the edge downstream value. The difference from x-mixing is that here the edge downstream value in the general case cannot be replaced by the upstream nodal value, since there are elements that change the enthalpy value. The system cannot be reduced to a purely nodal one; in addition, the system also includes the temperature T of the fluid.

HT-constraint:

$$H = H_{mod}(P, T, x), \tag{9}$$

$$H = H_{mod}(P, T_{prev}, x) + c_p(T - T_{prev}), \tag{10}$$

where H_{mod} is the thermodynamic model for enthalpy, $c_p = \partial H_{mod}/\partial T$ is the molar heat capacity calculated at point (P, T_{prev}, x). The equations (9)–(10) and (H, T) variables are introduced per node and edge.

The first equation relates enthalpy and temperature according to the thermodynamic model used. We use GERG2008 [15]–[17] as a concrete implementation of such relation. For software-technical reasons, it cannot be used directly; its call once per internal iteration produces too many total calls of GERG2008 module, resulting in significant slowdown. In addition, the equation is nonlinear, violating the desired linearity property of the Tmix subsystem. The second equation is a linearization of the first, it can be used in internal iterations, with a less frequent update of the coefficients. When using the workflow formulated above, (m, P, ρ) in all mix phases are considered as fixed parameters, updated in PM-phase. For H_{mod} and c_p, updates occur immediately before the start of the Tmix phase. In addition, to increase stability, the temperature is clamped to a given range, by default set to $[223.15, 423.15]K$.

Default element equation:

$$H_e = m_e > 0 ? H_{n1} : H_{n2} \tag{11}$$

formulates isenthalpic process [25], where the edge enthalpy is taken from the upstream node, similar to x-mixing. In this and further equations, the edge e goes from node n_1 to node n_2, conditions are written in C-notation: *x?y:z = if(x) then y; else z*. This model is applied to the most of element types, in particular, to valves, regulators, resistors and shortcuts; while the exceptional types are listed below.

Pipe equation:

$$\begin{aligned} &(m_e > 0 ? (H_{n1} - H_e)\mu_{n1}^{-1} : (H_{n2} - H_e)\mu_{n2}^{-1})|m_e| = \\ &= \pi D L c_{ht} (T_e - T_{soil}), \end{aligned} \tag{12}$$

the change of enthalpy over the pipe is equal to a heat exchange with the soil, eq.(33.3) in [28]. Here T_{soil} is soil temperature, D is pipe diameter, L is pipe length, and c_{ht} is heat transfer coefficient. The pipe should have sufficiently fine subdivision to model the heat exchange appropriately.

Compressor equation:

$$m_e > 0?(T_e - T_{n1}((|P_{n2}/P_{n1}|^{(\kappa-1)/\kappa} - 1)/\eta + \\ +1)z_{n1}/z_e) : (H_e - H_{n2}) = 0, \quad (13)$$

for positive flow, the change of temperature is described by eq.(38.51) in [28], or a similar formula (eq.(13-31)) without z-correction from [29]; otherwise, isenthalpic process is used. Here κ is isentropic exponent, η is efficiency, z is compressibility factor. This basic model is designed for gas transport, while for liquids, e.g., CO_2 pumps, customer-specific models can be used.

Coolers and heaters:

$$m_e > 0?(A_{set} > 0?(T_e - T_{set}) : (H_e - H_{n1})) \\ : (H_e - H_{n2}) = 0, \quad (14)$$

at the simplest modeling level, we implement these elements by clamp formulas: $T_e = \min(T_{n1}, T_{set})$ for coolers and $T_e = \max(T_{n1}, T_{set})$ for heaters. These formulas are piecewise-linear. Their linearization leads to the common formula above and the active set flag described by the following algorithm.

Algorithm (active set):

```
cooler:
  if(Aset==1&&He>Hn1) then Aset=0
  if(Aset==0&&Te>Tset) then Aset=1
heater:
  if(Aset==1&&He<Hn1) then Aset=0
  if(Aset==0&&Te<Tset) then Aset=1
```

Here $A_{set} = 1$ corresponds to an active mode, $A_{set} = 0$ to a standby mode. The algorithm is applied after Tmix-phase, its convergence is tracked.

III. Numerical Experiments

We performed a series of simulations on networks of different complexity levels to study in detail the effects of flow mixing, integration stability, and iteration convergence.

N1 network: the network shown in Figure 1 contains 100 nodes, 111 edges and is used for numerical experiments with the transport of natural gas and hydrogen. Detailed settings of supplies in the considered scenario are presented in Table I. Selected time discretization is $dt = 3 \cdot 10^4 s$, $nsteps = 100$. The network has a simple Y-shaped topology, with two supply nodes n99_gm and n56_gm, as well as a mixing node n89, where the flows from the supplies come together, and the rest of the network, ending with the most distant exit node n76.

Figure 2a shows the evolution of inverse molar mass. Figure 2b presents molar heat value, and Figure 2c demonstrates molar fraction of CH_4, representative for chemical composition in the considered test scenario. In all these plots, the values in supply nodes n99_gm and n56_gm are kept constant at set values. In stationary solution, the simple topology of the network leads to a single mixed state, formed in node n89 and propagated downstream to the rest of the network.

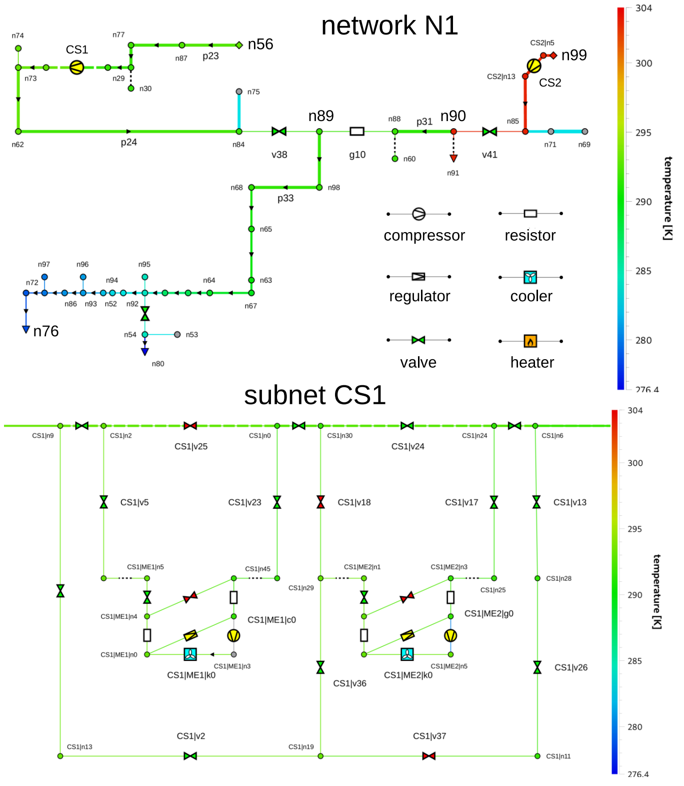


Figure 1. Test network N1.

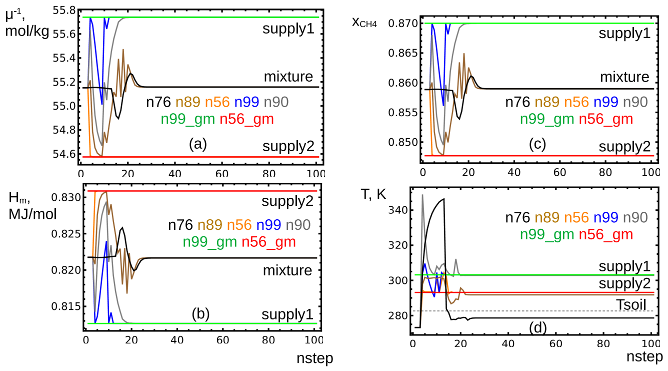


Figure 2. Simulation results (see text for details).

In the evolution, the values in all nodes tend either to supply values or to this mixed state. Interestingly, in the startup of the evolution, the curves perform several large oscillations between the boundary states, before they relax at the stationary state. This happens due to a complex distribution of flows at the startup phase.

Note that the graphs Figure 2a and Figure 2c have an identical shape, and Figure 2b has the same shape vertically reflected. This happens because there are only two supplies in the network, and the default composition is a linear combination of them. As a result, the trajectory of the system in x-space is limited to a 1-dimensional subspace. Graphs

TABLE I
SUPPLY SETTINGS IN VARIOUS SCENARIOS

scenario	entry	composition	temperature
N1 nat.gas	n99_gm	87% CH_4, 1% C_2H_6, 1% C_3H_8, 1% CO_2, 10% N_2	$303.15K$
N1 nat.gas	n56_gm	85% CH_4, 3% C_2H_6, 1% C_3H_8, 1% CO_2, 10% N_2	$293.15K$
dyn-pipe H_2	n0000	95% H_2, 5% N_2	$313.15K$
dyn-pipe CO_2	n0000	95% CO_2, 3% N_2, 2% O_2	$313.15K$

TABLE II
TESTING VARIOUS IMPLEMENTATIONS OF HEATERS ON N85 NETWORKS SET

implementation of heaters	num. of divergent cases
disabled	3
local	0
nonlocal	85
joined	2

Figure 2a-c are projections of this trajectory to different directions and therefore have the same shape.

