Chair of Cyber-Physical Systems in Production Engineering: Annual Report 2021

March 15, 2022

1 Introduction

"Designing safe, predictable, and high performance embedded platforms for next generation Cyber-Physical Systems."

Modern Cyber-Physical Systems (CPS) are the next generation of engineered systems in which computing, communication, and control technologies are tightly integrated. Applications include system automation, Internet of Things (IoT), smart buildings, smart manufacturing, smart cities, digital agriculture, robotics, and autonomous vehicles. The chair of Cyber-Physical Systems in Production Engineering was founded in September 2018.

In 2021, research activities of the Chair focused on following topics: a) develop new reinforcement learning strategies for CPS, path planning, and control, b) design and implement novel resource management policies for embedded real-time systems running on high-performance heterogeneous platforms, and c) design architectures for sandboxing controllers in CPS. Other research activities are also focusing on the secure and safe integration of machine learning algorithms with digital controllers for CPS.

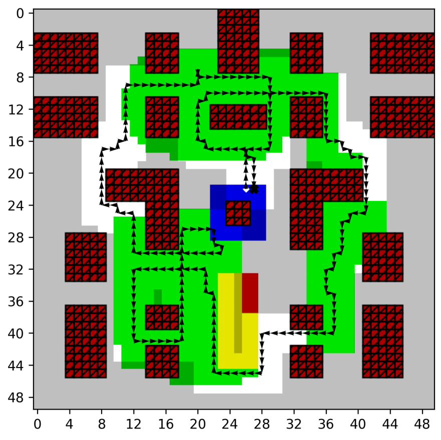
Members of the chair were involved in the peer review process of several international conferences/journals in real-time embedded systems and CPS, including RTSS 2021, RTAS 2021, RTAS 2022, DAC 2022, ECRTS 2021, IROS 2021, ICRA 2022, WCNC 2022, CDC 2021, RTCSA 2021, EMSOFT 2021, SAC 2022, ACC 2022, HSCC 2022, ICCPS 2022, ECC2022, DETECT 2021, EuroS&P, as well as ACM Transactions on Cyber-Physical Systems, IEEE Transactions on Automatic Control, IEEE Embedded Systems Letters, IEEE Transactions on Communications, IEEE Access, IEEE Transactions on Vehicular Technology, Journal of Measurement, Journal of Sensors, Journal of Nonlinear Analysis: Hybrid Systems, Leibniz Transactions on Embedded Systems, Journal of Innovations in Systems and Software Engineering.

2 Reinforcement Learning for Cyber-Physical Systems

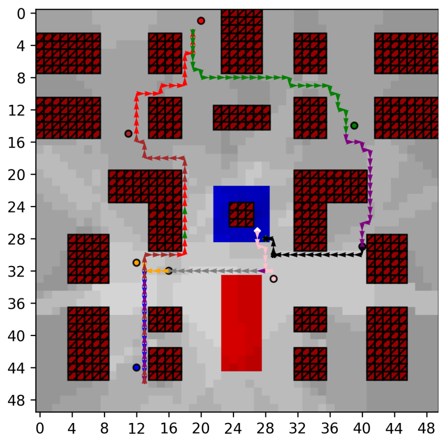
Reinforcement learning (RL) for planning and control of Cyber-Physical Systems is receiving increasing interest for its promise to solve complex optimization problems through interactions with the environment alone. To do so, the RL agent explores possible actions within an environment, receiving a reward. The goal of the agent is to maximize the cumulative reward it earns over time.

Our research in this direction can be split into two branches: a) Neural network architectures and state representations for UAV path planning missions and b) a distributed learning architecture for online learning capabilities in computationally constrained environments. Together the two branches lay out the methodology and infrastructure for reinforcement learning in various Cyber-Physical Systems.

2.1 UAV Path Planning



(a) Coverage path planning with green indicating the coverage target.



(b) Data harvesting with the devices to collect data from indicated with colored circles.

Figure 1: Example trajectories of reinforcement learning agents solving two distinct path planning problems.

If an RL agent is supposed to be applicable as a path planner for autonomous vehicles, such as unmanned aerial vehicles (UAVs), it has to be able to generalize over multiple scenarios. To generalize, the agents either have to find one robust solution that solves all possible problems or they need to adapt to differences in scenario parameters. In path planning, scenarios typically vary significantly such that there exists no one solution for all of them. Consequently, the agents have to adapt their policy to address scenario changes. This is only possible if the agents can observe or infer the varying parameters.

Our research in this field progressed into two directions: Scalability and multi-agent capabilities. The former was addressed in [41] for a single UAV to solve coverage path planning and data harvesting problems with a unified approach. In [7]the multi-agent case was added by using the map to share positional data among UAVs to allow for implicit coordination. The multi-agent approach further takes advantage of the scalability benefits from [41] for the single UAV.

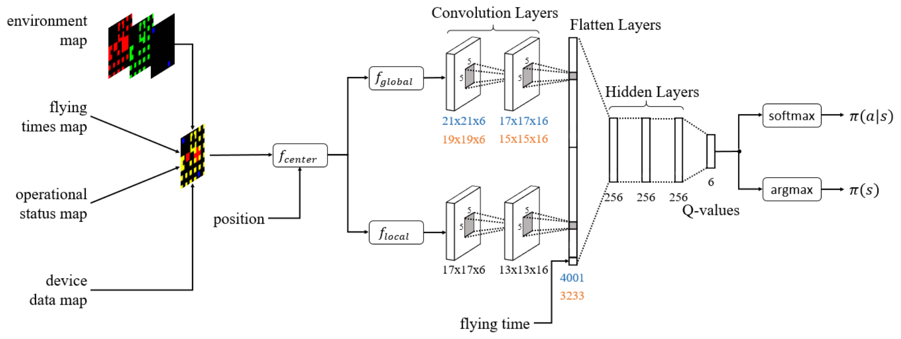


Figure 2: Neural network architecture for a deep Q-Network that estimates the state-action values of an agent in a multi-agent path planning scenario.

The map-based approach was shown in the previous year to be an enabler of generalization over scenario parameters. This was achieved by centering the map around the agent's position. This centering, while greatly simplifying the problem for the agent, significantly increased the observation space. In [41], we focussed on reducing the observation space again, without reducing generalization capabilities. The novel approach is to present the centered map as two inputs: a full-detail local map showing the agent's immediate surroundings and a compressed global map showing the entire environment with less detail. These two smaller maps contain sufficient information to allow for generalization, but their total sizes are significantly smaller than the previous observation space.

This approach allowed the agent to solve the coverage path planning (CPP) problem (Fig 1a) on significantly larger maps, learning in less time than before. Compared to previous work, the addition of map-centering to CPP allowed the agent to generalize over different target areas. In the data harvesting (DH) problem (Fig. 1b) the agent was also capable of generalizing over different sensor positions on larger maps.

The map-based path planning approach also allowed the easy integration into multi-agent scenarios where agents need to cooperate to solve a problem without communicating intention or explicitly coordinating actions. We addressed this problem in [7], by simply indicating the other agents' positions in separate map-layers as shown in Figure 2.

