1. Principal investigator, project title, site of research, date of research plan

Tk.D. Johan Ersfolk
Parallel scheduling of dynamic dataflow programs
Embedded Systems Laboratory, Department of Information Technologies,
Åbo Akademi University, Turku, Finland,
Period: 01.09.2015 – 31.8.2018 (36 months)

2. Rationale

The goal with this project is to develop programming methods that better target modern computer platforms. Modern platforms are more parallel, having several or even many processor cores, and more heterogeneous, having processor cores optimized for specific tasks. Portability of software between such architectures requires that the code can be adapted and optimized for each of the target platforms. For this to be possible, the program description such be such that order in which tasks are performed is not too strictly defined by the programmer but instead left for the deployment stage. Dataflow programming provides such a description but the tools and methods for adapting dataflow programs, especially to many-core systems, have not reached the point where dataflow programming could become a mainstream programming approach. In this project we will develop approaches where model checking is used to analyze and optimize dataflow programs. In other words, we apply formal methods (model checking) on software projects by improving development tools to allow more complex analysis on the software, hiding the formal methods in the development tools.

Dataflow has been proposed as a solution for the problem of how to describe software in a platform independent way while enabling transformation and analysis, as it encapsulates computations while explicitly defining dependencies and communication between nodes. This allows more complex analysis, and certain properties can be proven on individual actors while other properties on networks of actors. Dataflow programming and models are used in industry especially for modeling signal processing and multimedia applications. For example, Simulink and LabView are widely in research and development. Furthermore, the RVC-CAL dataflow language was standardizes as the language to be used for describing coding tools in the MPEG-RVC standard. These models provide descriptions that are easy to grasp on a high level due to the graphical representation of the directed graphs that the dataflow representation builds on; this representation also separates the computation tasks and highlights the communication, which makes the software components easier to reuse. While the dataflow representation provides several benefits to the programming of heterogeneous many-core architectures, it should be seen as a means to program certain parts of larger software projects, which are practical to be described as dataflow by having an identifiable data stream, while other languages and models can be used to interface the dataflow model

and interact with it. What this means is that dataflow programs need to be described such that the dataflow network has a predictable interface and behavior, and that these properties are possible to analyze and verify. This concerns properties related to scheduling such as whether or not a program can run with bounded memory, can be described with a manageable number of static schedules, and has predictable execution time for the different tasks.

Dataflow allows the programmer to construct programs without specifying unnecessary restrictions on the order in which different task should run, which results of a more flexible program description. The program is instead analyzed with static or quasi-static scheduling approaches which abstract any excess flexibility or dynamism describing the core behavior of the program which should correspond to the specifications of the application. This analysis makes the program more dependable, performance predictable, and reusable. Model checking can be used to verify interfaces between actors but also to find more efficient ways to run the application.

Dataflow is to some extent used in industry, however, the goal is to take dataflow programming to the next level, enabling larger software projects to utilize the benefits of dataflow programming, removing the current problems which mainly are a result of a lack of tools.

2.1 Significance of the research nationally and internationally

Many of the larger companies in the Finnish technology industry have activity in the fields where dataflow programming is relevant. As an example, video coding is relevant for some companies working with secure storage of people's data and video conferencing systems. In the more traditional industry, e.g. controllers of electrical or diesel engines are implemented using digital signal processing techniques, which in practice means that these in some sense are modeled as dataflow applications.

The traditional industry is slowly moving towards multi-core, and the IT industry many-core. For the companies that face the challenge with supporting and utilizing emerging platforms, and work close to the signal processing domain, can greatly benefit from efficient dataflow methods.

2.2 Previous research pertaining to the topic and how the research project links to it

Dataflow scheduling has been an active research topic for many years. The perhaps most well known approach, which is also widely used as it can be efficiently implemented, is SDF by Lee [11]. With SDF, as with many other dataflow models of computation (MoCs) the dataflow actors have restricted behavior which in turn enables static scheduling approaches. Many MoCs have been proposed among others: cyclo-static dataflow by [12], parameterized dataflow [13], and scenario-aware dataflow. These all provide different trade-offs between expressiveness and analyzability.

The other approach to dataflow scheduling, is to allow any kind of behavior within a dataflow actor, and by this provide the programmer an expressive programming model, while using complex tools to identify certain properties or patterns in the program. An example of a very expressive model is dataflow process network (DPN) [14], which is the model that the RVC-CAL language is based on. Existing work on the analysis and scheduling of such programs is either based on fitting individual actors to more restricted classes of MoCs and then using the techniques available for that MoC, examples of such works are Wipliez et al. [15] where RVC-CAL actors are classifies to SDF, CSDF, and PSDF to allow more efficient scheduling, Cedersjö et al. [16] where actors are translated into a special type of state machine called actor machine which then is used to optimize the scheduling and also is used for classifying the actors.

