2023 Design, Automation & Test in Europe Conference (DATE 2023)
Special Initiative "Autonomous Systems Design"

Information Processing Factory 2.0 – Self-awareness for Autonomous Collaborative Systems

Nora Sperling*, Alex Bendrick*, Dominik Stöhrmann*, Rolf Ernst*, Bryan Donyanavard†, Florian Maurer‡, Oliver Lenke‡, Anmol Surhonne‡, Andreas Herkersdorf‡, Walaa Amer§, Caio Batista de Melo§, Ping-Xiang Chen§, Quang Anh Hoang§, Rachid Karami§, Biswadip Maity§, Paul Nikolian§, Mariam Rakka§, Dongjoo Seo§, Saehanseul Yi§, Minjun Seo§, Nikil Dutt§, Fadi Kurdahi§

**Institute of Computer and Network Engineering, TU Braunschweig, Germany*
†Department of Computer Science, San Diego State University
‡School of Computation, Information & Technology, Technical University of Munich, Germany
§Center for Embedded and Cyber-physical Systems (CECS), University of California, Irvine

***Abstract*—This paper summarizes the talks of a special session on the IPF 2.0 project, a collaborative German-US research project that leverages self-awareness principles for the self-management of distributed systems of autonomous multiprocessor systems-on-chip (MPSoCs).**

***Index Terms*—autonomous systems, self-awareness, data centric**

I. Motivation and Background

Autonomous Systems Design must deal with several impairments that arise in designing and deploying complex autonomous systems. Chief among such impairments are: (1) the unexplainability conundrum of AI/ML leading to unbounded behavior, (2) the intractability of verifying the system under all possible use cases, leading to the inability to deal with unexpected situations (e.g., emergent behavior and unknown unknowns, or black swan events), (3) the inability to fully predict the behavior of humans entangled with such systems (e.g., self-driving cars interacting with human-driven ones), (4) the aging and other physical world interaction mechanisms that ultimately affect the system's operational parameters (e.g., energy, performance, reliability, safely and security) over time, (5) the inaccuracy of the models used at different levels of hierarchy to design the system, and (6) the need to reconcile conflicting operational parameters. The Information Processing Factory (IPF) project is a collaboration between research teams in the US (UC Irvine) and Germany (TU Munich and TU Braunschweig) applying self-awareness to the self-management of MPSoCs. This includes (a) self-reflection, i.e., awareness of the MPSoC's own hardware/software architecture, operational goals, and dynamic changes once deployed; (b) self-prediction of dynamic changes; and (c) self-adaptation to environment changes, optimizing operational parameters and protecting against unexpected situations with increased risk.

IPF 1.0 was first introduced in ESWEEK 2016 [1] as a paradigm to master complex dependable systems. The IPF paradigm applies principles inspired by factory management to the continuous operation and optimization of highly integrated embedded systems. A general objective is to identify a sweet spot between maximized autonomy among IPF constituent components and a minimum of centralized control to ensure guaranteed service even under strict safety and availability requirements. Emphasis is on self-diagnosis for early detection of degradation and imminent failures combined with unsupervised self-adaptation to meet performance and safety targets. IPF 1.0 produced high-grade results and spawned additional research projects [2].

As a follow-up project, IPF 2.0 expands the concept from a single self-organizing MpSoC towards distributed systems of autonomous MPSoCs with varying connectivity, in cyber-physical applications and the Internet of Things. It first covers locally interconnected systems, which are still controlled by a common plan, comparable to a larger factory, then takes a big step towards dynamically cooperating clusters with emergent behavior, comparable to the interplay of several factories in global supply chains. Vehicle platooning, with distributed collaborative planning, online verification and sensing, shall be used as common research scenario. IPF 2.0 has a system-of-systems structure in which several IPF 1.0 "factories" interact, providing an additional layer of abstraction with a data-centric approach.

