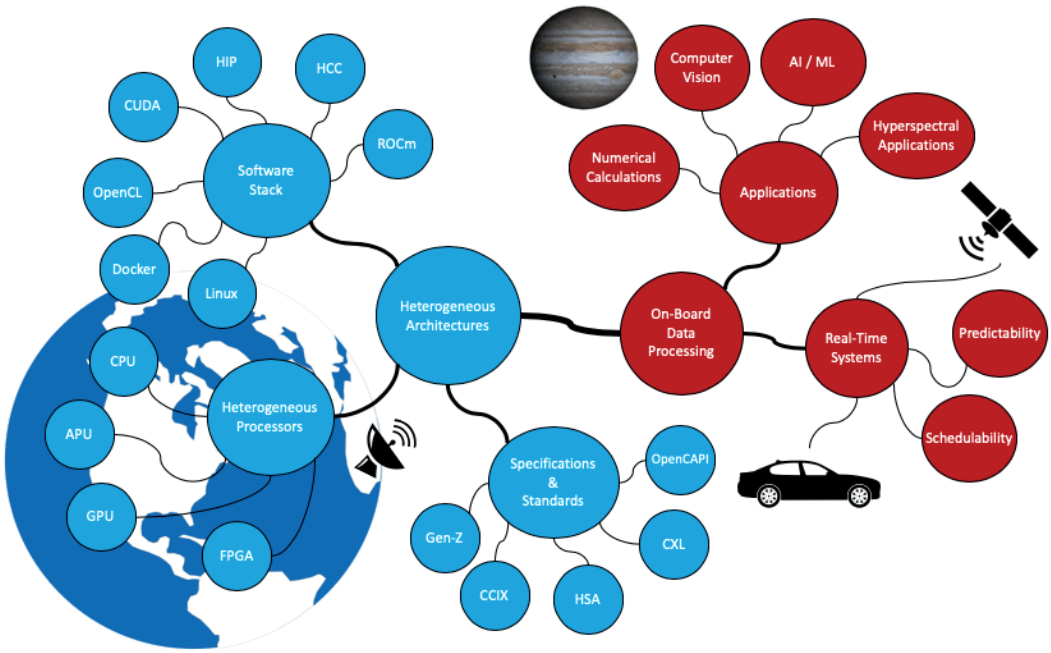
Mälardalen University Licentiate Thesis 286

Improving On-Board Data Processing using CPU-GPU Heterogeneous Architectures for Real-Time Systems

Nandinbaatar Tsog



MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Licentiate Theses
No. 286

IMPROVING ON-BOARD DATA PROCESSING USING CPU-GPU HETEROGENEOUS ARCHITECTURES FOR REAL-TIME SYSTEMS

Nandinbaatar Tsog

2019

School of Innovation, Design and Engineering

ISBN 978-91-7485-450-3
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

Abstract

This thesis investigates the efficacy of heterogeneous computing architectures in real-time systems. The goals of the thesis are twofold. First, to investigate various characteristics of the Heterogeneous System Architectures (HSA) compliant reference platforms focusing on computing performance and power consumption. The investigation is focused on the new technologies that could boost on-board data processing systems in satellites and spacecraft. Second, to enhance the usage of the heterogeneous processing units by introducing a technique for static allocation of parallel segments of tasks.

The investigation and experimental evaluation show that our method of GPU allocation for the parallel segments of tasks is more energy efficient compared to any other studied allocation. The investigation is conducted under different types of environments, such as process-level isolated environment, different software stacks, including kernels, and various task set scenarios. The evaluation results indicate that a balanced use of heterogeneous processing units (CPU and GPU) could improve schedulability of task sets up to 90% with the proposed allocation technique.

Sammanfattning

Denna avhandling undersöker effektiviteten hos heterogena datorarkitekturer i realtidssystem. Målet med avhandlingen är tvåfaldigt. Till att börja med, att undersöka olika egenskaper hos plattformar baserade på Heterogeneous System Architecture, med fokus på datorprestanda och strömförbrukning. Undersökningen är inriktad på tekniker som kan öka datorbehandlingssystemen ombord i satelliter och rymdskepp. För det andra förbättra användningen av heterogena arkitekturer genom att införa en teknik för statisk allokering av parallella programsegment.

Undersökningen och den experimentella utvärderingen visar att vår metod för effektiv användning av GPU-allokering för parallella programsegment är den mest energieffektiva jämfört med någon annan studerad allokering. Undersökningarna har genomförts i olika typer av miljöer, såsom processisolerad miljö, olika mjukvarustackar, inklusive kernel, och olika uppsättningsscenarier. Utvärderingsresultaten indikerar dessutom att en balanserad användning av heterogena beräkningsenheter (CPU och GPU) kan förbättra schemaläggningen för vissa program upp till 90% jämfört med de tidigare föreslagna allokeringsteknikerna.

Where there's a will there's a way

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors, my principal supervisor Professor Mikael Sjödin, industrial co-supervisor, Adjunct Professor Fredrik Bruhn, academic co-supervisors Professor Moris Behnam and Associate Professor Saad Mubeen, and co-authors (non-official supervisors) Dr. Matthias Becker and Dr. Harris Gasparakis, for working *parallel* with me, *executing* different kernels, applications, discussing *heterogeneous* ideas for these years.

I deeply appreciate my teachers, Sharaa, Munkhjargal, Erdene Natsagdorj, Professor Motomu Takeshige, Professor Shimizu, and Professor Yasushi Kato. Your words supported me to coming back to the academia.

Special thanks to my friends and advisors, Jakob Danielsson, Marcus Larsson, Tobias Andersson, Uyanga Ganbaatar, Batbuyan Batchuluun, Dr. Guillermo Rodriguez-Navas, Dr. Predrag Filipovikj, Filip Markovic, Ashalatha Kunnappilly, Mirgita Frasheri, Dr. Gabriel Campeanu, Leo Hatvani, Dr. Momo, Prof. Micke, Dr. Sara Abbaspour, Stefan Karlsson, Tugu, Batya, Mitsuteru Kaneoka, Koji Yamaguchi, and Prof. Ryu Funase. Your words, cheered me up a lot.

```
if ($id =~ /([a-z][a-z][a-z][0-9][0-9])/) {
    print <<MDH;
    To $1,
    Thank you for spending time with me at MDH.
    MDH
}
```

I appreciate the DPAC project for funding my doctoral study. I also appreciate Advanced Micro Devices, Inc. (AMD) and Unibap AB (publ.) for donating and providing the test platforms. In addition, I am thankful to Volvo CE, Saab, and SaraniaSat Inc. for providing the test data. Furthermore, I would like to thank my MOOCHA co-authors and family members.

