ECRTS 2024 Industrial Challenge Proposal: Elastic Scheduling for ARM AR HUD

Marion Sudvarg, Ao Li, Chris Gill, Ning Zhang
Washington University in St. Louis
(msudvarg, ao, cdgill, zhang.ning)@wustl.edu

June 2024

Abstract

This early stage proposal outlines a proposed solution to the scheduling problems presented by the AR HUD application from ARM's challenge to the ECRTS community. We suggest that the dataflow application can be modeled as a collection of tasks with harmonic period relationships reflecting the flow of data among them. By extending our prior work on elastic scheduling of dataflow applications with harmonic period constraints, which proved effective when applied to ORB-SLAM3 in resource-constrained environments, we believe that we can produce a period adaptation model to guarantee schedulability of the AR HUD's tasks even in dynamic environments and systems.

1 Introduction

Motivation: This short paper serves as an early stage proposal for the industrial challenge at the 36th Euromicro Conference on Real-Time Systems (ECRTS 2024). This year's industrial challenge was presented by ARM during ECRTS 2022 [1] and relates to an automotive augmented reality heads-up-display (AR HUD) to be projected in real time into the driver's field of view.

Challenge Background: The main aspects of the challenge are analysis of the worst-case execution times of each application task and optimization of the application to meet its desired real-time performance requirements, to include data-flow analysis, design of new scheduling policies, resource mapping, and performance isolation to prevent interference due to shared resource contention. This proposal focuses on **scheduling policy design**. It assumes that methodologies to characterize execution times are already well established.

The AR HUD implementation presented by ARM is illustrated in [1, Figure 2], which we reproduce here as Figure 1. It will be deployed on a system on chip (SoC) with a multicore CPU, a GPU, and possibly a dedicated neural processing unit (NPU). The software application uses OV^{2}SLAM [8] to determine the vehicle's orientation and trajectory and to generate a map of its surroundings. This is used to generate the projected images, which are oriented according to an estimate of the driver's head pose.

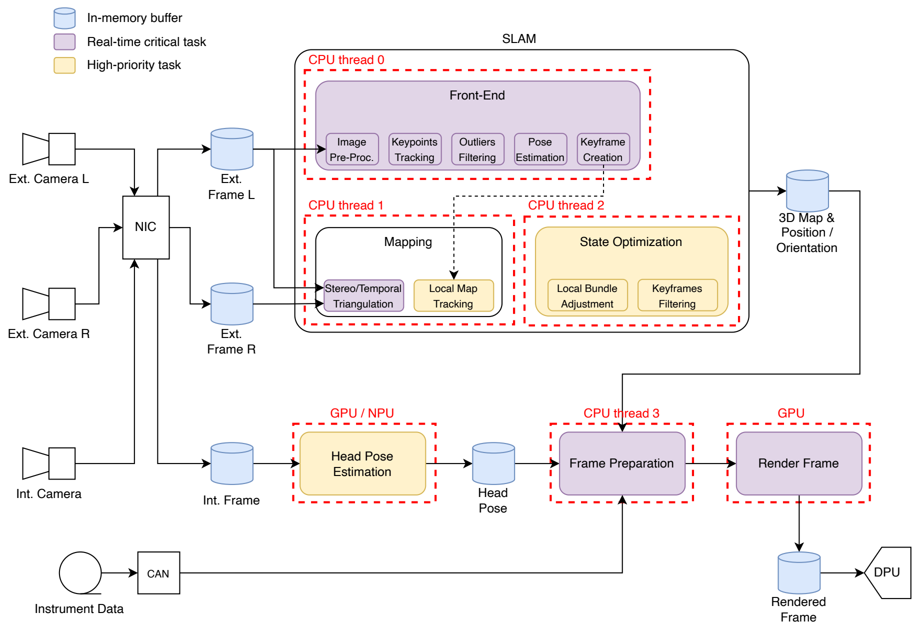

Figure 1: AR HUD software system design [1, Figure 2].

ARM's proposed solution decomposes the application into multiple critical and high-priority tasks, which are mapped onto concurrent CPU threads. The SLAM front-end tasks and stereo/temporal triangulation are considered critical, and must run at the same rate as camera frame acquisition. On the other hand, local map tracking is high-priority but non-critical, as jobs can be aborted if a new keyframe becomes available before the previous one is processed; however, it is handled by the same thread as the critical triangulation task. Another thread performs bundle adjustment and filters redundant keyframes; these are also considered non-critical, as they are optimization tasks. Another thread runs the critical task of preparing the image frame to project on the HUD.

The challenge, then, is to construct a scheduling policy that enables the desired timing semantics of the application to be met: critical tasks should complete according to the specified framerate, while high-priority tasks should also run when possible. Guarantees should be maintained even in dynamic environments where workloads might be dependent on external events or inputs (e.g., in crowded, dense urban driving scenarios). Simply prioritizing based on criticality may result in poor system utilization [14, 4, 2]; the scheduling policy should be aware of such pitfalls, and allow (when possible) concurrent execution of general-purpose or background tasks.

