A Fast Evaluation Approach of Data Consistency Protocols within a Compilation Toolchain

Loïc Cudennec, Safae Dahmani, Guy Gogniat, Cédric Maignan, Martha Johanna Sepulveda

Procedia Computer Science

Volume 80, 2016, Pages 2297–2301

ICCS 2016. The International Conference on Computational Science

A Fast Evaluation Approach of Data Consistency Protocols within a Compilation Toolchain

Loïc Cudennec[1], Safae Dahmani[2], Guy Gogniat[3], Cédric Maignan[3], and Martha Johanna Sepúlveda[4]

[1] CEA, LIST, Saclay, France
[2] Sorbonne Universités, UPMC Paris 06, UMR 7606, LIP6, Paris, France
[3] University of Bretagne-Sud, LabSTICC, Lorient, France
[4] Institute for Security in Information Technology, Technical University of Munich, Germany

Abstract

Shared memory is a critical issue for large distributed systems. Despite several data consistency protocols have been proposed, the selection of the protocol that best suits to the application requirements and system constraints remains a challenge. The development of multi-consistency systems, where different protocols can be deployed during runtime, appears to be an interesting alternative. In order to explore the design space of the consistency protocols a fast and accurate method should be used. In this work we rely on a compilation toolchain that transparently handles data consistency decisions for a multi-protocol platform. We focus on the analytical evaluation of the consistency configuration that stands within the optimization loop. We propose to use a TLM NoC simulator to get feedback on expected network contentions. We evaluate the approach using five workloads and three different data consistency protocols. As a result, we are able to obtain a fast and accurate evaluation of the different consistency alternatives.

Keywords: Many-core, Network-on-chip, Data-consistency Protocols, Analytical Evaluation

1 Introduction

Today, many-core processors embed thousands of processing cores on a single chip with complex distributed memory systems (288-core Kalray MPPA, 72-core Tilera GX, 64-core Adapteva Epiphany, 52-core Intel Xeon Phi). While hardware support can still enforce consistency between local $L1$ and $L2$ cache memories, there is a need to provide a global system that federates larger memories such as scratchpads and memories that are shared within clusters of processing elements (PE). As far as we know, most of the large many-core chips do not provide unified consistent distributed memories. Towards the growing complexity of data consistency protocols, software distributed shared memory (DSM) can help manage memories onto large parallel

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.421

processors. Several protocols for many-core processors [1,2,5] have been proposed. One conclusion is that there is no unique protocol that fits to all applications, many-core architectures and running contexts. Instead, we propose to use a multi-protocol platform that allows to choose for each access to a shared data a specific protocol that is tightly parameterized. However, the protocol decision relies on the developer of the application, with explicit information given to the compiler for each memory access. Here we propose to automate this decision process at compile time, by deeply analyzing the application source code. A regular compilation system may not afford to simulate with a high level of precision and to process all possible solutions. In order to overcome such drawback, in this work we propose the use of high-level cycle-accurate system model for evaluating the consistency protocols in MPSoCs. As a result, different system alternatives can be rapidly evaluated with higher precision, due to the contention-aware capability of our framework. Our framework extended the light-weight SystemC transaction-level modeling cycle-accurate network-on-chip simulator [6] with three libraries: i) several MPSoC consistency protocols; ii) goals and cost functions; and iii) memory access traces. The results obtained with our framework are used within the optimization loop in order to improve and guide the selection of the best configuration.

2 Contribution: A Time accurate *NoC* Model to evaluate Cache Consistency Protocols

The Cache Validator tool reads a memory access trace of an application and calculates the traffic generated within the *NoC*, given a particular data consistency protocol and a *NoC* topology. It is based on an analytical model, meaning that statistics are calculated given input data and calculation rules: no real execution of the application is performed at this point. However, such an input memory access trace is obtained using a dynamic binary instrumentation tool [4] from a previous execution. The quality of the analytical evaluation directly depends on the quality of the trace in terms of code coverage and shared data access interleaving. The obtained traffic includes all exchanged messages (control and data messages) involved in a memory access request. Different execution platform configurations can be explored using the Cache Validator model according to some parameters: chip size (number of cores), network topology (interconnections between the routers) and cache size (number of memory blocks). In this work, we propose to extend the analytical evaluation with a cycle-accurate *NoC* model that can highlight network contention, while keeping the computing time affordable to an optimization loop. The evaluation of different consistency protocols in MPSoCs is performed by means of the extended TLM-NoC simulator [6]. The basic version of the simulator consists on a modular SystemC-TLM cycle accurate simulation environment composed by a set of libraries that allow the description and configuration of NoC-based MPSoCs. It is composed by: i) Traffic generators and consumers, which emulate the behavior of the computation and storage components of the MPSoC. Different traffic patterns are obtained by modifying the nature, topology and type of the data exchanged; ii) Routers, which include a rich set of parameters and which can be interconnected according to the desired topology; iii) Monitors, that annotate all the communication events; and iv) Analysis tools, which quantify a set of metrics according to the annotated values by the monitor. The configuration of such parameters determines the architecture. This allows to evaluate each solution according to specific requirements. In order to be able to evaluate the consistency protocols, we have enhanced the Traffic generators and consumers, monitors and analysis tools. The Cache Validator generates the memory access traces depending on the current consistency protocol. Then, the consumers emulate the behav-