Figure 2d shows temperature dependence in selected nodes. During startup evolution, strong heating occurs due to the inverse Joule-Thomson (JT) effect and the influence of the $\partial P/\partial t$-term in (8). With further evolution, the temperature in nodes close to supplies tends to the corresponding constant temperature values of the incoming fluid. In more detail, in the considered scenario, after each supply there is a compressor station, the outlet temperature of which is regulated by a cooler. The outlet temperature of the cooler is set to the same value as that of the corresponding supply. The temperature in network nodes remote from the supply tends to a constant value, slightly below $T_{soil} = 283.15K$, due to the influence of the JT-effect.

N85 networks set: contains 85 realistic natural gas networks, obtained for benchmarking from our industrial partner. The networks are highly resolved, containing up to 4 thousands of nodes each. We used these networks for numerical experiments testing the stability of simulation with a different implementation of heaters. Unlike coolers, which control their own output temperature, heaters must control the temperature in an adjacent element, the regulator. In dynamic formulation of the problem, especially at low flows, heaters do not have time to regulate their temperature in order to constantly ensure the set temperature values in the regulator. This leads to divergences. We have tested several options for implementation of heaters, shown in Table II. For disabled heaters, 3 scenarios out of 85 are divergent. For the most stable implementation option, when heaters control their own local temperature, all scenarios are convergent. If the heaters try to control the temperature nonlocally, in the attached regulators, all scenarios diverge, making such implementation impossible. For our selected option, the heaters are joined with regulators, the unified element controls its own output temperature, 2 scenarios out of 85 are divergent, slightly better than the complete disabling of the heaters. On average, simulating one converging scenario from the N85 set takes about one minute on a 2.6 GHz CPU 16 GB RAM computer.

Hydrogen and carbon dioxide pipelines: this is one of our standard test cases, $L = 150km$ $D = 0.5m$ horizontally laid pipeline, transporting gaseous H_2 or CO_2 in liquid or supercritical phase. The case supports variable spatial discretization, for the considered scenario selected to $nsubdiv = 50$. Time discretization is the same as for N1 network. Supply setting is presented in Table I. The considered scenario has a single fluid composition and is used mainly for testing of the temperature modeling. The dynamic simulation starts from $T_{soil} = 283.15K$ and a different $T_{set} = 313.15K$ at the pipeline entry. The simulation converges to stationary solution with nearly exponential fall of temperature from T_{set} to T_{soil}. For CO_2, an observed stronger deviation from the exponent is due to JT-effect and the nonlinear enthalpy model.

Convergence of iterations: in our implementation, we use the globally convergent Newton's solver with Armijo line search rule [30], applied at every time step. For linear problems, it just forwards the solution to the underlying sparse linear solver, that for non-degenerate problems converges in 1 iteration. Due to proper initialization, at the first time step all phases converge in 0 iteration, just keeping the starting values. This provides a good method to test that all variables are correctly initialized. At the second time step, all mix phases also converge in 0 iteration, while in the last PM phase the network filling begins, and PM phase starts to increase its iteration number. For N1 network and H_2/CO_2 pipe scenarios, all mix phases are solved in 1 iteration on intermediate timesteps, as it should be for non-degenerate linear systems; and in 0 iteration at the last timesteps, due to convergence to stationary solution. For large N85 networks, Tmix phase can have intermediately 2-3 iterations, indicating the remaining degeneracy or the disbalance of scaling factors in Tmix system. This effect is planned to be studied in more details, with the application of principal component analysis [31].

The numerical experiments performed show that the purpose of this work has been fully achieved, the modeling has been extended to include mixing flows and is working for scenarios of varying complexity. The modeling in our system is presented in open text form, as a list of variables and equations, which both we and the users can freely modify. This distinguishes us from the existing solutions, in which the modeling is usually hardcoded within the system. We also provide numerical stability of the modeling and the solution algorithms, which allows us to solve large realistic scenarios in fluid transport simulation.

IV. CONCLUSION AND FUTURE WORK

This paper considered the modeling of mixing flows in dynamic simulation of pipeline fluid transport. Mixed characteristics include molar mass, heat value, chemical composition and temperature of the transported fluids. In the absence of chemical reactions, the modeling is based on the universal conservation laws for molar flows and total energy. The modeling leads to a system of differential algebraic equations, including linear molar mixing formulas, nonlinear temperature-energy relationships, and piecewise-linear element equations for coolers and heaters. In our approach, for nonlinear relations, linearization is carried out in the vicinity of the previous integration step, piecewise-linear relations are reduced to linear ones using the active set method. The resulting sequence of linear systems is solved by a sparse linear solver, typically in one iteration per integration step. The functionality and stability of the developed approach have been tested in a number of realistic network scenarios.

Numerical experiments on the moderate size N1 network allow us to follow the mixing processes in detail, including the evolution of molar mass, heat value, chemical composition, and temperature. Experiments on the N85 set of large-scale networks demonstrate the stability of the developed methods and its sensitivity to such details as nonlocality of equations used in the implementation of heaters. Hydrogen and carbon dioxide pipeline scenarios are used for testing the temperature modeling and the convergence of simulation. Due to the linearity of the mixing equations, their solution is typically carried out in one iteration, representing a minor overhead to solving the main system of nonlinear equations describing the distribution of pressures and flows over the fluid transport networks.

Our future plans include studying the balance of scaling factors in the temperature mixing system for large-scale networks and applying the developed methods to more complex hydrogen and carbon dioxide scenarios.

Acknowledgments

The work has been partly supported by Fraunhofer research cluster CINES and partly by BMBF project TransHyDE, grant no. 03HY201M. We also acknowledge support from Open Grid Europe GmbH in the development and testing of the software.

References

[1] T. Clees, I. Nikitin, and L. Nikitina, "Making network solvers globally convergent", Advances in Intelligent Systems and Computing, vol. 676, 2018, pp. 140-153.

[2] M. Anvari et al., "Stability of dynamic fluid transport simulations", J. Phys.: Conf. Ser., vol. 2701, 2024, 012009.

[3] A. Baldin et al., "On advanced modeling of compressors and weighted mix iteration for simulation of gas transport networks", Lecture Notes in Networks and Systems, vol. 601, 2023, pp. 138-152.

[4] T. Clees et al., "MYNTS: Multi-physics network simulator", in Proc. of SIMULTECH 2016, International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 179-186, SciTePress, 2016.

[5] A. Baldin, T. Clees, B. Klaassen, I. Nikitin, and L. Nikitina, "Topological reduction of stationary network problems: example of gas transport", International Journal On Advances in Systems and Measurements, vol. 13, 2020, pp. 83-93.

[6] T. Clees et al., "Efficient method for simulation of long-distance gas transport networks with large amounts of hydrogen injection", Energy Conversion and Management, vol. 234, 2021, 113984.

[7] M. Anvari et al., "Simulation of pipeline transport of carbon dioxide with impurities", in Proc. of INFOCOMP 2023, the 13th International Conference on Advanced Communications and Computation, pp. 1-6, IARIA, 2023.

[8] A. Baldin et al., "Universal translation algorithm for formulation of transport network problems", in Proc. SIMULTECH 2018, International Conference on Simulation and Modeling Methodologies, Technologies and Applications, vol. 1, pp. 315-322.

[9] J. Nikuradse, "Laws of flow in rough pipes", NACA Technical Memorandum 1292, Washington, 1950.

[10] P. Hofer, "Error evaluation in calculation of pipelines", GWF-Gas/Erdgas, vol. 114, no. 3, 1973, pp. 113-119 (in German).

[11] J. Saleh, ed., Fluid Flow Handbook, McGraw-Hill 2002.

[12] D.-Y. Peng and D.P. Robinson, "A new two-constant equation of state", Ind. Eng. Chem. Fundam, vol. 15, 1976, pp. 59-64.

[13] G. Soave, "Equilibrium constants from a modified Redlich-Kwong equation of state", Chemical Engineering Science, vol. 27, 1972, pp. 1197-1203.

[14] ISO 12213-2: Natural gas – calculation of compression factor, International Organization for Standardization, 2020.

[15] ISO 20765-2: Natural gas – Calculation of thermodynamic properties – Part 2: Single-phase properties (gas, liquid, and dense fluid) for extended ranges of application, International Organization for Standardization, 2020.

[16] O. Kunz and W. Wagner, "The GERG-2008 wide-range equation of state for natural gases and other mixtures: An expansion of GERG-2004", J. Chem. Eng. Data, vol. 57, 2012, pp. 3032-3091.

[17] W. Wagner, Description of the Software Package for the Calculation of Thermodynamic Properties from the GERG-2008 Wide-Range Equation of State for Natural Gases and Similar Mixtures, Ruhr-Universität Bochum, 2022.

[18] E. Egger and J. Giesselmann, "Stability and asymptotic analysis for instationary gas transport via relative energy estimates", Numerische Mathematik, Springer 2023.

[19] J. K. van Deen and S. R. Reintsema, "Modelling of high-pressure gas transmission lines", Appl. Math. Modelling, vol. 7, 1983, pp. 268-273.

[20] C.-Y. Chang, S.-H. Wang, Y.-C. Huang, and C.-L. Chen, "Transient response analysis of high pressure steam distribution networks in a refinery", in Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP), May 28-31, 2017, Taipei, Taiwan, pp. 418-423, IEEE 2017.