Finally and foremost, I would like to express my greatest gratitude to my wife Bolormaa, son Ananda, parents, sister Nandin, and Jouni for your continuous love, supports and encourages.

Nandinbaatar Tsog
Sala, 2019

List of Publications

Papers included in thesis[1][2]

Paper A: *Intelligent Data Processing using In-Orbit Advanced Algorithms on Heterogeneous System Architecture* – Nandinbaatar Tsog, Moris Behnam, Mikael Sjödin, Fredrik Bruhn. In the Proceedings of the 39th International IEEE Aerospace Conference, AeroConf 2018.

Paper B: *Static Allocation of Parallel Tasks to Improve Schedulability in CPU-GPU Heterogeneous Real-Time Systems* – Nandinbaatar Tsog, Matthias Becker, Fredrik Bruhn, Moris Behnam, Mikael Sjödin. In the Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society, IECON 2019.

Paper C: *Using Docker in Process Level Isolation for Heterogeneous Computing on GPU Accelerated On-Board Data Processing Systems* – Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn. In the Proceedings of the 12th IAA Symposium on Small Satellites for Earth Observation, IAASmallSat 2019.

Paper D: *A Trade-Off between Computing Power and Energy Consumption of On-Board Data Processing in GPU Accelerated Real-Time Systems* – Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn. In the Proceedings of the 32nd International Symposium on Space Technology and Science, ISTS 2019.

[1] A licentiate degree is a Swedish graduate degree halfway between master and doctoral degrees.

[2] The included papers have been reformatted to comply with the licentiate thesis settings.

Papers not included in the thesis

Paper V: *Moon Cubesat Hazard Assessment (MOOCHA) - An International Earth-Moon Small Satellite Constellation* - Alexandros Binios, Janis Dalbins, Sean Haslam, Rusnė Ivaškevičiūtė, Ayush Jain, Maarit Kinnari, Joosep Kivastik, Fiona Leverone, Juuso Mikkola, Ervin Oro, Laura Ruusmann, Janis Sate, Hector-Andreas Stavrakakis, Nandinbaatar Tsog, Karin Pai, Jaan Praks, René Laufer. In the Proceedings of the 12th IAA Symposium on Small Satellites for Earth Observation, IAASmallSat 2019.

Paper W: *Using Heterogeneous Computing on GPU Accelerated Systems to Advance On-Board Data Processing* - Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn. In the European Workshop on On-Board Data Processing, OBDP 2019.

Paper X: *A Systematic Mapping Study on Real-time Cloud Services* - Jakob Danielsson, Nandinbaatar Tsog, Ashalatha Kunnappilly. In the Proceedings of the 1st Workshop on Quality Assurance in the Context of Cloud Computing, QA3C 2018.

Paper Y: *Advancing On-Board Big Data Processing Using Heterogeneous System Architecture* - Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn. In the Proceedings of the ESA/CNES 4S Symposium 2018, 4S 2018.

Paper Z: *Real-Time Capabilities of HSA Compliant COTS Platforms* - Nandinbaatar Tsog, Matthias Becker, Marcus Larsson, Fredrik Bruhn, Moris Behnam, Mikael Sjödin. In the Proceedings of the 37th IEEE Real-Time Systems Symposium (WiP) , WiP RTSS 2016.

Contents

Part I

Thesis

Chapter 1

Introduction

Currently the demand for increased capacity for on-board computation in different vehicles such as satellites and cars is skyrocketing. Many vehicles are deployed with high data-rate sensors such as stereo cameras, radars and lidars which often generate many tens or hundreds of gigabytes of data per second. To be able process these amounts of data in real-time and with limited resources in terms of space and energy the chip manufacturers have started to develop chips with heterogeneous computational units. So called heterogeneous architecture. However, these adaptations make the systems more complex, e.g., developing the systems with heterogeneous processing units increases unpredictable behaviours compared to single-core processing computing platforms.

Heterogeneous architectures employ different types of processing units Central Processing Unit (CPU), Graphics Processing Unit (GPU), Field-Programmable Gate Array (FPGA), Digital signal processor (DSP), to mention a few. These architectures provide massive computation capabilities and robustness to their applications. However, in this thesis, we tackle the challenges to make efficient use of the different resources due to various bottlenecks and difficulties in the architectures. In this regard, sharing of data between different processing units is one of the main bottlenecks. The time for transferring data between the memories of the different processing units could even be prolonged compared to the processing time on the slowest processing unit. Furthermore, each processor has its toolchain, technique, and specialty that the developers need to master. For example, there are different architectures of CPUs such as x86 and RISC. Similarly, GPUs are programmed with different programming languages, e.g., Nvidia's

GPUs accept only CUDA[1] while AMD GPUs allow OpenCL[2], HIP[3]. As a consequence of these challenges, it becomes a daunting task for a meager-skilled developer to fully utilize these platforms without loss of performance. With respect to these difficulties, Heterogeneous System Architecture (HSA) Foundation[4] has initiated a new standard, Heterogeneous System Architecture, to support ease of using heterogeneous computing for common development processes [13]. In this thesis, HSA-compliant GPU accelerated heterogeneous architectures are considered for on-board computers.

The emerging trends in automotive and aerospace domains such as self-driving cars and intelligent satellites respectively attach significance to the on-board data processing as the real-time decision making is crucial in these systems. For example, in small satellite systems, several independent subsystems/tasks (e.g., processing of mission experiments in-orbit and normal operations) that share the same processors can interfere with each other due to SWaP (size, weight and power) limitations, which can result in the system failure. Furthermore, the life cycle perspective for software is tricky, since the continuous development and continuous integration (CD/CI) of software or migration to new hardware is tedious. These complexities prevent to use heterogeneous architectures or limit to adopt a new type of heterogeneous architectures in embedded system applications that require timing predictability and energy efficiency.

