An algorithmic skeleton for hybrid NUMA-aware loop tiling and its derivatives

Al. Levchenko

SPbPU Supercomputer Center, St. Petersburg, Russia

The modern design of automatic loop transformation algorithms remains challenging due to the growing complexity of deep memory hierarchies against the background of the current efforts toward exascale and beyond. Overcoming the problem of modeling locality and non-uniformity of memory access is a crucial step to deliver performance portability for a wide range of the computational kernels across multi-level memory. Target architectures of this class introduce special conditions to satisfy the automatic loop transformation algorithms, regardless of whether these techniques have relied on affine transformations in the polyhedral compilation framework or non-affine transformations on abstract syntax tree (AST). Singular characteristics of modern and future multi-level memory architectures will inevitably require an automatic parameterized definition of tiled loop characteristics in hierarchical tiling strategy to map it according to the configuration of caches, NUMA nodes and translation lookaside buffer. Another important criteria is a possibility to capture locality-enhancing features of polyhedral optimizers like Pluto [1]. Moreover, the resulting optimized code should be experimentally evaluated on a real large-scale system with deep memory hierarchies using polyhedral benchmark suite.

This paper introduces an algorithmic skeleton for the hybrid NUMA-aware loop tiling and the first experimental results of the derived algorithms based on it. The prime objective of the proposed approach is to achieve the performance portability for the computational kernels on HPC machines with deep memory hierarchies. To this end, it is essential to improve tendencies in program behavior to access neighboring memory regions near the regions that have been recently accessed, or to reference the same recently referenced memory addresses, i.e., to improve spatial or temporal locality, respectively. This goal is achieved via solving the problem of combining affine scheduling for locality with hierarchical parametric tiling performed in a traditional syntactic/AST-based manner. Loop tiling, a fundamental transformation used to exploit spatial and temporal data reuse, is applicable at both stages of the following two approaches. The proof of concept for the hybrid affine+syntactic approach was proposed by Shirako et al. (2014) [2]. Complementing the prior research of performing loop tiling in the combined approach of this sort, this work contributes to (i) prioritization of spatial and temporal data locality and (ii) modeling of non-uniformity effects of memory access. The Pluto polyhedral parallelizer and locality optimizer [1] algorithm was used to prioritize the locality by iteratively finding linearly independent hyperplanes based on the objective to minimize dependence distances. The stage of the AST-based approach allows to use a parameterized tiling strategy when tile sizes are symbolic parameters of different configurations of hierarchical memory architecture. In this way, it is possible to generate various parameters of tiles, such as size, shape or form, when the effective set of the tiles is determined on the fly. Parameterized tiling provides a near-optimal performance of the code via changing the granularity of the tile computations at runtime in accordance with the memory hierarchy. It is this feature that has formed the concept of NUMA-aware tiling. This method starts with calculation of the tile size coefficients and shape finding for the certain level of hierarchy in the direct memory access cost model. Further, the groups of memory hierarchy levels are defined, with tile size selection (TSS) and tile shape selection (TSHS) algorithms performed for the groups. The implementation of the proposed approach as a skeleton corresponds to the existing scientific efforts of the recent years in the aspects of designing permutational target-specific TSS/TSHS algorithms, primarily for stencil computations. The variants of Acoherent Non-Uniform Loop Tiling (ACNULT) algorithm were designed on the basis of the proposed skeleton. ACNULT(1,2) implements TSS and

```
Input: Intermediate representation of program P
Output: Intermediate representation of optimized P code
begin
    /* Polyhedral transformations with locality
       prioritization                                  */
    begin
        P := loop fusion;
        P := loop permutation;
        { P := experimental loop skewing/tiling };
    end
    /* Loop transformations for non-uniformity         */
    begin
        P := loop skewing;
        P := modeling DMA costs for available levels of
          memory hierarchies;
        P := loop tiling with ACNULT(1) with TSS;
        P := loop tiling with ACNULT(2) with TSHS;
    end
end
```

Figure 1. Overall algorithmic skeleton pipeline extended with ACNULT variants

TSHS algorithms, respectively. Acoherence in this context means that the algorithm is based solely on hardware-based cache coherence schemes, and it works with the globally addressable shared memory of the multi-machine macronode in the same way as with the memory of a general-purpose cluster node. Figure 1 illustrates an algorithmic skeleton.

A preliminary experimental evaluation of ACNULT derivatives was performed using multi-machine macronode with cache coherent non-uniform memory access, which is part of the large-scale ccNUMA system. Even the smallest machine memory included L1d/L1i/L2/L3 cache levels along with a three-tier NUMA nodes hierarchy and configurable L4-cache for a scalable directory based on cache coherence protocol. PolyBench/C kernels were executed with EXTRALARGE dataset size to provide highest workload. At the early stage, stencil computations formed a set of kernels of interest. One aspect of the stencils is the limitation of memory bandwidth due to the low ratio of floating-point operations per byte of data read from memory. Therefore, performance improvement of the stencils will inevitably require a data movement reduction. The highest speed-ups (relative to non-optimized code) were gained with ACNULT(1) for stencils, in particular, with alternating direction implicit solver, 2-D finite different time domain kernel, 1-D Jacobi stencil computation, 2-D Jacobi stencil computation, and 2-D Seidel stencil computation. Improving the locality of these methods by the ACNULT-like techniques will raise the performance of a wide range of computational kernels across deep memory hierarchies. In the nearest future, the work will focus on multi-level tiling in the polyhedral model.

References

1. Aravind Acharya, Uday Bondhugula, Albert Cohen. Polyhedral auto-transformation with no integer linear programming // 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2018). Philadelphia, PA, USA. June 18–22, 2018. P. 529–542. DOI: 10.1145/3192366.3192401

2. Jun Shirako, Louis-Noël Pouchet, Vivek Sarkar. Oil and water can mix: an integration of polyhedral and AST-based transformations // SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans, LA, USA. November 16–21, 2014. P. 287–298. DOI: 10.1109/SC.2014.29