Mitigating Computational Constraints via Adaptive Control and Resource Allocation Co-design*

Linh Thi Xuan Phan
University of Pennsylvania

Ricardo G. Sanfelice
University of California, Santa Cruz

I. Introduction and Motivation

Most engineering systems of today involve physics and computing systems for their operation. These cyber-physical systems (CPS) exchange data of different nature (analog, digital, etc.) and dimension (small data packets, video streams, etc.) between its components [1], [5]. The resources required for the processing of the data and for making decisions change over time. *One size fits all* type of solutions are not adequate, as they lead to a costly, oversized system that is not necessarily tailored to the application at hand.

Many applications of CPS require the control algorithm to change its behavior to adapt to events in the environment. A widely adopted paradigm consists of defining different modes of operation for the system and the controller algorithm, and include a decision maker that selects the suitable mode of operation for the current conditions. In most cases, the control algorithm used for each mode is significantly different structurally (e.g., a PID, MPC, or neural network controller), or uses totally different control parameters (e.g., adaptive cruise control). As a result of the changes in the system and control algorithm, the required platform resources change. Unfortunately, the unavoidable computational constraints makes adaptive resource allocation challenging.

To cope with such changes, resources need to be reallocated properly to optimize performance and resource use. A similar situation emerges when the control algorithm consists of the combination of several controllers. In such a setting, the decision maker may activate or deactivate different sub-controllers as the system operates (e.g., on a highway, pedestrian detection is not typically needed), which results in different subset of tasks running. Motivated by these real-world scenarios, designers are typically interested in using multiple control tasks, deployed on a multicore platform, and in resources (e.g., CPU, cache, memory, and memory bandwidth) that can be dynamically adapted. To materialize such a need, co-design of the controllers and the decision maker that allocate resources to them is critical to optimize performance while minimizing resource use.

State-of-the-art research in CPS has considered this problem to a limited extent. For example, control-platform co-design methods have been proposed that enable a control algorithm to change its behavior (e.g., sampling time) based on the available CPU or network resource. Independently, real-time multi-mode techniques have been developed to ensure timing predictability of multi-mode systems. However, research in these two directions is typically *disconnected from one another*, and each focuses exclusively on either CPU (on a single core platform) or network. None of the existing work considers shared resources on modern multicore hardware or heterogeneous architectures.

*This research is supported in part by NSF grants CNS-1703936, CNS-1750158, CNS-1955670, ECS-1710621, CNS-1544396 and CNS-2039054, ONR grant N00014-20-1-2744, AFOSR grants FA9550-19-1-0053, FA9550-19-1-0169 and FA9550-20-1-0238, and by ARO grant W911NF-20-1-0253.

II. Challenges and Opportunities in Co-design of Adaptive Control and Resource Allocation

A. *Compositional co-design with control-resource interfaces*

An important challenge to control and resource allocation co-design is complexity. Due to sharing, different controllers (implemented as tasks) executing concurrently in the system can substantially interfere with one another and cause deadline misses by accessing shared resources, such as cache, memory and memory bandwidth on a multicore platform. In addition, different types of resources on a platform are often interdependent – e.g., the larger the shared cache, the less the demands on memory bandwidth – and so are their effects on tasks' execution. Further, analyzing a system that dynamically updates its behavior at run time is generally much more difficult than analyzing a static system. Finally, while considering the controllers and resource allocation jointly improves performance and resource optimization, it also increases the design space and analysis complexity.

One promising approach to addressing this complexity is through novel adaptive interfaces for *compositional co-design*. Such interfaces holistically, and succinctly, capture the interdependence between the characteristics and resource needs of the controller, as well as the resource availability and reconfiguration/reallocation capabilities of the platform. The interfaces should also enable run-time adaptation – besides exposing the dynamic nature of the system across different levels, they should additionally inform feasible run-time changes to the control behavior and resource allocation. Based on such an interface, control algorithms and resource allocation strategies can be concurrently developed and analyzed, and will later be composed (and potentially redesigned) in an efficient manner. Prior work on real-time compositional analysis [3], [7] and assume-guarantee interfaces [6] should provide a useful first step.

B. *Co-synthesis of modes and mode transitions*

Our approach to modeling adaptive CPS is by using the *multi-mode formalism* [4]. At the high level, the system operates in multiple modes, where each mode represents a distinct control behavior and/or resource allocation, and each mode transition captures triggering conditions for a new control behavior/strategy or resource reallocation. This explicit modeling of run-time adaptation via mode transitions is beneficial as it facilitates formal analysis (which is essential for safety) and systematic exploration of the design space. A key question here is how to efficiently construct a mode structure for the system that balances the trade-off between analysis efficiency, performance, safety guarantee, and resource optimization.

