The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
21-23 September 2017, Bucharest, Romania

Constrained Mixed-Critical Parallelization for Distributed Heterogeneous Systems

Robert Höttger, Mustafa Özcelikörs, Lukas Krawczyk, Philipp Heisig, Carsten Wolff, Burkhard Igel
IDiAL Institute - Dortmund University of Applied Sciences and Arts,
{robert.hoettger, mustafa.ozcelikors, lukas.krawczyk, philipp.heisig, carsten.wolff, igel}@fh-dortmund.de
www.idial.institute

***Abstract* – Distributing software effectively to multi-core, many-core, and distributed systems has been studied for decades and still advances successively driven by domain specific constraints. Programming vehicle ECUs is one of the most constrained domains that just recently approached the need for concurrency due to advanced driver assistant systems or autonomous driving approaches. In this paper, various challenges for such systems are outlined, discussed, and solutions are given upon instruction precise modeling, affinity constrained distribution, and effective software parallelization. The solutions are compared upon bare-metal and OS based implementations while considering fixed priorities for sequential, OS based, and APP4MC scheduling. The latter case has been published at ~~[1]~~ and evolved to consider affinity constraints, SWC-based partitioning and communication cost related mapping. Results show that using APP4MC based distributions on a distributed heterogeneous system outperforms available approaches for mixed-critical applications.**

***Keywords* – Constrained parallelization, heterogeneous systems, distributed systems, multicore, parallel embedded systems**

I. Introduction

Especially the automotive domain requires lots of constraints originating from different safety, security, timing, or similar requirements. The verification, validation, testing, and simulation stack requires dozens of tools and the consideration of established architectures, standards, models, and assessments to address product line supporting, consistent, modular, and scalable software. However, these efforts lack in transparency likewise the comprehensive understanding of applications. Recent approaches such as APP4MC [2] already address this challenge and try to provide a common adaptable platform based on AUTOSAR [3] ~~while providing a standardized data model~~. Any specific commercial or proprietary tooling is supposed to be integrable in order to provide seamless interaction with provided tooling such as product line management, requirements engineering, partitioning, mapping, testing, and more.

~~In this paper, we use the open source APP4MC environment in order to address both industrial and research related models while evaluating our new developments not only regarding model results but also to validate result among a specific use case described in section III~~. The RC-Car is an advanced demonstrator that features 16 100MHz logical cores (XMOS ExplorerKit, XCore-200) combined with four cores clocked at 1200MHz (Raspberry Pi 3, ARM CoreTex-A53). With using APP4MC, we show that automatic OS driven distributions not always create the most efficient results.

The further remainder of this paper is structured as follows: Section II roughly describes the environment this contribution is based upon and its specifics among constraints and instructions. Afterwards, Section III illustrates capabilities and properties of the RC-Car demonstrator as well as its distributed architecture and concurrency capabilities. Subsequently, Section IV evaluates the results and benefits gained from using APP4MC for an automatic, constrained based parallelization of a mixed critical application. Finally, Section V concludes this paper's contributions and discusses benefits, disadvantages, possible extensions, and promising outlooks that potentially advance the current approach.

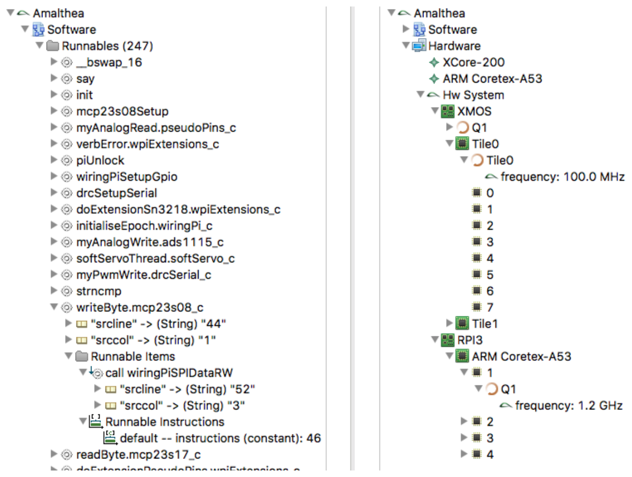
II. APP4MC

APP4MC is an open source project for developing multi- and many-core systems primarily for the automotive domain. Instead of describing its different purposes and features, the focus of this section is on the required steps to utilize APP4MC parallelization capabilities [1], i.e. partitioning and mapping. Those processes have been advanced in recent years to consider affinity, allocation, and safety constraints among others. Dealing with safety constraints ensures that tasks with a specific Automotive Safety Integrity Level (ASIL) are allocated to processing units (cores) featuring the corresponding ASIL level. Allocation constraints are considered the same way, except that instead of ASIL levels properties such as Floating Point Unit (FPU) connectivity, cache memory size, processing frequency or others are investigated. Affinity constraints are evaluated within the partitioning process