The diagram shows the full neural network architecture of the agents learning a multi-agent data harvesting scenario. The addition of the flying times map, indicating the remaining energy budget of the other UAVs and the operational status map, showing whether the other UAVs are still active or landed (or crashed), was sufficient to enable generalization for the multi-agent scenario.

The agents showed to have learned implicit collaboration by spatially separating the problem based on agent position only. A great example can be seen in Figure 3, where three agents were spawned on the blue area indicated with the white diamonds. Based solely on their initial position the agents spatially split the map into three sections, which simplified the data collection problem for each agent.

Orthogonally to the reinforcement learning based algorithmic improvements we further invested efforts into a long-endurance solar UAV, the UIUC-TUM Solar Flyer [14] [15] [13]. The ultimate goal is to integrate the map-based path planning into the solar UAV testbed, to conduct coverage path planning missions. Additionally, as a preparation for multi-agent use cases, we developed a multi-level clustering algorithm [29] that can potentially support communication and coordination among multiple UAVs.

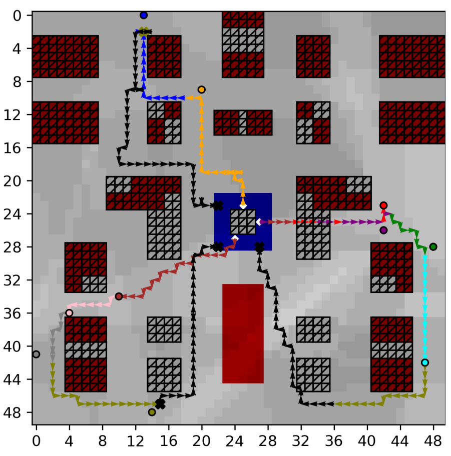


Figure 3: Example trajectory of three agents collecting data from IoT devices by spatially dividing the problem.

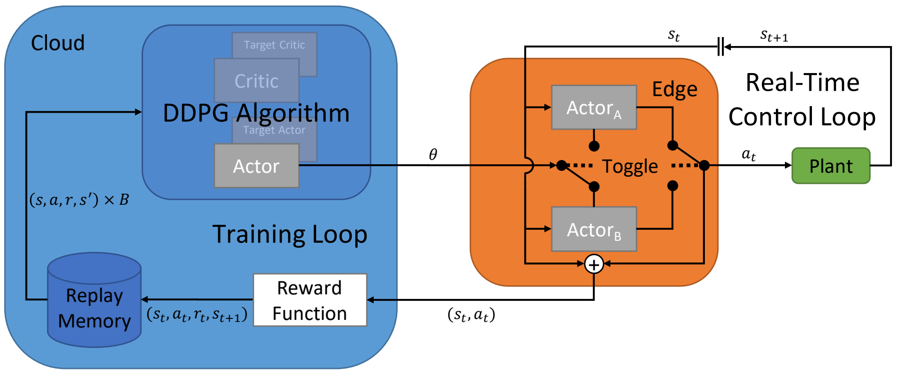


Figure 4: Cloud-edge distributed training architecture.

2.2 Could-Edge Distributed Training Architecture

Deep reinforcement learning (DRL) enables robots to master complicated tasks with impressive performances in the robotic control domain, e.g., locomotion, grasping, and manipulation. However, due to the high demand for training data for DRL, direct training on physical systems presents many challenges. Collecting training data in the real world is expensive concerning time and labor. Human supervision is usually needed to reset the system and monitor its hardware maintenance and safety status.

To address these real-world training challenges, some work improved DRL algorithms to reduce the learning sensitivities to hyperparameter settings, making training on physical systems more stable. Approaches of off-policy training with replay memory and model-based reinforcement learning are proposed to increase real-world sampling efficiency. Moreover, training using demonstrations and scripted policies can further ease real-world exploration.

In contrast, modern simulations can simulate complex systems and various environments. Recent sim-to-real approaches pretrain agents in simulations and then directly deploy the learned policies for real-world applications without further training. Simulation-based training boosts sampling efficiency significantly since simulations run faster than physical systems and can be further improved via parallel training. Additionally, the system can reset automatically without human intervention. Simulation training also does not require hardware maintenance and safety measures, drastically reducing the need for human supervision. Unfortunately, direct deployment of simulation-trained agents often fails in real-world applications due to the dynamic divergence between the real world and the simulated environment, the so-called *reality gap*. The *reality gap* arises mainly from system under-modeling, where the complex dynamics, e.g., the contact and friction effects, are difficult to measure and model. Moreover, computation and communication delays and environmental noise introduce extra modeling errors.

To bridge the *reality gap*, system identification approaches aim to create more accurate simulations, mitigating modeling errors. Techniques like domain randomization and domain adaptation can improve the agents' robustness in simulations, resulting in a successful direct real-world deployment. Nevertheless, in many scenarios, the domain randomization strategies might not be feasible since there might not exist a single policy that can solve all problems within the domain.

Pretraining in simulation to learn a sub-optimal policy and then continuing the training in the real world can take advantage of the efficient simulation training and the real dynamics. With this approach, all the advantages of simulation learning can be adopted, and the *reality gap* can be closed while training in the real world. We follow the continuous sim-to-real training paradigm in this work, as we believe it is the most promising approach for real-world DRL.

However, the continuous sim-to-real training paradigm suffers from two main problems. First, DRL requires high-performance computation and benefits from dedicated devices such as GPUs or TPUs for its training loop. The plant cannot have a high-performance device onboard in many real-time control systems, e.g., unmanned aerial vehicles or other mobile robots, due to power, weight, and space constraints. It cannot be controlled directly from a remote high-performance device, as perfect, loss-less, and low-delay communication cannot be guaranteed. The second problem is that the transfer of a pretrained agent to the real physical system is non-trivial. The dynamics can change abruptly, making the value estimates of the DRL agent inaccurate, leading to deteriorating performance.

In [10], we addressed the real-world training problem by introducing a novel distributed cloud-edge architecture, as shown in Figure 4. The real-time control loop on the edge is decoupled from the computationally intensive training loop on the cloud. The agent on the edge collects experiences in real-time, sending them to the trainer on the cloud, which periodically updates the edge with the optimized parameters. The agent is double-buffered such that the real-time inference loop is not interrupted by the policy updates. We addressed the sim-to-real transfer problem by delaying the neural network training at the beginning of the real-world interactions. Specifically, we start the policy optimization later than the value estimate training to avoid policy deterioration based on unstable value estimates.

We evaluate our approach on a physical inverted-pendulum system controlled by a DRL agent deployed on a Raspberry Pi 4B. The training loop is offloaded to a high-performance workstation. The agents are pretrained in intentionally under-modeled simulations to induce different levels of the *reality gap* to analyze the sim-to-real transfer. The real-world experiments show that our architecture can adapt the pretrained DRL agents to unseen dynamics consistently and efficiently. We further analyze the impact of the neural network optimization delays, highlighting their necessity. Additionally, we investigate the relevance of the edge update frequency on training performance to show that the architecture can work in constrained bandwidth settings, albeit with an impact on training time.