The special session at DATE 2023 includes four talks that are elaborated in the following sections. The first section outlines the challenges when moving from self-organizing local systems in IPF 1.0 to autonomous systems collaboration in IPF 2.0. It takes commercial vehicle platooning as a use case, and explains how the self-aware truck control systems collaborate towards a platoon-level runtime verification that continuously supervises the state of a platoon, even under a changing platoon formation and external disturbance, e.g., by intersecting traffic participants. Future autonomous cyber-physical systems will be data centric: generating, processing, and storing enormous amounts of dynamic data. This trend will affect both local autonomy and autonomous systems collaboration. The second section outlines the resulting challenges and discusses how self-aware caching can help in mastering the resulting communication and data management requirements. A cornerstone in data management is memory technology. High data dynamics put a particular burden on memory power consumption. The third section will propose approaches to mitigate the energy cost of data management. The fourth section addresses a particular problem of

978-3-9819263-7-8/DATE23/© 2023 EDAA

autonomous systems, the lack of explainability, often resulting from the underlying machine learning technology. It evaluates the use of learning classifiers, a successful IPF 1.0 machine learning technology, in the context of autonomous systems collaboration.

II. Trust, But Verify: Towards Self-Aware, Safe, Autonomous Self-Driving Systems

In the IPF project 1.0 [2], self-awareness and self-healing are essential factors for the long-term normal operation of the system. Known system failures and predictions of the system's future state must be observable, and the IPF system has the following components for this purpose; Runtime Verification (RV) and Inference Engine (IE). The system utilizes both proactive and reactive methods for monitoring. Runtime Verification (RV) – especially when dealing with online monitoring – takes into account finite executions of increasing size. For this, the monitor must be designed to take into account the execution in an incremental way [3]. RV can respond nicely to predefined and predictable system states but may pose difficulty in detecting unknown behaviors. An envisioned way to detect imminent hazards is by means of an inference engine (IE) [4]. The engine is coupled with RV. An RNN with long short-term memory (LSTM), useful in predicting the time series data, is then employed to examine the obtained trace data. RNN LSTM-based weights are obtained using trace data at program execution and data consisting of normal/abnormal annotation (tag). The obtained weight is synthesized with the inference engine configuration and updated at runtime by the IE. Through this, IE can be used to predict hazards at runtime.

IPF 2.0 evolves existing systems a step further. First, there needs to be a complement to RV and IE. RV is the best detection method for defined properties, and IE helps to predict the known possible system state, but it is challenging to detect unknown abnormal system behavior. Anomaly detection (AD) techniques can find an anomaly or outlier that is significantly different from the remaining data, and thus flag anomalous execution in computer systems. For example, network intrusion detection, credit card fraud detection, sensor network error detection, medical diagnostics, and many other fields are well-known areas utilizing the AD technique [5]. AD can be particularly useful at runtime to detect an execution out of the normal range, and thus can play a significant role in detecting emergent behavior. However, since the important system state is not preserved and is only determined based on stateless input values, further optimization and backtracking of the system state are almost impossible by use of AD alone.

Figure 1 represents all of the above components and shows a flow of how IE, AD, and RV interact. Traces are created from system components such as the CPU, memory controller, bus, and bus peripherals and typically generate data equivalent to several gigabits per second. For IE and AD using machine learning techniques, feature extraction is required. The process reduces the flood of trace data and better represents the intrinsic state of the system. Conversely, in the case of RV, which needs all the exact status of the system, this trace is directly used and filtered internally. IE is a proactive mechanism that will

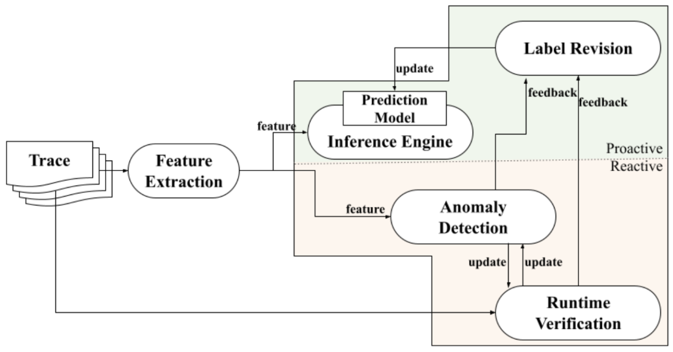


Fig. 1. Details of system traces as input, RV, IE, and AD components and how the components interact

inform the system before it fails. In the case of IE, we have a pre-trained model offline, which is updated by the detection results (feedback) of RV and AD. AD is used to determine unknown system conditions. AD is learned offline using the known normal state of the system and reports to the system whenever the system is not in the known normal state. If RV detects a failure and AD recognizes it as normal, an online update of AD is required. RV shows a 100% detection rate for defined properties. If AD detects an unknown abnormal operation, it is required to reflect the property. The system knows the proposition related to the current property, and when the AD detects an abnormal operation, it creates/updates a new property.