ior of each data access. Each consistency protocol will present a different data injection. New monitors include the ability to track each transaction due the consistency protocol. Finally, the analysis tool allows to quantify the access latency of the different workloads. By using the enhanced framework, it is possible to observe the NoC contention impact on the system performance. As a study case, we consider a MPSoC composed by 8x8 tiles interconnected by a mesh/torus NoC of 32-bit width channels. Each tile emulates the behavior of a processor and a L1 cache. Only one tile located at the bottom right corner of the MPSoC emulates the behavior of the main memory. In order to introduce the contention consideration in our study we define different types of packets. Note that the extended framework can evaluate a wide set of MPSoC configurations under several consistency protocols.

3 Experimental results

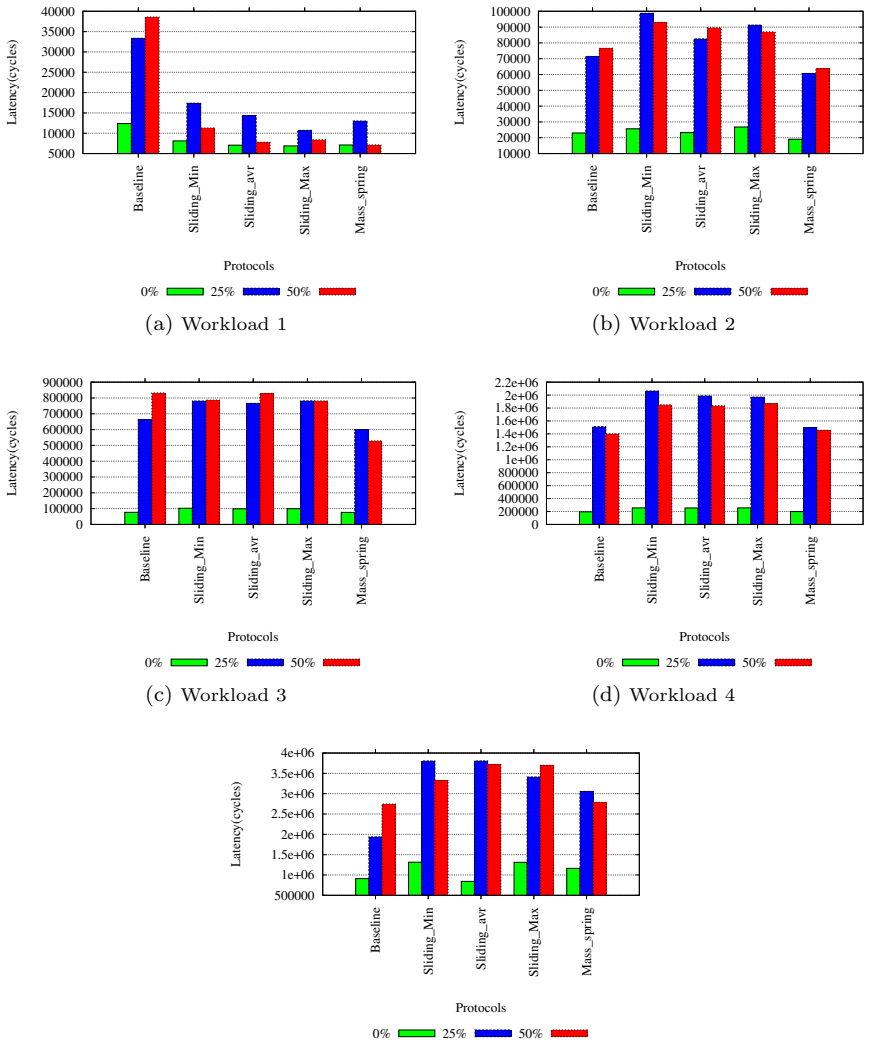
We studied two types of protocols: *Baseline* and *Sliding-based protocols*. The *baseline* protocol refers to a directory-based cache consistency protocol [3] widely used for multicore architectures. The *Data Sliding* protocol [1] is a cache cooperative approach where each core is allowed to use its neighbor's cache in order to lower load concentration in hot spots. The *mass-spring* [2] sliding-based protocol is an *N-chance forwarding* protocol with a variable migration radius using the *mass-spring* physical model.

