2024 Design, Automation & Test in Europe Conference (DATE 2024)
Special Initiative "Autonomous Systems Design"

Constraint-aware Resource Management for Cyber-Physical Systems

Justin McGowen, Ismet Dagli, Neil T. Dantam, and Mehmet E. Belviranli
Computer Science Department at Colorado School of Mines
{jmcgowen, ismetdagli, ndantam, belviranli}@mines.edu

***Abstract*—Cyber-physical systems (CPS) such as robots and self-driving cars pose strict physical requirements to avoid failure. Scheduling choices impact these requirements. This presents a challenge: how do we find efficient schedules for CPS with heterogeneous processing units, such that the schedules are resource-bounded to meet the physical requirements?**

We propose the creation of a structured system, the Constrained Autonomous Workload Scheduler, which determines scheduling decisions with direct relations to the environment. By using a representation language (AuWL), Timed Petri nets, and mixed-integer linear programming, our scheme offers novel capabilities to represent and schedule many types of CPS workloads, real world constraints, and optimization criteria.

***Index Terms*—accelerator, autonomous systems, gpu, heterogeneous, cyber-physical systems**

I. Introduction

Embedded heterogeneous computing systems offer improved latency and energy use in Cyber-Physical Systems (CPS), but to do so under the physical requirements of a CPS requires a way to simultaneously consider both different processors and physical requirements. CPS have two factors that fundamentally limit the ability to handle their requirements. The first factor is physical—*e.g.*, braking distance or battery charge. The other limiting factor is computational—*i.e.*, moving and processing data to make a decision takes time and energy. Many recent CPS embed powerful heterogeneous system-on-chips (SoC), such as NVIDIA's Xavier or Tesla's FSD, which can improve these computational limits. A key property of these SoCs is that they employ a variety of processing units (PUs): often a general purpose CPU and various domain specific accelerators (DSAs).

Autonomous systems place multiple requirements on resource use. Systems with time-critical components or strict constraints on energy or power place upper bounds on resource use for any PU (*i.e.*, CPUs or DSAs); if computation takes more time, energy, *etc.*, such a system might fail. For example, an aerial drone may fly at 50mph, when—considering the true computation time—it only has enough time to react if flying at 40mph. Such requirements may pose a variety of constraints and objectives to optimize, such as, minimizing power while maintaining safety, flying as fast as safely possible, or minimizing latency. Finding the optimal task schedule under such tradeoffs is non-trivial.

Scheduling for diversely heterogeneous systems has broadly been investigated over the last decade [1], [2], [7]. A limited number of studies [5], [6], [8] have structurally approached timing in CPS computation by building models relating physical constraints and computational elements. However, these studies are restricted to a specific physical constraint and do not address domain-independent criteria.

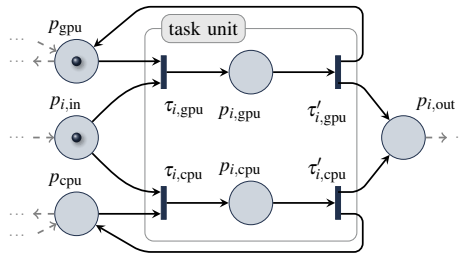

Fig. 1: Proposed Petri net construction. *(a)* Petri net "unit" for one task. Each unit has multiple "paths", one for each PU. The unit operates by (1) firing $\tau_{i,xpu}$ to consume tokens for its input and a selected PU and setting a token on the place to remember the selected PU, and (2) firing $\tau'_{i,xpu}$ to set a token in the output place and restore the PU availability token. In this example, the GPU is available.

In this study, we propose a constraint-based autonomous workload modeling and scheduling technique for CPSs with embedded heterogeneous PUs. Our approach offers a generalized solution to the heterogeneous (*i.e.*, multi-DSA) scheduling problem that accounts for the physical constraints using a novel representative language (AuWL), Timed Petri nets, and mixed-integer linear programming. Our scheduler creates schedules that are optimal with respect to the user-defined objective (including time) and profiled executions, and the schedules maintain the constraints defined in AuWL, as computation can impact time, power, or other factors limited by the physical requirements.