Given the new focus in IPF 2.0, we investigate a novel way to perform RV on (distributed) systems that can dynamically self-configure at runtime. When shifting the target platform to dynamic, distributed systems that process large volumes of data, RV becomes more challenging due to the need to support failover and resource sharing. Furthermore, many emerging systems can self-reconfigure and self-organize at runtime, increasing the difficulty in defining an expected global behavior. Besides, as systems complexity increases, so does the challenge of performing runtime verification at the architecture level. These issues point to a critical gap in the applicability of RV to distributed self-reconfigurable cyber-physical systems that need to be addressed. Truck platooning is an example of a distributed autonomous system that requires a modified RV methodology due to its runtime self-configuration. As each truck operates independently, it generates a separate trace, which makes it challenging to have a centralized monitor as traces may have an undefined global state when considering the entire platoon.

Figure 2 shows an initial framework for RV over a distributed data-centric truck platoon. Each truck in the system stores its local data for verification in a local datastore implemented on top of a Database Management System (DBMS). Each datastore consists of (i) a Distributed Runtime Monitor (DRM), (ii) an Event Trigger (ET), and (iii) a Temporal Query Translator (TQT). The DRM performs a given query specification and returns whether it was violated. The ET checks new data as it is inserted into the datastore for a defined condition, and when the condition is satisfied, it can trigger further actions. The TQT enables temporal queries using window-based queries on timestamped data and spatial queries by combining user-defined queries with system knowledge. These different datastores can work together to verify properties across the distributed system. When a property needs to be verified across the entire platoon, it is dynamically instantiated with the platoon's current state, as the trucks in the platoon can change over time. Then, this dynamic query is sent to all trucks that verify its result considering their local data, and the individual verification results are combined for a global result regarding the dynamic global query. By distributing the verification of safety properties, the RV can accommodate the dynamism of data-centric design flows such as a truck platoon.

III. Vehicle as a Cache – A Data Centric Platform for the IPF Paradigm

Data management in vehicles currently must respond to two major trends, dynamics and data volume. We first elaborate on dynamics. While in the past, vehicle software was developed as part of a rigorous V-model vehicle design process that ended in a static, possibly configurable software artifact, recent vehicles follow a more agile software design style with frequent updates and service oriented software architectures (e.g. [6]) all the way to software updates and configuration "over-the-air" after deployment. This is partly due to vehicle software customization and partly due to a plethora of new cloud and edge based services including advanced navigation, vehicle cooperation, teleoperation, or advanced passenger support and infotainment. New players, such as Amazon or Microsoft, provide the necessary infrastructure effectively using the vehicle as a wireless front-end IT platform that is connected via a future V2X network (cp. [7]). In consequence, future vehicles will be subject to software dynamics with very different profile and timing.

While vehicle software updates and customization involve critical functions, but can be constrained to times of low vehicle activity, the new applications change dynamically at run-time. The resulting dynamic load meets a vehicle platform with limited and varying computing and communication resources. In consequence, the current, static vehicle resource mapping must be extended by an in-field adaptive resource planning. It must operate under reliability, safety and timing constraints, must consider evolving environment, functions and emergence and should include predictive risk management, online verification,

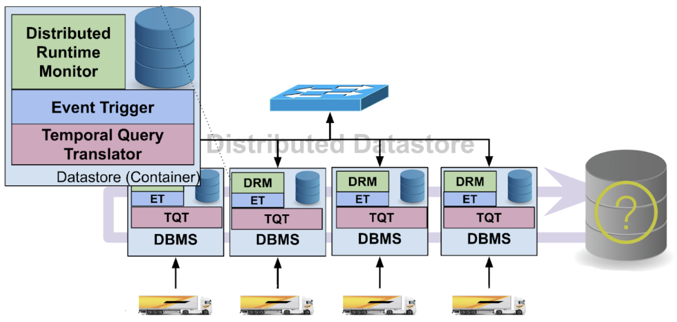


Fig. 2. An overview of IPF 2.0 runtime verification based on distributed database management systems (DBMS)

platform adaptation, and data management. This is a typical use case for the Information Processing Factory paradigm.