Our Proposal: Buttazzo's elastic real-time scheduling model provides a framework for dynamic task adaptation to guarantee schedulability even on a system that becomes overloaded [5, 6]. In recent work presented at RTAS, we proposed a new extension of elastic scheduling to systems that constrain task periods to be harmonic, then demonstrated an application of this model to two dataflow applications [13]. Our framework allows ORB-SLAM3 [7], running atop ROS2 [12], to adjust task periods in response to changes in available CPU utilization. Compared to a baseline implementation where jobs may be dropped if they overrun their deadlines, or even an integrated ML-based implementation that intelligently drops input frames and keyframes in response to overload, our elastic scheduling framework produced more accurate maps of the environment.

We propose to extend our model to address ARM's challenge. The dataflow semantics of the AR HUD application pipeline can be reflected by harmonic period relationships among its separate tasks. For example, the SLAM front-end must execute at the same rate as frame acquisition; the local map tracking task's period should be some integer multiple of the front-end's. Given a fast enough CPU, all tasks can execute at their maximum desired rates (e.g., 60 Hz, equivalent to the fastest possible camera frame acquisition rate). However, if the system cannot sustain this (e.g., due to complex environments or interference tasks), task periods and the harmonic relationships among them can be adjusted. For example, the front-end's period can be increased, and the map tracking task's period can be set to double that of the front-end. This means that only every other frame is processed as a keyframe, but this keeps timing deterministic, and it avoids wasted execution due to dropping a partially-executed keyframe if a new one arrives.

Open Challenges: Applying our model to ARM's AR HUD application imposes new challenges not addressed in our prior work in [13]. Chief among these are an extension of the harmonic elastic scheduling model to *multiprocessor* scheduling; our work considered only scheduling on a uniprocessor system. We also need to consider the implications of dynamic execution times. The problem of period assignment under the harmonic elastic scheduling model is NP-hard, but we demonstrated an efficient algorithm for online adaptation in response to changes in available CPU utilization. An algorithm that can respond to both dynamic resource availability *and* execution times is needed for this challenge. We discuss these further in Section 3.

2 Background

Buttazzo's elastic model for implicit-deadline tasks on a uniprocessor [5, 6] characterizes each task $\tau_i=(C_i, T_i^{\min}, T_i^{\max}, T_i, E_i)$ by five non-negative parameters: C_i (the task's worst-case execution time); $T_i^{\min}$ (the task's minimum period, i.e., its nominal value when executing at the desired rate in an uncompressed state); $T_i^{\max}$ (its maximum period, beyond which correct behavior is not maintained); T_i (the task's assigned period, $T_i^{\min} \leq T_i \leq T_i^{\max}$); and E_i (a constant representing "the flexibility of the task to vary its utilization" [5]). From these parameters are derived the corresponding values $U_i{=}C_i/T_i$, $U_i^{\min}{=}C_i/T_i^{\max}$, and $U_i^{\max}{=}C_i/T_i^{\min}$.

Chantem et al. formulated elastic scheduling as a constrained optimization problem. If a system is overloaded, then task utilizations are reduced ("compressed") by assigning them the values that satisfy the following:

$$\min_{U_i} \quad \sum_{i=1}^{n} \frac{1}{E_i}(U_i^{\max} - U_i)^2 \tag{1a}$$

$$\text{s.t.} \quad \sum_{i} U_i \leq U_D \quad \text{and} \tag{1b}$$

$$\forall_i, \quad U_i^{\min} \leq U_i \leq U_i^{\max} \tag{1c}$$

where U_D is the schedulable utilization bound of the system. In [13], we considered the case that periods must also remain harmonic, adding the following additional constraint:

$$\forall_{i,j}, \quad T_i/T_j \in \mathbb{N}^+ \text{ or } T_j/T_i \in \mathbb{N}^+ \tag{2}$$

We also proposed a variation of the problem, **"the ordered harmonic elastic problem"**, where task periods must be selected to respect some total ordering assigned a priori. This is a natural restriction in several applications, including in the dataflow pipelines found in many SLAM systems.

Though the optimization problem given by Expressions 1 and 2 is NP-hard in general, we demonstrated that offline construction of a lookup table enables polynomial-time reassignment of task periods satisfying the ordered harmonic elastic problem in response to online changes in available CPU utilization. We therefore propose a similar approach for ARM's AR HUD application, though this will require extensions to our prior model. The next section outlines the proposed approach.

3 Elastic Scheduling Model

In this proposal, we focus only on CPU-based scheduling and execution. In a full system, those tasks on the GPU/NPU must be considered as well, but if they have dedicated resources (i.e., if head pose estimation has exclusive access to an NPU, and frame rendering to a GPU), these do not contribute to the challenge of scheduling.