3 Predictable and high-performance resource management of CPS on heterogeneous platforms

The widespread use of artificial-intelligence (AI) algorithms in many Cyber-Physical Systems (CPS) such as autonomous cars, drones, and smart robots has driven the integration of specialized hardware accelerators (e.g., GPUs, FPGAs) on high-performance multiprocessor boards. Towards ensuring safety and real-time requirements, these heterogeneous multiprocessor systems-on-chips (MPSoC) pose unprecedented challenges. In fact, the implementation of complex CPS using these platforms generates increasing volumes of real-time (e.g., imaging) data flows causing the hardware memory hierarchy (the DRAM, the interconnect, and the cache hierarchy, especially the last level cache shared among multiple cores) to become a bottleneck and a source of temporal unpredictability. This phenomenon is further aggravated by the presence of accelerators (GPUs/FPGAs) that can independently access memory with high-bandwidth requests. Traditional techniques to allocate and optimize the execution of real-time tasks on safety critical CPS do not consider the heterogeneity of the computing elements and the complexity of MPSoCs' memory hierarchy. In addition, classical task models widely adopted in the real-time scheduling domain fail to capture the parallelization and heterogeneous computing needs of the new workloads.

At the integration level, developers and integrators are daunted by the task of finding the right trade-offs between selecting the appropriate scheduling policies, assigning real-time tasks to (heterogeneous) cores, selecting the size of cache partitions, and determining adequate bandwidth to allocate to each communicating resource. Our work tackles these challenges with techniques that could be rapidly adopted by the industry, and that aim to practically simplify the deployment of real-time workload on MPSoC without sacrificing neither predictability nor performance. On the platform side, we research, develop, and evaluate techniques to restore isolation and temporal predictability of safety critical software. We specifically target solutions (e.g., [45, 44, 23]) that prove to perform well "in practice", and we focus our integration effort at both Operating System (OS) and Hypervisor levels. Hypervisors (e.g., [1, 30, 37, 18, 17]) have become the de-facto industry standard to ensure isolation in certified partitioned safety-critical systems, but do not provide satisfactory isolation and predictability properties when contention at cache, interconnect, and DRAM level is considered. To facilitate the evaluation and the adoption of these techniques by the industry, in addition to publications (e.g., [36, 22]), we actively participate in the development of open source hypervisors (e.g., [1]) to make the developed techniques not only readily available, but also supported by an active community [6]. On these topics, we also actively collaborate with highly skilled international teams [28, 42], which pursue objectives close to ours.

On the application and deployment side, our research considers optimization and scheduling techniques to hide the complexity of the configuration space from the integrators, while enforcing isolation (enacted via the above-mentioned low level solutions) and ensuring the real-time properties of the workload. Problems under analysis are: the optimization of real-time task allocation under simultaneous consideration of cache-size, core, and bandwidth constraints; scheduling of real-time resources with different quality-of-service guarantees to applications with multiple criticality levels (e.g., [32]); support the effective debugging of timing properties for cyber-physical systems to restore system schedulability (e.g., [33]); automated synthesis of distributed embedded controllers and its integration at tooling level (e.g., [34]).

In the remainder of this section, we present more details on our recent works [36, 33, 22]. These papers are representative of our research efforts in 2021 towards predictable and high-performance resource management of CPS on heterogeneous platforms. We further discuss our work-in-progress software development efforts towards predictable memory management for operating systems.

3.1 Predictable and high-performance inter-VM communication on MPSoC [36]

Simultaneously ensuring real-time properties and high-performance on current multiprocessor systems on a chip (MPSoC) is a challenging task. In order to sustain the heavy data requirements of artificial intelligence (AI) algorithms, heterogeneous MPSoCs feature not only multicore architectures, but also integrate special purpose accelerators such as FPGAs and GPUs.

Previous research [23, 27, 44, 45] has focused on isolating the access to the memory hierarchy for independent virtual machines (VM), thus restoring the possibility of a practical real-time analysis. By design though, the strength of these approaches (i.e., ensuring isolation) is also their major drawback when it comes to establishing communication channels —which, by definition, violate isolation -–between VMs.

Our approach proposes an architecture [36] that enables predictable communication for VMs on MPSoCs under explicit consideration of the memory interference caused by VM communication. Furthermore, to ensure high data-transfer rates, we make use of Direct Memory Access (DMA) engines and we analyze the memory interplay between such engines and cores when they concurrently access the interconnect.

Our communication framework transparently supports unmodified VMs by integrating into hypervisor solutions (the industry standard to partition complex software systems). Furthermore, the architecture ensures maximum portability by implementing the API between the framework and unmodified VMs using the virtio standard.

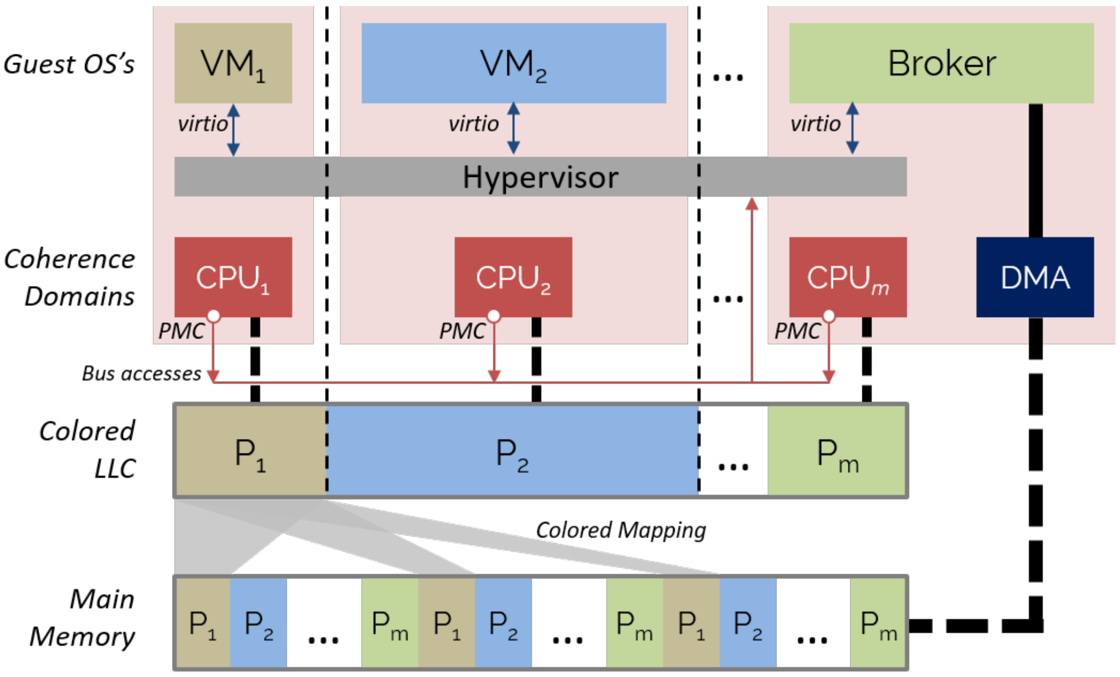


Figure 5: Proposed broker-based architecture for inter-VM communication on MPSoC.

