CalMAR -a Multi-Application Dataflow Runtime

Lionel Morel, Manuel Selva, Kevin Marquet, Coralie Saysset, Tanguy Risset

Work-in-Progress: CalMAR - a Multi-Application Dataflow Runtime

Lionel Morel
Univ Lyon, INSA Lyon, CITI, F-69621 Villeurbanne, France
lionel.morel@insa-lyon.fr

Manuel Selva
INRIA CAMUS, Icube lab, Univ of Strasbourg
manuel.selva@inria.fr

Kevin Marquet
Univ Lyon,INSA Lyon, Inria, CITI, F-69621, Villeurbanne, France
kevin.marquet@insa-lyon.fr

Coralie Saysset
Univ Lyon, INSA Lyon, CITI, F-69621 Villeurbanne, France
coralie.saysset@insa-lyon.fr

Tanguy Risset
Univ Lyon,INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France
tanguy.risset@insa-lyon.fr

ABSTRACT

In this paper, we propose a new dataflow runtime that dynamically manages several dataflow applications. We propose an implementation of such a runtime called CalMAR built on top of the RVC-Cal environment, and validate its efficiency compared to the RVC-CAL traditional approach.

Keywords

dataflow programming, runtime system, load-balancing

1. INTRODUCTION

As the processor industry continues the performance race, multi-core processors are generalizing to personal computers and handheld devices. The historical thread-based programming model is still the main approach used to program such systems. However, dataflow programming has proven to be very efficient in many application domains including networking, video, sound and digital signal processing.

Existing dataflow runtime systems focus either on the efficient execution of a *single* data-flow application or on scenarii where applications are known a priori. New contexts such as video streaming servers and software-radio appear where the whole system is dynamic and where dataflow applications are deployed on-demand. This raises of the problem of executing an a priori unknown number of dataflow applications concurrently on the same multi-core system.

The goal of this work is to define the requirements for such a *dataflow multi-application runtime*, and to introduce CalMAR, our prototype implementation based on RVC-Cal [1].

EMSOFT'17 Companion, October 15–20, 2017, Seoul, Republic of Korea
ISBN 978-1-4503-2138-9.
DOI: https://doi.org/10.1145/3125503.3125562

We illustrate the efficiency of CalMAR with various scenarios where several instances of video decoders are executed simultaneously.

2. PROBLEM STATEMENT

Many different models of computation have been proposed in the dataflow paradigm, providing a trade-off between static analyzability and program flexibility. Our work can be applied to static or dynamic dataflow models of computation. As a target architecture model, we focus on *shared–memory multi-core* architectures.

When executing a program, the main task of the dataflow runtime system is to drive actors' execution while maximizing the applications throughput. To that end, the runtime first distributes the workload by *mapping* each actor to a particular core. On each core, the runtime then *schedules* each actor's execution. To efficiently execute a single dataflow application, existing runtimes execute on top of an operating system that provides threads. Typical mapping strategies are *one-thread-per-core*.

Executing several dataflow applications in parallel with this solution results in bad performances because the number of context-switches increases drastically and because load-balancing is performed independently by several runtimes, without a global knowledge of the system. In particular, applying the one-thread-per-core for each application (leading to what we call *concurrent runtimes*) leads to very poor performances. *Hardware partitioning*, where each application executes on a separate set of cores is not flexible: it is limited to executing at most N applications on N cores.

Based on these observations, we promote the design of a runtime built for executing any number of dataflow applications concurrently on a multi-core architecture. This runtime should be able to dynamically load dataflow applications and provide efficient mapping and scheduling strategies dealing with actors of all the applications present at a given time in the system. To limit the effects of context switching, we design our runtime with as many threads as there are cores in the system independently of the number of dataflow applications (Fig. 1).

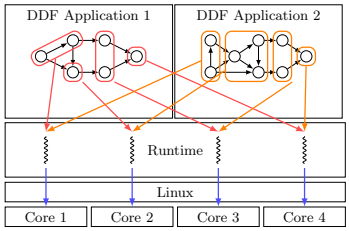
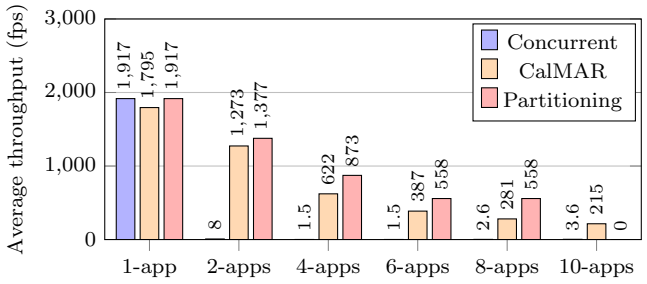
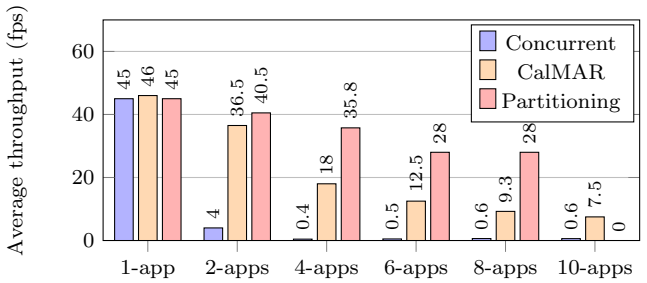


Figure 1: Global approach used in CalMAR.