As part of the AMALTHEA4public project, this work has been funded by the German Federal Ministry of Education and Research - BMBF, under funding no. $01|S14029K$

since runnables, i.e. smallest executable units, have to be kept in the same partition that are later on resolved to tasks. This paper's application, i.e. the RC-Car, features all of those constraints. Affinity has to be kept for instance regarding the light system state task and the Pulse Width Modulation (PWM) signal generation (timer) tasks since their distribution would cause implementation and communication overheads. Since there are tasks that continuously check the core utilization on each X-Core Tile, those tasks feature an allocation constraint to each X-Core Tile. Finally, due to the fact that the RC-Car performs image processing from a camera as well as real time tasks for motor control, steering, and bluetooth based communications, we define all tasks that are crucial for safe driving as being safety critical (safety constraint) and bind them to the real time capable XMOS board.

Apart from the constraints, software and hardware descriptions are required. While the hardware model is created manually, a C/CPP parser was used to receive necessary AMALTHEA software model data, i.e. runnables, labels, label accesses, activations, runnable calls, Interrupts Service Routines (ISRs), Semaphores, and more for the Raspberry Pi (RPI) implementations. Excerpts of those models are shown in Figure 1. While the RPI already features 247 Runnables and more, XMOS implementations were modeled on a higher level upon threads only, ~~that correspond tasks. Those count 30 in total where profiling was performed via counting assembler instructions and using XXX for profiling.~~



```
Amalthea
  Software
    Runnables (247)
      __bswap_16
      say
      init
      mcp23s08Setup
      myAnalogRead.pseudoPins_c
      verbError.wpiExtensions_c
      piUnlock
      wiringPiSetupGpio
      drcSetupSerial
      doExtensionSn3218.wpiExtensions_c
      initialiseEpoch.wiringPi_c
      myAnalogWrite.ads1115_c
      softServoThread.softServo_c
      myPwmWrite.drcSerial_c
      strncmp
      writeByte.mcp23s08_c
        "srcline" -> (String) "44"
        "srccol" -> (String) "1"
        Runnable Items
          call wiringPiSPIDataRW
            "srcline" -> (String) "52"
            "srccol" -> (String) "3"
          Runnable Instructions
            default -- instructions (constant): 46
      readByte.mcp23s17_c
```

```
Amalthea
  Software
  Hardware
    XCore-200
    ARM Coretex-A53
    Hw System
      XMOS
        Q1
        Tile0
          Tile0
            frequency: 100.0 MHz
          0
          1
          2
          3
          4
          5
          6
          7
        Tile1
      RPI3
        ARM Coretex-A53
          1
            Q1
              frequency: 1.2 GHz
          2
          3
          4
```

Figure 1. AMALTHEA software and hardware models (excerpts) for the RC-Car

The created scripts or applications need to be modeled precisely with APP4MC in order to get more accurate results. The Linux OS provides a set of important kernel-level and development tools for processes, threads, and binaries that allow us to model and manage systems. Making use of Linux OS tools has been an important part of the whole development process while modeling with APP4MC.

The first step that is needed in order to properly model the runnables and tasks is getting the proper number of instructions by either looking at disassembly of binary objects or using profiling tools. In order to profile the dynamic processes, ~~which are one of the greatest parts in our application,~~ the tool '~~perf'~~ has been used. ~~A linux system bash script has been created in order to get the dynamical instructions by making use of this tool.~~ One could also make use of the kernel folders in Linux such as 'proc' in order to define constraints for the processes while modeling them in APP4MC. Furthermore, the runnables within C/C++ applications are disassembled and analyzed by using the tool '~~objdump',~~ which creates disassembled runnables from binary files.

Linux provides other tools such as 'top' and 'taskset' which are also crucial to the development of multi-core systems. The tool 'top' is used for the process monitoring, whereas the tool 'taskset' is used for core pinning/mapping to manage the multi-core distribution. In the RCCAR, the aforementioned tools are used in order to manage a multi-process system distributed on multiple cores. The mapping on the RCCAR is done automatically by a created Python script which parses an included text file and makes the distribution using Linux's 'taskset'.

