Chapter 5
Conclusions

Locality of memory accesses is one of the most important aspects to be considered when designing an architecture or developing software. With the introduction of multicore architectures, the memory hierarchy had to evolve to able to provide the necessary bandwidth to several cores operating in parallel. With this evolution, memory hierarchies started to present several caches in the same level, some levels shared by multiple cores, and other private to a core. Another important step was the incorporation of a memory controller inside the processor, in which multiprocessor systems presented NUMA characteristics. Due to the introduction of such technologies, the performance of memory hierarchies and the systems as a whole were even more dependent on memory locality. In this context, techniques such as sharing-aware thread and data mapping are able to increase memory locality and thereby performance. Our experiments indicate performance improvements of up to 200% in a scientific application.

Lots of related work on the area of sharing-aware mapping has been proposed, with a wide variety of characteristics and features. The majority of the proposals perform only static mapping, which are able to handle only applications whose memory access behavior keeps the same along different executions. Most work also only handles thread or data mapping alone, not both together. Most related work that is able to handle both thread and data mappings and operate online, during the execution of the application, have a high trade-off between accuracy and overhead. To achieve a higher accuracy, they have to increase the overhead of their memory access behavior detection as well. Some proposals are able to achieve high accuracy with low overhead, but require special hardware support.

E. H. M. Cruz et al., *Thread and Data Mapping for Multicore Systems*, SpringerBriefs in Computer Science, https://doi.org/10.1007/978-3-319-91074-1_5