[21] T.-P. Azevedo-Perdicoúlis, F. Perestrelo, and R. Almeida, "A note on convergence of finite differences schemata for gas network simulation", in Proceedings of the 22nd International Conference on Process Control, June 11-14, 2019, Štrbské Pleso, Slovakia, pp. 274-279, IEEE 2019.

[22] M. Chaczykowski and A. J. Osiadacz, "Dynamic simulation of pipelines containing dense phase/supercritical CO2-rich mixtures for carbon capture and storage", International Journal of Greenhouse Gas Control, vol. 9, 2012, pp. 446-456.

[23] P. Aursand, M. Hammer, S. T. Munkejord, and Ø. Wilhelmsen, "Pipeline transport of CO2 mixtures: models for transient simulation", International Journal of Greenhouse Gas Control, vol. 15, 2013, pp. 174-185.

[24] S. T. McCoy and E. S. Rubin, "An engineering-economic model of pipeline transport of CO2 with application to carbon capture and storage", International Journal of Greenhouse Gas Control, vol. 2, 2008, pp. 219-229.

[25] M. J. Moran and H. N. Shapiro, Fundamentals of Engineering Thermodynamics, John Wiley and Sons, 2006.

[26] C. Himpe, S. Grundel, and P. Benner, "Next-gen gas network simulation", in: Progress in Industrial Mathematics at ECMI 2021, pp. 107-113, Springer 2022.

[27] C. Himpe, S. Grundel, and P. Benner, "Model order reduction for gas and energy networks", J. Math. Industry, vol. 11:13, 2021, pp. 1-46.

[28] J. Mischner, H.-G. Fasold, and K. Kadner, gas2energy.net, Systemplanung in der Gasversorgung, Gaswirtschaftliche Grundlagen, Oldenbourg Industrieverlag GmbH, 2011 (in German).

[29] GPSA Engineering Data Book, 14th Edition, Gas Processors Suppliers Association, 2016.

[30] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations, SIAM, 1995.

[31] A. Baldin et al., "Principal component analysis in gas transport simulation", in Proc. of SIMULTECH 2022, International Conference on Simulation and Modeling Methodologies, Technologies and Applications, pp. 178-185, SciTePress, 2022.

Comparing Fault-tolerance in Kubernetes and Slurm in HPC Infrastructure

Mirac Aydin, Michael Bidollahkhani, Julian M. Kunkel
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)
Universität Göttingen
Göttingen, Germany
e-mail: {mirac.aydin | julian.kunkel}@gwdg.de , michael.bkhani@uni-goettingen.de

***Abstract*—In this paper, we explore the fault-tolerance mechanisms in Kubernetes and Slurm within High-Performance Computing (HPC) infrastructures. As computational workloads and data requirements continue to expand, ensuring reliable and resilient HPC systems becomes increasingly critical. Our study examines the strategies employed by Kubernetes and Slurm to handle failures, maintain system stability, and provide continuous service. We present a comparative analysis, highlighting the strengths and limitations of each system in various failure scenarios. We review and synthesize findings from existing literature and case studies to infer the effectiveness of these fault-tolerance mechanisms. Through this comprehensive review, we provide insights into the current state of fault-tolerance in Kubernetes and Slurm and propose recommendations for enhancing resilience in HPC environments.**

Keywords-fault-tolerance; Kubernetes; Slurm; High-Performance Computing; HPC; resilience; comparative analysis; literature review; case studies.

I. Introduction

HPC systems are integral to a wide range of scientific and industrial applications, driving advancements in fields, such as climate modeling, bioinformatics, and complex simulations. As the demand for computational power and data processing continues to grow, maintaining the reliability and resilience of HPC infrastructures becomes increasingly crucial. Given that a single protocol failure can potentially disrupt the entire HPC system, fault-tolerance, which is the capacity of a system to maintain operational functionality in the presence of failures, is a critical factor in ensuring the robustness and reliability of such systems [1].

Kubernetes and Slurm are two prominent orchestration and workload management systems used in HPC environments. Kubernetes, originally designed for managing containerized applications [2], has gained traction for its flexibility and scalability [3] [4] [5]. Slurm, on the other hand, is a traditional HPC workload manager known for its efficiency in scheduling and resource management [6] [7]. Both systems offer mechanisms to handle failures, but their approaches and effectiveness can vary significantly.

This paper aims to explore and compare the fault-tolerance mechanisms of Kubernetes and Slurm in the context of the EU DECICE Project [8]. The DECICE Project, funded by the Horizon Europe program, aims to develop an AI-based, open, and portable cloud management framework for the automatic and adaptive optimization and deployment of applications across a federated infrastructure, encompassing HPC systems, cloud, edge, and IoT devices. A key objective of DECICE is to enhance the resilience and efficiency of this compute continuum through dynamic scheduling and fault-tolerance mechanisms.

By reviewing and synthesizing findings from existing literature and case studies, we seek to understand how Kubernetes and Slurm manage common failure conditions, such as node crashes, network partitions, and resource exhaustion, within the framework envisioned by DECICE. Our goal is to provide a comprehensive analysis that highlights the strengths and limitations of each system's fault-tolerance capabilities, contributing to the project's objectives of developing an intelligent management plane and achieving high levels of performance and energy efficiency.

The remainder of this paper is structured as follows: Section II provides a background on fault-tolerance in HPC systems and an overview of Kubernetes and Slurm. Section III discusses the methodology used for our literature review and comparative analysis. Section IV presents the results of our review, detailing the fault-tolerance mechanisms and their effectiveness. Finally, Section V offers conclusions and recommendations for enhancing fault-tolerance in HPC environments using Kubernetes and Slurm, aligned with the goals of the DECICE Project.

By understanding the current state of fault-tolerance in these systems, we aim to contribute valuable insights that can inform future developments and best practices in HPC infrastructure management, ultimately supporting the broader aims of the DECICE Project.

II. Background

In this section, we provide an overview of the key components and mechanisms relevant to fault-tolerance in HPC systems. We begin by outlining the common faults encountered in HPC environments, discussing their types and origins to set the context for the fault-tolerance strategies required. Following this, we explore the architectures and fault-tolerance mechanisms of Kubernetes and Slurm, two prominent orchestration and workload management systems used in HPC. Understanding these foundations is crucial, as Kubernetes and Slurm represent different approaches to managing workloads and ensuring system resilience. This background sets the stage for a detailed comparison of their fault-tolerance capabilities in subsequent sections. The historical evolution and importance of robust fault-tolerance mechanisms are underscored by the increasing demand for reliable and efficient HPC systems, which are critical for advancing scientific research and industrial applications. HPC systems, while immensely powerful, are not immune to faults and failures. Knowing the common types of faults in HPC environments and their origins is crucial for developing effective fault-tolerance mechanisms.

A. Types of Failure in HPC

1) Hardware Failures: Hardware failures are one of the most common and impactful faults in HPC systems. In the paper [9], the authors found out an increase in failure probability as high as 170X for environmental failures and nearly 10X for software failures. These can include:

- **Node Failures**: These occur when individual compute nodes malfunction due to issues, such as overheating, power supply problems, or component wear and tear.
- **Network Failures**: HPC systems rely on complex network infrastructures for inter-node communication. Failures in network hardware, such as switches or routers, can disrupt these communications.
- **Storage Failures**: HPC applications often require high-throughput and low-latency access to large datasets. Failures in storage devices or file systems can lead to significant performance degradation or data loss.

Figure 1 provides a histogram of the failure frequency across different components in an HPC system. The x-axis lists various components prone to failures, including applications (APPL), CPU cores (CORE), controllers (CTRL), disk storage (DISK), cooling fans (FAN), hypervisors (HSV), operating systems (OS), power supplies (PS), and scientific backplanes (SCI_BP). The y-axis represents the frequency of failures observed in each component. Each bar in the histogram corresponds to the failure frequency of a specific component, with the height of the bar indicating how often failures occurred for that component. The histogram shows that disk storage (DISK) has the highest frequency of failures among all components, followed by controllers (CTRL) and power supplies (PS). Other components, such as applications (APPL), CPU cores (CORE), and hypervisors (HSV) have relatively lower failure frequencies. This figure highlights the critical areas within an HPC system that are more prone to failures, emphasizing the need for robust fault-tolerance mechanisms, especially for components with high failure frequencies. Understanding these failure patterns can help in designing more resilient HPC systems and implementing effective predictive maintenance strategies.

2) Software Failures: Software-related faults can arise from bugs, configuration errors, or resource management issues. Common software faults include:

- **Application Crashes**: Applications running on HPC systems may crash due to bugs, unhandled exceptions, or invalid inputs.
- **Operating System Failures**: The operating system on compute nodes can experience kernel panics or other critical failures that disrupt node operations.
- **Middleware Failures**: HPC systems often utilize middleware for job scheduling, resource allocation, and data management. Failures in these middleware components can lead to job delays or failures.