Of course, planning is only effective where resources are sufficient. However, the second trend is a dramatically growing data volume. Even though the CAN-based vehicle network of the past is currently replaced by a TSN Ethernet backbone with data rates at and above 1 Gbit/s, they face data rates and latency requirements of high resolution sensors (cameras, LIDAR, radar, ...) that grow even faster. Dynamic vehicle mode dependent network configuration and service integration can improve network utilization, in particular where V2X communication is involved [8], but communicating all data that are relevant to active services will typically exceed the available bandwidth. As an indicator, we take the in-vehicle memory size required for automated driving that is predicted to grow from 8GByte today to multiple TByte in 2030 [9]. A large share of those data is critical and subject to real-time requirements, such as sensor data (radar, camera, LIDAR), status information (perception, traffic, real-time maps) or related to vehicle coordination. Such data are subject to a high "volatility", meaning their relevance depends on data age requiring continuous updates, like in factory management or logistics. As an indicator for data volatility, we might take the DRAM requirements, which are predicted to exceed 50GBytes already in 2025, for driving automation alone [10].

There are solutions for time critical data management in distributed systems, with the DDS middleware as the most prominent example [11]. However, DDS handles the distribution of complete data objects, such as camera frames. The resulting data rates [9] will likely be prohibitive, on all levels of systems hierarchy, in ECU, in vehicle and V2X. We see two alternatives, reducing object size by compression and/or quality reduction, or smart object caching. The first alternative is beyond the IPF project, the second alternative, caching, is an interesting research topic. As an example, we take a traffic light recognition application from the popular Autoware platform [12]. The application searches a traffic light in a 2048x2048 camera frame, based on region information from a 3D map of the intersection. The region information, vehicle position and angle, are used to reduce the search to a small attraction box covering about 1.2% of the camera frame [13]. As the vehicle moves, the box will move in the sequence of camera frames, but updates are only needed for the box content, significantly reducing the required data rate requirements if object caching only transmits the dynamically selected box. This becomes especially relevant if camera images are processed in different

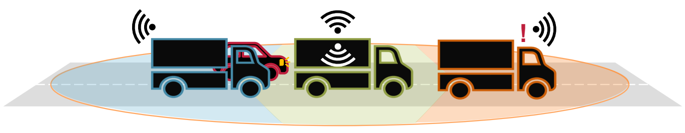


Fig. 3. Collaborative platooning use case: Following trucks share relevant sensor data with Platoon Leader

The collaborative platooning use case, as shown Fig. 3, combines both trends that were explained above. The leading truck makes its front camera/LIDAR data available to the following trucks and the following trucks can provide their side and rear camera and/or LIDAR views for sensor fusion providing a complete surround view and perception of other vehicles and obstacles, even where they are hidden from the leading truck. Because the relevant sensor data depends on the scene, the number of platoon vehicles and the position of objects in the environment, self-aware object caching seems to be an interesting option.

The missing link is reliable wireless communication. Using remote cameras still requires adhering to the same stringent data age requirements for sensor data processing as known from in-vehicle networks. Hence, low object latencies need to be ensured to mitigate errors in sensor fusion and object detection that could lead to accidents. For that purpose, we have developed an application level error correction protocol for short latencies of 100ms that was already used in a valet parking example where infrastructure cameras were used to enhance vehicle based perception [8], [14]. The protocol is combined with a safety layer that falls back to the vehicle sensors if the error rate is unacceptably high, an approach that can also be used in platooning. The reliable protocol is bidirectional and, hence, compatible to object caching. This shall be investigated as future work.