(a) Comparison with multiple instances of MPEG4-SP.



(b) Comparison with multiple instances of HEVC.

Figure 2: Comparing concurrent runtimes, hardware isolation and CalMAR.

3. PROTOTYPE IMPLEMENTATION

Our prototype is called *Cal Multi-Application Runtime* (CalMAR). It is built on top of the *Orcc* environment [2]. It assumes applications have been designed in the RVC-Cal language and compiled with Orcc. CalMAR uses Orcc's data structures to represent applications, actors, FIFOs and it adapts mapping strategies to cope with several applications.

Once CalMAR is started, the user interacts with it through a small set of commands. The most important one is the `load` command that allows to load a new application into the runtime, applies the mapping heuristic to the set of applications in the system, including the newly loaded one, and resumes the execution of all applications present.

In terms of mapping, we have first experimented with an application-wise load-balancing (*LB*) implemented in Orcc: the heuristics is applied iteratively to each application present, without taking into account any information about other applications. We have also implemented a global load-balancing strategy (*gLB*) which takes all the actors of all the applications present and maps them using this global knowledge.

Actor scheduling in CalMAR is for now an exact replication of the scheduling strategies offered by the Orcc runtime. In our experiments we only use the round-robin strategy.

	Configuration 1		*Configuration 2*	
	3 HEVC	5 MPEG4-SP	5 HEVC	3 MPEG4-SP
LB (fps)	25.22	27.7	16	19.85
gLB (fps)	27.8	26.2	20.75	19.6

Table 1: Throughput comparison of LB and gLB load balancing policies, for two different configurations mixing HEVC and MPEG4-SP.

4. EXPERIMENTAL VALIDATION

We evaluate CalMAR using an HEVC and an MPEG4 *Simple Profile* video decoding applications, several instances of which are loaded into CalMAR. Results were obtained on a Dell PowerEdge C6220 II node of the grid5000 experiment platform[1], which comprises an Intel Xeon E5-2660 v2, with 8 cores running a Linux kernel 3.2.

In a first scenario, we execute instances of the same application and compare the throughput obtained with concurrent runtimes, hardware partitioning and CalMAR. The number of instances is noted $n \in [1, 10]$. The number of cores is 8. For partitioning, each of the n applications considered is allocated to $\lfloor 8/n \rfloor$ cores. Fig. 2a (resp. Fig. 2b) shows a comparison for $n = 1$ to $n = 10$ instances of MPEG4-SP (resp. HEVC). For each application the average throughput, in frames per second, over the different instances is reported.

We make the following observations. First, concurrent runtimes should not be used. Second, hardware partitioning, when it is possible, is better than CalMAR. However, partitioning does not scale as the number of instances becomes greater than the number of cores. Moreover, guessing the best partitioning is hard in general and not practical when applications are admitted dynamically.

In another scenario we run different combinations of applications in CalMAR with both LB and gLB strategies applied and compare the throughput obtained by type of applications. Table 1 compares the two strategies when running 5 (resp. 3) instances of HEVC and 3 (resp 5.) instances of MPEG4-SP. The results show that our gLB strategy gives better performances for applications with many complex actors such as HEVC.

5. CONCLUSION

We present the principles of a fully-dynamic runtime dedicated to executing efficiently dataflow applications. Many paths of research open ahead of us. We plan to study mapping strategies allowing some form of partitioning. We also work on integrating quality-of-service requirements on applications and building qos-aware actors' mapping and scheduling strategies. Finally, we plan to extend CalMAR to take both dataflow and non-dataflow applications into account.

6. REFERENCES

[1] I. Amer, C. Lucarz, G. Roquier, M. Mattavelli, M. Raulet, J.-F. Nezan, and O. Deforges. Reconfigurable video coding on multicore. *Signal Processing Magazine, IEEE*, 26(6):113–123, 2009.

[2] H. Yviquel, A. Lorence, K. Jerbi, G. Cocherel, A. Sanchez, and M. Raulet. Orcc: Multimedia development made easy. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 863–866. ACM, 2013.

[1] https://www.grid5000.fr/