II. Petri Nets for Heterogeneous Scheduling

A Petri net [3] is a directed, bipartite graph.

Definition 1: A Petri net is the tuple $\mathcal{N} = (P, T, E)$, where, P and T is the finite set of *place* nodes and *transition* nodes, respectively, and $E \subseteq (P \times T \cup T \times P)$ are *edges* between places and transitions. Each place P may contain a number of *tokens*. We call the number of tokens contained in all places a configuration or *marking* of the Petri net. When a particular transition *fires*, it changes the marking by decrementing the tokens at incoming places and incrementing tokens at outgoing places.

In our application of Petri nets (as shown in Figure 1), we use places to represent the availability of a PU. Transitions specify the possible changes in resources. The incoming places to a transition represent resources that are acquired or consumed, and the outgoing places represent resources that are released or produced. The Petri net captures the parallelism by allowing transitions to fire in any order, as long as their incoming places have positive token counts.

III. Proposed Methodology

The input to our proposed scheme is a specification that includes (1) *the control flow graph* (CFG) represented using

979-8-3503-4859-0/DATE24/© 2024 EDAA

```
model AuWL_example {
    constraint (= total-power 30)
    constraint (= velocity 5)
    constraint (= motor-power (* velocity velocity))
    constraint (< ENERGY 50)
    constraint (< POWER (- total-power motor-power))
    objective  (- TIME)
    data camera, lidar
    data obj_bounding_boxes, localized_position, route
    op object_detection {in=camera;out=bounding_boxes}
    op localization {in=lidar;out=localized_position}
    op route_planning {in=obj_bounding_boxes,
                       localized_position; out=route}
}
```

Fig. 2: An example AuWL program, defining the primary operations, their dependencies, CPS constraints and the objective for a simplified model of an autonomous vehicle.

operations in AuWL file, (2) the necessary *performance criteria* (*e.g.*, a constraint on time, energy or power), and (3) the *profiling data* for estimated runtimes and energy consumption of tasks on available PUs. The output is a heterogeneous schedule that includes (1) *the set of tasks*, (2) *the ordering of tasks*, and (3) *the mapping of tasks to PUs*.

We use the system specification (CFG, performance criteria, and profiling data) to construct a Petri net as an intermediate representation of the CPS. Our method utilizes the Petri net to generate a set of constraints and objectives corresponding to valid and optimal (with respect to the objective) schedules. We find a solution to these constraints using a state-of-the-art solver (specifically, Z3 [4]) to obtain a schedule that satisfies the physical requirements. Z3 guarantees optimality given an objective, selecting from the wider set of Pareto-optimal schedules.

Our proposed methodology introduces the Autonomous Workload Language (AuWL) representation. AuWL describes the data flow of tasks (*i.e.*, the CFG) and the necessary schedule criteria (*e.g.*, minimizing computation time, meeting an energy budget). Figure 2 illustrates an example AuWL representation for a simplified autonomous scenario, which must minimize execution time under several constraints. The Petri net captures the structure of control flow choices and resource use, facilitating the subsequent generation of constraints. We achieve this by constructing a set of timing constraints for valid schedules using the information in AuWL representation.

IV. Evaluation

We evaluate our proposed scheme on a search-and-rescue simulation with a quadcopter (3DR Iris) in a burning house. The drone must explore the house to monitor fires and discover those in need of help. This environment induces 3 constraints: *Heat*, *Power* and *Stopping Distance*. As the computing platform, we target NVIDIA's Xavier AGX and NX. They both provide two accelerators: GPU (low-latency) and DLA (low-power). We run and profile a variety of neural networks (NN) supported by TensorRT on both GPU and DLA.