This thesis investigates the characteristics of heterogeneous computing architectures focusing on computing potential and power consumption. We explore how the different processing units behave in terms of computing potential and power consumption by considering different types of applications covering computer vision, heavy mathematical algorithms as well as AI applications. The exploration is considered to employ the different type of environments such as legacy system vs HSA-compliant system, non-isolated environment vs process-level isolated environment, and a system with integrated accelerator vs a system with discrete accelerator. Furthermore, the thesis proposes a technique for the static allocation of parallel segments of tasks to suitable processing units while considering timing predictability and balanced usage of resources on heterogeneous computing architectures. Note that the heterogeneous architectures include multiple heterogeneous processing units, and each of them consists of multiple cores. In this thesis, CPU-GPU heterogeneous

[1]CUDA https://developer.nvidia.com/cuda-zone
[2]OpenCL https://www.khronos.org/opencl/
[3]HIP https://gpuopen.com/compute-product/hip-convert-cuda-to-portable-c-code/
[4]HSA Foundation http://www.hsafoundation.com

architectures are considered. Moreover, the investigation of the characteristics of heterogeneous architectures extends to test scenarios that are applied to the proposed technique. The test scenarios consider how allocations of parallel segments to either GPU and CPU change the power consumption while ensuring the system is schedulable.

1.1 Thesis Goal and Research Challenges

The overall goal of this thesis is *to facilitate the timing predictability of real-time applications on heterogeneous computing architectures while maintaining their computing performance and energy efficiency.* To achieve the overall goal, we target two core research challenges, while focusing on the aspects of timing predictability, computing performance, and energy efficiency of heterogeneous architectures for on-board data processing of embedded system applications.

> **Research Challenge 1:** *Which characteristics of processing units employed in CPU-GPU platforms affect the computation time and power consumption?*

> **Research Challenge 2:** *How to improve schedulability of real-time applications running on heterogeneous computing architectures by manipulating the extracted characteristics?*

1.2 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 introduces the background information including on-board data processing, Heterogeneous System Architecture (HSA), and metrics in focus. Furthermore, Chapter 2 presents system model and architecture. Chapter 3 describes research description in terms of research methodology, technical contribution, and thesis contribution. Related work is discussed in Chapter 4. The first part completes with Chapter 5 which discusses conclusions and future directions. The second part of the thesis is compilation of four papers from Chapter 6 to Chapter 9.

Chapter 2

Background and System Model

This section introduces necessary technical concepts and background information that support the context of the thesis. Further, system model and architecture is presented.

2.1 On-Board Data Processing

Space offers an ideal environment for real-time system applications as many on-board functions in spacecraft and satellites are constrained by real-time requirements. Any failure may end-up in a catastrophic result as it is not possible to fix the devices in orbit or in deep space. In order to reduce delays and have more predictable activities, the role of on-board data processing becomes significant. However, due to SWaP (size, weight and power) constraints along with radiation hardiness problem in space, the development of space systems usually encounter limitations which are not always experienced on the earth. The design and development of the on-board computer need to overcome these limitations.

As a heterogeneous architecture, the combination of CPU+FPGA and/or CPU+DSP is broadly employed for on-board computers in space [20]. However, these combinations could not support massive amount of computations required by the intelligent on-board data processing systems.These heterogeneous architecture combinations complemented by GPUs can overcome the above mentioned limitation by running multiple parallel executions and faster memory accesses. Thus, heterogeneous architectures that include CPU, GPU and FPGA can offer an efficient computation solution for on-board data processing systems. In such an architecture, an FPGA can be used for receiving sensors' data with shorter delays, a CPU can act as a controller between the

FPGA and GPU, and the GPU can process heavy computations. Unibap AB's[1] e22xx family products[2] evolved from GIMME3[3] provide a good example of such heterogeneous platforms. GIMME3 is invented by Mälardalen University and Unibap AB, and it is compliant with the HSA. The reference platforms used in this thesis are compatible with e22xx family products.

2.2 Heterogeneous System Architecture

The role of heterogeneous computing has been growing dramatically in industrial applications [1]. Employing multiple types of processing units makes the embedded systems robust. However, different types of specifications and designs of the processing units bring complexities for the development process from cost and timing perspective. To tackle these problems, HSA Foundation [12] has been established by multiple leading hardware vendors to develop the Heterogeneous System Architecture (HSA) specification for reducing the complexity of heterogeneous computations and providing the developer-friendly environments.

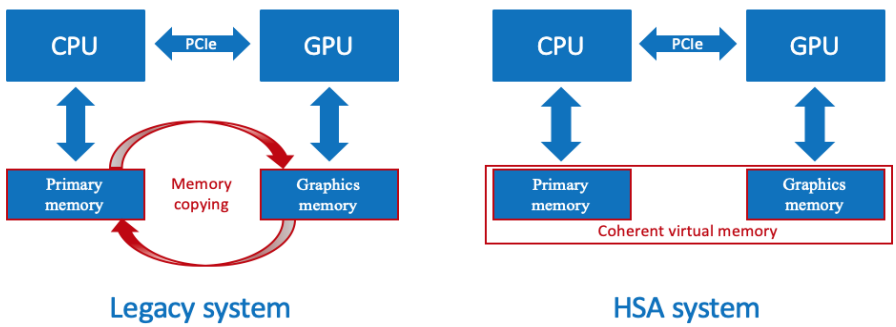


Figure 2.1: Memory structure between a non-HSA system and an HSA system.

Memory handling: The HSA aims to ease the development process on the heterogeneous platform by providing a development environment to the programmers that is similar to the environment for traditional systems, i.e., homogeneous systems. For example, the HSA supports well-known open source compilers, LLVM and GCC, and gives access to the memory spaces of all the

[1] Unibap AB (publ.) https://unibap.com/

[2] e22xx family products https://unibap.com/product/e22xx-compute-module/

[3] GIMME3 https://www.es.mdh.se/projects/364-GIMME3___Semi_fault_tolerant_next_generation_high_performance_computer_architecture_based_on_screened_industrial_components

processing units using only pointers in a virtual memory space. The HSA provides unified coherent memory for host and devices that saves time for transferring data between different physical memories, i.e., there is no memory copy between different physical memories, e.g., primary and graphics memories (see 2.1).

Software stack: As a part of the HSA, AMD introduces an initiative GPUOpen, an open source software stack, including, but not limited to, kernel level driver, runtime environment, profiling tools, computer vision and machine learning libraries such as ROCm[4], CodeXL[5], AMD OpenVX[6], MIOpen[7] as well as Tensorflow[8] on AMD GPUs. From TensorFlow 2.0, AMD has fully upstreamed their support. Only patches for APU support is needed. In addition, AMD's HSA compliant software stack is able to be used with Docker [24], which provides Continuous Integration and Continuous Deployment (CI/CD) implementation for easing the development process.

