Reactive Scheduling of Computational Resources in Control Systems

Gera Weiss
Ben-Gurion University
Beer-Sheva, Israel
geraw@cs.bgu.ac.il

Hodai Goldman
Ben-Gurion University
Beer-Sheva, Israel
hodaig@cs.bgu.ac.il

ABSTRACT

1. INTRODUCTION

Cyber-physical systems (CPS) technologies of integrating software and control are at the heart of many critical applications (c.f. [14]). These technologies aim at handling issues that emerge when the integration of software and hardware brakes the traditional abstraction layers: when researchers and practitioners are required to consider a unified view that includes both software and hardware. An example of such an issue is the challenge of dynamic assignment of computational resources to software based controllers discussed in, e.g., [3, 21, 23]. While the computation burden required by the control loops can be ignored in many situations, this is not always the case. A main motivating example studied in this paper is vision based control, where computer vision algorithms acquire state information to be used in a feedback loop (c.f. [7, 9, 19]). Unlike conventional sensors such as accelerometers, gyros, compasses, etc., a visual sensor requires significant processing of the acquired image to prepare the state information for feedback. Since typical cyber-physical application, such as robot control, consist of many control loops, responsible for different aspects of the system, that run simultaneously and share the same computational resources, the computer vision algorithms cannot always be invoked in full power. Alternatively, we propose in this paper a mechanism to dynamically trade CPU consumption vs. measurement accuracy so that data acquisition algorithms run in full power only when the control loop requires accurate data.

A main challenge in forming mechanisms for the integration of software and control lies in the design of efficient interfaces for integrating the engineering diciplines involved (c.f. [23]). Components with clearly specified APIs, such as Java library classes, allow designers to build complex systems effectively in many application domains. The key to such modular development is that an individual component can be designed, analyzed, and tested without the knowledge of other components or the underlying computing platform. When the system contains components with real-time requirements, the notion of an interface must include the requirements regarding resources, and existing programming languages provide little support for this. Consequently, current development of real-time embedded software requires significant low-level manual effort for debugging and component assembly (cf. [11, 13, 18]). This has motivated many researchers to develop compositional approaches and interface notions for real-time scheduling (cf. [5, 6, 8, 15–17, 20, 22]).

In this paper we present an approach, a proof-of-concept tool, and a case-study in scheduling computations in embedded control systems. Our approach is based on the automata based scheduling approach, suggested in [1,2,24], where automata are proposed as interfaces that allow the dynamicity and efficiency of desktop operating systems with the predictability of real-time operating systems. The approach allows for components to specify the CPU resources that they need in a way that gives an application agnostic scheduler the freedom to choose schedules at run-time such that the needs of all the components are met, even of components that were added only at run-time. The main contributions of this paper relative to the earlier work in this direction is: (1) We propose an extension of the automata based scheduling frame work that allows to direct the schedule based on the state of the controllers; (2) We propose a technique, based on the theory of Kalman Filters, for designing reactively scheduled controllers; (3) We report on our experience with improving the performance of real-time a vision-based control system (a quadrotor that stabilizes itself in front of a window).

2. ARCHITECTURE: AUTOMATA BASED REACTIVE SCHEDULER

As proposed in earlier work on automata based scheduling (cf. [1,2,24]) we aim at a development process where a system is built as a composition of a set of components where each component is a class in, say C++, accompanied with an automaton. Our addition here is that we allow the automata to be guarded, i.e., the automaton acts as a specification of a reactive system that tells the scheduler which functions of a component it may run in each slot depending on the dynamic state of the controllers.

For scheduling tasks within a slot, we adopt the Logical Execution Time (LET) abstraction [12] that allows deterministic external interface even when the underlying physical execution layer introduces bounded nondeterminism.

The relation between logical and physical task execution is depicted Figure 1 (adopted from [10]). The diagram depicts an execution of one task in one logical execution slot. Logically, the task is released at the beginning of the slot and terminates at the end of it, i.e., the task processes the inputs captured at the release time and its output are only made available to the outside world at the logical termination time. The actual execution, as shown in the figure, may start and finish anywhere in the logical execution slot and the execution may be interrupted by higher priority tasks. However, the external interface remains deterministic provided only that all tasks always finish within logical execution slots.

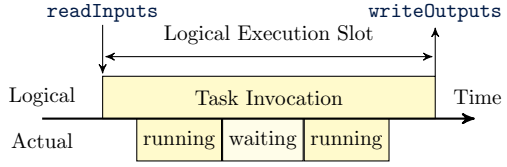

Figure 1: Execution of a task in an execution slot.

Diverging from the way LET is applied in other architectures (c.f. [5, 12]), our architecture executes a dynamic assignment of tasks to logical execution slots. Specifically, the set of tasks that run in each logical execution slot is chosen from the composition of all resource specifications of the components. The scheduler has a complete view of resource requirements and can allocate them accordingly. For example, the system we can react to an increased need of CPU of one of the components by reducing the non real-time computations.