The proposed solution (depicted in Figure 5) adopts a broker-based approach where a privileged VM (broker) mediates and schedules the communication flows between VMs. This enables

a) asynchronous communication between VMs, b) straightforward DMA integration, c) centralized scheduling and real-time analysis of the communication flows. In particular, the analysis of the communication flows enables designers to bound the maximum bandwidth to assign to cores and to the DMA, in order to meet the communication deadline constraints (latency) while avoiding over-utilization of the DRAM memory controller (thus causing unpredictable latencies). The combined evaluation of DMA and core regulation mechanisms allows designers to enforce appropriate bandwidth levels to meet the requirements of the analysis.

Our first prototype of the architecture [36, 35] makes use of basic building blocks (cache-coloring [27], MemGuard [45], and hardware-enforced QoS [38]) that have proved to perform well to guarantee isolation and that have been previously integrated on the Jailhouse Open Source hypervisor branch maintained at the Chair in collaboration with the Boston (US) and UniMoRe (IT) universities. In order to keep the runtime overheads to a minimum, we have implemented the virtio support layer into the lightweight FreeRTOS operating system. The viability of the approach has been then assessed by measuring the overheads and bandwidth of the prototype, and by comparing the experimental real-time latency against the one determined by our overhead-aware schedulability analysis.

The prototype is only a first step towards achieving predictable and high-performance communication on MPSoCs. As future research, we plan to integrate our solution into mainstream operating systems such as Linux (to enable even wider adoption), and to study the interplay between different factors (e.g., DMA transfer sizes, DMA/core communication setup latency) that could impact the overheads and the analysis of the framework. Furthermore, we plan to extend the approach to support not only inter-VM communication, but also communication with (virtualized) hardware devices.

3.2 Identifying Unexpected Inter-core Interference Induced by Shared Cache [22]

In modern high-performance multiprocessor system-on-a-chips (MPSoCs), caches have become a crucial piece of hardware bridging the gap between the speed of the processing elements and the main memory. The demand for high-performance systems has prompted the emergence of non-blocking caches, a type of cache capable of handling several concurrent accesses to main memory while hiding the cache-miss penalty.

Unfortunately, by nature, high-performance non-blocking shared caches decision taking is opaque and unpredictable. As proven by the amount of research conducted on cache management for real-time applications, understanding the cache behavior is of the utmost importance for safety-critical hard real-time systems where timing constraints must be respected and guaranteed. The two main sources of unpredictability imputed to the last-level cache (LLC) are (1) the inter-core cache line eviction and (2) the opaque management of shared internal resources.

Unpredictability introduced by inter-core cache line eviction is well-studied and can be successfully prevented by enforcing the spatial isolation of the processing elements through way-based or set-based partitioning [19, 27]. On the other hand, inter-core interferences caused by shared internal resources such as the Miss Status Holding Registers or the write-back unit are rare. Recent studies have highlighted the huge impact on the system's predictability if these shared internal resources are saturated. [9] and [43] show that victim tasks can exhibit a 346-fold increase in execution-time.

In [22], we show the existence of another source of inter-core interference which, contrary to the aforementioned sources, is not due to the saturation of the memory sub-system. Instead, we experimentally highlight how the response time of a cacheable target memory impacts all the processing elements connected to the shared non-blocking cache. Specifically, if a target

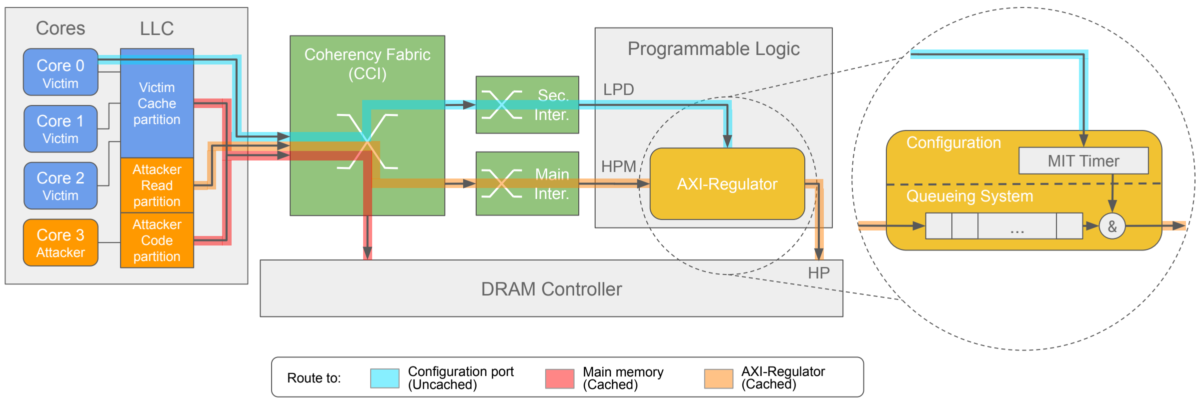


Figure 6: Schematic overview of the victim-attacker system model and architecture deployed on a PS-PL platform.

memory acknowledges a single transaction coming from one processing element but waits to deliver the response, the execution time of tasks running on independent processing elements is also impacted.

In [22], we draw the details of a victim-attacking system model emphasizing the above mentioned source of interference and discuss how we created it using state-of-the-art isolation techniques on a widely available out-of-the-shelf platform. The outcome, shown in Figure 6, makes use of Jailhouse [1] (Time Isolation), Cache-Coloring on Jailhouse [6] (Cache spatial isolation), and PLIM [31, 21] (Data path to main memory isolation).

Experiments performed on the ARM Cortex-A53 core cluster (Xilinx' UltraScale+ ZCU102 development board) show that the execution time of a victim task can be increased by a factor of 10. Furthermore, we show that if the target memory acknowledges a single read transaction but never provides a response, the whole core cluster is frozen indefinitely. Even though evidence gathered hints at a defect in the non-blocking shared cache controller, this early research item does not provide a definitive answer regarding the source of the observed interference. In [22], we set the premises for a wider and deeper investigation.

3.3 Timing Debugging for Cyber-Physical Systems [33]

An important pillar in the design of time-critical systems, viz., "what to change when timing properties are not satisfied?" is surprisingly very weak in terms of the relevant techniques and literature that are available. We aim to address this important question of "what should be changed?" and to propose a new class of timing debugging techniques. Our approach relies on the observation that many time-critical software tasks implement feedback control loops, and therefore the changes to satisfy schedulability constraints could be evaluated from a system-level perspective, viz., how do those changes impact the dynamics of the feedback control loop? Given this integrated approach of ensuring the timing properties of software tasks by accounting for the (physical) dynamics of the system being controlled, one can view this as a Cyber-Physical Systems (CPS) approach to timing debugging.