Our experiments show that our methodology can find a schedule predicted to meet the constraints as long as the drone stays within the region where there are valid schedules. The drone adapts to the varying conditions with these schedules depicted in Figure 3. When the drone, right before passing close to the fire, goes around the corner and skirts the wall, the corner reduces obstacle distance and required latency (Schedule 1). As the drone gets near the heat source, a more

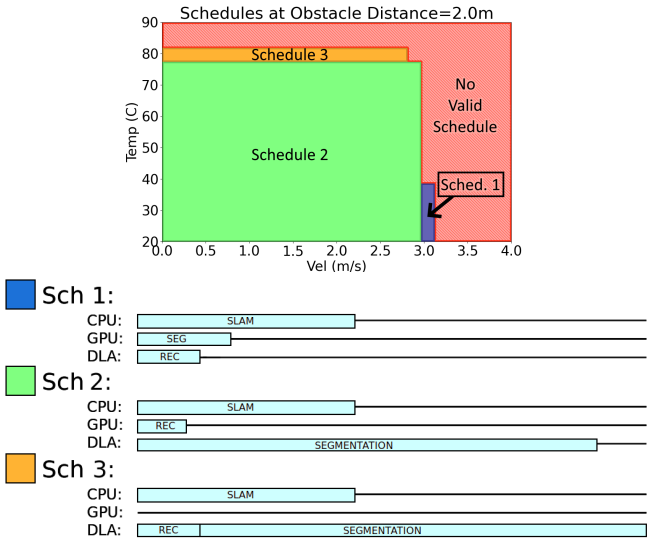

Fig. 3: The set of schedules generated for the case study. CAuWS can pre-compute multiple schedules for varying physical parameters, creating a policy such as "if speed is less than 3m/s and ambient temperature less the 76, run NN 1 on the GPU and NN 2 on the DLA simultaneously". Schedule 1 is fast but uses more energy, while Schedule 3 is slow but more energy efficient, due to the choice of PUs.

energy-efficient schedule must be chosen (Schedule 3). In other cases, it defaults to Schedule 2. Past schedulers, which do not consider physical constraints, could only choose one of these possible schedules and would violate physical constraints for at least 25% of the length of the path. If the path went significantly closer to the wall or fire, there would be no possible schedules regardless of which scheduler is used.

V. Conclusion

We presented a scheme to represent and generate schedules for heterogeneous devices that satisfy changing physical constraints. Our scheme works for a wide variety of devices, and its capability is demonstrated on a heterogeneous compute platform controlling a simulated drone. Our proposed scheme offers foundation to address an expanded set of scheduling problems that take physical constraints into account.

Acknowledgements

This material is based upon work supported by NSF under Grant No. CCF-2124010. Any opinions, findings, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Ismet Dagli and Mehmet Belviranli. Shared memory-contention-aware concurrent dnn execution for diversely heterogeneous socs. In *PPoPP'24*.

[2] Ismet Dagli, Alexander Cieslewicz, Jedidiah McClurg, and Mehmet E Belviranli. Axonn: Energy-aware execution of neural network inference on multi-accelerator heterogeneous socs. In *DAC*, 2022.

[3] Cassandras et al. *Introduction to discrete event systems*. Springer, 2008.

[4] Leonardoet De Moura et al. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008.

[5] Ramyad Hadidi et al. Quantifying the design-space tradeoffs in autonomous drones. In *ASPLOS*, pages 661–673, 2021.

[6] Srivatsan Krishnan et al. Sky is not limit: A visual performance model for cyber-physical co-design in autonomous machines. *IEEE CAL*, 2020.

[7] Van Craeynest et al. Scheduling heterogeneous multi-cores through performance impact estimation (pie). In *ISCA*, 2012.

[8] Zishen Wan et al. Analyzing and improving fault tolerance of learning-based navigation systems. In *DAC*, 2021.