ROCm is an open source software stack including kernel driver and consists of multiple modules which support GPU computing. CodeXL is an open source development tool suite that supports profiling and debugging for the different processors such as CPU, GPU, and APU. MIOpen, an alternative to Nvidia's CuDNN, is an open source machine learning library that is developed to exert full potential of ROCm software stack as well as heterogeneous computing. Along with MIOpen, Tensorflow is available to run with ROCm driver. Tensorflow is an open source machine learning platform initiated by Google. OpenVX is an open and royalty-free standard computer vision library which is designed by the Khronos Group. It is portable across different vendors and hardware types, therefore, AMD's OpenVX version is able to run on ROCm driver. Moreover, OpenVX and OpenCV complement each other to perform as perfect computer vision library. OpenVX has to be implemented by hardware vendors while OpenCV is supported by a strong open source community.

Interconnection: While HSA covers interconnection between different devices, there exist the following four specifications focused on the interconnection and buses. CCIX[9] (Cache Coherent Interconnect for Accelerators), OpenCAPI[10] (Open Coherent Accelerator Processor Interface), Gen-Z[11] and

[4]ROCm https://github.com/RadeonOpenCompute/ROCm
[5]CodeXL https://gpuopen.com/compute-product/codexl/
[6]OpenVX https://gpuopen.com/compute-product/amd-openvx/
[7]MIOpen https://gpuopen.com/compute-product/miopen/
[8]Tensorflow https://rocm.github.io/tensorflow.html
[9]CCIX https://www.ccixconsortium.com/
[10]OpenCAPI https://opencapi.org/
[11]Gen-Z https://genzconsortium.org/

CXL[12] (Computer Express Link). The first three specifications are done by the industry top vendors excluding Intel. On the other hand, CXL is proposed by an Intel-driving consortium, and both AMD and ARM are announced to join this consortium. In this sense, CXL has a vast potential that could be upgraded to the industry de-facto standard for the interconnection between host and devices. The relation between HSA and CXL could be described as follows. While HSA is located at a high level, which is close to the developers for reducing the complexity of the development of heterogeneous systems, CXL is directly focused on hardware devices, which is at a low level. This indicates the possibility of the co-existence of HSA and CXL.

2.3 Metrics

In this thesis, the following metrics are considered in the investigation and experimental evaluation of heterogeneous computing architectures: *computing performance*, *energy efficiency*, and *timing predictability*. Use of these metrics in different research methods is explained in Section 3.1.

Computing performance describes how fast tasks are calculated on the given processing units. Hence, this metric presents computing potential of platforms/processing units. A unit of time, [*second*] or [*s*], is used for this metric and less computation time shows faster computing performance. There are other ways to express this metric such as FLOPS (floating-point operations per second). In the contributed papers, these expressions are used to give information about the reference platforms at a glance.

Energy efficiency introduces how less power is consumed by the given processing units to compute tasks. A unit of energy, [*Joule*] or [*J*], is chosen to describe this metric. A smaller value of power consumption corresponds to more energy efficiency. Energy efficiency is a metric to consider power consumption on the systems/platforms.

Timing predictability presents how systems fulfill the given timing constraints. We consider a system is predictable if all tasks in this system meet their timing requirements [27]. Timing predictability has been considered as "timing predictability of a system is related to proving, demonstrating or verifying the fulfillment of the timing requirements that are specified on the system [25]". In order to consider the timing requirements of tasks in task sets, we conduct schedulability analysis of task sets.

[12]CXL https://www.computeexpresslink.org/

2.4 System Model and Architecture

We consider a system, which consists of applications comprising of task sets, environments (i.e., containers, virtual machines), and a hardware platform as shown in Figure 2.2. A task consist of parallel and sequential segments. The parallel segments are represented by the fork/join model. Sequential segments should be executed only on CPU while parallel segments can be executed in parallel on GPU or on CPU sequentially. In this thesis, we consider an extension of the fork-join task model by adopting the notion of heterogeneous segments. A heterogeneous segment can either be mapped to a GPU for parallel execution (alternative B) or to a CPU for sequential execution of the same code segment (alternative A), see Figure 2.3.

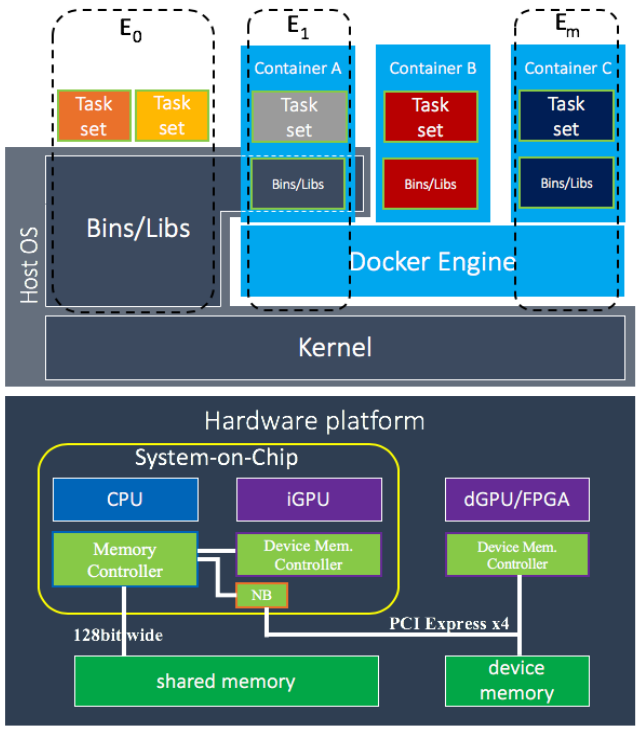


Figure 2.2: System Architecture.