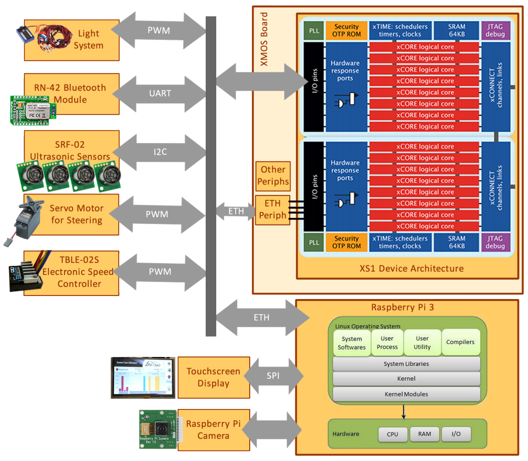
Before the partitioning and mapping processes can be configured and executed, one of the most challenging key property has to be added to runnables i.e. the above mentioned affinity, allocation, and safety constraints.

III. RC-Car

The current ~~Raspberry Pi distribution~~ includes processes for the touchscreen, ethernet communication, core utilization reader, mjpg streamer, vnc application, apache2, OpenCV [4] image processing, and additional cyclewaster. The latter processes can be added in order to push the core utilization to a maximum and consider high workload. The effective distribution provided by APP4MC is here crucial to provide deadline violation free program execution.

The XMOS tasks focus on necessary real time implementations. Scheduling on the XMOS is non preemptive and allows a better determinism as well as easier Worst Case Execution Time (WCET) reasoning due to less context switching and less scheduling jitter [5]. However, since our number of tasks exceed the total number of available cores on the XMOS, we had to define high priority tasks, that run on dedicated cores, and low priority tasks that run concurrently with other low level tasks in cooperative multitask manner on common cores (via *combinable* functions in XMOS terminology). The amount of high priority tasks thereby defines the number of cores available for the low priority tasks.

Figure 2 shows the basic RC-Car architecture that uses different communication APIs such as Ethernet, I2C, or UART and various actuators and sensors to interact with its environment. Its current application is to approach autonomous driving scenarios.



~~Figure 2. RC-Car architecture, interfaces, sensors, and actuators~~

IV. Evaluation and Related Work

The following chart (FIgure 3) shows results of three different software distribution scenarios on the RC-Car obtained from separate utilization tracker threads.
For clarity purposes, the XMOS logical cores were combined to the corresponding tiles since their separation would result in less clarity as well as lessen the physical accordance. Hence, each color on a XMOS tile can consume up to 12,5% maximal tile utilization, such that a tile is only fully utilized, if all 8 cores fully express 12,5% core utilization. The distributed sequential parallelization (left part of Figure 3) shows that each board (RPI & XMOS) has one fully utilized processing unit. Consequently, applications and tasks can hardly meet their deadlines and execute with very low throughput. In contrast to that, when using the Linux Completely Fair Queuing (CFQ) scheduler without explicitly defining priority or scheduling classes on the RPI as well as the round robin scheduler on the XMOS via `par` statements, one can experience great benefits due to concurrent progressing on both hardware units (see mid part of Figure 3). Nevertheless, for this case programmers still need to manually separate the real time applications from the non-critical programs and map them to the corresponding processing elements (critical programs to the XMOS, non-critical to the RPI). It is important to note here, that the non-critical tasks still have deadlines to meet in order to assess effectiveness i.e. that results are produced within a limited time period predefined upon requirements.

However, as pointed out earlier in this paper, the automatic distributions are not always the prior choice when targeting efficiency. Deadlines may be met comprehensively (the effect is the same), but long slack times waste processing cycles that could be spent otherwise. The APP4MC distribution (right part of Figure 3) not only considers any types of constraints but also can be configured to balance load as much as possible such that e.g. the total frequency (here on the RPI) could be scaled down to safe energy. Assuming a global Dynamic Voltage and Frequency Scaling (DVFS) feature on the RPI, we measured up to 9% less energy consumption when commonly reducing the CPU frequencies from 1200MHz to 800MHz while still meeting all deadlines.
[6] [7] [8] [9]

V. ~~Conclusion~~

We successfully validated that using APP4MC for efficient mixed-critical software parallelization to distributed heterogeneous systems eases development processes on various levels. Error prone and time consuming manual software distribution can be avoided and utilizing APP4MC results even outperforms OS or compiler based automatic approaches. While the constraint modeling requires certain efforts, it considers crucial requirements emerging from timing, safety, or security demands that are inevitable in modern industrial developments. Our experiments among an