3.1 Network contention and sliding radius analysis

Network contention evaluation mainly depends on the ability of the system to replay consistency protocol data and control messages in the NoC simulator with realistic timings. This is a problematic issue because messages are generated by the Cache Validator tool without timings: There only exists a causal dependency relation between them. However, the NoC simulator is able to replay timed messages. We therefore propose to partition the set of messages (in a contiguous way), each subset being sequentially fired, and all messages within a subset being fired in parallel. Partitioning can describe different parallelism levels: from a full sequential trace (one message in each subset) to a full parallel trace (all messages in a single subset). Table 1 gives the contention rates corresponding to different parallelism levels for the Sliding protocol with a radius of migration equals to 1. We observe that the parallelism level generates more contention for simple traces (workloads 1 to 3). More complex traces (workloads 4 and 5) generate, even when triggering messages sequentially, network contention close to hot spots and key nodes.

Workloads - Para. level	0%	25%	50%	75%	100%
Workload 1	39%	39%	28%	45%	57%
Workload 2	83%	88%	93%	89%	95%
Workload 3	80%	76%	87%	87%	89%
Workload 4	66%	76%	73%	80%	66%
Workload 5	97%	81%	71%	78%	92%

Table 1: Contention rate for different parallelism levels from 0% (full sequential) to 100% (full parallel).



(a) Workload 1

(b) Workload 2

(c) Workload 3

(d) Workload 4

(e) Workload 5

Figure 1: Access latency comparison between different workloads using several consistency protocols and different parallelism levels (0, 25 and 50%). Lower is better.

3.2 A comparative study of cache management protocols

In this section, we evaluate 5 protocol configurations for the 5 workloads, with 3 levels of parallelism. Protocols are: the 4-state MESI baseline, the Data sliding with radius set to minimum (1-hop), average (7-hop) and maximum (14-hop), and the Mass-spring protocol. Results are given in Figure 1. First, we observe that there is a notable difference between results obtained using the (0%) sequential simulation (close to pure analytical evaluation) and the parallel simulations. This is particularly true for the first workload in which severe contention is highlighted for the *baseline* protocol and not for the others. In some situations, such a difference can clearly

modify the choice of the consistency protocol in the compilation toolchain. Second, we observe that the protocols perform differently depending on the complexity of the workloads. For a simple workload (Figure 1a), the Data sliding and Mass-spring protocols perform far better than the Baseline protocol. When increasing the complexity of the workloads, we can observe that the Data sliding protocols are not efficient, the Baseline is undecided (it depends on the hot spots localization), and the Mass-spring always provides good performance. These information are then used to decide what consistency protocols the system should use or not. We notice, for example, in figure 1a corresponding to the less stressed workload that the cooperative sliding based protocols are more efficient than the *baseline* one. Moreover, the *mass-spring* protocol that defines dynamically the migration radius of each data provides a better performance. Whereas, the figure 1e shows that both of the *baseline* and *mass-spring* protocols are more efficient than fixed-radius sliding protocols. This means that for *workload 5* these protocols are not the best choice.

4 Conclusion

The main contribution of this paper is a method to evaluate network contention within a compilation toolchain for data consistency protocol decision. The model relies on a configurable TLM-based *NoC* platform that generates several performance evaluation metrics such as the number of cycles needed to perform remote memory accesses. Despite the lower accuracy of such an analysis process compared to a full execution of the application onto the targeted platform, the proposed approach gives tangible clues to choose and configure cache consistency protocols. Processing times are also kept within reasonable bounds in order to be integrated within the multi-protocol compilation platform.

References

[1] Safae Dahmani, Loïc Cudennec, and Guy Gogniat. Introducing a data sliding mechanism for cooperative caching in manycore architectures. *Proceedings of the 18th International Workshop on High-Level Parallel Programming Models and Supportive Environments*, pages 335–344, 2013.

[2] Safae Dahmani, Loïc Cudennec, Stéphane Louise, and Guy Gogniat. Using the spring physical model to extend a cooperative caching protocol for many-core processors. *Proceeding of the IEEE International Symposium on Embedded Multicore/Many-core*, 2014.

[3] Daniel Lenoski, James Laudon, Kourosh Gharachorloo, Anoop Gupta, and John Hennessy. *The directory-based cache coherence protocol for the DASH multiprocessor*, volume 18. ACM, 1990.

[4] C.K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V.J. Reddi, and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *ACM SIGPLAN Notices*, volume 40, pages 190–200. ACM, 2005.

[5] Jussara Marandola, Stéphane Louise, Loïc Cudennec, Jean-Thomas Acquaviva, and David Bader. Enhancing Cache Coherent Architectures with Access Patterns for Embedded Manycore Systems. In *International Symposium on System-on-Chip 2012 (SoC 2012)*, Tampere, Finlande, October 2012. Tampere University of Technology, Department of Computer Systems, IEEE.

[6] Johanna Sepúlveda, M Strum, and JC Wang. A tlm-based network-on-chip performance evaluation framework. In *Proc. 3rd Symposium on Circuits and Systems, Colombian Chapter*, pages 54–60, 2007.