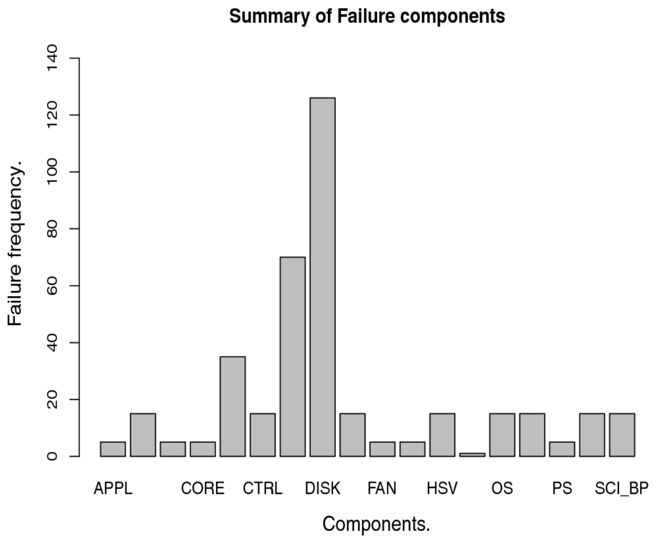


Figure 1. Summary of HPC hardware components' failure [10].

3) Human Errors: Human errors are another significant source of faults in HPC systems. These can include:

- **Configuration Errors**: Incorrect configuration of hardware, software, or network settings by administrators can lead to system instability or inefficiencies.
- **Operational Mistakes**: Mistakes made during system maintenance, updates, or job submissions can cause unintended downtimes or performance issues.

4) Environmental Factors: Environmental factors, although less common, can also affect HPC systems. These include:

- **Power Outages**: Unexpected power outages can lead to abrupt system shutdowns, data corruption, and hardware damage.
- **Cooling Failures**: HPC systems generate substantial heat and rely on efficient cooling mechanisms. Failures in cooling systems can cause overheating and subsequent hardware failures.

B. Origins of Faults in HPC

The origins of faults in HPC systems are varied and can be traced back to several underlying causes:

1) Complexity and Scale: HPC systems are inherently complex, comprising thousands of interconnected nodes, sophisticated networking, and vast storage solutions. This complexity increases the likelihood of faults due to the sheer number of components and interactions involved.

2) Technological Limits: As HPC systems push the boundaries of current technology to achieve greater performance, they also encounter the limitations of that technology. This can include issues like hardware failures due to higher operational stress and software bugs that surface under extreme conditions.

3) Evolution and Upgrades: HPC systems continuously evolve with new hardware and software upgrades. However, integrating new components with existing infrastructure can introduce compatibility issues, configuration errors, and unforeseen bugs.

4) Operational Environment: The operational environment of HPC systems, including physical location, power supply stability, and cooling efficiency, can significantly impact their reliability. Adverse conditions in these areas can exacerbate fault occurrences.

Understanding these common faults and their origins is essential for developing robust fault-tolerance strategies. The subsequent sections will explore how Kubernetes and Slurm address these challenges, providing insights into their fault-tolerance mechanisms within HPC infrastructures.

C. Kubernetes

Kubernetes is an open-source platform that helps to manage containerized workloads and services. It supports easy configuration and automation. With a quickly growing community, there are plenty of services, support options, and tools available [11].

1) Kubernetes Architecture: Figure 2 provides a high-level overview of Kubernetes architecture. A cluster is made up of compute machines, called nodes, and a control plane. The control plane manages the cluster, while the nodes run user applications. Typically, the control plane operates on a separate physical device, but it can also share a device with a compute node or be spread across multiple devices for redundancy. Pods, which are the scheduling units in Kubernetes, house the application containers. The following sections detail the Kubernetes components [12]:

- **API Server:** Exposes the Kubernetes API via HTTP.
- **Controller manager:** Manages all standard controllers for Kubernetes resources, like pods, services, and deployments. Each controller ensures that the resource's current state matches the desired state.
- **etcd:** A persistent distributed key-value store, which stores resources and cluster configuration information.
- **Scheduler:** Component responsible for determining the most suitable node for a given pod.
- **kubelet:** Agent that runs on every node within a Kubernetes cluster. It communicates via a REST API with the API server and handles interactions with the underlying container runtime.
- **Container runtime:** Component responsible for managing the containers' lifecycle. Examples of runtimes are CRI-O [13] and Containerd [14].
- **kube-proxy:** Facilitates communication with pods and implements the Service concept in Kubernetes, which abstracts the IP addresses of application pods to provide a unified method for exposure.

D. Fault Tolerance in Kubernetes

By default, Kubernetes offers two mechanisms for ensuring reliability: self-healing and replication. Self-healing involves

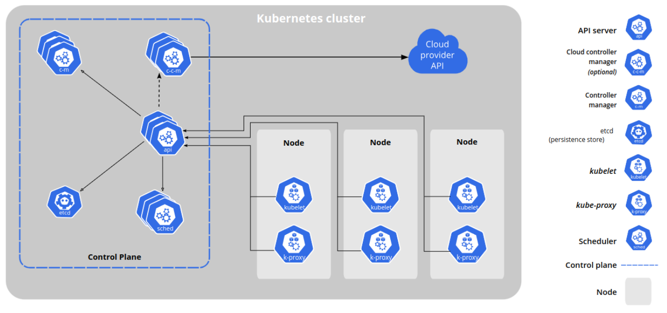


Figure 2. Kubernetes components [15].

Kubernetes restarting or replacing containers that fail, terminating unresponsive containers based on user-defined health checks, and delaying client access until containers are ready to serve [11]. Replication ensures a specified number of pod replicas are constantly operational, guaranteeing continuous availability of pods or homogeneous pod sets [16].

Different fault-tolerance mechanisms can be also be implemented into Kubernetes. They are described below:

1) Horizontal Pod Autoscaler (HPA): Kubernetes boasts autoscaling as a crucial feature. This capability allows containerized applications to function reliably without constant manual intervention. A core autoscaling tool within Kubernetes is the HPA.

HPA guarantees high availability by dynamically adjusting the number of running pods, which are the execution units within the cluster. When triggered, HPA seamlessly adds new pods to distribute the workload, preventing negative impacts on existing pods.

To make autoscaling decisions, Kubernetes relies on metrics, which are statistics gathered from pods, applications, host machines, and the overall cluster. By default, Kubernetes uses Resource Metrics, which primarily monitor CPU and memory usage of pods and host machines. To enhance HPA's performance and flexibility, you can incorporate Customizable Metrics with the help of external software [17].

2) CRIU: CRIU (Checkpoint/Restore in Userspace) is a critical tool for managing process state in Linux environments. It empowers users to capture a running container or individual application at a specific point in time by performing a checkpoint. This checkpoint essentially freezes the process, recording its entire state, including memory, registers, and open file descriptors. This captured data can then be leveraged to restore the application and resume execution precisely from the checkpointed state. CRIU unlocks a multitude of powerful functionalities within the container orchestration landscape. These functionalities include live migration of containers or applications across machines without downtime, facilitating seamless application rollbacks through the creation of snapshots, and enabling remote debugging of running processes for efficient troubleshooting [18].

3) RAFT Protocol: Raft ensures that all replicas within the cluster maintain identical state machines, guaranteeing data integrity even in the face of node failures. It leverages a passive replication approach, where each node can assume one of three roles: leader, follower, or candidate [19]:

- **Leader:** The cluster elects a single leader node responsible for steering communication. The leader receives incoming requests, processes them, replicates state machine changes across follower nodes, and relays responses back to clients.
- **Follower:** While a leader is active, all other nodes transition to a follower state. Followers passively wait for the leader to transmit state machine updates, which they then apply to maintain consistency.
- **Candidate:** In the event of a leader failure, follower nodes transition to a candidate state and initiate a voting process to elect a new leader, ensuring rapid recovery and continued cluster operation.

E. SLURM

Slurm is a robust, open-source system designed to manage clusters and schedule jobs on Linux platforms, suitable for both large-scale and smaller deployments. It operates seamlessly without requiring kernel modifications and functions as a self-contained framework. As a cluster workload manager, Slurm performs three key roles: it allocates exclusive or non-exclusive access to compute nodes for users over defined time periods, facilitates the initiation, execution, and monitoring of jobs (especially parallel tasks) across allocated nodes, and manages resource contention by prioritizing pending work in a queue. These capabilities make Slurm essential for efficiently orchestrating cluster operations and optimizing job performance across diverse computing environments [20].

The main components of Slurm are detailed below, with a sample architecture depicted in Figure 3:

- **slurmctld:** This component monitors resource states, decides when and where to initiate jobs and job steps, and handles nearly all user commands, excluding database-related operations.
- **slurmd:** As Slurm's compute node daemon, slurmd is responsible for executing actual work on the node.
- **slurmdbd:** This component records accounting information and centrally manages certain configuration details, such as limits, fair share information, Quality of Service (QoS) settings, and licenses [21].

F. Fault Tolerance in SLURM

In the realm of HPC, fault-tolerant software is crucial in HPC environmeparamounts. Job resubmissions due to errors significantly impact resource utilization, leading to longer queue times for all users. While Slurm offers inherent fault tolerance, it primarily focuses on hardware issues rather than software errors. This ensures that malfunctioning jobs, exceeding resource limits or encountering failures, are isolated and prevented from disrupting other running jobs. However, the responsibility remains with users and developers to craft robust software, minimizing the need for job resubmissions caused by execution-time errors [1].