Acknowledgment

The authors would like to express their appreciation to the AMALTHEA4public consortium for sharing experience, knowledge, and know-how. Special thanks to

References

[1] R. Höttger, L. Krawczyk, and B. Igel, “Model-Based Automotive Partitioning and Mapping for Embedded Multicore Systems,” in *International Conference on Parallel, Distributed Systems and Software Engineering*, ser. ICPDSSE’15, vol. 2. World Academy of Science, Engineering and Technology, 2015, pp. 2643–2649.
[2] APP4MC consortium and contributors, “Application Platform Project for Multi Core,” 2017. [Online]. Available: http://eclip.se/a8
[3] AUTOSAR consortium, “Automotive Open System Architecture,” 2017. [Online]. Available: http://www.autosar.org
[4] OpenCV consortium and contributors, “Open Source Computer Vision Library - A Common Infrastructure for Computer Vision Applications,” 2017. [Online]. Available: http://opencv.org
[5] XMOS Ltd., “xCORE Achitecture,” 2017. [Online]. Available: https://www.xmos.com/download/private/xCORE-Architecture-Flyer%281.1%29.pdf
[6] A. Liang, L. Xiao, Y. Li, Z. Zhang, and L. Ruan, “Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Power-scalable Cluster,” in *International Conference on High Performance Computing and Communications*. IEEE, 2013, pp. 6–11.
[7] D. Socci, P. Poplavko, S. Bensalem, and M. Bozga, “Multiprocessor Scheduling of Precedence-constrained Mixed-Critical Jobs,” in *18th International Symposium on Real-Time Distributed Computing*. IEEE, 2015, pp. 198–207.
[8] A. Kritikakou, C. Pagetti, O. Baldellon, M. Roy, and C. Rochange, “Run-time Control to Increase Task Parallelism in Mixed-Critical Systems,” in *26th Euromicro Conference on Real-Time Systems*. IEEE, 2014, pp. 119–128.
[9] X. Xiao, G. Xie, R. Li, and K. Li, “Minimizing Schedule Length of Energy Consumption Constrained Parallel Applications on Heterogeneous Distributed Systems,” in *TrustCom-/BigDataSE/-ISPA*. IEEE, 2016, pp. 1471–1476.

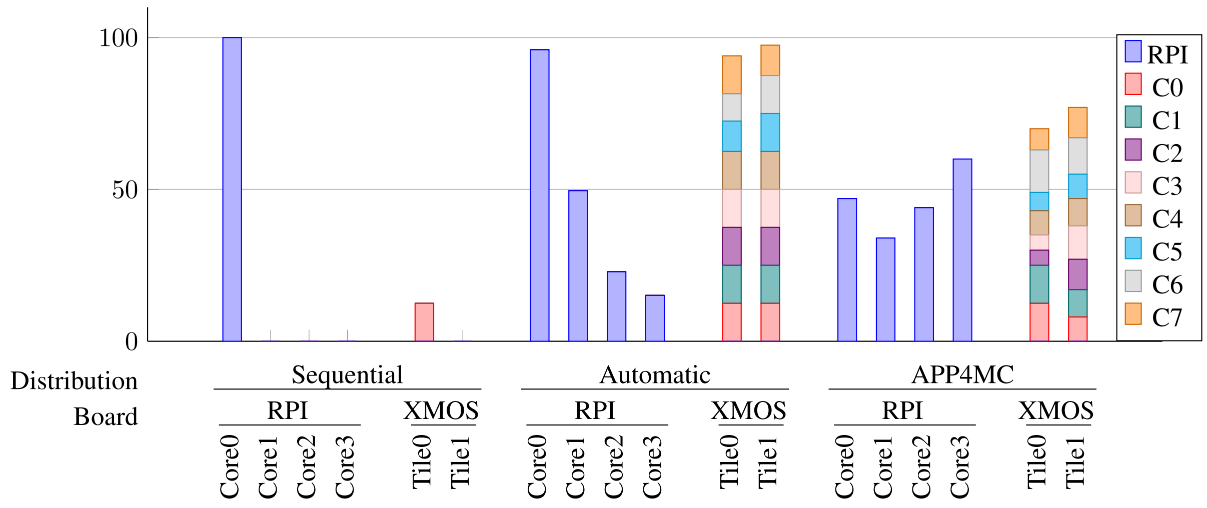


Figure 3. Core utilization in % (y-axis) for sequentail, automatic, and APP4MC distribution