3. SIMULATIONS

4. PROPOSED METHODOLOGY

1. **Controller design:** Based on the separation principle, we propose to design the controller to achieve the control objectives assuming a perfect observation. In practice, this may not be feasible because controller designs such as PID require a system to experiment with. In this case, as demonstrated in the case-study below (see Section 5), we propose to work with one of the observation modes. If the system is close to linear, this should result with a near optimal design.

2. **Observers design:** Specification of sensor modes and observers design

3. **Performance analysis:** Now, we can perform some experiments with the different observer modes and analyze transient behaviors. Specifically, as shown in the case study **??**, we can measure how long it takes for the error to accumulate after switching to a lesser observation mode and formulate how this error affects the control objectives.

4. **Scheduling automata design:** Base on the analysis we can specify the resource scheduling requirements in the form of *specification automaton*. The goal is to design flexible specification that allow dynamic scheduling in order to adapt the environment and the system state, this will improve the system efficient.

5. APPLICATION: STABILIZING A QUADROTOR IN FRONT OF A WINDOW

The case study we used to test our concept is the development of a controller that stabilizes a quadrotor [**?**] in front of a window. We implement an autonomous controller for that task and evaluated its performance.

The part of the controller that we focused on is the vision based position controller. Specifically, the main controller, that we will describe below, uses a standard low-level angular controller and a simple image processing algorithm that identifies the position of the corners of the window in the image plane[1]. Its goal is to regulate the position of the quadrotor by tilting it. Note that rotations of the quadrotor generate a more-or-less proportional accelerations in a corresponding direction. A main challenge for this controller is that the computer vision algorithm takes significant time to compute relative to the fast control loop. We can decrease computation time by lowering the resolution, but this also increase the measurement noise. We will demonstrate how

[1]In the experiment, to simplify the image processing algorithm, we marked the corners of the window with led lights.

adaptive scheduling of the resolution can serve for balancing resource consumption vs. control performance.

5.1 Observer Design

We first implemented an observer based on the work of Efraim at al. [**?**]. The observer gets the positions of the widow corners, enumerated clockwise starting from the top left corner noted by p_1, p_2, p_3, p_4, and extract 4 quantities based on the shape and location of the window corners in the image plane: S_x , s_y , V_d and sz. **center of mass:** S_x and S_y represent the window "center of mass" in the image plane along the image x and y axes, respectively. S_x and S_y are normalized to the range of $[-1, 1]$. S_x is used to measure the roll angle of the window center (for stabilize the roll axis), and S_y is used to measure the altitude of the drone (for stabilizing the throttle). **window size:** sz is the sum of the vertical edges of the window, sz is used to measure the distance of the drone from the window and then to regulate the roll angle [2]. **Vertical difference:** $V_d = \frac{y_1 - y_4 - (y_2 - y_3)}{y_1 - y_4 + (y_2 - y_3)}$, were y_i is the vertical position of p_i in the range of $[-1, 1]$ (0 is the center of the image) first is used to measure the angular position of the drone in relation to the window (θ), then θ and sz are used to calculate the horizontal position parallel to the window surface (x position) of the drone (see Figure **??**).

After measuring the relative position and yaw attitude as usual we add estimator (filter) to get the best state estimation. As shown in the simulation at Section 3 we should use Kalman filter estimator, in our case we have non-linear system... For simplicity, inspiring from linear Kalman filter [**?**], we use two steps estimator that *predict* the current state evolved from the previous state using a linearized model of the system ($\hat{x_{k|k-1}}$) and then *update* the prediction with current state measurement from image explained before, noted by z_k. This is a simplified kalman filter, a known technique as show in [**?**], where The result estimation (noted by $\hat{x}_{k|k}$) is a complementary filter of the prediction ($\hat{x}_{k|k-1}$) and the measurement (z_k): $\hat{x_{k|k}} = K\hat{x}_{k|k-1} + (1-K)z_k$.

As describe in Section **??**, the image processing have few different operation modes, every mode use different image resolution, better image resolution results in more accurate measurement but consume more processing time, changing operation modes allows us to control the trade off between mesurment quality and processing time as shown in Table **??**. The K gain is tuned different for each mode (resolution) for achieving the best estimation in each mode. For simplicity, we examine only two modes, *High quality* with resolution of 960p and *Low quality* with resolution of 240p.

[2]we use fixed size window and convert sz to distance (in meter in our case) based on this window, in the general case the distance is relative to the window size

5.2 Controller Design

We used a simple Proportional Integral Derivative (PID [4]) controller that we tuned by experimenting with the quadrotor in our lab. Based on the separation principle [**?**], we tuned the parameters of our controller only with the highest resolution observer and used them for all the other observers without any modification.

5.3 Analysis and Specification Automata

The objective of the system is to maintain stable hovering in front of the window. Hence, the performances of the system is measured by the amount of deviation from the center line of the window in the critical axis x (see Figure **??**). Our goal is to achieve maximum performance with minimal amount of processing (CPU percentage). The main consumer of computation resources in this system is the heavy observation tasks.