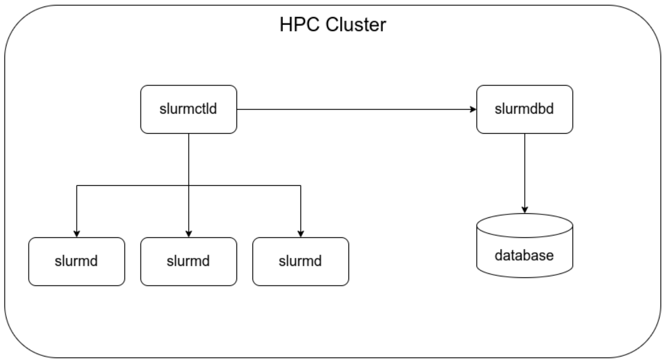


Figure 3. Typical Slurm architecture.

III. Methodology

In this section, we describe the methodology used for our literature review and comparative analysis of fault-tolerance mechanisms in Kubernetes and Slurm. We detail our approach to data collection, analysis techniques, and the criteria used for evaluating the effectiveness of fault-tolerance strategies.

A. Literature Review

To collect relevant academic papers, technical reports, and case studies on fault-tolerance in Kubernetes and Slurm, we searched various databases including IEEE Xplore, Google Scholar, and ACM Digital Library. We used search terms, such as "fault-tolerance in Kubernetes," "fault-tolerance in Slurm," "HPC fault-tolerance," and "resilience in HPC systems." We reviewed and summarized key findings and methodologies used in these studies to understand the current state of research and identify gaps for further exploration.

B. Comparative Analysis Framework

To collect logs from a Kubernetes cluster, including server logs and logs from objects, such as pods, deployments, and services, the EFK stack (Elasticsearch, Fluentd, and Kibana) has been installed. Elasticsearch [22] is a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Fluentd [23], a fast, lightweight, and highly scalable logging and metrics processor, handles log forwarding. Kibana [24] provides a data visualization dashboard for Elasticsearch. Together, these tools enable the collection and analysis of Kubernetes-related system logs. Figure 4 illustrates the general architecture and workflow of the EFK stack.

In the context of Slurm workload management, agent logs generated by both the slurmctld control daemon and slurmd worker daemons on corresponding servers were collected and analyzed thoroughly. This analysis aimed to identify errors and malfunctions within HPC systems. Through a systematic examination of these logs, recurring patterns and underlying causes of system failures were identified.

We systematically evaluated each system against specific criteria, tracking essential aspects throughout the process. The analysis included metrics and benchmarks, such as Mean

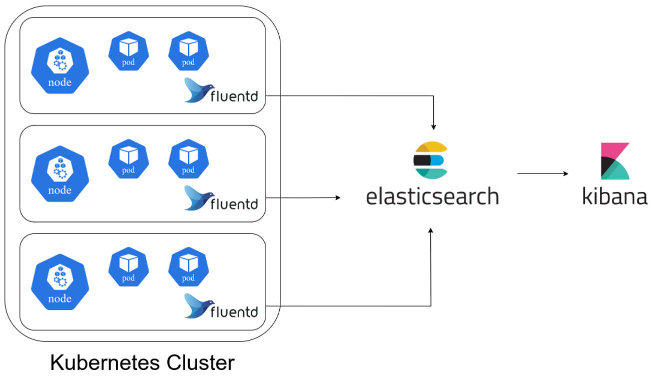


Figure 4. EFK stack on Kubernetes.

Time to Recovery (MTTR), detection accuracy, and resource utilization rates. Table I presents a comparison of these criteria for Kubernetes and Slurm.

As previously mentioned, various tools have been employed to collect logs and data from Slurm and Kubernetes. Table II provides a detailed list of these tools, outlining their functionalities and the specific data they collect.

C. *Log Files Processing Steps*

The processing of log files involved several intricate steps to ensure accurate labeling and analysis of the log entries, facilitating the identification and categorization of fault-tolerance mechanisms in HPC environments:

1) Data Cleaning and Preprocessing: Initially, raw log data from both Kubernetes and Slurm environments were cleaned to remove any extraneous information and ensure consistency in formatting. This involved:

- Removing duplicate entries to prevent data redundancy.
- Parsing timestamps and standardizing date-time formats.
- Filtering out non-informative log entries, such as routine status updates that do not indicate errors or significant events.

2) Labeling Using Language Models: To categorize the log entries, we employed a Large Language Model (LLM) for automated labeling. The steps included:

- Extracting messages from the cleaned log files.
- Using the LLM to generate unique labels for each log entry, indicating the type of failure or operational state.
- Implementing a similarity check using TF-IDF vectorization and cosine similarity to avoid redundant labels. If a new log entry was found to be 70
- Periodically saving the progress to avoid data loss during processing.

The following pseudocode summarizes this process:

```
for each log entry in cleaned_log_data:
    if is_similar(log_entry, existing_entries):
        assign existing label
        else:
            label = generate_label_with_LLM(log_entry,
             existing_labels)
        update existing_labels and existing_entries
    save progress periodically
```

3) Parallel Processing with GPU Acceleration: Given the large volume of log data, processing was accelerated using parallel computing techniques on GPUs. This approach significantly reduced the time required for vectorization and similarity computations:

- Utilizing CuPy for GPU-based array operations to handle large-scale vector computations efficiently.
- Employing parallel processing libraries, such as Dask to distribute the workload across multiple GPU cores.

4) Generating Summary Statistics and Visualizations: After labeling the log entries, summary statistics were computed to understand the distribution and frequency of different failure types. Visualization tools were employed to create charts and graphs depicting these distributions:

- Bar charts showing the number of log entries per label.
- Pie charts illustrating the percentage distribution of each label category.

The results were saved in both graphical and tabular formats to facilitate further analysis and reporting.

IV. Results

This section presents the findings from our literature review and comparative analysis. We detail the fault-tolerance mechanisms of Kubernetes and Slurm, highlighting their strengths and limitations in different failure scenarios.

A. *Fault-Tolerance Mechanisms in Kubernetes*

Kubernetes employs several fault-tolerance mechanisms to ensure system resilience:

- **Self-Healing**: Kubernetes restarts failed containers, replaces containers, and kills containers that do not respond to health checks.
- **Replication**: Ensures a specified number of pod replicas are running at any time.
- **HPA**: Adjusts the number of pods based on CPU/memory usage or custom metrics, improving availability.
- **CRIU**: Enables checkpointing and restoring of container states, supporting live migration and snapshots.
- **RAFT Protocol**: Ensures consistency among replicas, with leader election and state synchronization mechanisms.

Case studies have shown these mechanisms to be effective in maintaining high availability and quick recovery in various failure scenarios [12] [19].

B. *Fault-Tolerance Mechanisms in Slurm*

Slurm's fault-tolerance mechanisms focus primarily on hardware faults:

- **Node Failover**: Automatically reassigns jobs from failed nodes to healthy ones.
- **Job Checkpointing**: Allows jobs to be checkpointed and restarted from the last checkpoint in case of failure.
- **Health Checks**: Monitors node health and takes corrective actions, such as draining or rebooting unhealthy nodes.

TABLE I. DETAILED COMPARISON OF FAULT DETECTION, EVALUATION, AND TOLERANCE FACTORS IN KUBERNETES AND SLURM ENVIRONMENTS

Factor	Kubernetes	Slurm
MTTR	MTTR in Kubernetes is calculated by adding up all the downtime in a specific period and dividing it by the number of incidents.	MTTR in SLURM is calculated by taking the total repair time resulting from a particular failure and dividing it by the total number of repairs that are performed during a specific period.
Fault Detection Accuracy	Kubernetes has high fault detection accuracy due to liveness, readiness, and startup probes.	SLURM has moderate fault detection accuracy, primarily relying on node health checks and job statuses.
Resource Utilization Overhead	Kubernetes has a moderate overhead (10-15%) from extensive monitoring and replication.	SLURM has lower overhead (5-10%) due to simpler fault detection mechanisms.
Recovery Mechanisms	Kubernetes has built-in recovery mechanisms. For example, if a pod fails, Kubernetes automatically restarts it.	SLURM has mechanisms for recovery, but specific details are not readily available.
Failure Types and Frequencies	Specific data on failure types and frequencies for Kubernetes is not readily available.	Specific data on failure types and frequencies for SLURM is not readily available.

TABLE II. FAULT DETECTION TOOLS USED IN SLURM AND KUBERNETES ENVIRONMENTS

Environment	Tools	Functionality	Data Collected
SLURM	`sacct` [25]	Job accounting and fault detection.	Job accounting data.
	`sview` [26]	Graphical interface for fault detection and monitoring.	Slurm configuration, job, step, node and partitions state information.
	`Slurm simulator` [27]	Simulation for PdM and fault detection.	Effects of different Slurm parameters on HPC resource performance.
Kubernetes	`kubectl` [28]	Command line tool for fault detection and management.	Kubernetes cluster's control plane data.
	`Elasticsearch` [22]	Distributed search engine.	Different types of searches on collected data.
	`Fluentd` [23]	Data collector for fault detection.	System and component logs in the cluster.
	`Kibana` [24]	Visualization and monitoring tool for fault detection.	Visualization dashboards, cluster metrics.

- **Job Requeueing**: Jobs that fail due to node failures can be requeued and run on different nodes.

