Cache locking content selection algorithms for ARINC-653 compliant RTOS

Alexy Torres Aurora Dugo[1], Jean-Baptiste Lefoul[1], Felipe Gohring de Magalhaes[1], Dahman Assal[2] and Gabriela Nicolescu[1]

Polytechnique Montréal[1], MANNARINO Systems and Software[2]

Context

- Avionic software -> **real time, determinism and safety**
  - ARINC-653 standard defines partitioned architectures
  - Partitions have multiple threads executing concurrently
- Modern architectures generate **interferences**[1]
  - Resource sharing
  - Contention
- One of the **main** interference channels are caches[2]
- **Private cache** levels are intra-partition interferences
- Partition's threads can **evict** each others cache data and generate unwanted **delays in execution**.

Objectives

- Reduce cache related interferences in single-core and multi-core architectures
- Reduce contention in shared caches
- Improve private cache performances in critical real-time embedded systems

Contribution

- Propose a memory tracing framework, capable of analyzing memory accesses of real-time tasks
- Define new approaches to select the cache lines to lock based on different criteria
- Integrate cache locking mechanism in an ARINC-653 compliant RTOS

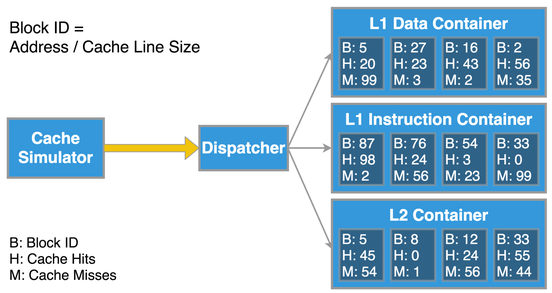
Contact

Alexy Torres Aurora Dugo:
alexy.torres-aurora-dugo@polymtl.ca

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

MANNARINO®

Greedy approach



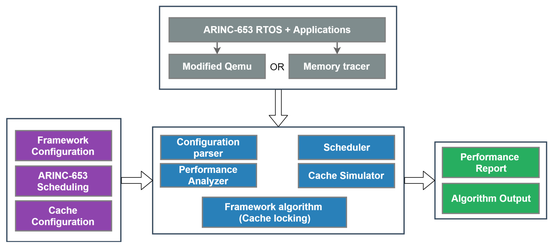
	WAY 1	WAY 2	WAY 3	WAY 4	
SET 0	76				W= 234
SET 1	5				W= 453
SET 2	2				W= 332
SET 3	27				W= 45

	WAY 1	WAY 2	WAY 3	WAY 4	
SET 0	76	0	560		W= 102
SET 1	5	165	345		W= 550
SET 2	2	342	674		W= 564
SET 3	27	43	543		W= 234

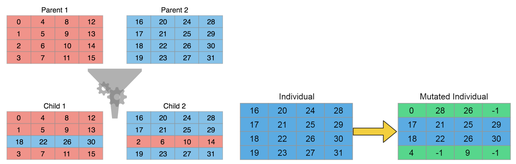
	WAY 1	WAY 2	WAY 3	WAY 4	
SET 0	76	0			W= 142
SET 1	5	165			W= 334
SET 2	2	342			W= 310
SET 3	27	43			W= 123

	WAY 1	WAY 2	WAY 3	WAY 4	
SET 0	76	0	560	64	W= 460
SET 1	5	165	345	321	W= 455
SET 2	2	342	674	22	W= 554
SET 3	27	43	543	123	W= 604

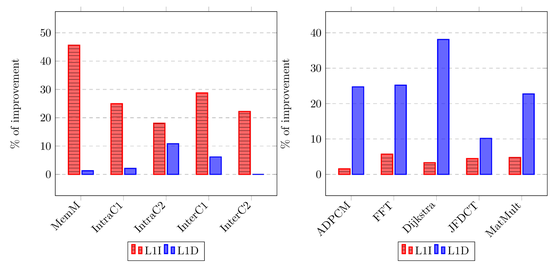
Memory Analyzer Framework



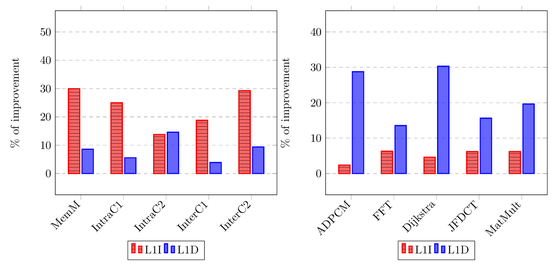
Genetic approach



Result



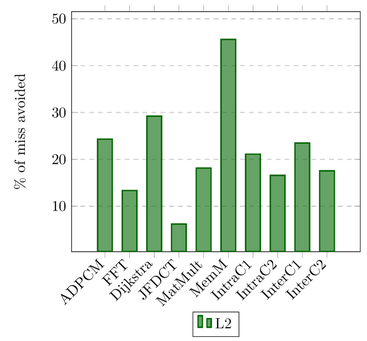
Cache miss improvement using PLRU policy with a reduction of instruction miss up to 45% and data miss up to 38%.



Cache miss improvement using PRR policy with a reduction of instruction miss up to 30% and data miss up to 31%.

Application	Not locked	Locked	Improvement (%)
FFT	868.09	0.55	99.93
ADPCM	1123.80	9.78	99.13
Dijkstra	664.39	10.55	98.49
MatMult	342.13	0.54	99.84
MemM	4031.65	442.01	89.04

Standard deviation of execution time (CPU cycles). Results show that we significantly reduce the execution time standard deviation by 97.29% on average.



L2 accesses avoided for PRR policy. Our algorithm allows to reduce the workload on L2 cache by 45% in some cases and by 31.5% on average.

Conclusion

- In this work we **proposed**
  - A novel approach to **select cache locking content**
  - A memory analysis **framework**

The framework allows the integration of algorithms to consume these traces and generate configuration file to be used by the RTOS.

- We **compared** two approaches:
  - A **greedy** approach
  - A **genetic** algorithm

The greedy approach performs better in our context

- We **integrated** our solution in an existing ARINC-653 compliant RTOS.

We were able to **reduce** cache misses in **private** caches by up to 45% and 25% on average.

References

[1] I. Bate, P. Conmy, T. Kelly, and J. McDermid. Use of modern processors in safety-critical applications. *The Computer Journal*, 44(6):531–543, 2001.

[2] R. Fuchsen. How to address certification for multi-core based ima platforms: Current status and potential solutions. In *29th Digital Avionics Systems Conference*, pages 5.E.3–1–5.E.3–11, Salt Lake City, UT, USA, Oct 2010.

Acknowledgements

The authors would like to thank the industrial partner MANNARINO Systems & Software for their support and their help. We also would like to thank MITACS and CRIAQ for founding this research.

Mitacs CRIAQ