Configurable Data Structure Layout for Memory Hierarchies

Máté Karácsony, Máté Tejfel

Dept. of Programming Languages and Compilers,
Eötvös Loránd University, Budapest
`{kmate,matej}@elte.hu`

Developments of the last decades resulted in an increasing gap between processor and memory speeds [?]. Therefore in high-performance computations the main bottleneck is memory access. With the spread of multi-core processors and data-parallel applications this effect only increases as multiple devices are contending for the same resource at the same time. A practical solution to this problem is to implement memory hierarchies utilizing multiple levels of memories with different capacities and access profiles. The most common implementations use hardware controlled cache memories. However, for embedded and low-power systems it is more beneficial to apply programmable scratchpad memories instead [?]. The overall performance of scratchpad-aware software heavily depends on how the data is distributed and accessed in different memory layers.

Systems with programmable memory hierarchies are usually programmed using low level languages. In these languages, the distribution of data in different memory layers is reflected by the data structure definitions and the operations on those structures [?]. Choosing the data layout for optimal performance is clearly a set of design decisions. The validation of these choices is usually possible only by profiling the software. Therefore further optimizations or feature extensions often lead to changes in these design decisions. Finally these changes will be reflected in the source code as well, often causing significant non-trivial modifications.

We introduce *configurations over data structures* to enable deferring these design decisions. Configurations enable fine-grained description of data layout of an application. We demonstrate their use in synthesizing data structure definitions and accessing code through implementations in C macros and C++ templates. We also present how to use configurations to describe source-to-source transformations, or to specify data layout as a built-in language construct in a domain-specific language named Miller [?].

References

[1] Carlos Carvalho. The gap between processor and memory speeds. In *Proceedings of IEEE International Conference on Control and Automation*, 2002.

[2] Preeti Ranjan Panda, Nikil D. Dutt, and Alexandru Nicolau. Efficient utilization of scratch-pad memory in embedded processor applications. *Proceedings of the 1997 European conference on Design and Test. IEEE Computer Society*, 1997.

[3] Nieplocha, Jarek, Robert Harrison, and Ian Foster. Explicit management of memory hierarchy. *Advances in High Performance Computing.* Springer Netherlands, 1997. 185-199.

[4] Németh, Boldizsár, and Zoltán Csörnyei. Stackless programming in Miller. *Acta Universitatis Sapientiae, Informatica 5.2*, 2013. 167-183.