3.1 Quantify period ranges and characterize relationships

We begin by identifying the acceptable period ranges for each task, as well as the relationships among those periods imposed by the dataflow semantics of the application. We work from the outside in by considering the camera framerate as well as the requirements for visually smooth frame rendering.

Per the challenge details in [1], frame acquisition is expected be performed in the range 30–60 frames per second; the SLAM front-end task should therefore be assigned a period in the range 17–33ms. The minimum periods of all other tasks should also be 17ms: ideally, and without resource constraints, all tasks should be invoked for every camera frame at 60 fps.

The maximum period of frame preparation should be based on the lowest framerate that still delivers a sufficiently smooth experience to the driver (e.g., 100ms). Characterizing maximum acceptable periods for stereo/temporal triangulation, map tracking, bundle adjustment, and keyframe filtering require domain knowledge and understanding of the algorithms used, although empirical studies may be useful in finding an upper bound that still produces useful results. For example, in our studies of ORB-SLAM3, we found that map accuracy rapidly increased for image processing periods above 200ms [13, Fig. 7]. For illustrative purposes, we reproduce the result in Figure 2.

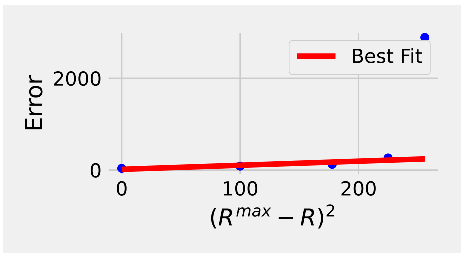

Figure 2: Upper-bounding image processing period for ORB-SLAM3 [13, Fig. 7].

An ordering may also be imposed on task periods based on system requirements. Again, domain knowledge is required here, but we propose to constrain the ordering as follows:

$$\text{Front-End} \leq \text{Triangulation} \leq \text{Frame Prep.} \leq \text{Tracking} \leq \text{Bundle Adj.} \leq \text{Keyframe Filter}$$

3.2 Identify Elastic Constants

Though the elastic constant E_i assigned to each task τ_i is intended to represent "the flexibility of the task to vary its utilization" [5], the elastic objective minimized in Expression 1a gives rise to a more quantitative notion of elasticity based on the first-order error induced by individually decreasing the rate of each task from its fastest desired value. We introduced this idea in [13] and demonstrated that it is effective in assigning elastic constants for the tasks that compose ORB-SLAM3 (see Figure 2 as an example).

This assumes, however, a metric by which application outcome can be quantified. For SLAM in isolation, we used the mean translational error of the produced map as our error metric in [13]. A similar notion may be applied to the AR HUD, e.g., error in predicted vehicle orientation and trajectory, or error in the map of its surroundings, might define the loss against which we measure elastic constants. However, there is also a penalty in terms of driver experience associated with decreasing the framerate of the display; incorporating human notions of "importance" will require further input from the stakeholders of this challenge.

3.3 Scheduling

We suggest that each CPU-based task should be assigned its own thread. This allows for independent scheduling, and also reduces potential issues related to mixed-criticality scheduling (e.g., whereby a critical task needs to execute, but its thread is busy with a non-critical workload).

On a single processor, tasks can be scheduled according to a fixed-priority rate monotonic scheme or with EDF; since periods are harmonic, both achieve a utilization bound of 1. Under the assumption that task execution times have already been characterized (for this proposal, we are *not* addressing this aspect of the challenge), utilizations corresponding to the minimum periods can be computed. If the total utilization exceeds the bound, our techniques in [13] can assign harmonic periods that optimize the elastic objective (Expression 1a) within the constraints of schedulability.

However, effective multiprocessor scheduling is necessary to realize this application. Under the fluid scheduling model [3], each task is assigned a fraction of a processor at each instance in time. It is a convenient abstraction where the schedulable utilization bound U_D is equal to the number of processor cores. However, it often remains impractical in real systems [11]. A more practical solution is global EDF scheduling, where at any instant in time, those jobs with the earliest *absolute* deadlines are selected for execution. Goossens et al. showed [9, Theorem 5] that a set Γ of implicit-deadline tasks is schedulable on m processors if:

$$\sum_{\tau_i \in \Gamma} U_i \leq m - (m-1) \cdot \max_{\tau_i \in \Gamma} \{U_i\} \tag{3}$$

Our approach in [13] for solving the problem on a uniprocessor naturally extends to this schedulability condition. The approach involves enumerating all possible sequences of integer multiples among task periods, given their acceptable ranges. As a simple example, if task τ_1 can take periods in the range $[2,3]$, τ_2 can take periods in the range $[4,5]$, and τ_3 can take periods in the range $[5,8]$ then $\{1,1\}$ and $\{1,2\}$ are the possible sequences of multipliers. For a given utilization bound, the period assignment that arises from each sequence (if one exists) for that bound is input to Expression 1a; the assignment that minimizes that objective is then used.