IV. Computational Self-Awareness for Energy-Efficient Memory Systems

The vehicular challenges for data dynamism and volume (as highlighted earlier) are exacerbated by the heterogeneity and timeliness of safety-critical data. Fig. 4 shows an exemplar computational pipeline for an Autonomous Vehicle (AV), where data in the perception-planning-control pipeline is: 1) generated from multiple external (e.g, cameras, LIDAR, radar) and internal sensors (e.g., CANbus polling) sensors; 2) transformed through a computational pipeline (e.g., perception, planning); and 3) output to inform control and actuation.

The nature of the AV computational pipeline makes the raw sensor data it operates on timing constrained and safety critical, and the intermediate data representations increase the volume and dynamism of the data used. For instance, data from cameras can be used in object detection to identify obstacles, and in structure from motion to reconstruct 3D environments. Each process generates different data structures, with variable criticality and timeliness. For a fast moving vehicle to operate

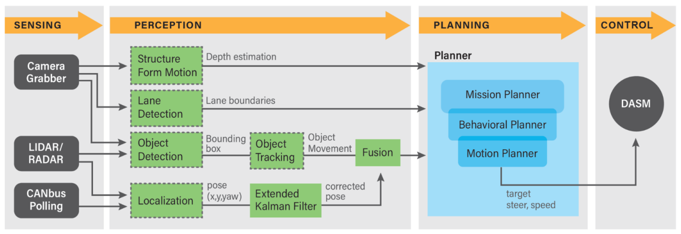


Fig. 4. Example perception-planning-control task pipeline for autonomous vehicles [15]

safely, obstacles must be detected quickly, and are relevant for short periods of time. For visual representation of anomalous event playback (e.g., an accident), 3D representations of 2D objects can improve quality of user experience, and may be stored for reuse. Critical control signals may be low in volume but high in timing criticality. This significant volume of dynamic data must be stored, managed, transformed, and transmitted across memories within single, locally-connected, and distributed computing platforms. The data storage and movement consumes a significant amount of energy, posing additional challenges for energy efficiency while meeting critical timing requirements across the computing stack and network.

We deploy Computational Self-Awareness (SA) principles to: a) build self-models of the system at multiple levels of the abstraction stack; b) enable introspective analysis for control loops to respond quickly, enhanced with reflective loops that consider both past as well as potential future outcomes for proactive planning; and c) perform cross-layer sense-making to fuse disparate and heterogenous data for holistic analysis and predictions. In the data-centric IPF 2.0 context, the implications on memory and energy efficiency can be studied at three levels:

1) Single Computational Platform. When applications execute on a single computational platform, data flows across multiple layers of the memory hierarchy from on-chip caches, to off-chip main memory, and to storage. The transformation, movement, and storage of this highly dynamic data remains a key performance and energy bottleneck. Cross layer optimization across the abstraction stack is essential to meet critical performance needs while ensuring energy efficiency. In [16] we demonstrated how self-awareness principles can be used to design an energy-efficient memory subsystem, and explored different degrees of self-awareness. We further exploited approximations at the architectural level to tradeoff memory accuracy for energy efficiency [17] . Using Computational SA, we coordinated multiple levels of memory hierarchy in a RISC-V system to achieve significant energy gains for data centric applications. At the operating system level, in [18] we demonstrated a self-aware proactive and adaptive swapping policy for individual computational platforms executing multi-application pipelines. Furthermore, we proposed a demand-layering scheme that minimizes the memory footprint of DNN inferencing model parameters through layer-by-layer loading and execution [19].

2) End-to-End Applications on Multiple Platforms. AVs typically use multiple computational platforms (ECUs) to execute end-to-end (sensors-to-actuators) compute pipelines. The data flows as a Directed Acyclic Graph (DAG) for each sensors-to-actuators path, and each path in the DAG is typically constrained by an end-to-end deadline. Tasks may execute on specialized processors such as GPUs and accelerators, as well as general-purpose multicore processors. System designers need a framework to perform rigorous analyses, explore alternative schedules, task and data mappings, and perform energy optimizations, all while adhering to criticality constraints. We developed the Chauffeur [15] open-source framework that empowers designers to explore the end-to-end tradeoffs between design constraints, power budgets, real- time performance requirements, and accuracy of self-driving workloads. For instance, we investigated exploiting the dynamic slack obtained by adaptively relaxing end-to-end deadlines based on different driving scenarios [20] (e.g., driving at a lower speed does not require the same actuation interval as when driving at high speeds). We demonstrated over 30% energy reduction for typical driving scenarios based on the Bosch WATERS Industrial Challenge [21].