The hardware platform employs a heterogeneous architecture, which may include three types of processing units; host device as CPU, integrated accelerators such as integrated GPU (iGPU), and discrete accelerators such as discrete GPU (dGPU) and/or FPGA (see Figure 2.2). We assume that the hard-

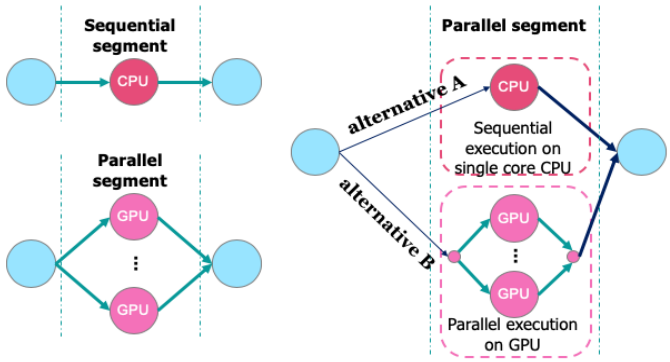


Figure 2.3: Sequential, parallel and alternative executions of parallel segments of tasks.

ware platform is HSA-compliant. In some cases, to narrow the setting, at least processing units in system-on-chip (SoC) side should be compliant with the HSA. From the power utilization perspective, on one hand, host device and integrated accelerators share the same power controller, i.e., both turn on and off at the same. On the other hand, discrete accelerators have a dedicated power controller for each, which means that the combinations, "Host-and-Integrated-Accelerators" and "Host-and-Discrete-Accelerators", have different amount of power consumption. Furthermore, from the memory structure perspective, "Host-and-Integrated-Accelerators" connects to the shared memory while "Host-and-Discrete-Accelerators" accesses to the virtual shared memory. In this thesis, the following three reference platforms are considered: CPU+iGPU, CPU+iGPU+FPGA, and CPU+dGPU.

On top of the hardware platform, three different types of environments are able to run. The first environment, a host environment, consists of operating system (OS) kernel, OS libraries, and task sets (i.e., workloads). There are two types of containers using either the same libraries with the host environment or custom-made libraries. In Figure 2.2, the environment E_1 shares the same libraries with the host environment E_0 while the environment E_m has its own custom made libraries. In this thesis, Docker [24] software stack brings a container on top of the Linux OS. In order to use the HSA, the host and container environments should include AMD's HSA-compliant driver ROCm. CPU scheduling is realized by a partitioned fixed-priority scheduler and we further assume that the execution on CPUs is preemptable. In contrast, the execution on GPU is not preemptable. GPU allows to execute tasks with non-preemptive fixed priority scheduling.

Chapter 3

Research Description

3.1 Research Methodology

The scientific method [7] provides how to facilitate new questions and formulate the problems. Holz et al. [11] discuss the four major steps (problem formulation, propose solution, implementation and evaluation) of the research methodology that we adopt in our research. The research methodology that is used in our research is described in Figure 3.1.

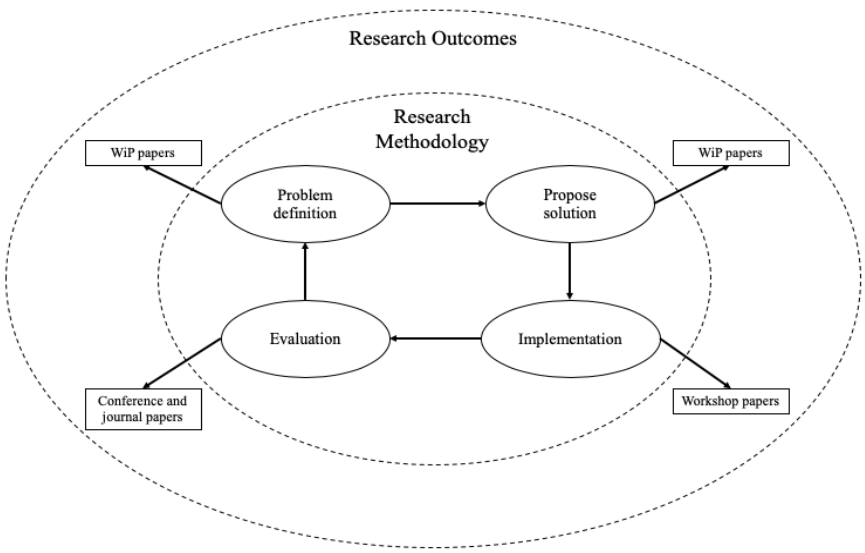


Figure 3.1: Research Methodology.

Problem definition. As first step, we have done a review of both the state

of the art and practice including the reason/problem for initiation of our research which has not been studied in academia. In addition with discussion between the involved parties, the research goal(s) are formulated as an outcome of problem formulation step. In Paper A, we have conducted a review of the state of the practices. How heterogeneous and parallel computing introduced in the state of art technologies such as HSA, CUDA, OpenCL, OpenMP and so on. The literature review has been done in order to understand how academia deals with heterogeneous computing in real-time systems in Paper B. The referenced papers in the collected papers in this thesis are included in the literature review study as well. As a result, we found some ideas for contributions.

Propose solution. After survey study, we chosen the most relevant works which help to consolidate our ideas. In Paper B, we extracted a model from real applications and proposed a solution using the model to improve the existing solutions. In Papers A, C, and D, we have identified the metrics which are used to understand the characterization of heterogeneous architectures.

Implementation. The implementation step results with empirical studies based on either real implementation (Papers A, C, and D) and simulation using the state of the art analysing tool (Paper B). Real implementations help to perform benchmarking study, i.e., measurement based experiments.

Evaluation. Comparison studies using the introduced metrics are considered in the evaluation step. Depending on the results of the evaluation step, the problem formulation and proposed solution could be revised and continued with the later steps. This process is iterated until the results are acceptable. The results/outcomes of each step could be presented as papers, reports and presentations in work-in-progress sessions, workshops, conferences and journals.

3.2 Technical Contributions

The thesis provides three technical contributions, which address the research challenges presented in Section 1.1.

Contribution 1: This contribution helps to understand the characteristics of heterogeneous architectures. By using the metrics, computing performance and energy efficiency, we characterize the differences of processing units, CPU and GPU. As our prior knowledge, some tasks are suitable for parallel computing while some tasks are executable only in a sequential manner. Computer vision and machine learning applications are examples of parallelizable applications, and we consider these type of applications in Papers A, C and D. Our investigations aim to understand what kind of applications (i.e., under what

kind of condition, applications) are suitable to run on GPU compared to CPU or vice versa while consuming less energy. As a result, the execution of parallel applications on GPU boosts up to 238 times computing performance and consumes 13.5 times less energy, compared to CPU. Although applications that include the smaller number of parallel executions are suitable to run on CPU from the computing performance aspect, all parallel applications consumed less energy on GPU compared to CPU in the reference platforms. This contribution addresses Research Challenge 1.

