Accelerating Hardware Coherence Using Programmer Input in Multi/Manycore Systems

Keywords : *Manycore, cache coherence, RISC-V*

1 Context

To improve performance, a general purpose processor associates a private cache memory to each of its cores, in order to keep a subset of data close to the execution units of the core and thereby accelerate data access. The chip also features a larger cache shared by all cores. This architecture is depicted in Figure 1.

To facilitate the development of parallel applications, the various cache memories implemented on chip provide data coherency. That is, a given memory address may be cached in several private caches only if the associated cores are only reading the data. Any write to the data requires invalidating all existing copies in the private caches (except for that of the writer). A cache that lost its copy following a write will have to reload the new version of the data (e.g., from memory). In general purpose processors, coherency is handled by hardware and is completely transparent to the programmer. If that were not the case, software would have to manage coherency explicitly for the program to be correct, which would significantly slow parallel application development down.

However, the hardware handling coherency is forced to make sub-optimal choices as it does not have a global vision of access and sharing patterns across the system. For instance, when data is loaded from memory, a core can aks to insert it into its private cache either with read permission or both read and write permissions. Depending on the access and sharing patterns, the correct decision (for performance or chip traffic) is not always the same. If the data is not

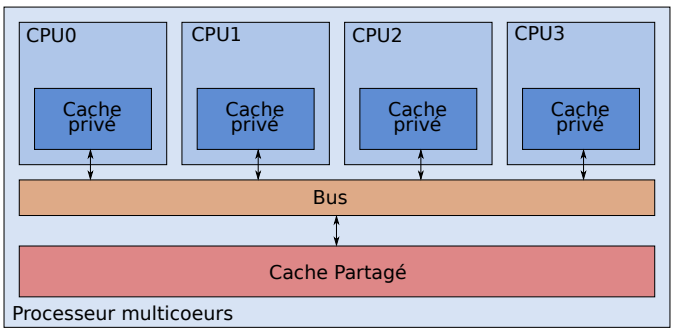

FIGURE 1 – 4-core multicore processor, each core features its own private cache and all cores share a last level shared cache memory.

shared, the core should obtain write permission in order to avoid sending a second request asking for write permission in the future. If the data is shared but the core is only going to read it, then, the core should ask for read permission only. Asking for write permission would imply invalidating the other copies and would be wasteful in this case. In addition, generally speaking, depending on the number of cycles between when a data is read by a core and when it is written by that core, it can be more interesting to obtain the write permission early (when reading) in order to not slow down the write if it is close in time, or late (when performing the write), in order to allow other cores to keep their copies as long as possible.

Those access and sharing patterns are known by the developer. It would therefore be interesting to express those patterns in the source code to help the hardware make correct decisions at runtime, rather than just trying to guess what that decision might be. The RISC-V instruction set being open source and extensible, it provides us with an opportunity to study this technique, by adding instructions conveying with what access and sharing patterns a data is being manipulated.

2 Objectives

The thesis is built around three items. First, reviewing the litterature on sharing patterns in multicore programs as well as hardware techniques to identify them will allow the candidate to identify patterns that we would want to express via dedicated instructions. Second, quantifying the frequency at which such patterns occur at runtime in typical multicore workloads (PARSEC [1]) will be needed, in order to confirm the usefulness of such patterns and prioritize which patterns to support. Finally, the candidate will study the performance gain that the introduction of new instructions will bring by i) Adding support for those instructions in gcc or LLVM (through intrinsic) ii) Adding those instructions in the PARSEC benchmarks and iii) By simulating PARSEC benchmarks on a multicore processor model to which the candidate will have added support for the new RISC-V instructions, both in the processor core and the blocks handling cache coherency. For this last step, a high level simulator will be used (gem5 [2]), and hardware will not be developped using VHDL or Verilog.

3 Expected Skills

3.1 Personal Skills

The candidate should be aware that a PhD programme is vastly different from a BS or MS programme. Pursuing a PhD requires strong motivation and the ability to focus on a specific topic for three years.

— The candidate should expect to be autonomous in developing software, experiments, and analyzing results.
— The candidate should be able to clearly express their ideas and conclusions, and to motivate their research directions.
— The candidate should be open to constructive criticism from their peers and supervisors.

3.2 Technical Skills

— Programming : C/C++. Strong knowledge and understanding of data structures, testing and debugging tools.

— Linux scripting : Python, bash or other.
— At least basic level in computer architecture (caches, virtual memory, pipelining) and Instruction Set Architecture concepts. Advanced level is a plus.
— At least basic level in cache coherence concepts (coherence protocols, snooping vs. directory)
— Understanding of synchronization concepts for parallel programming (threads, locks...)
— Strong level of english is required as scientific articles are written and presented in english.

4 General Information

Funding The 3-year PhD programme is funded by Inria Défi Cocorisco, as a result the employer is Inria. Monthly salary is 2082 euros "brut" (1675 euros "net") in the first year and 2190 euros "brut" (1760 euros "net") in the second and third year.

Start Date Applications are welcome as soon as possible, with a starting date in September 2025 at the earliest.

Requirements The candidate should hold a Master of Science in Computer Science or Computer Engineering, or any equivalent degree.

Direction The thesis will be directed by Prof. Frédéric Pétrot (Grenoble INP) and co-supervised by Associate Prof. Julie Dumas (Grenoble INP) and Arthur Perais (CNRS).

Institution The PhD will be hosted in the TIMA laboratory in the SLS team : https://tima.univ-grenoble-alpes.fr/research/sls. At the start of the PhD, SLS will have become the MADMax Inria team (Inria center of Grenoble). TIMA welcomes applicants with diverse backgrounds and experiences. We regard gender equality and diversity as a strength and an asset.

Doctoral School The PhD programme is tied to the MSTII Doctoral School of Université of Grenoble : https://edmstii.univ-grenoble-alpes.fr/mstii-doctoral-school/.

Teaching During the 3 years, teaching (labs and practicals) is possible but english classes are the exception, so if the candidate wishes to teach, fluency in french is highly preferable. Teaching is remunerated at around 40€ gross per hour in front of students.

Mandatory Training During the 3 years, the candidate will be expected to attend 120 hours of training, split in three topical buckets of 40h : Scientific, Transversal and Professional Project. More information : https://edmstii.univ-grenoble-alpes.fr/formation/.

5 Contact :

Please send your resume and a couple paragraphs providing context for your application (no need for a full cover letter, we just want to know why you are interested) :

Arthur Perais, Arthur.Perais@univ-grenoble-alpes.fr
and
Julie Dumas Julie.Dumas@univ-grenoble-alpes.fr

References

[1] https://parsec.cs.princeton.edu/parsec3-doc.htm
[2] https://www.gem5.org/