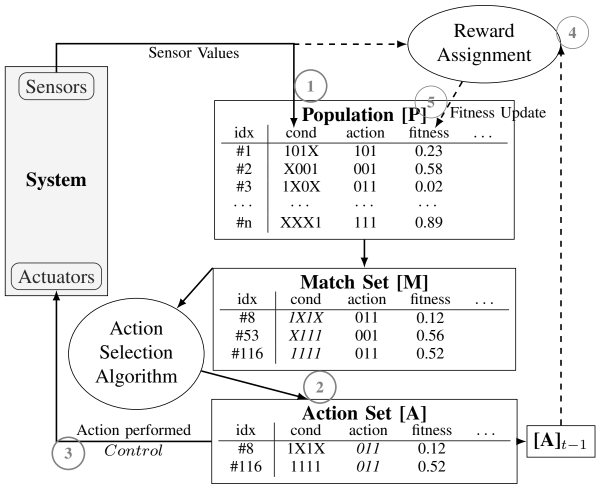
3) Dynamic Data Management for Distributed, Networked Platforms. As AV computational platform architectures evolve from stand-alone AVs (using CAN and/or TSN Ethernet for multiple on-board ECUs) to cloud/edge services and V2X scenarios, data must be intelligently orchestrated between multiple AVs across the end-edge-cloud networks. In [22] we orchestrated realtime inference requests across an end-edge-cloud network to reduce latency while maintaining accuracy. Managing computation alone is no longer sufficient: our studies show that in realtime applications, processing and data both impact performance and efficiency [23], and therefore must be managed together. In IPF 2.0, the vehicles as cache paradigm brings the data in the AV pipeline to the forefront for a vehicle. Distributed runtime verification generates even more safety-critical data that must be managed in addition to sensor data for a network of vehicles. When performing data management in safety-critical systems, it is important that information and decisions are fast, efficient, and interpretable (see Section V). We must revisit memory and storage policies for efficient streaming-based data management both across the abstraction stack and network.

V. LCT - Turning ML Decision Making Explainable

Prior sections on safe and verifiable self-aware systems explain the importance of predictable system states. As unknown abnormal behavior is challenging to detect, system designers should try to minimize unforeseeable output results of system components. One known source of unpredictable system responses is ML decision making.

Deep learning disrupted the domains of probabilistic inference and statistical optimization for applications such as object and speech recognition, image classification or, generally speaking, the modeling of complex, non-linear and high-dimensional relationships [24]. However, it is also well known that such deep neural network-based (DNN) machine learning (ML) engines are "black boxes" which do not allow making correlations between individual neuron weights and selected output labels. In contrast, rule-based classifier systems within the MAPE control loop of embedded systems [25], e.g. as shown in Fig. 5, do make the Match Set determination deterministic for any sensor inputs and thus, provision an explicit containment for the finally applied action/label. Nonetheless, applying a fitness-weighted randomization function on the Match set affects the particular rule selection (incl. action) which results in the Action set. Within the Action Set all rules have identical action.

Within the IPF project, we developed a hardware learning classifier table (LCT) agent for DVFS (dynamic voltage



Population [P]

idx	cond	action	fitness	...
#1	101X	101	0.23	
#2	X001	001	0.58	
#3	1X0X	011	0.02	
...	...	...	...	
#n	XXX1	111	0.89	

Match Set [M]

idx	cond	action	fitness	...
#8	*1X1X*	011	0.12	
#53	*X111*	001	0.56	
#116	*1111*	011	0.52	

Action Set [A]

idx	cond	action	fitness	...
#8	1X1X	*011*	0.12	
#116	1111	*011*	0.52	

Fig. 5. LCT data structures and information flow

frequency scaling) control of a RISC core (System) that is provisioned with a variable processing performance target (IPS_{target}/[instr. per second]) and a hard power constraint (P_{constr}/[W]) [26]. Current operation values of the core are sensed and assessed against these IPS_{target}/P_{constr} specs by a Reward Assignment function. All rules in the Action Set are rewarded with a fitness increase in case an applied action (that varies the core frequency f by a specific step size) lowered the gap between current IPS and IPS_{target} while the power P stays within the P_{constr} limit. Rules that lead to an increased gap towards IPS_{target} or even violate the P_{constr} are penalized with a fitness value decrease. We refer to [27] for more in-depth description on the fitness update.