The problem we study in this work is the most basic timing debugging problem: given a set of periodic tasks with implicit deadlines, i.e., deadlines equal to periods, which periods should be changed in order to make the task set schedulable? See Figure 7. Since the earliest deadline first (EDF) scheduling policy is known to be optimal in this setting, we consider this for scheduling

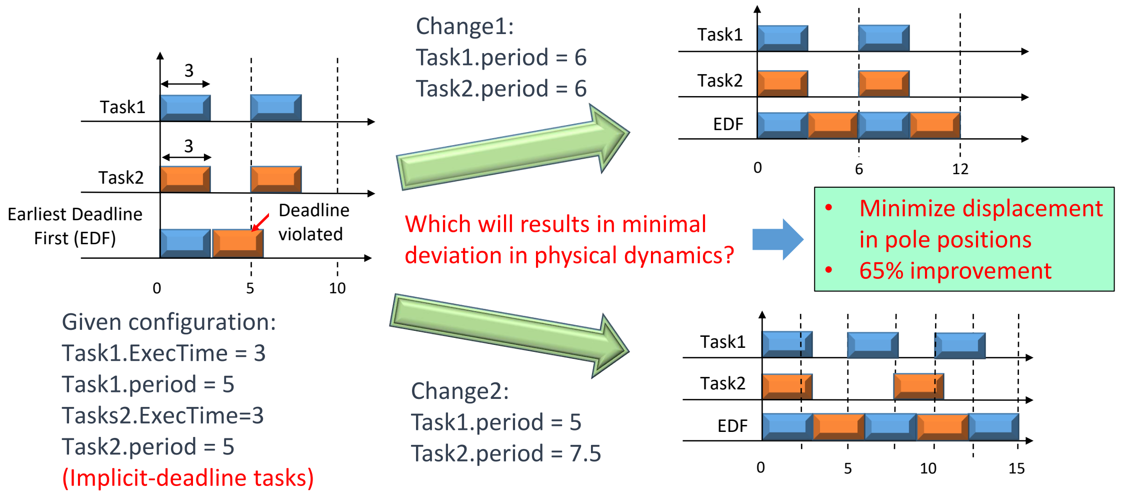


Figure 7: A simple timing debugging problem: How to change the task periods to make the task set schedulable?

the tasks on a single processor. The design flow assumed in this work is the usual one — first the control strategy, along with its associated parameters (e.g., sampling periods and gain values) are designed. This is followed by their implementation as software tasks, and in this process the schedulability test fails. Now, it is required to adjust their periods in order to make them schedulable, while minimally changing their targeted dynamics (as planned by the control designers in the first step). Changing the task periods together with the gain values leads to a co-design problem that has been studied recently in different setups [34, 32, 11] and is not the focus of this work.

Our proposed approach here hinges on the fact that the dynamics of the closed-loop system depend on the location of the system poles [16], and by changing the task periods these poles also shift. Therefore, our approach is to change those task periods that result in the minimum shift in the poles, which translates into an optimization problem that we solve using an efficient breadth-first search. For a case study, we show that our proposed scheme can minimize the average deviation in plant dynamics by more than 65% compared to a naive approach where the utilization of each task is reduced equally. A system has multiple poles and depending on their location they impact the system dynamics differently. A change in the task period might have a different impact on how much each of these poles changes. Similarly, how should the notion of dynamics be quantified? They can be quantified using metrics like peak overshoot, settling time, or the root sum square difference between the trajectories of how the state of the system evolves for two different periods. How these metrics depend on the shift in the system poles is complex and system dependent. All of these aspects lead to the complexity of the debugging process, which we have highlighted.

The goal of this work is merely to initiate a discussion on this topic, and towards this we have shown that even the simplest setup that we consider here opens up many different design issues. A more real-life setup could involve multiple (heterogeneous) processors and communication buses on which different tasks are mapped and scheduled. The timing debugging question could then expand to: how should task and message mapping and scheduling, periods of these tasks, the granularity with which they are partitioned, and possibly even their offsets, be changed in order to satisfy the required timing and schedulability constraints? We believe that this topic is mostly unexplored and, hence, there are a lot of research opportunities for the future.

3.4 Predictable Memory Management in Operating Systems [WiP]

Recent research in the area of predictable memory management at hypervisor level on complex system-on-a-chip architectures (e.g., cache coloring [27] or MemGuard [45]) use static partitioning and temporal isolation techniques to reduce interference between different virtual machines (VMs). However, these techniques often consider the VMs as black boxes that must be properly contained at hypervisor level to control interference. Likewise, the operating systems inside the VMs are also often not aware of the regulation control mechanisms of the underlying hypervisor layer. This strict separation simplifies reasoning about the isolation properties of the resulting overall system, but also leads to suboptimal system utilization.

The objective of our research in predictable memory management for operating system is to extend the scope of the mechanisms to control interference from the VM / OS level to the task / application level. This effectively turns a VM into a white box from the hypervisor's point of view, and lets the operating systems inside the VMs anticipate the regulation mechanisms of the hypervisor. We expect benefits both from the regulation point of view as well as from the performance point of view. Our techniques would allow, for example, handling of memory resources with different quality-of-service (QoS) properties (e.g., different cache partitions), and propagate the appropriate type of resource to applications.

To this end, we are extending a microkernel operating system [48] with a predictable memory management layer that suits both memory allocation and usage patterns of complex applications. Here we are integrating different memory allocators and support for partitioned virtual memory. In the future, we plan to integrate into memory regulation mechanisms at hypervisor level and provide support for upcoming hardware-based regulation such as MPAM [3]. Our current prototype targets the ARM architecture and integrates well into the environment used in our previous research [36], namely the Xilinx ZCU102 Ultrascale platform, the Jailhouse Hypervisor, and a predictable guest operating system on top of the hypervisor.

We plan to submit publications on the topic of predictable memory management from virtual machine level down to task level at top-tier conferences and release the code of the prototype as open source.

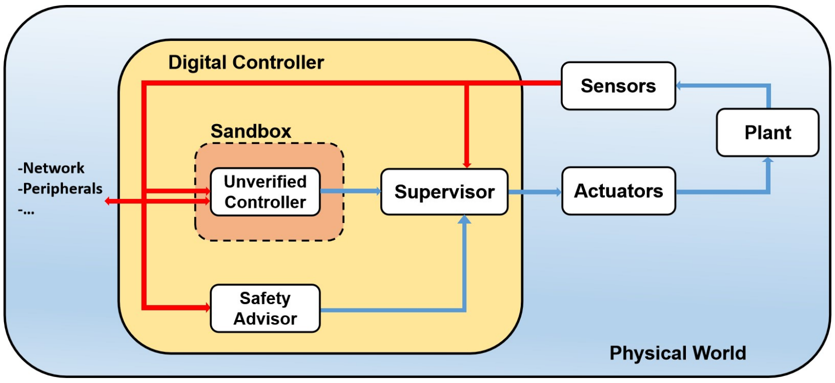


Figure 8: Safety Advisor - Supervisor (Safe-visor) Architecture for sandboxing (AI-based) unverified controllers.