Contribution 2: Based on the achievements of contribution 1 and literature review, we propose a solution which tackles to improve the state of the art as well as the state of the practice in better schedulability of the task sets. We extract our model for heterogeneous processors from real applications and apply it to the real-time analysis proposed for CPU-GPU heterogeneous platforms. The solution improves the schedulability of task sets up to 90% compared to the existing solutions. This contribution is discussed in Paper B in detail and addresses Research Challenge 2.

Contribution 3: Once we have an understanding of heterogeneous architectures and the proposed solution which is timing predictable, the evaluation between the state of the art and the proposed solution is conducted. This contribution addresses a trade-off between the following two situations. On one hand, we know the balanced use of CPU and GPU improves schedulability of task sets (from contribution 2). On the other hand, however, GPU consumes less energy compared to CPU even computing performance is better in CPU compared to GPU (from contribution 1). Thus, in this contribution, we consider all three metrics, timing predictability, energy efficiency and computing performance. Papers C and D address this contribution. As a result, we confirm that there is no notable decrease in computing potential using Docker in process-level isolation. Contribution 3 addresses both research challenges 1 and 2.

3.3 Thesis Contribution

This dissertation is a compilation of the articles, called *kappa* in Nordic countries [21], and these articles are introduced in this section.

3.3.1 Paper A

Intelligent Data Processing using In-Orbit Advanced Algorithms on Heterogeneous System Architecture
Nandinbaatar Tsog, Moris Behnam, Mikael Sjödin, Fredrik Bruhn.

Status: This paper has been published in the Proceedings of the 39th International IEEE Aerospace Conference, March 2018.

Summary: The rapid increase in commercial usages of small satellites and CubeSats requires timely and accurate data dissemination to end-users. In order to fulfill these requirements, we consider that the intelligent and automation of on-board data processing is essential. Thus, in this paper, the advantages of intelligent on-board processing using advanced algorithms for Heterogeneous System Architecture (HSA) compliant on-board data processing systems are explored. Furthermore, we conduct an experimental study to evaluate the performance analysis by using image recognition algorithms based on an open source intelligent machine library 'MIOpen' and an open standard 'OpenVX'.

The main contributions of this paper are as follows:

- Investigate the computing performance of on-board data processing on the heterogeneous architecture by running image processing algorithms which use both MIOpen framework of high performing machine learning primitives and OpenVX vision library.
- How the concurrent executions of multiple advanced algorithms affect the worst-case execution time (WCET) of other parallel running tasks which of expresses the quality of the on-board heterogeneous system.
- Energy efficiency of HSA compliant GPU based computation compared to CPU only computation for the feature tracking with the different workloads.

My contribution: The candidate was the main driver of the work. The co-authors contributed by valuable discussions and reviewing the paper.

3.3.2 Paper B

Static Allocation of Parallel Tasks to Improve Schedulability in GPU Accelerated Real-Time Systems

Nandinbaatar Tsog, Matthias Becker, Fredrik Bruhn, Moris Behnam, Mikael Sjödin.

Status: This paper has been accepted to the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON 2019), October 2019.

Summary: Allocating the right computation segments to the right processing units encounters with complexity and is crucial while the

computing world is currently undergoing a paradigm shift towards heterogeneous computing. In this paper, the execution of the parallel segments of tasks has performed a pre-runtime static allocation either sequentially on CPU or in parallel using a GPU. This allows improving any unbalanced use of GPU accelerators in a heterogeneous environment. The overuse of accelerators such as GPU ends up with a bottle-neck of the entire system execution. The experimental results show that the allocation scheme improves the system schedulability up to 90%.

The main contributions of this paper are as follows:

- The overusing of GPU might bring a negative impact to the schedulability of real-time systems
- Improve this problem by offloading GPU computing to CPU.
- The following three heuristic approaches are adapted: Non-Greedy Resource Allocation Heuristic Approach (NHA), Speedup Classifier based Heuristic Approach (SHA), and Min-Min Approach (MMA).
- The synthetic experiments show up to 90% of improvement of the schedulability of the real-time systems.

My contribution: The candidate was the main driver of the work. The co-authors contributed by valuable discussions and reviewing the paper.

3.3.3 Paper C

Using Docker in Process Level Isolation for Heterogeneous Computing on GPU Accelerated On-Board Data Processing Systems
Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn.
Status: This paper has been published in the Proceedings of the 12th IAA Symposium on Small Satellites for Earth Observation, May 2019.

Summary: The operation of satellites may fall into critical conditions when the on-board data processing interferes strongly to the basic operation functionalities of satellites while the technological advancements make the on-board data processing possible on small satellites and CubeSats. In order to prevent falling these critical conditions, we present an experimental study of the process level isolation of on-board payload data processing from the basic operations of satellites using Docker. This study continues with the prior study [28] on parallel allocation method, which improves the schedulability of the entire system up to 90%. Based on this allocation

method, the comparison study has been conducted between the non-isolated and isolated environments.

The main contributions of this paper are as follows:

- An experimental study of an isolation technique using Docker for on-board data processing.
- Comparison study of a parallel allocation method between the non-isolated and isolated environments.

My contribution: The candidate was the main driver of the work. The co-authors contributed by valuable discussions and reviewing the paper.

3.3.4 Paper D

A Trade-Off between Computing Power and Energy Consumption of On-Board Data Processing in GPU Accelerated Real-Time Systems
Nandinbaatar Tsog, Mikael Sjödin, Fredrik Bruhn.
Status: This paper has been published in the Proceedings of the 32nd International Symposium on Space Technology and Science (ISTS), June 2019. The paper has invited to submit to the Transactions of JSASS, Aerospace Technology Japan, and is under review now.

Summary: On-board data processing using heterogeneous processing units is one of the prior on-orbit activities that it improves the performance capability of in-orbit space systems such as deep-space exploration, earth and atmospheric observation satellites, and CubeSat constellations. However, on-board data processing encounters with higher energy consumption compared to traditional space systems. Because traditional space systems employ simple processing units such as micro-controllers or a single-core processor as the systems require no heavy data processing on orbit. In this paper, we introduce HSA compliant platforms employed with heterogeneous processing units for on-board data processing and perform the detailed study of the platform.

- Understanding suitable mapping from heterogeneous processing units to tasks under limited energy budget.
- Conduct observations of energy consumption in GPU accelerated real-time systems while using the mapping method for the balanced use of heterogeneous processors.

- The following observations are performed. Compiler vs Computing potential, Power consumption vs Programming manner, and Power consumption vs Execution manner.

My contribution: The candidate was the main driver of the work. The co-authors contributed by valuable discussions and reviewing the paper.