The other point of view is to identify repeatable firing sequences from clusters of actors and translate the behavior of the cluster to a finite state machine. Such methods have been proposed by Boutellier et al. [17], Ersfolk et al. [5], which both identify the necessary scheduling points of a program while finding static schedules between these points.

The work mentioned above mostly relates to scheduling of dataflow programs on single-core architectures. Multi- or many-core scheduling of dataflow programs is a more difficult problem which has only been solved for special cases. Of course, in a dataflow program, each actor can run concurrently with the other actors as long as its input requirements are fulfilled, and, the work in [10] can be used to adapt the size of actors such that each actor can run on its own core, unfortunately, this type of scheduling is not enough for scheduling dataflow on the new many-core architectures. Instead, what is needed is an approach for scheduling the individual tasks on the available cores. Piat et al. presented a tool/method [18] for analysis and scheduling of dataflow actors on multi-core system. This method enables static compile-time scheduling which gives minimal overhead and maximal utilization of the available cores, however, actors currently need to have static or cyclo-static behavior.

2.3 How the research project links to other research by the PI

Previous work by the PI has concentrated on using model checking for optimizing single-core parts of dataflow programs by applying efficient scheduling on dataflow programs, or parts of such, that are mapped to a single processor core. While this also is applicable to multi-core, by applying it separately on program parts that are mapped to individual cores, and thereby achieving a multi-threaded program, this kind of parallelism will not scale to future many-core platforms and it does not give a solution to how to handle load balancing.

The previous work by the PI, however, provides background knowledge regarding how model checking can be used with dataflow programs and has provided the basic set of tools for defining model checking tasks for dataflow programs. The goals of the previous work were mainly related to showing that dataflow programs can be described as model checking without the state space becoming too large to handle and showing that the scheduling

overhead can be radically decreased by using such quasi-static scheduling methods. Compared to this previous work, this project takes this analysis one step further including parallelism and communication overheads in the analysis.

3. Objectives and expected results

3.1 Research objectives

The overall objective of the project is to enable efficient execution of programs described in a high-level language (e.g. RVC-CAL) on emerging many-core heterogeneous architectures. The challenge with this objective is that some architectures work well with task parallelism while other requires data parallelism in order to be efficient. We claim that while dataflow as such corresponds to task parallelism, when an algorithm has data parallelism the dataflow program can be described such that this parallelism also can be identified from the dataflow description. For this, we will develop methods for utilizing both data- and task parallelism from dataflow programs.

The main objectives are the following.

1) To identify parallelism in schedules, meaning that the schedules are partial orders instead of total orders. The goal with this is to enable parallel run-time systems to with a minimal overhead utilize the platform.

2) Develop cost functions that can be used to find efficient schedules with the model checker. Low scheduling overhead and enough parallelism are essential, however, for efficient execution properties such as the memory hierarchy should be considered in the scheduling phase by scheduling based on a cost function corresponding to cache behavior.

3) Investigate which specifications are needed for making the dataflow programs easier to analyze. In order to fully utilize the potential of a dataflow program, more formal specification of the behavior of the actors should be used to describe the intended behavior of the actors. The benefit of this is both that the actors can be verified to work correctly and, more importantly in this context, it enables more efficient scheduling as more is known about the program. In other words, we will investigate how dataflow programs should be designed and specified in order to enable efficient execution on parallel architectures.

3.2 Hypotheses

A dataflow program contains parallelism which can be utilized on a many core architecture. The efficient firing (execution) sequences of dataflow actions on many cores cannot be left to chance but must be found at compile-time. Methods based on model checking can be used to analyze the program state space and firing sequences, taking the architecture in to account, producing performance predictable software.

3.3 Expected impacts of research

One of the bigger problems in the software industry at the moment is how to program parallel architectures. With heterogeneous architectures and the number of cores reaching tens to hundreds of cores, synchronizing these manually is not anymore an option. The research contributes to closing the gap between efficient scheduling of static dataflow programs [18] on multi-core and the expressive dynamic dataflow programs [19]. This means that we can combine programming languages that are expressive enough for industrial applications with execution and run-time systems that efficiently run smaller tasks (or dataflow actors) on multi-core platforms.