4 Safe-visor Architecture for Sandboxing (AI-based) Unverified Controllers in Safety-critical Cyber-Physical Systems

Modern Cyber-Physical Systems (CPS) have received significant attention in the past few years since they provide powerful frameworks for modeling various applications, including transportation systems, power grids, manufacturing plants, and so on. Nowadays, high-performance but unverified controllers like artificial intelligence-based (a.k.a. AI-based) controllers or black-box controllers from third parties are expected to be employed in CPSs to accomplish complex missions. However, the application of such controllers makes it increasingly challenging to ensure the overall safety of CPSs. Particularly, ensuring safety is crucial for safety-critical applications, where system failures (e.g., collision) are not acceptable. This issue motivates us to investigate control architectures that allow the application of unverified controllers for functionality, while formal safety guarantees can still be provided.

In this project, we exploit the idea of the sandbox from the community of computer security to design a correct-by-construction architecture, namely Safe-visor architecture, for sandboxing (AI-based) unverified controllers. This architecture is illustrated in Figure 8. In general, the Safe-visor architecture specifies verifiable safety rules for the unverified controller to provide safety guarantees for physical systems. Concretely, this architecture employs a Safety Advisor and a Supervisor (Safe-visor). At runtime, the Safety Advisor is responsible for providing an advisory input that only enforces the safety property of interest. Meanwhile, the Supervisor checks the input given by the unverified controller according to the same safety property. The unverified inputs would only be accepted when they follow the safety rules; otherwise, the Supervisor would accept the advisory input given by the Safety Advisor. It is worth noting that the unverified controller is supposed to accomplish tasks that are much more complex than only maintaining the system's safety. Accordingly, the Safety Advisor would only be used if the unverified controller tries to perform harmful actions. As a result, one can exploit the functionalities offered by the unverified controller while preventing the system from becoming unsafe by utilizing our proposed architecture.

Compared with our preliminary results in [47], we are currently [46, 25] able to handle a larger class of safety specifications, which can be expressed by the accepting languages of deterministic finite automata (DFA) [4], instead of simple invariance properties as in [47]. Here, we provide

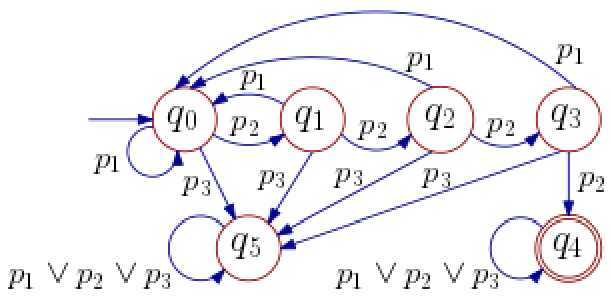


Figure 9: DFA with accepting state q_4, alphabet $\{p1, p2, p3\}$ and labeling function L, where $L(y) = p_1$ when output y of the system is within in $(3, 10]$, $L(y) = p_2$ when y is within $[0, 3]$ and $L(y) = p_3$ when y is within $(-\infty, 0)$ or $(10, +\infty)$. In English, this property requires that (i) Within 8 time instances, the system output should reach $[0, 3]$ and then stay within $[0, 3]$ for 3 time instances after it reaches $[0, 3]$; (ii) the output should not exceed $[0, 10]$.

an example of DFA in Figure 9. Additionally, we develop our new approaches on top of an approximate probabilistic relation [20] between the original system and its finite abstraction that is utilized for constructing the controllers of interest. With this type of relation, the synthesized architecture can reduce the error between the original model and its abstraction and, hence, enforce the desired properties with a less conservative probabilistic guarantee.

The new results are applied to a controlled problem of a DC motor. For demonstration purposes, we train a controller using deep reinforcement learning with deep deterministic policy gradient (DDPG) algorithms [26] for the DC motor to track the desired angular velocity. The experiments are performed via MATLAB 2019b, on a machine with Windows 10 operating system (Intel Xeon E-2186G CPU running at 3.8 GHz and using 32 GB of RAM). Without sandboxing the AI-based controller, all output sequences violate the desired specification. Meanwhile, by sandboxing the AI-based controller, all output sequences satisfy the desired property, while 84.73% of inputs from the unverified controller can be accepted. In other words, the unverified AI-based controller can still be applied most of the time when it does not endanger the system's safety.

5 Work in progress 6D Pose Estimation

In the current year we continued our effort to setup a robotics laboratory aimed at the automation of industrial processes with a special focus on unstructured environments and varying production lines. Beside the procurement and setup of the necessary infrastructure, i.e. a FANUC CRX robot, a GPU cluster for deep learning, a data acquisition set up for 3D models generation and other tools, we have also progressed in the integration of the different systems. In particular, we are currently developing: an API for deep-neural-networks-based control of the industrial robot, a simulation environment for the generation of training data for grasping and manipulation, a photogrammetry and reconstruction pipeline to produce models for the aforementioned simulation, and high dynamic range simulation for deep reinforcement learning with realistic visual rendering. We plan to complete most of the above development work early next year which will lay the basis for our future research goals. The first goal is to publish a comprehensive training dataset based on both simulated data and real pictures to provide end-to-end training from stereo images to depth and object poses. Further work on deep-learning-based grasping methods in production lines will be based on this training database, as well as on the online simulation that we are working on.

6 Basic Information of the Chair of Cyber-Physical Systems in Production Engineering

Prof. Dr. Marco Caccamo
Contact
www.mw.tum.de/cps
mcaccamo@tum.de
Tel: +49.89.289.55170

Management

Prof. Dr. Marco Caccamo, Director

Administrative Staff

Anke Harisch, Secretary

Research Scientists

- Tomasz Kloda, Dr.
- Andrea Bastoni, Dr.
- Debayan Roy, Dr.
- Alexander Züpke, Dr.
- Ohchul Kwon, Dr.
- Mirco Theile, M.Sc.
- Bingzhuo Zhong, M.Sc.
- Or Dantsker, M.Sc.
- Denis Hoornaert, M.Sc.

- Jiyang Chen, M.Sc.
- Hongpeng Cao, M.Eng.
- Daniele Bernardini, M.Sc.
- Binqi Sun, M.Sc.
- Gero Schwäricke, M.Sc.
- Richard Nai (MS student)
- Federico Wyrwal (MS student)
- Andres Zapata Rodriguez (MS Student)
- Lukas Dirnberger (MS Student)
- Tobias Mascetta (MS Student)
- Ziyuan Qin (MS Student)

Research Focus

- Safety-critical cyber-physical systems
- Real-time systems
- Scheduling and schedulability analysis
- Secure and safe integration of machine learning with CPS
- Reinforcement learning for CPS

Competence

- System-level programming
- Embedded system software design
- Hardware and software integration on FPGAs
- Real-time operating systems
- Reinforcement learning for CPS

Infrastructure

- 3 DOF helicopter
- Embedded and FPGA multi-core development platforms
- High-performance servers
- Linear inverted pendulum
- Fused filament fabrication, dual-head 3D printer
- F1/10 autonomous cars

Collaborations

- University of Illinois at Urbana-Champaign, USA
- Boston University, USA
- University of Colorado Boulder, USA
- University of Waterloo, Canada
- University of Modena, Italy
- Federal University of Santa Catarina, Brazil
- EURECOM, Sophia Antipolis, France