3.4 Mapping between Research Challenges, Contributions and Publications

A mapping among the research challenge, contributions and publications included in this thesis is illustrated in Figure 3.2. The Research Contribution 1, encapsulated in Papers A, C and D addresses Research Challenge 1. The Research Contribution 2 included in Paper B addresses Research Challenge 1. Whereas, Research Contribution 3, packaged in Papers C and D, addresses both research challenges.

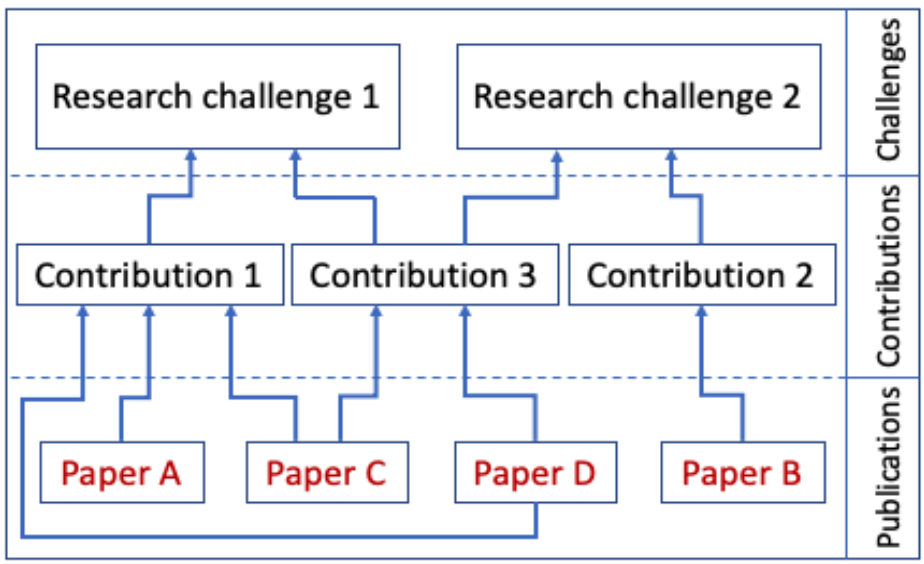


Figure 3.2: Mapping among the research challenges, contributions and publications.

Chapter 4

Related Work

HSA is a new architecture and there are less related works, while heterogeneous computing is researched a lot. Recently, Bruhn et al. [5] presented the design, analysis, and benchmarking of a heterogeneous computing platform built with COTS components targeting space environments. According to the authors, the platform shows 1.7 times larger computational performance than state-of-the-art platforms for the space industry. The authors describe the importance of using the HSA enabled architecture in the heterogeneous computing platforms for future work. A preliminary theoretical work has been done for discussing the importance of HSA in real-time systems and its opportunities. Behnam et al. [3] discuss some of the opportunities on HSA as a standard for next-generation high-performance systems. The authors propose some challenges such as the architecture of modelling support, and allocation and policing by the system software.

There exist several approaches handling the usage of GPU in real-time systems. Kim et al. presented TimeGraph [16], RGEM [15] and Gdev [17] along with zero-copy I/O processing for low-latency GPU computing [14]. TimeGraph is a user- and kernel-space real-time GPU scheduler that allows requests from tasks to access GPU by their priorities. RGEM introduces how to utilize a user-space GPU scheduler by splitting a memory-copy transaction into smaller pieces to allow preemption during the data-copy operations. Gdev presents direct access to system memory from GPU by using a zero-copy technique [14]. Self-suspending task techniques [6, 4] are widely investigated in real-time systems, and it is applicable to GPU accelerated real-time systems. Most of these works focus on compensating the limitation of early existing GPU hardware and device drivers such as a zero-copy technique for accelerators' memory and slicing tasks into smaller pieces for allowing preemption. However, these limitations are solved by coming new technologies such as

unified memory, zero-copy and preemption technologies in CUDA [10] and Heterogeneous System Architecture (HSA) [12].

The works of Elliott et al. [9, 8] and Kim et al. [18, 19] consider worst-case timing behavior in GPU accelerated real-time systems. GPUSync [9] handles the schedulability of tasks in GPU accelerated real-time systems using a synchronization based approach for the GPU management. Kim et al. [18, 19] present a Server-Based Approach which improves the locking demerit of synchronization based approaches. They present a GPU server that controls the GPU access by its queue management.

The research of modeling focusing on sequential and parallel tasks such as fork-join [22, 26] and DAG [29, 23] is active. Recently, Baruah [2] introduced $if\text{-}then\text{-}else$ concept using conditional DAG task modeling, which is useful for the heterogeneous computing. The topology of this model is similar to a heterogeneous segment discussed in Paper B, however, the if-then-else targets the control flow of the application whereas we focus on design-time mapping of a fork-join task model on heterogeneous platforms.

Chapter 5

Conclusion and Future Work

This thesis conducted an investigation to understand heterogeneous architectures using the metrics timing predictability, computing performance, and energy efficiency. We have been addressing the following overall goal; to facilitate the timing predictability of heterogeneous architectures while maintaining their computing performance and energy efficiency. First, we have investigated the characteristics of processing units in the heterogeneous architectures focusing on computing potential and power consumption. Our measurement based result shows that GPU consumes less power consumption compared to CPU even CPU performs better computing performance compared to GPU. Then, the next contribution is that our new heterogeneous computing model improves the schedulability of task sets up to 90% compared to the state-of-the-art solutions. We have performed the measurements based on the schedulability analysis of task sets. Our last contribution addresses the pros and cons considering all three metrics, timing predictability, energy efficiency, and computing performance.

As a future work, we plan to validate the proposed techniques on industrial case studies. The need for the practical use of heterogeneous architectures increases dramatically as they required in more and more computer vision and AI applictions. Thus, HSA-compliant ROCm is one of the strongest candidates for the practical use as it is an open-source software stack including kernel drivers. ROCm is adopted in mainline Linux kernel and the thesis confirms that it performs very stable from Linux kernel 5.0. However, Linux is suitable for average and best case scenarios, although Linux is a de-facto OS for many embedded systems. Hence, we will introduce a real-time GPU scheduler customizing ROCm for an HSA-compliant GPU accelerated platforms. Further, understanding real-time properties of HSA specification is crucial, and we aim to investigate it from both theoretical and practical sides. Another in-

teresting research direction for future work is to investigate memory management of platforms that employ heterogeneous processing units including heterogeneous CPUs and heterogeneous accelerators. There already exist different shared memory architectures that take into account for heterogeneous processing units such as uniform memory access (UMA), non-uniform memory access (NUMA) as well as heterogeneous uniform memory access (hUMA). We plan to investigate the different use case scenarios of these architectures with heterogeneous processing units.