At any instance in time, the composition and content of the address-mapped Population table (containing all rules and corresponding fitness values) can be accessed/sampled via a CPU control interface. Alternatively, changes or updates in the table can be streamed as a trace for either on-the-fly analysis or post-experiment processing in case of capturing the trace in CPU memory. More importantly, the LCT approach to machine learning allows:

- Following the evolution of fitness values assigned to rules (and their specific actions/labels) and thus, reasoning whether condition-action pairs make sense.
- Observing explicitly how new rules inserted into the Population table at runtime evolve in terms of fitness and what impact the removal of rules has on the system.
- Studying the influence of different objective and reward assignment functions as well as differences between immediate reward assignment policies for individual actions (single-step learning) vs. for a bounded series of actions (multi-step learning).
- Analyzing, correlating and explaining the causal dependencies in the behavior of multiple LCT agents in multi-core CPU systems.
- Manually adding and editing the conditions and actions of rules based on best practice expert knowledge or design-time as well as run-time explorations.
- Time series traces on applied rules document how their

fitness varies, how often individual rules were selected and in general, allows reasoning about what control behavior performs well or not so well under given environmental conditions and workloads.

The following use case scenario discusses some of the above properties in situations we observed during simulations and measurements on FPGA prototype systems. More examples will be presented during the talk of the special session. In the DVFS use case, we occasionally observed certain rules with generally high fitness but transient transitions towards lower fitness values. In hindsight, this phenomenon could be explained after doing in-depth time series analysis on fitness and CPU parameter traces. These investigations revealed that the action of a specific rule, occasionally brought the CPU operation point (OP) – given by its frequency / voltage pair – into a pre-defined "margin zone", close to P_{constr} (see Fig. 6). Hence, a follow-up action had to "recover" the OP into a known

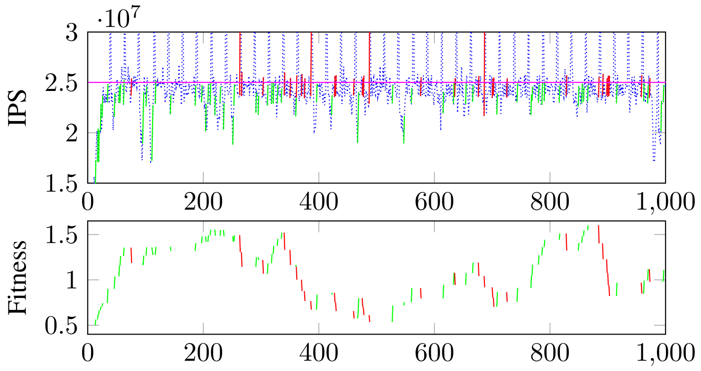


Fig. 6. Action implication on optimization goal IPS w.r.t. $P < P_{constr}$: green (improvement), red (degradation), blue (different rule action)

safe state (i.e., the OP before the critical action was applied) and all rules in the previous action set received a negative reward [26]. It turned out that if applying this rule only when there is sufficient power distance to the margin zone (which implied an adjustment of the condition section in the specific rule), the issue got resolved. Such an analysis as well as the manual intervention (which alternatively or complementary can also be augmented with an automated heuristic for new rule generation) isn't possible if the ML inference is based on weights on few or numerous neural layers.

VI. Conclusions

The IPF 2.0 project will continue to exploit computational self-awareness principles to address the challenges of steering, managing, and storing the large volumes of heterogeneous data within and across networked cyber-physical systems and the emerging edge/cloud infrastructure. The project will ensure safety with a view to dynamically optimize operational goals in a context-sensitive manner, using interpretable and explainable data-driven methods.

VII. Acknowledgments

We acknowledge the financial support from the DFG Grant ER168/32-2, HE4584/7-2 as well as NSF Grants CCF-1704859 and ECCS-2028782.