Lectures

- Concepts and Software Design for Cyber-Physical Systems
- Tutorial Concepts and Software Design for Cyber-Physical Systems
- Advanced Seminar on Safe Cyber-Physical Systems
- PhD-Seminar on Real-Time Cyber-Physical Systems
- Cyber-Physical Systems Lab: Autonomous Applications
- Simplex: Fault-Tolerant Control Strategy for Real-Time Cyber-Physical Systems - Laboratory

Humboldt Sponsored Research

Selected Publications 2021

Journal

- Bingzhuo Zhong, Abolfazl Lavaei, Hongpeng Cao, Majid Zamani, and Marco Caccamo. Safe-visor architecture for sandboxing (AI-based) unverified controllers in stochastic Cyber–Physical Systems. *Nonlinear Analysis: Hybrid Systems*, 43:101110, 2021
- Rohan Tabish, Jen-Yang Wen, Rodolfo Pellizzoni, Renato Mancuso, Heechul Yun, Marco Caccamo, and Lui Raymond Sha. An analyzable inter-core communication framework for high-performance multicore embedded systems. *Journal of Systems Architecture*, page 102178, 2021
- Harald Bayerlein, Mirco Theile, Marco Caccamo, and David Gesbert. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open Journal of the Communications Society*, 2:1171–1187, 2021

Conference proceeding

- Gero Schwaericke, Rohan Tabish, Rodolfo Pellizzoni, Renato Mancuso, Andrea Bastoni, Alexander Zuepke, and Marco Caccamo. A real-time virtio-based framework for predictable inter-VM communication. In *2021 IEEE International Real-Time Systems Symposium (RTSS)*, 2021
- Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV path planning using global and local map information with deep reinforcement learning. *arXiv preprint arXiv:2010.06917*, 2020
- Bruna Arruda Araujo, Giovani Gracioli, Tomasz Kloda, Denis Hoornaert, and Marco Caccamo. Implementation and evaluation of adaptive cache insertion policies for real-time systems. In *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–8. IEEE, 2021
- Or D. Dantsker, Mirco Theile, and Marco Caccamo. Long endurance flight testing results for the UIUC-TUM solar flyer. In *AIAA AVIATION 2021 FORUM*, page 3196, 2021
- Or D. Dantsker, Mirco Theile, Marco Caccamo, and Seongyong Hong. Integrated power simulation for a solar-powered, computationally-intensive unmanned aircraft. In *AIAA Propulsion and Energy 2021 Forum*, page 3317, 2021
- Or D. Dantsker, Marco Caccamo, and Renato Mancuso. Energy system instrumentation and data acquisition for flight testing a long-endurance, solar-powered unmanned aircraft. In *AIAA Propulsion and Energy 2021 Forum*, page 3721, 2021
- Ayoosh Bansal, Jayati Singh, Micaela Verucchi, Marco Caccamo, and Lui Sha. Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles. *arXiv preprint arXiv:2106.04146*, 2021
- Denis Hoornaert, Shahin Roozkhosh, Renato Mancuso, and Marco Caccamo. Work in progress: Identifying unexpected inter-core interference induced by shared cache. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 517–520, 2021
- Jiyang Chen, Tomasz Kloda, Ayoosh Bansal, Rohan Tabish, Chien-Ying Chen, Bo Liu, Sibin Mohan, Marco Caccamo, and Lui Sha. SchedGuard: Protecting against schedule leaks using linux containers. *arXiv preprint arXiv:2104.04528*, 2021
- Abolfazl Lavaei, Bingzhuo Zhong, Marco Caccamo, and Majid Zamani. Towards trustworthy AI: safe-visor architecture for uncertified controllers in stochastic Cyber-Physical Systems. In *Proceedings of the Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems*, pages 7–8, 2021
- Ohchul Kwon, Gero Schwäricke, Tomasz Kloda, Denis Hoornaert, Giovani Gracioli, and Marco Caccamo. Flexible cache partitioning for multi-mode real-time systems. In *Design, Automation and Test in Europe Conference (DATE 2021)*, 2021
- Debayan Roy, Clara Hobbs, James H Anderson, Marco Caccamo, and Samarjit Chakraborty. Timing debugging for Cyber-Physical Systems. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, page 1893–1898. IEEE, 2021

- Jonathan Ponniah, Mirco Theile, Or Dantsker, and Marco Caccamo. Autonomous hierarchical multi-level clustering for multi-UAV systems. In *AIAA Scitech 2021 Forum*, page 0656, 2021

- Debayan Roy, Licong Zhang, Wanli Chang, Dip Goswami, Birgit Vogel-Heuser, and Samarjit Chakraborty. Tool integration for automated synthesis of distributed embedded controllers. *ACM Trans. Cyber-Phys. Syst.*, 6(1), nov 2021

Software in Development

- Jailhouse Cache-coloring. Jailhouse ML.
  https://groups.google.com/g/jailhouse-dev/c/K4rqZxpxa0U

- Virtio prototype (see [36]). https://github.com/gschwaer/rt-virtio

References

[1] Siemens AG. Jailhouse hypervisor. https://github.com/siemens/. Accessed: 2021-02-08.

[2] Bruna Arruda Araujo, Giovani Gracioli, Tomasz Kloda, Denis Hoornaert, and Marco Caccamo. Implementation and evaluation of adaptive cache insertion policies for real-time systems. In *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–8. IEEE, 2021.

[3] ARM. Arm architecture reference manual supplement. memory system resource partitioning and monitoring (MPAM) for Armv8-A. https://developer.arm.com/docs/ddi0598/latest.

[4] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[5] Ayoosh Bansal, Jayati Singh, Micaela Verucchi, Marco Caccamo, and Lui Sha. Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles. *arXiv preprint arXiv:2106.04146*, 2021.

[6] Andrea Bastoni. Jailhouse Cahe-Coloring Proposal. https://groups.google.com/g/jailhouse-dev/c/K4rqZxpxa0U.

[7] Harald Bayerlein, Mirco Theile, Marco Caccamo, and David Gesbert. UAV path planning for wireless data harvesting: A deep reinforcement learning approach. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[8] Harald Bayerlein, Mirco Theile, Marco Caccamo, and David Gesbert. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open Journal of the Communications Society*, 2:1171–1187, 2021.

[9] Michael Bechtel and Heechul Yun. Denial-of-service attacks on shared cache in multicore: Analysis and prevention. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 357–367, 2019.

[10] Hongpeng Cao, Mirco Theile, Federico G Wyrwal, and Marco Caccamo. Cloud-edge training architecture for sim-to-real deep reinforcement learning. *arXiv preprint arXiv:2203.02230*, 2022.

[11] Wanli Chang, Licong Zhang, Debayan Roy, and Samarjit Chakraborty. *Control/Architecture Codesign for Cyber-Physical Systems*. Number 37. Springer Netherlands, 2017.

[12] Jiyang Chen, Tomasz Kloda, Ayoosh Bansal, Rohan Tabish, Chien-Ying Chen, Bo Liu, Sibin Mohan, Marco Caccamo, and Lui Sha. SchedGuard: Protecting against schedule leaks using linux containers. *arXiv preprint arXiv:2104.04528*, 2021.