The results from the research are directly usable in the existing tool-chains and will partly be evaluated as part of these. This means that the results directly are accessible and can be used by partners both in industry and academia.

4. Research methods and material, support from environment

4.1 Research methods

For the research to be successful, we need to develop the mathematical foundation, build experimental tools, and perform experiments and measurements. The mathematical foundation describes the rules for correct scheduling and parallelism; by showing that some rules hold for a program or part of it, efficient scheduling can be applied and the parallelism can be exploited. The mathematical foundation must be evaluated from two points of view; first, correctness is obviously required, secondly, we need to evaluate how well these correspond to how dataflow programs typically are implemented, and thereby evaluate how well the parallelism of the programs can be exploited.

The experimental tools are part of compiler/development tool chains, and will be functional to the extent that dataflow programs can be transformed and compiled to take advantage of a set of chosen platforms. The research will be done within the available open source RVC-CAL tools and compilers. The concrete implementation work involves analysis passes that identify parallel schedules in the dataflow program, as well as instrumentation of the model checkers to take more knowledge regarding the dataflow program and the target platform into account when searching for schedules. The purpose of the implementation work is, on the one hand, to show that the analysis is feasible in practice, and on the other hand, to enable experiments and measurements.

Finally, experiments on relevant embedded platforms are performed, not only to show that the approach is useful but also to learn about how dataflow applications can be adapted to different types of architectures. This part opens the possibility to collaboration with our partners at INSA-Rennes regarding run-time systems.

While the research above is divided into distinct steps, in practice the research is performed as an iterative process in order to minimize the risks of insufficient methods that does not fit the practical applications and architectures.

4.2 Research material

The main tools needed for the work, Orcc (open rvc-cal compiler), Preesm, and Spin, are open source software projects. Two of these tools, Orcc and Preesm, are authored by our partners it INSA Rennes which means that it is easy to get support and also access to the code repositories.

Additionally to software, some developments boards with embedded processors are provided by ÅA.

5 Ethical issues

The research does not raise any ethical issues.

6. Implementation: schedule, budget, distribution of work

6.1 Work Package 1: Mathematical Foundation for Schedulability and Parallelism

The first work package extends the theoretical work in [thesis] to parallel schedules and develops the scheduling towards covering more types of actors than in the previous work. The main contribution is taking an RVC-CAL actor, which by definition is sequential, and transforming it in to a schedule containing parallel sets of computation tasks. The approach is based on separating different aspects that decides the order in which an RVC-CAL program executes and relaxing the sequential execution of actor firing when some rules apply.

Work package 1 is mainly concerned with the deciding when static scheduling can be applied and whether or not a static schedule allows parallelism. The feasibility of the first part of this, regarding static scheduling, has already been shown in the previous work, however, this part will be extended to, instead of covering a subset of programs, be applicable to any program. This can be done by instead of analyzing which programs follow certain rules and only allow scheduling these, generalize the choice of scheduling rules to allow any program with the worst case that the transformed program is identical to the original program. The parallelism of the programs is the more interesting and the main part of this work package and the theoretical work is mainly related to modeling the firings of actors and analyzing the closure, that is, which calculations are required to precede the actor firings.

6.2 Work Package 2: Analysis Algorithms and Tools

The second work package is concerned with constructing algorithms that can identify structures defined in the first work package. The difference between these is that the mathematical foundations descries when static scheduling or parallelism is possible while

algorithms are needed to check whether some rules hold for an actor. The algorithms that are developed as part of this work package use static analysis as well as abstract interpretation and various formal methods such as SMT, model checking, and theorem proving to prove that a set of rules hold for the actor or set of actors.

6.2.1 Work Package 2.1: Partitioning, schedulability, choice of firing rules

This work package takes the ideas from work package 1 and develops algorithms that can be implemented in the development tools that are used within the research group for the work related to dataflow development. Compared to the mathematical foundation, the algorithms are conservative in the sense that these attempt to prove a certain property, however, failure does not mean that the property does not hold, but merely that it could not be proven to hold. This means that there is a need for different algorithms that attempt to prove such properties using different information about the program or using different approaches. In other words, there is a rather large search space to explore regarding algorithms.

6.2.2 Work Package 2.2: Model Checking Strategies and Instrumentation

The second part if this work package concerns the strategies the model checking part uses to find schedules and how it decides *how good* a schedule is. In practice this means that the scheduling tool is made more aware about both the dataflow program and the target architecture. Dataflow aware, in this case, means that the relevant firing rules, the need for potential delay tokens, etc. need to be identified such that the states the model checker searches for directly can be generated without any assistance from the developer. Platform aware means that when a schedule is found, the model checker has some values describing different aspects such as how well the schedule will fit the cache.