These mechanisms help mitigate the impact of hardware failures, but software faults and application-level failures require additional user intervention [1].

C. Error Distribution in Slurm and Kubernetes

The scenario analyzed in this subsection involves a critical examination of node failures during high-intensity computational tasks. To clarify, the analysis considers both the immediate impact on job execution and the subsequent recovery processes initiated by the fault-tolerance mechanisms in Kubernetes and Slurm.

Based on the data collected from a Slurm environment, we identified and categorized various errors. Figure 5 illustrates the distribution of different error types encountered. The most frequent error was the General Warning, comprising 64.95% of all errors. This was followed by PrologRunningError (21.13%), NodeError (13.61%), and OutOfMemoryError (0.14%). Other errors, such as JobCancelError, PartitionError, JobExitError, and TimeLimitExhaustedError each accounted for less than 0.1% of the total errors. The least frequent errors included LogrotateError, TopologyError, and NodeListError, each constituting approximately 0.0002% of the total errors.

In contrast, the Kubernetes environment exhibited a different error distribution 6. The most frequent error identified was the HTTP2StreamClosedError, which accounted for 20.08% of the total errors. This was followed by StreamClosedError (8.01%) and KubernetesRejectedConnection (4.25%). Additionally, errors such as KubernetesAPIEndpointConnectionFailure, ContextCanceledFailure, and ErrorSyncingHelmClusterRepo were also significant, each contributing to around 4.01% of the total errors. Less frequent issues included ConnectionRefusedError, DialTimeoutError, and etcdConnectionFailure, which occurred with a frequency of less than 1%. These findings highlight the critical areas where Kubernetes fault-tolerance mechanisms must focus, particularly on managing API server errors and stream-related failures.

The comparison of these distributions underscores the varying challenges faced by Slurm and Kubernetes environments, emphasizing the need for tailored fault-tolerance strategies in each system.

This distribution highlights the prevalence of certain errors over others, emphasizing the need for targeted fault-tolerance strategies that prioritize the most common and impactful issues.

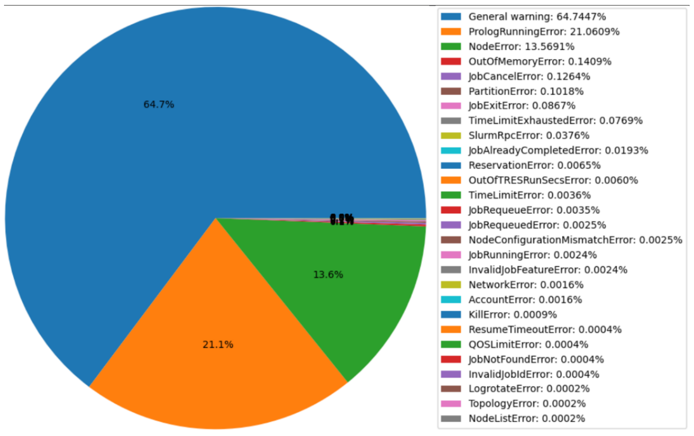


Figure 5. Distribution of error types in GWDG SCC Node Clusters Case study.

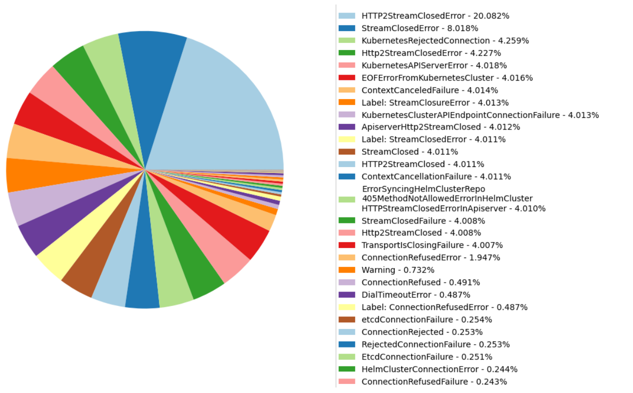


Figure 6. Distribution of error types in EU DECICE Project Kubernetes Clusters Case study.

D. Comparative Analysis

Our comparative analysis highlights the following differences:

- **Recovery Time**: Kubernetes generally achieves faster recovery times due to its self-healing and replication mechanisms. Slurm's recovery time depends on the effectiveness of node failover and job checkpointing.
- **Fault Detection**: Kubernetes has robust fault detection capabilities with its health checks and RAFT protocol. Slurm relies more on node health checks, which may not detect all types of faults promptly.
- **System Overhead**: Kubernetes introduces some overhead due to its extensive monitoring and replication processes. Slurm has lower overhead but may require more manual intervention for fault management.

Kubernetes excels in scenarios requiring high availability and rapid recovery, making it suitable for dynamic and scalable environments. Slurm is highly effective in traditional HPC setups with stringent resource management needs but may need enhancements for better software fault-tolerance.

V. Discussion

In this section, we interpret the results of our comparative analysis, discussing their implications for HPC infrastructure management. We also explore potential improvements and future directions for enhancing fault-tolerance in Kubernetes and Slurm.

A. Implications for HPC Management

Our findings suggest that:

- Kubernetes is well-suited for environments that require high scalability and rapid fault recovery. Its self-healing and autoscaling capabilities provide robust fault-tolerance with minimal manual intervention.
- Slurm remains a strong candidate for traditional HPC environments, particularly where job scheduling and resource management efficiency are highly valued.
- Combining elements of both systems could potentially yield a more comprehensive fault-tolerance strategy for HPC infrastructures, leveraging Kubernetes' dynamic fault management with Slurm's efficient resource scheduling.

In addition to the error distribution analysis in Slurm, a similar examination of error distribution in Kubernetes clusters reveals distinct patterns. Errors related to pod restarts, node unresponsiveness, and network partitions are prevalent. These findings underscore the importance of tailored fault-tolerance strategies in Kubernetes, comparable to those implemented in Slurm, to mitigate these common issues. Key considerations for choosing between Kubernetes and Slurm include the specific fault-tolerance needs, the complexity of workloads, and the desired level of automation in fault management. To address the limitations of both systems effectively, we recommend exploring practical solutions such as implementing dynamic fault-tolerance mechanisms tailored to specific failure scenarios. For instance, as discussed in [29], AI plays a crucial role in predictive maintenance strategies, integrating machine learning algorithms to predict potential failures and automatically adjusting resource allocations could enhance system resilience. Additionally, adopting hybrid models that combine the strengths of Kubernetes and Slurm may offer a balanced approach to fault tolerance in diverse HPC environments.

B. Future Directions

Future research and development could focus on the following areas:

- Enhancing Slurm's software fault-tolerance capabilities, possibly by integrating Kubernetes-like self-healing and replication mechanisms.
- Developing hybrid models that combine the strengths of Kubernetes and Slurm for improved fault-tolerance in diverse HPC environments.
- Exploring emerging technologies, such as machine learning for predictive fault detection and proactive fault management in HPC systems.

Potential enhancements to existing mechanisms in Kubernetes and Slurm could also involve better integration with AI-based monitoring tools and more granular control over fault-tolerance policies, enabling more effective and efficient fault management.

VI. Conclusion

The goal of this study was to compare the fault-tolerance mechanisms in Kubernetes and Slurm within HPC infrastructures. By examining the logs retrieved from GWDG SCC Node Clusters managed by Slurm, we identified and categorized various failures to understand their prevalence and impact. Our findings indicate that the most prevalent failure type is the General Warning, which accounts for 64.95% of all errors. This is followed by PrologRunningError (21.13%) and NodeError (13.61%), highlighting common issues related to job initialization and node malfunctions. Less frequent errors include OutOfMemoryError (0.14%), JobCancelError (0.13%), and PartitionError (0.10%). The rarest errors, such as LogrotateError, TopologyError, and NodeListError, each constitute approximately 0.0002% of the total errors, indicating specific system or configuration issues that occur infrequently.

Kubernetes can effectively handle these types of failures through its robust fault-tolerance mechanisms. Self-healing capabilities can automatically restart failed containers, addressing General Warnings and NodeErrors. HPA and resource limits can mitigate OutOfMemoryErrors by adjusting resources dynamically. Kubernetes' replication and RAFT protocol ensure high availability and data consistency, which can reduce the impact of network and configuration errors.

Slurm provides node failover and job checkpointing to handle node and job-related errors. Health checks and job requeueing mechanisms help maintain system stability by reallocating jobs from failed nodes to healthy ones, thereby addressing PrologRunningErrors and NodeErrors. For resource management issues like OutOfMemoryError and PartitionError, Slurm's resource scheduling and management features can be tuned to optimize usage and prevent such failures.

While our recommendations provide a strong foundation, concrete guidelines for achieving fault tolerance are necessary. These guidelines should include best practices for configuring and tuning fault-tolerance mechanisms, such as setting appropriate thresholds for autoscaling and designing robust health check probes.

By understanding these error distributions and leveraging the fault-tolerance mechanisms of Kubernetes and Slurm, HPC administrators can enhance system resilience and fault-tolerance. This targeted approach will improve the overall reliability and efficiency of HPC infrastructures. Future research should focus on integrating advanced fault detection and management technologies to further bolster the resilience of HPC systems.