[13] Or D. Dantsker, Marco Caccamo, and Renato Mancuso. Energy system instrumentation and data acquisition for flight testing a long-endurance, solar-powered unmanned aircraft. In *AIAA Propulsion and Energy 2021 Forum*, page 3721, 2021.

[14] Or D. Dantsker, Mirco Theile, and Marco Caccamo. Long endurance flight testing results for the UIUC-TUM solar flyer. In *AIAA AVIATION 2021 FORUM*, page 3196, 2021.

[15] Or D. Dantsker, Mirco Theile, Marco Caccamo, and Seongyong Hong. Integrated power simulation for a solar-powered, computationally-intensive unmanned aircraft. In *AIAA Propulsion and Energy 2021 Forum*, page 3317, 2021.

[16] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems*. Prentice-Hall, Inc., USA, 9th edition, 2000.

[17] Bosch GmbH. ETAS RTA Hypervisor. https://www.etas.com/en/products/rta-vrte.php. Accessed: 2021-02-08.

[18] SYSGO GmbH. PikeOS Hypervisor. https://www.sysgo.com.

[19] Giovani Gracioli, Ahmed Alhammad, Renato Mancuso, Antônio Augusto Fröhlich, and Rodolfo Pellizzoni. A survey on cache management mechanisms for real-time embedded systems. *ACM Comput. Surv.*, 48(2), nov 2015.

[20] Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511, 2020.

[21] Denis Hoornaert, Shahin Roozkhosh, and Renato Mancuso. A memory scheduling infrastructure for multi-core systems with re-programmable logic. In Björn B. Brandenburg, editor, *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, volume 196 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:22, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[22] Denis Hoornaert, Shahin Roozkhosh, Renato Mancuso, and Marco Caccamo. Work in progress: Identifying unexpected inter-core interference induced by shared cache. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 517–520, 2021.

[23] T. Kloda, M. Solieri, R. Mancuso, N. Capodieci, P. Valente, and M. Bertogna. Deterministic memory hierarchy and virtualization for modern multi-core embedded systems. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 1–14, 2019.

[24] Ohchul Kwon, Gero Schwäricke, Tomasz Kloda, Denis Hoornaert, Giovani Gracioli, and Marco Caccamo. Flexible cache partitioning for multi-mode real-time systems. In *Design, Automation and Test in Europe Conference (DATE 2021)*, 2021.

[25] Abolfazl Lavaei, Bingzhuo Zhong, Marco Caccamo, and Majid Zamani. Towards trustworthy AI: safe-visor architecture for uncertified controllers in stochastic Cyber-Physical Systems. In *Proceedings of the Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems*, pages 7–8, 2021.

[26] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations(Poster)*, 2016.

[27] R. Mancuso, R. Dudko, E. Betti, M. Cesati, M. Caccamo, and R. Pellizzoni. Real-time cache management framework for multi-core architectures. In *2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 45–54, 2013.

[28] Renato Mancuso. Boston University. http://www.bu.edu/cs/profiles/renato-mancuso/.

[29] Jonathan Ponniah, Mirco Theile, Or Dantsker, and Marco Caccamo. Autonomous hierarchical multi-level clustering for multi-UAV systems. In *AIAA Scitech 2021 Forum*, page 0656, 2021.

[30] The Linux Foundation Projects. ACRN hypervisor. https://projectacrn.org/. Accessed: 2021-02-08.

[31] Shahin Roozkhosh and Renato Mancuso. The potential of programmable logic in the middle: Cache bleaching. In *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 296–309, 2020.

[32] Debayan Roy, Sumana Ghosh, Qi Zhu, Marco Caccamo, and Samarjit Chakraborty. Goodspread: Criticality-aware static scheduling of CPS with multi-qos resources. In *41st IEEE Real-Time Systems Symposium, RTSS 2020, Houston, TX, USA, December 1-4, 2020*, page 178–190. IEEE, 2020.

[33] Debayan Roy, Clara Hobbs, James H Anderson, Marco Caccamo, and Samarjit Chakraborty. Timing debugging for Cyber-Physical Systems. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, page 1893–1898. IEEE, 2021.

[34] Debayan Roy, Licong Zhang, Wanli Chang, Dip Goswami, Birgit Vogel-Heuser, and Samarjit Chakraborty. Tool integration for automated synthesis of distributed embedded controllers. *ACM Trans. Cyber-Phys. Syst.*, 6(1), nov 2021.

[35] Gero Schwaericke. A real-time virtio-based framework for pre-dictable inter-VM communication (software). https://github.com/gschwaer/rt-virtio.

[36] Gero Schwaericke, Rohan Tabish, Rodolfo Pellizzoni, Renato Mancuso, Andrea Bastoni, Alexander Zuepke, and Marco Caccamo. A real-time virtio-based framework for predictable inter-VM communication. In *2021 IEEE International Real-Time Systems Symposium (RTSS)*, 2021.

[37] Green Hills Software. GHS Integrity. https://www.ghs.com/products/rtos/integrity_virtualization.html.

[38] Parul Sohal, Rohan Tabish, Ulrich Drepper, and Renato Mancuso. E-WarP: A system-wide framework for memory bandwidth profiling and management. In *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020.

[39] Rohan Tabish, Jen-Yang Wen, Rodolfo Pellizzoni, Renato Mancuso, Heechul Yun, Marco Caccamo, and Lui Raymond Sha. An analyzable inter-core communication framework for high-performance multicore embedded systems. *Journal of Systems Architecture*, page 102178, 2021.

[40] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV path planning using global and local map information with deep reinforcement learning. *arXiv preprint arXiv:2010.06917*, 2020.

[41] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV path planning using global and local map information with deep reinforcement learning. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 539–546, 2021.

[42] Unimore. HiPeRT Lab. https://hipert.unimore.it.

[43] Prathap Kumar Valsan, Heechul Yun, and Farzad Farshchi. Addressing isolation challenges of non-blocking caches for multicore real-time systems. *Real-Time Systems*, 53(5):673–708, 2017.

[44] H. Yun, R. Mancuso, Z. P. Wu, and R. Pellizzoni. PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, page 155–166, 2014.

[45] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. Memory bandwidth management for efficient performance isolation in multi-core platforms. *IEEE Transactions on Computers*, 65(2):562–576, 2016.

[46] Bingzhuo Zhong, Abolfazl Lavaei, Hongpeng Cao, Majid Zamani, and Marco Caccamo. Safe-visor architecture for sandboxing (AI-based) unverified controllers in stochastic Cyber–Physical Systems. *Nonlinear Analysis: Hybrid Systems*, 43:101110, 2021.

[47] Bingzhuo Zhong, Majid Zamani, and Marco Caccamo. Sandboxing controllers for stochastic Cyber-Physical Systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 247–264. Springer, 2019.

[48] Alexander Zuepke. Turning futexes inside-out: Efficient and deterministic user space synchronization primitives for real-time systems with IPCP. In Marcus Völp, editor, *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, volume 165 of *Leibniz International Proceedings in Informatics (LIPIcs)*, page 11:1–11:23, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.