Bibliography

[1] H. Andrade, L. E. Lwakatare, I. Crnkovic, and J. Bosch. Software challenges in heterogeneous computing: A multiple case study in industry.

[2] S. Baruah. Resource-efficient execution of conditional parallel real-time tasks. In M. Aldinucci, L. Padovani, and M. Torquati, editors, *Euro-Par 2018: Parallel Processing*, pages 218–231. Springer International Publishing, 2018.

[3] M. Behnam, F. Ciccozzi, M. Sjödin, and F. Bruhn. Software architecture for next generation hyperparallel cyber-physical hardware platforms. pages 1–4, 2015.

[4] K. Bletsas, N. Audsley, W.-H. Huang, J.-J. Chen, and G. Nelissen. Errata for three papers (2004-05) on fixed-priority scheduling with self-suspensions. *Leibniz Transactions on Embedded Systems*, 5(1):02–1–02:20, 2018.

[5] F. Bruhn, K. Brunberg, J. Hines, L. Asplund, and M. Norgren. Introducing radiation tolerant heterogeneous computers for small satellites. *IEEE Aerospace Conference Proceedings*, 2015-June:1–10, 2015.

[6] J.-J. Chen, G. Nelissen, W.-H. Huang, M. Yang, B. Brandenburg, K. Bletsas, C. Liu, P. Richard, F. Ridouard, N. Audsley, R. Rajkumar, D. de Niz, and G. von der Brüggen. Many suspensions, many problems: a review of self-suspending tasks in real-time systems. *Real-Time Systems*, Sep 2018.

[7] G. Dodig-Crnkovic. Scientific methods in computer science. In *Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, April 2002.

[8] G. A. Elliott and J. H. Anderson. Globally scheduled real-time multiprocessor systems with gpus. *Real-Time Systems*, 48(1):34–74, Jan. 2012.

[9] G. A. Elliott, B. C. Ward, and J. H. Anderson. Gpusync: A framework for real-time gpu management. In *34th IEEE Real-Time Systems Symposium (RTSS)*, pages 33–44, Dec 2013.

[10] M. Harris. "Unified Memory for CUDA Beginners." June 19, 2017. Available: https://devblogs.nvidia.com/unified-memory-cuda-beginners/ [Oct 16, 2018].

[11] H. J. Holz, C. State, E. Bay, A. Applin, D. Joyce, H. Purchase, C. Reed, C. State, and E. Bay. What are they and how should we teach them.pdf. pages 96–114, 2006.

[12] HSA Foundation. "Heterogeneous System Architecture.". Available: http://www.hsafoundation.com/ [Oct 16, 2018].

[13] HSA Foundation. HSA Foundation - ARM, AMD, Imagination, MediaTek, Qualcomm, Samsung, TI. Available online at: http://www.hsafoundation.com (accessed 6 November 2018).

[14] S. Kato, J. Aumiller, and S. Brandt. Zero-copy i/o processing for low-latency gpu computing. In *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 170–178, April 2013.

[15] S. Kato, K. Lakshmanan, A. Kumar, M. Kelkar, Y. Ishikawa, and R. Rajkumar. Rgem: A responsive gpgpu execution model for runtime engines. In *32nd IEEE Real-Time Systems Symposium (RTSS)*, pages 57–66, Nov 2011.

[16] S. Kato, K. Lakshmanan, R. Rajkumar, and Y. Ishikawa. Timegraph: Gpu scheduling for real-time multi-tasking environments. In *USENIX Conference on USENIX Annual Technical Conference (USENIXATC)*, pages 2–2. USENIX Association, 2011.

[17] S. Kato, M. McThrow, C. Maltzahn, and S. Brandt. Gdev: First-class gpu resource management in the operating system. In *USENIX Conference on Annual Technical Conference (USENIXATC)*, pages 37–37. USENIX Association, 2012.

[18] H. Kim, P. Patel, S. Wang, and R. R. Rajkumar. A server-based approach for predictable gpu access control. In *23rd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–10, Aug 2017.

[19] H. Kim, P. Patel, S. Wang, and R. R. Rajkumar. A server-based approach for predictable gpu access with improved analysis. *Journal of Systems Architecture*, 88:97–109, 2018.

[20] G. Lentaris, K. Maragos, I. Stratakos, L. Papadopoulos, O. Papanikolaou, D. Soudris, M. Lourakis, X. Zabulis, D. Gonzalez-Arjona, and G. Furano. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. *Journal of Aerospace Information Systems*, 15(4):178–192, 2018.

[21] L. Lundahl. On the choice of thesis format and on writing the" kappa" of a thesis of publications. *WORK IN PROGRESS. Umeå universitet: November*, 2010.

[22] C. Maia, M. Bertogna, L. Nogueira, and L. M. Pinho. Response-time analysis of synchronous parallel tasks in multiprocessor systems. In *22nd International Conference on Real-Time Networks and Systems (RTNS)*, pages 3:3–3:12. ACM, 2014.

[23] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. C. Buttazzo. Response-time analysis of conditional dag tasks in multiprocessor systems. In *27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 211–221, July 2015.

[24] D. Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), Mar. 2014.

[25] S. Mubeen, E. Lisova, and A. V. Feljan. A perspective on ensuring predictability in time-critical and secure cooperative cyber physical systems. In *20th IEEE International Conference on Industrial Technology ICIT 2019, 13 Feb 2019, Melbourne, Australia*, number 20, 2019.

[26] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu, and C. D. Gill. Parallel real-time scheduling of dags. *IEEE Transactions on Parallel and Distributed Systems*, 25(12):3242–3252, Dec 2014.

[27] J. A. Stankovic and K. Ramamritham. What is predictability for real-time systems?, 1990.

[28] N. Tsog, M. Sjödin, and F. Bruhn. Using Heterogeneous Computing on GPU Accelerated Systems to Advance On-Board Data Processing. In *European Workshop on On-Board Data Processing (OBDP2019)*, Amsterdam, the Netherlands, 2019. ESTEC.

[29] J. D. Ullman. Np-complete scheduling problems. *J. Comput. Syst. Sci.*, 10(3):384–393, June 1975.