We observe that the multi-mode abstraction applies at nearly all levels across the system hierarchy. For instance, the system may operate in different modes that each consist of different subsets of control software components, each control software may switch between different controllers, each controller may transition between different control parameters with different resource needs, and the platform may switch between different configurations in response to events such as hardware/software failures. Based on this insight, a potential direction is to use different techniques for constructing modes and transitions that best suit each level and compose the results. For this, we anticipate that existing techniques from an interdisciplinary set of domains could be adapted—e.g., fault-tolerance techniques can provide a way for constructing different hardware/software configurations in response to faults; hybrid control design and learning methods can be extended to synthesize verifiable controllers and their switching semantics; and safety assurance and real-time scheduling techniques can be adapted to derive system-level modes and transitions.

C. Unified control-resource mode change protocols

During run-time adaptation, as the system changes the control behavior and resource allocation by moving to a new mode, it enters a transient stage in which jobs from both modes co-exist. This co-execution of jobs from old and new modes may lead to a temporal overload that can cause jobs to miss their deadlines. Simultaneously, the system may need to perform transitional activities that add delay, such as saving and loading state variables for consistency and functional correctness. Hence, appropriate mechanisms for handling transient stages, otherwise known as mode change protocols (MCPs), are crucial to the overall performance and safety of the system. By enforcing a certain execution behavior of a mode transition – such as aborting certain jobs, or delaying the release of new jobs – such protocols minimize potential overload to avoid timing and safety violations.

MCPs have been studied extensively in the real-time community; however, prior work focuses exclusively on timeliness and CPU resource scheduling. There is a need for novel unified protocols that holistically consider control, scheduling and resource allocation. Specifically, they should define mode change actions not only with respect to CPU scheduling but also for the reallocation of the different shared resources on modern hardware. Furthermore, they should tightly integrate the mode switch semantics of the control algorithm with that of the scheduling and resource allocation. For example, the switching mechanism between different controllers needs to account for the mode change actions and resulting delay at the scheduling and resource allocation in the implementation layer, and vice versa. To be effective, the protocols must also be low in run-time overhead. New control synthesis and mode change analysis techniques will also need to be developed to ensure safety and control performance under such unified MCPs.

D. Efficient and safe run-time systems for co-design

Theoretical foundations for adaptive control and resource allocation co-design aren't sufficient—they can only realize their full potential when accompanied by efficient run-time systems for testing, experimental evaluation, and design space exploration. Building such a system is particularly challenging in the presence of run-time adaptation: to automatically adapt to run-time changes, the system needs to perform a broad set of scheduling and resource allocation actions, such as migrate tasks between hardware components, redeploy tasks (e.g., stop some existing tasks and spawn some new tasks), update tasks' parameters, and reconfigure resource allocations. Such actions need to be done promptly (i.e., with minimal delay), efficiently (i.e., without incurring high overhead), and safely (i.e., without breaking control and timing guarantees). Ideally, the run-time system should also be expressive (e.g., easier to specify and implement a broad spectrum of modes and mode transition behaviors) and support composable co-design (e.g., allows for efficient construction of a system by composing mode structures and mode change actions at various levels of granularity). An interesting direction is to develop OS and hypervisor primitives and resource allocation protocols that optimize run-time adaptation actions, as well as corresponding overhead-aware formal analysis. Prior work on run-time systems for multi-mode virtualization and MCP design [2] can serve as initial building blocks in this direction.

III. Summary

This abstract envisions a new co-design paradigm for adaptive CPS on modern hardware, in which adaptive control methods and run-time resource allocations are jointly developed. We have discussed several key challenges and opportunities, as well as potential directions towards this vision. If successful, co-design with run-time adaptation as a first-class citizen will enable more advanced control capability, optimize resources, improving performance while guaranteeing safety.

References

[1] R. Alur. *Principles of cyber-physical systems*. MIT press, 2015.

[2] T. Chen and L. T. X. Phan. SafeMC: A system for the design and evaluation of mode change protocols. In *RTAS*, 2018.

[3] L. T. X. Phan, I. Lee, and O. Sokolsky. Compositional analysis of multi-mode systems. In *ECRTS*, 2010.

[4] L. T. X. Phan, I. Lee, and O. Sokolsky. A semantic framework for mode change protocols. In *RTAS*, 2011.

[5] R. G. Sanfelice. *Analysis and Design of Cyber-Physical Systems: A Hybrid Control Systems Approach*, pages 3–31. CRC Press, 2015.

[6] H. T.A., M. M., and P. V. Assume-guarantee reasoning for hierarchical hybrid systems. In *HSCC*, 2001.

[7] M. Xu, L. T. X. Phan, I. Lee, O. Sokolsky, S. Xi, C. Lu, and C. Gill. Cache-aware compositional analysis of real-time multicore virtualization platforms. In *RTSS*, 2013.