6.2.3 Work Package 2.3: Specification and Verification

The third part of this work package is an analysis of the gap between the mathematical foundation and what can be resolved from the program text. In order to close this gap, more careful specification of the RVC-CAL is demonstrated and used to allow parallel scheduling of *difficult* applications. Simultaneously, the specifications are used to prove that the dataflow program works according to the specification, either using model checking or theorem proving. In practice, the specifications will in this work be described with some simple format such as XML, as the format itself is not the goal but to decide which information or specifications are relevant.

6.3 Work Package 3: Code Generation and Run-time Systems

The third work package is concerned with efficiently executing dynamic dataflow program on parallel architectures. The relevant run-time systems are divided into two parts where the first one is the default code generation strategy that will be used in the project while the second part is an attempt to speed-up the research by collaboration.

6.3.1 Work Package 3.1: Composition, Scheduling Format and OpenCL

This part of the research is concerned with generating efficient programs from the analysis results obtained. In previous work, the result of the scheduling stage has been a finite state machine with sequential static schedules as the transitions; this format in turn has been used to perform actor composition. In this work package, we develop new models for describing the parallelism that has been identified, such that parallel code can be generated for OpenCL.

6.3.2 Work Package 3.2: PREESM and Open Event Machine

There is a rather interesting connection between the analysis part of this project and the piSDF model used in PREESM [18]. This work package is intended to work as a research boost of these two, so far independent, approaches. The information that is derived from the RVC-CAL programs in this work is to some extent comparable to the piSDF description that is used in PREESM. Currently PREESM is used to schedule and design parameterized static dataflow graphs of parallel architectures, while this project targets dynamic dataflow programs. The successful connection between these works will open the opportunity to use PREESM for scheduling the existing RVC-CAL applications, simultaneously it will enable our approach to be used to find statically schedulable parts of program while using PREESM for the parallel static scheduling and using the Open Event Machine as the run-time for the programs.

6.4 Work Package 4: Tool Demonstrator and Experiments

The project results in a set of tools that are used with the Orcc compiler or alternatively with the PREESM tool. The goal with this work package is to concentrate on the usability part of the tools, making these easy to use with the RVC-CAL compiler. There are two reasons why this is important, 1) it makes the results usable in practice, resulting in a larger impact, 2) it makes it easy to experiment with applications and platforms, which in turn makes it easier to experiment with how programs can be fitted to platforms.

The second part of this work package is therefore running experiments of various platforms with as many RVC-CAL programs as possible. The goal with the experiments is to learn about how different design choices affect performance on parallel architectures. The tool demonstrator enables experiments regarding design space exploration for dynamic dataflow programs with a flexibility that currently is not available in any tool.

6.5 Justifications for the total cost estimate

We seek funding for one (1) post-doctoral researcher for three (3) years. The salary cost is 112222 € (for 3 person years). For mobility and travel costs we have budgeted it for 28000 €. We have included VAT cost in other costs. We have used levels 5/3 for year 2015 for post doctoral researcher of the salary system for universities in estimating yearly costs and, furthermore, we estimated the rise in salary to be about 2000 € per year considering an increase in performance.

7. Research team, collaboration

The project is performed at the Embedded Systems Laboratory at Åbo Akademi University. The lab has several years of experience with dataflow programming, and has collaborated with several groups which currently are among the main contributors in the field. For this work, the most important collaborations are with EPFL Switzerland and INSA-Rennes France, with which the group has collaborated on several related research papers during the previous years, among others: [1], [2], [3], [4], [8], and [9]. As part of the collaboration between these groups, there has been a continuous exchange of visiting researchers.

The lab is part of the Department of Information Technologies at Åbo Akademi University, which has a strong background in formal methods. Also, the leader of Embedded Systems Laboratory Prof. Johan Lilius has a strong connection to formal methods taking part in the Rodin EU project [7].

8. Researcher training and research careers

The purpose with the project on a personal level for the PI is to develop the research skills to achieve a docentship by the end of the project.

9. Mobility plan

The project will include two half year visits to two research groups. The exact times for the visits have not yet been decided, however, we have a history of continuously organizing visits between the groups. As part of this project, the PI will visit Marco Mattavelli's group at EPFL in Switzerland, which has a strong background in video coding and dataflow programming and design space exploration. This visit will be used to further strengthen the collaboration between the groups. The PI will also visit J-F Nezan's group at INSA-Rennes in France, which is the author of two of the more important tools regarding RVC-CAL, namely Orcc and PREESM. This visit will be used to produce concrete tools that will be usable by the programmers using RVC-CAL.