Acknowledgment

The authors gratefully acknowledge the Göttingen Scientific Compute Cluster (SCC) for providing crucial cloud computing resources that greatly aided this research. We extend our special thanks to Felix Stein, Mojtaba Akbari, and Jonathan Decker for their exceptional support and contributions to the DECICE Project, including Felix's assistance with code optimization for analysis and their collective work on the project's infrastructure development.

This research was performed in the context of the DECICE project, funded by the European Union (grant agreement no. 101092582). Additionally, this project was supported by the Federal Ministry of Education and Research (BMBF), Germany, under grant no. 01|S22093A for the AI service center KISSKI.

References

[1] K. Klenk and R. J. Spiteri, "Improving resource utilization and fault tolerance in large simulations via actors," *Cluster Computing*, vol. 27, no. 6323–6340, 2024.

[2] G. Sayfan, *Mastering Kubernetes: Master the art of container management by using the power of Kubernetes*, 2nd ed. Packt Publishing Ltd., 2018.

[3] L. Dewi, A. Noertjahyana, H. Palit, and K. Yedutun, "Server scalability using kubernetes," in *2019 4th technology innovation management and engineering science international conference (TIMES-iCON)*, IEEE, 2019, pp. 1–4.

[4] G. Turin *et al.*, "A formal model of the kubernetes container framework," in *International Symposium on Leveraging Applications of Formal Methods*, Springer International Publishing, 2020, pp. 558–577.

[5] S. Muralidharan, G. Song, and H. Ko, "Monitoring and managing iot applications in smart cities using kubernetes," in *The Tenth International Conference on Cloud Computing, GRIDs, and Virtualization*, 2019, pp. 1–7.

[6] M. D'Amico, "Scheduling and resource management solutions for the scalable and efficient design of today's and tomorrow's hpc machines," Ph.D. dissertation, Universitat Politècnica de Catalunya, 2021.

[7] N. A. Simakov *et al.*, "A slurm simulator: Implementation and parametric analysis," in *High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation: 8th International Workshop, PMBS 2017, Denver, CO, USA, November 13, 2017, Proceedings 8*, Springer International Publishing, 2018, pp. 197–217.

[8] J. M. Kunkel *et al.*, "Decice: Device-edge-cloud intelligent collaboration framework," in *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 2023, pp. 266–271.

[9] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail," in *2013 43rd annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, IEEE, 2013, pp. 1–12.

[10] B. Mohammed, I. Awan, H. Ugail, and M. Younas, "Failure prediction using machine learning in a virtualised hpc system and application," *Cluster Computing*, vol. 22, pp. 471–485, 2019.

[11] *Kubernetes concepts - overview*, https://kubernetes.io/docs/concepts/overview/, Accessed: 2024-06-28.

[12] H. Schmidt, Z. Rejiba, R. Eidenbenz, and K.-T. Förster, "Transparent fault tolerance for stateful applications in kubernetes with checkpoint/restore," in *2023 42nd International Symposium on Reliable Distributed Systems (SRDS)*, IEEE, 2023, pp. 129–139.

[13] *Cri-o - lightweight container runtime for kubernetes*, https://cri-o.io/, Accessed: 2024-06-30.

[14] *Containerd - an industry-standard container runtime with an emphasis on simplicity, robustness and portability*, https://containerd.io/, Accessed: 2024-06-30.

[15] *Kubernetes components*, https://kubernetes.io/docs/concepts/overview/components/, Accessed: 2024-06-28.

[16] *Replication controller*, https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/, Accessed: 2024-06-28.

[17] T. T. Nguyen, Y. J. Yeom, T. Kim, D. H. Park, and S. J. Kim, “Horizontal pod autoscaling in kubernetes for elastic container orchestration,” *Sensors*, vol. 20, no. 16, p. 462, 2020.

[18] *Criu: Checkpoint/restore in userspace*, https://criu.org/Main_Page, Accessed: 2024-06-28.

[19] G. M. Diouf, H. Elbiaze, and W. Jaafar, “On byzantine fault tolerance in multi-master kubernetes clusters,” *Future Generation Computer Systems*, vol. 109, pp. 407–419, 2020.

[20] *Slurm overview*, https://slurm.schedmd.com/overview.html, Accessed: 2024-06-28.

[21] M. A. Jette and T. Wickberg, “Architecture of the slurm workload manager,” in *Workshop on Job Scheduling Strategies for Parallel Processing*, Springer Nature Switzerland, 2023, pp. 3–23.

[22] *Elasticsearch - a distributed, restful search and analytics engine*, https://www.elastic.co/elasticsearch, Accessed: 2024-06-30.

[23] *Fluentd - an open source data collector*, https://www.fluentd.org/, Accessed: 2024-06-30.

[24] *Kibana - a data visualization dashboard*, https://www.elastic.co/kibana, Accessed: 2024-06-30.

[25] *Slurm workload manager - sacct*, https://slurm.schedmd.com/sacct.html, Accessed: 2024-06-30.

[26] *Slurm workload manager - sview*, https://slurm.schedmd.com/sview.html, Accessed: 2024-06-30.

[27] *Slurm simulator*, https://ubccr-slurm-simulator.github.io/, Accessed: 2024-06-30.

[28] *Command line tool (kubectl) | kubernetes*, https://kubernetes.io/docs/reference/kubectl/, Accessed: 2024-06-30.

[29] M. Bidollahkhani and J. M. Kunkel, “Revolutionizing system reliability: The role of ai in predictive maintenance strategies,” in *CLOUD COMPUTING 2024: The Fifteenth International Conference on Cloud Computing, GRIDs, and Virtualization*, ISBN: 978-1-68558-156-5, 2024.

FEM Modeling for PCB Assembly Simulation

Ming-Hsiao Lee, Jiunn-Horng Lee, Chih-Min Yao, Jen-Gaw Lee
National Center for High-performance Computing, NARL
30076 Hsinchu City, Taiwan
Email: {9303103, jhlee, 9403116, jglee}@narlabs.org.tw

***Abstract*— A Printed Circuit Board (PCB) is a multi-layered composite, which consists of several layers of copper circuits and dielectric materials. It is used as a backbone to carry and connect various electronic components, i.e., PCB Assembly (PCBA), to achieve certain functions. Currently, most of the products need PCBAs to fulfill their functions. Nevertheless, because the components, e.g., chips, are decreasing in size, the accuracy and reliability of PCBAs have become critical issues, either in manufacturing processes or in actual uses. One of the serious problems is the warpage of the PCB, which is induced by the thermal mismatch due to unevenly distributed circuits and multi-materialled components during the manufacturing process, which experiences a large temperature change. This may cause defects and failures in the assemblies, e.g., the chips and PCB are not well connected due to the dislocation. It is helpful to simulate in advance to evaluate possible defects. However, since the circuits on a PCB are very tiny, compared with the size of the PCB or the components, it is unpractical and not feasible to build a Finite Element Method (FEM) model to include all the details of circuits and components, which are attached to the PCB with a big amount of solder joints. To avoid the difficulties, a new effective modeling approach, which adopts equivalent material properties, is proposed for PCBA's simulation. In this approach, the multi-layered PCB and the attached components are modeled with a moderate mesh, while the circuit's and solder joint's effects are still included. With the proposed approach, the analysis model can be useful and efficient for simulating the PCBA to evaluate the manufacturing process and structural characteristics of the PCBA.**

Keywords- PCB; PCBA; Warpage; Trace mapping.

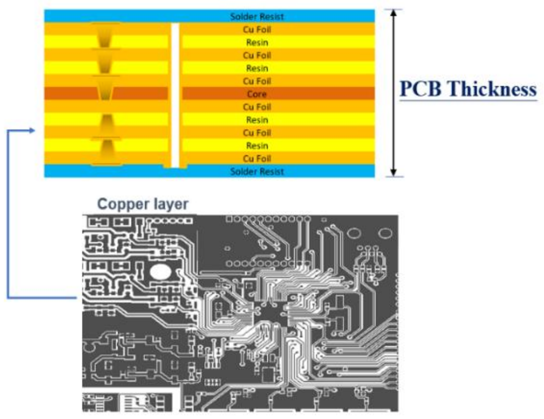


Figure 1. A typical stack-up of a PCB.

I. Introduction

A PCB is a multi-layered composite, which includes many layers of copper circuits and dielectric materials, as shown in Figure 1. Currently, the circuit trace width on the board is getting smaller. Due to the unevenly distribution of the circuits on different copper layers, the layer's stiffnesses, and thermal expansion coefficients are also unevenly distributed, which affect the characteristics of the board's structural behaviors, especially causing warpage due to thermal mismatch. Therefore, to evaluate the design, the distribution of circuits should be taken into account during the structural analyses. Nevertheless, the trace width and thickness are so tiny, i.e., in the order of microns or sub-microns, much smaller than that of the PCB board, i.e., normally in the order of centimeters. Moreover, one PCB may consist of tens of trace layers and dielectric layers. If all the traces are also meshed, the element size could be extremely small, i.e., in the order of microns or sub-microns, and then the whole model could reach billions of elements. This is not a workable case. This type of case actually is a multi-scaled problem. To simulate this type of problem, often an equivalent material approach [1] is adopted. Here, as proposed, first, the multi-layered PCB structure is meshed independently of the circuits so that the mesh size is not affected by the width of the traces. Instead, a moderate element size can be chosen. Then, the circuit part adopts an equivalent material approach; it is to calculate the area ratio of copper circuits distributed on each element of the circuit layers, so called trace mapping, as shown in Figure 2.