Analyzing the test results shown in Table **??** we see that High quality observation mode provide 0.4 meter error tolerance meaning we can fly this mode in 1.2 meter wide corridor[3], with the cost of 30% CPU usage. With adaptive scheduling specification we lower the CPU usage without significant worsening of High quality performances.

From the Low quality experiment graph we can see that measurement post-fit residual ($\tilde{y}_{k|k}$) is accumulate proportional to the deviation from the center of x axis, and we can use $\tilde{y}_{k|k}$ vales to predict

TODO - position ($x_{k|k}$) derived schedulers
TODO - complex / agregated schedulers
TODO - gyro derived??

5.4 Results

TODO - graphs and tables

6. EXPERIMENT SETUP

The drone is controlled by a Raspberry pi with the navio [**?**,**?**] that runs a modified ArduPilot [**?**] software.

7. CONCLUSIONS

Conclusion goes here.

Acknowledgments

Acknowledgement goes here.

8. REFERENCES

[1] R. Alur and G. Weiss. Regular specifications of resource requirements for embedded control software. In *Proceedings of the 14th IEEE Real-time and Embedded Technology and Applications*, 2008.

[3]Experiments was done with 0.4 meter wide quad-rotor

[2] R. Alur and G. Weiss. RTComposer: a framework for real-time components with scheduling interfaces. In *Proceedings of the 8th ACM international conference on Embedded software*, pages 159–168. ACM, 2008.

[3] K.-E. Arzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, pages 4865–4870. IEEE, 2000.

[4] K. J. Åström and T. Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[5] J. Auerbach, D. F. Bacon, D. T. Iercan, C. M. Kirsch, V. T. Rajan, H. Roeck, and R. Trummer. Java takes flight: time-portable real-time programming with exotasks. In *Proceedings of the conference on Languages, Compilers, and Tools for Embedded Systems*, pages 51–62, 2007.

[6] A. Chakrabarti, L. de Alfaro, T. Henzinger, and M. Stoelinga. Resource interfaces. In *Embedded Software, 3rd International Conference*, LNCS 2855, pages 117–133, 2003.

[7] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor. A vision-based formation control framework. *IEEE transactions on robotics and automation*, 18(5):813–825, 2002.

[8] L. de Alfaro and T. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering (FSE)*, pages 109–120. ACM Press, 2001.

[9] H. Efraim, S. Arogeti, A. Shapiro, and G. Weiss. Vision based output feedback control of micro aerial vehicles in indoor environments. *Journal of Intelligent & Robotic Systems*, 87(1):169–186, Jul 2017.

[10] E. Farcas, C. Farcas, W. Pree, and J. Templ. Transparent distribution of real-time components based on logical execution time. In *Proceedings of the conference on Languages, Compilers, and Tools for Embedded Systems*, pages 31–39, 2005.

[11] T. Henzinger and J. Sifakis. The embedded systems design challenge. In *FM 2006: 14th International Symposium on Formal Methods*, LNCS 4085, pages 1–15, 2006.

[12] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: a time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, 2003.

[13] E. Lee. What's ahead for embedded software. *IEEE Computer*, pages 18–26, September 2000.

[14] E. A. Lee. Cyber physical systems: Design challenges. In *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on*, pages 363–369. IEEE, 2008.

[15] A. Mok and A. Feng. Towards compositionality in real-time resource partitioning based on regularity bounds. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 129–138, 2001.

[16] J. Regehr and J. Stankovic. HLS: A framework for composing soft real-time schedulers. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 3–14, 2001.

[17] A. Sangiovanni-Vincetelli, L. Carloni, F. D. Bernardinis, and M. Sgori. Benefits and challenges for platform-based design. In *Proceedings of the 41th ACM Design Automation Conference*, pages 409–414, 2004.

[18] S. Sastry, J. Sztipanovits, R. Bajcsy, and H. Gill. Modeling and design of embedded software. *Proceedings of the IEEE*, 91(1), 2003.

[19] O. Shakernia, Y. Ma, T. J. Koo, and S. Sastry. Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian journal of control*, 1(3):128–145, 1999.

[20] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *Trans. on Embedded Computing Sys.*, 7(3):1–39, 2008.

[21] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[22] L. Thiele, E. Wanderer, and N. Stoimenov. Real-time interfaces for composing real-time systems. In *Proceedings of the 6th ACM/IEEE International Conference on Embedded Software*, pages 34–43, 2006.

[23] G. Weiss and R. Alur. Automata based interfaces for control and scheduling. In *International Workshop on Hybrid Systems: Computation and Control*, pages 601–613. Springer, 2007.

[24] G. Weiss and R. Alur. Automata based interfaces for control and scheduling. In *Proceedings of the 10th workshop on Hybrid Systems: Computation and Control*, 2007.

APPENDIX

Appendix goes here.