10. Bibliography

[1] Ersfolk Johan, Roquier Ghislain, Lilius Johan, Mattavelli Marco. *"Modeling Control Tokens for Composition of CAL Actors"*. Conference on Design and Architectures for Signal and Image Processing (DASIP'13). 2013.

[2] Boutellier Jani, Ghazi Amanullah, Silvén Olli, Ersfolk Johan. "*High-Performance Programs by Source-Level Merging of RVC-CAL Dataflow Actors".* Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '13). 2013.

[3] Ersfolk Johan, Roquier, Ghislain, Lund Wictor, Mattavelli Marco, Lilius Johan. "*Static and Quasi-static Compositions of Stream Processing Applications from Dynamic*

Dataflow Programs". Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'13). 2013

[4] Ersfolk Johan, Roquier Ghislain, Lilius Johan, Mattavelli Marco. *"Scheduling of Dynamic Dataflow Programs based on State Space Analysis"*. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '12). 2012.

[5] Ersfolk Johan, Roquier Ghislain, Jokhio Fareed, Lilius Johan, Mattavelli Marco. *"Scheduling of Dynamic Dataflow Programs with Model Checking"*. Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '11). 2011.

[6] Dahlin Andreas, Ersfolk Johan, Yang Guifu, Habli Haitham, Lilius Johan. *"The Canals Language and its Compiler"*. Proceedings of the 12th International Workshop on Software and Compilers for Embedded Systems (SCOPES-09). 2009.

[7] RODIN EU project: Rigorous Open Development Environment for Complex Systems.

[8] Simon Holmbacka, Erwan Nogues, Maxime Pelcat, Sébastien Lafond and Johan Lilius. "*Energy Efficiency and Performance Management of Parallel Dataflow Applications".* Conference on Design and Architectures for Signal and Image Processing (DASIP'14). 2014.

[9] Dahlin, Andreas and Jokhio, Fareed and Gorin, Jérôme and Lilius, Johan and Raulet, Mickaël. "*Interfacing and Scheduling Legacy Code within the Canals Framework*". Proceedings of the 2011 Conference on Design and Architectures for Signal and Image Processing (DASIP). 2011

[10] Ersfolk, Johan, *"Scheduling Dynamic Dataflow Graphs with Model Checking"* TUCS Dissertations 181. 2014

[11] E.A. Lee and D.G. Messerschmitt. *"Static scheduling of synchronous dataflow programs for digital signal processing"*. Computers, IEEE Transactions on, C-36(1):24 - 35, January 1987.

[12] G. Bilsen, M. Engels, R. Lauwereins, and J.A. Peperstraete. *"Cyclostatic dataflow"*. In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on, volume 5, pages 3255 - 3258 vol.5, May.

[13] B. Bhattacharya and S.S. Bhattacharyya. *"Parameterized dataow modeling of dsp systems. In Acoustics, Speech, and Signal Processing"*, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on, volume 6, pages 3362 -3365 vol.6.

[14] E.A. Lee and T.M. Parks. *"Dataflow process networks"*. Proceedings of the IEEE, 83(5):773 - 801, May 1995.

[15] Matthieu Wipliez and Micka□el Raulet. *"Classification of dataow actors with satisfiability and abstract interpretation"*. IJERTCS, 3(1):49 - 69, 2012.

[16] Gustav Cedersjö and Jörn Janneck. *"Actor Classification using Actor Machines"*. In Signals, Systems and Computers (ASILOMAR), 2013 Conference Record of the Forty Seventh Asilomar Conference on, 2013.

[17] Jani Boutellier, Mickaël Raulet, and Olli Silvén. *"Automatic hierarchical discovery of quasi-static schedules of rvc-cal dataflow programs"*. Journal of Signal Processing Systems, 71(1):35 - 40, 2013.

[18] J. Piat, M. Raulet, M. Pelcat, Pengcheng Mu, and O. Deforges. *"An extensible framework for fast prototyping of multiprocessor dataflow applications*". In Design and Test Workshop, 2008. IDT 2008. 3rd International, pages 215 - 220, Dec 2008.

[19] J. Eker and J. Janneck. "*CAL Language Report"*. Technical Report ERL Technical Memo UCB/ERL M03/48, University of California at Berkeley, December 2003.