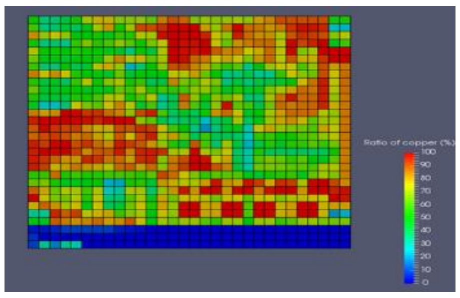


Figure 2. Copper (trace) ratio distribution.

After the copper ratio on each element of the circuit layer is obtained, calculate its equivalent material properties, such as Young's modulus and the coefficient of thermal expansion, according to the copper ratio [2][3][4] (Zwemer et al. and Bajaj et al. tackle only the PCB with the equivalent material property approach). More than that, the relationship between the equivalent material properties and the copper ratio can be also derived by a numerical test measure, as the one shown in Figure 3. It is like tensile testing, i.e., test models with various copper ratios are analyzed to inversely derive their equivalent material properties, such as Young's modulus. Then, the relationship between the Young's modulus and the copper ratio is obtained and used to calculate the equivalent material properties for each circuit layer element. Once the generated FEM models including equivalent material properties for each element on different trace layers are ready (all are generated automatically with the self-developed programs), the FEM model can be solved with available solvers, e.g., Calculix (an open-source software) [5]. With this approach, although the circuit layer effect has been included, the mesh model is still moderate and suitable for the analysis. Similarly, PCB with mounted chips (PCBA) can be simulated in the same way. However, the chips and the solder bump layers, which mount the chips onto the PCB, also need to take advantage of equivalent material approach in order to solve the cases efficiently.

The rest of the paper is structured as follows. In Section II, we present the simulation processes and results. We finish in Section III with some concluding remarks.

II. RESULTS

A PCBA, as shown in Figure 4, usually is manufactured in batches; each batch includes several pieces of the final PCBA modules, as shown in Figure 5. The warpage could happen during the manufacturing processes, e.g., the Surface Mount Technology (SMT) process, which experiences a great temperature change. This is mainly induced by the thermal mismatch due to the unevenly distribution of circuits, multi-materialled chips, the shielding frames, etc. Figure 5 shows warpage of a batch panel of PCBAs. If the warpage is large, it may cause defects and failures in the assemblies, e.g., the chips and PCB are not well connected due to the dislocation. In addition, in many circumstances during the use, the cyclic temperature change, as a cyclic loading, could also cause fatigue failures. This is also a typical problem for the PCBA.

The structure of a PCBA is not only extremely complicated, but also multi-scaled, so simulating the PCBA has been a challenging problem. However, the proposed effective modeling approaches for the tiny circuits and solder joints can effectively solve these difficulties. It can distinctly reduce the size of the analysis model and efficiently handle the effect of the circuits and solder joints for the simulation. With the effective modeling approaches, evaluating the structural behaviors of PCBAs becomes simple and efficient. Moreover, even irregular-shaped PCBs can be handled, as shown in Figure 6.

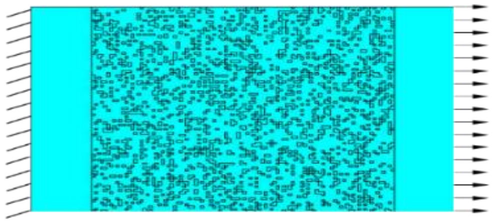


Figure 3. Calculation setting for equivalent material proprieties.

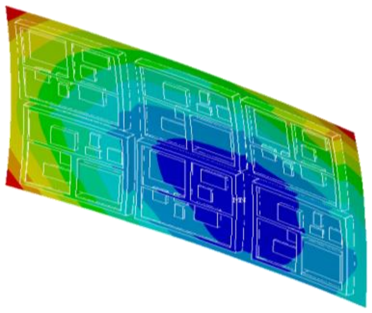


Figure 5. Warpage of a PCBA panel.

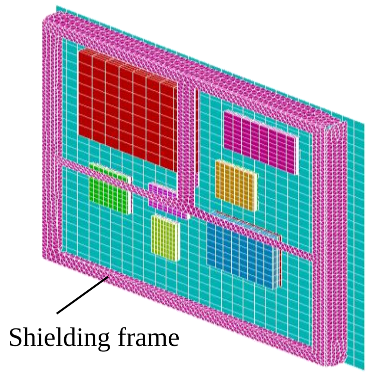


Figure 4. A typical PCBA with shielding frame.

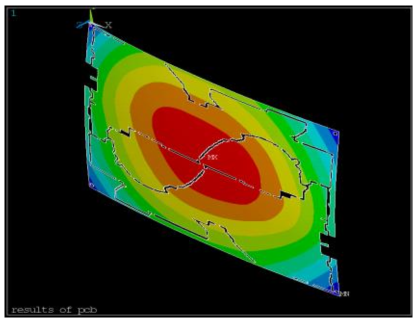


Figure 6. Warpage of an irregular-shaped PCB.

To verify the adopted approaches, a real test case was conducted, as shown in Figure 7. A typical PCBA case was conducted to go through a standard manufacturing process, i.e. a temperature change. Although the PCBA module (60 mm x 60 mm) includes many chips, for the simulation, only four main chips, i.e., one CPU (Central Processing Unit), two DRAMs (Dynamic Random-Access Memory), one PMIC (Power Management IC), are included. Other components or chips are very small and ignored. This PCB consists of, totally, 21 layers, including 10 trace layers. The equivalent material properties, e.g., Young's modulus, of each element on each trace layer are calculated according to the derived copper ratio. With all these data and model, the analysis was conducted. The comparison of the maximum warpage (out-of-plane displacement) between the simulation and experiment is shown in Table 1. It shows the difference is around 17%. For such a complicated case, this can validate the proposed approaches.

TABLE 1. MAXIMUM WARPAGE COMPARISON

	Simulation (μm)	**Experiment (μm)**
Maximum Warpage	169	140

In addition, a PCBA case with shielding frames was also conducted. The analyses have found some warpages seriously affected by shielding frames, as shown in Figures 4 and 5. Although the original purpose of the Electromagnetic Compatibility (EMC) shielding frame is to prevent Electromagnetic Interference (EMI) or Radio Frequency Interference (RFI), not for structural considerations; however, they cause some side effects. This finding also shows that the proposed approaches can be useful for the evaluation and design.

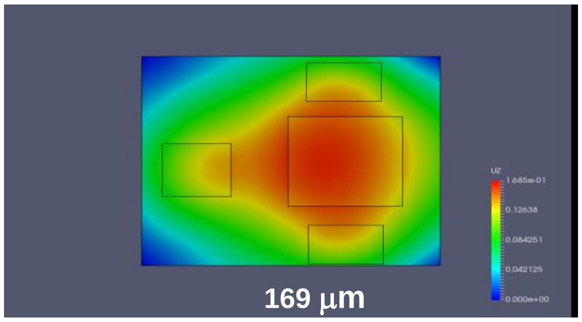


Figure 7. Warpage of a tested PCBA.

III. CONCLUSION

The structure of a PCBA is not only extremely complicated, but also multi-scaled, so simulating the PCBA has been a challenging problem. However, the proposed effective modeling approaches for the tiny circuits and solder joints can effectively solve these difficulties. It can distinctly reduce the size of the analysis model and efficiently handle the effects of the circuits and solder joints for the simulation. With the effective modeling approaches, evaluating the structural behaviors of PCBAs becomes simple and efficient. The test case shown above proves the effectiveness of the approaches. In addition, the fact that the warpage is affected seriously by the shielding frame has been found, although the original purpose of the shielding frame is not for structural considerations. This finding also demonstrates that the proposed approaches can be helpful. Nevertheless, because a PCBA consists of many chips, which may come from different vendors, the data of the internal structures and used materials of the chips may not be sufficient. In this situation, the accuracy of the results would be affected.

REFERENCES

[1] H. C. Cheng, K. N. Chiang, and M. H. Lee, "An Effective Approach for Three-Dimensional Finite Element Analysis of Ball Grid Array Typed Packages," Journal of Electronic Packaging, ASME, vol. 120, pp. 129-134, 1998.

[2] D. Zwemer et al., "PWB Warpage Analysis and Verification using an AP210 Standards-Based Engineering Framework and Shadow Moir," EuroSimE 2004, Brussels, Belgium, pp. 121-131, May 10-12, 2004.

[3] M. Bajaj et al., "Automating Thermo-Mechanical Warpage Estimation of PCBs/PCAs Using a Design-Analysis Integration Framework," Mentor U2U, San Jose, CA, USA, May 3-5, 2006. Available from: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=32287

[4] D. H. Kim, S. J. Joo, D. O. Kwak and H. S. Kim, "Warpage Simulation of a Multilayer Printed Circuit Board and Microelectronic Package Using the Anisotropic Viscoelastic Shell Modelling Technique That Considers the Initial Warpage," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 6, no. 11, pp.1667-1676, 2016.

[5] Calculix: https://www.dhondt.de/. [accessed Sept. 2024]

Copyright (c) IARIA, 2024. ISBN: 978-1-68558-184-8