Since the task with the maximum utilization does not change for a given sequence of multipliers, the approach only needs to be modified slightly for the schedulability condition in Expression 3. Rather than a fixed utilization bound, the utilization bound is recomputed for each sequence for the given number of available cores.

3.4 Online Adaptation

Our goal is not merely to assign periods to tasks offline to produce a schedulable system. Rather, we would like to use elastic scheduling to enable *online* task adaptation. This may be necessary under two types of scenarios.

1. Changes in Available Resources: The amount of CPU utilization available to the AR HUD application's tasks may change for a number of reasons. For example, a CPU core might go offline, CPU speed may throttle down in DVFS-enabled systems, or other tasks (e.g., the aggressor workloads mentioned in the challenge [1] or background execution) may consume CPU time.

In these cases, it should be straightforward under the uniprocessor, fluid, and global EDF scheduling paradigms to adapt task periods efficiently. Extending our techniques in [13] to fluid and global EDF should be straightforward, as we argue above; nonetheless, efforts to formalize these algorithms are ongoing.

2. Changes in Execution Time: Individual task execution times may change for several reasons. Workloads may depend on the environment; for example, keypoint tracking may require more computation in crowded driving environments where more object features are detected. Workloads may also vary with shared resource contention; e.g., interference with access to caches, the memory bus and DRAM row buffers, I/O subsystems, and OS queues can increase execution times significantly. Existing tools can help to quantify the extent to which such interference is possible, e.g., our own attack-based tool PolyRhythm [10].

The algorithms we have developed and proposed for polynomial-time task adaptation assume that task execution times remain fixed. If execution times change, the lookup tables used may have to be regenerated. We found that this can be done in under 10ms for systems of up to 8 elastic tasks [13]; since the AR HUD application only has 6 CPU-based elastic tasks, this may still be feasible online. However, further evaluation is needed.

4 Conclusion

In this early stage proposal, we have suggested applying our harmonic elastic task model from [1] to the ARM AU HUD application. We look forward to discussing these ideas with the stakeholders at ARM and the broader real-time systems community. If these ideas have merit, we will develop the scheduling algorithms more formally, and apply and evaluate them in the software packages that ARM has provided.

References

[1] Matteo Andreozzi, Giacomo Gabrielli, Balaji Venu, and Giacomo Travaglini. Industrial Challenge 2022: A High-Performance Real-Time Case Study on Arm. In Martina Maggio, editor, *34th Euromicro Conference*

on Real-Time Systems (ECRTS 2022), volume 231 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[2] S. Baruah, V. Bonifaci, G. DAngelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *2012 24th Euromicro Conference on Real-Time Systems*, pages 145–154, 2012.

[3] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, Jun 1996.

[4] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo D'Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Nicole Megow, and Leen Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, 61(8):1140–1152, 2012.

[5] Giorgio C. Buttazzo, Giuseppe Lipari, and Luca Abeni. Elastic Task Model for Adaptive Rate Control. In *IEEE Real-Time Systems Symposium*, 1998.

[6] Giorgio C. Buttazzo, Giuseppe Lipari, Marco Caccamo, and Luca Abeni. Elastic Scheduling for Flexible Workload Management. *IEEE Transactions on Computers*, 51(3):289–302, March 2002.

[7] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[8] Maxime Ferrera, Alexandre Eudes, Julien Moras, Martial Sanfourche, and Guy Le Besnerais. OV^{2}SLAM : A fully online and versatile visual SLAM for real-time applications. *CoRR*, abs/2102.04060, 2021.

[9] Joël Goossens, Shelby Funk, and Sanjoy Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Systems*, 25(2):187–205, Sep 2003.

[10] Ao Li, Marion Sudvarg, Han Liu, Zhiyuan Yu, Chris Gill, and Ning Zhang. Polyrhythm: Adaptive tuning of a multi-channel attack template for timing interference. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 225–239. IEEE, 2022.

[11] James Orr and Sanjoy Baruah. Multiprocessor scheduling of elastic tasks. In *Proc. of 27th International Conference on Real-Time Networks and Systems*, pages 133–142. ACM, 2019.

[12] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[13] Marion Sudvarg, Ao Li, Daisy Wang, Sanjoy Baruah, Jeremy Buhler, Chris Gill, Ning Zhang, and Pontus Ekberg. Elastic scheduling for harmonic task systems. In *Proceedings of the 30th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'24)*. IEEE Computer Society Press., 2024.

[14] Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *28th IEEE international real-time systems symposium (RTSS 2007)*, pages 239–243. IEEE, 2007.