References

[1] N. Dutt, F. J. Kurdahi, R. Ernst, and A. Herkersdorf, "Conquering mpsoc complexity with principles of a self-aware information processing factory," in *IEEE/ACM/IFIP CODES*, 2016.
[2] E. Rambo *et al.*, "The self-aware information processing factory paradigm for mixed-critical multiprocessing," *IEEE Transactions on Emerging Topics in Computing*, 2020.
[3] M. Seo and F. Kurdahi, "Efficient tracing methodology using automata processor," *ACM TECS*, 2019.
[4] Zhang *et al.*, "Predicting failures in embedded systems using long short-term inference," *IEEE Embedded Systems Letters*, vol. 13, 2020.
[5] C. C. Aggarwal, "Outlier analysis," in *Data mining*, Springer, 2015.
[6] C. Ebert and J. Favaro, "Automotive software," *IEEE Software*, vol. 34, pp. 33–39, may 2017.
[7] Microsoft, "Empowering automotive innovation," Jan. 2017.
[8] R. Ernst *et al.*, "Application-centric network management – addressing safety and real-time in v2x applications," *ACM TECS*, 2022.
[9] Semiconductor Research Corportation (SRC), "The decadal plan for semiconductors," 2021.
[10] M. Huonker (Mercedes), "Infotainment and autonomous vehicles – the challenges of storage," in *Flash Memory Summit 2019*, 2019.
[11] Object Management Group (OMG), "Data distribution service, version 1.4." OMG Document Number formal/2015-04-10, March 2015.
[12] The Autoware Foundation, "Traffic light map based detector," 2022.
[13] Y. a. Lu, "Traffic signal detection and classification in street views using an attention model," *Computational Visual Media*, vol. 4, no. 3, 2018.
[14] J. Peeck *et al.*, "A middleware protocol for time-critical wireless communication of large data samples," in *2021 IEEE (RTSS)*, IEEE, 2021.
[15] B. Maity *et al.*, "Chauffeur: Benchmark suite for design and end-to-end analysis of self-driving vehicles on embedded systems," *ACM TECS*, vol. 20, no. 5s, pp. 1–22, 2021.
[16] B. Maity, B. Donyanavard, and N. Dutt, "Self-aware memory management for emerging energy-efficient architectures," in *2020 IEEE IGSC*, IEEE, 2020.
[17] B. Maity, B. Donyanavard, *et al.*, "Seams: Self-optimizing runtime manager for approximate memory hierarchies," *ACM TECS*, vol. 20, no. 5, 2021.
[18] D. Seo, B. Maity, *et al.*, "Proswap: Period-aware proactive swapping to maximize embedded application performance," in *2022 IEEE NAS*, IEEE, 2022.
[19] M. Ji *et al.*, "Demand layering for real-time dnn inference with minimized memory usage," *arXiv preprint arXiv:2210.04024*, 2022.
[20] S. Yi, T.-W. Kim, *et al.*, "Energy-efficient adaptive system reconfiguration for dynamic deadlines in autonomous driving," in *2021 IEEE ISORC*, IEEE, 2021.
[21] A. Hamann, D. Dasari, *et al.*, "Waters industrial challenge 2017," in *WATERS 2017*, 2017.
[22] S. Shahhosseini, D. Seo, *et al.*, "Online learning for orchestration of inference in multi-user end-edge-cloud networks," *ACM TECS*, 2022.
[23] S. Hernández, J. Araujo, *et al.*, "Cross-layer configuration optimization for localization on resource-constrained devices," in *2021 IEEE/RSJ IROS*, IEEE, 2021.
[24] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
[25] E. Rambo *et al.*, "The information processing factory: A paradigm for life cycle management of dependable systems," in *CODES+ ISSS 2019*, IEEE, 2019.
[26] F. Maurer, B. Donyanavard, *et al.*, "Emergent control of mpsoc operation by a hierarchical supervisor/reinforcement learning approach," in *DATE 2020*, IEEE, 2020.
[27] B. Donyanavard, T. Mück, *et al.*, "Sosa: Self-optimizing learning with self-adaptive control for hierarchical system-on-chip management," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019.