Automated Parallelization of Non-uniform Convolutions on Chip Multiprocessors

Y. Zhang[1], M. Kandemir[1], N. Pitsianis[2] and X. Sun[2]
[1]Department of Computer Science and Engineering, Pennsylvania State University, PA, 16802, USA
[2]Department of Computer Science, Duke University, NC, 27708, USA
{yuz123@psu.edu, kandemir@csepsu.edu, Nikos.P.Pitsianis@Duke.edu, xiaobai@cs.duke.edu}

Introduction

This paper introduces an approach for automatic parallelization of unequally-spaced convolutions on chip multiprocessors (CMPs). CMPs are very promising candidates for digital processing in signal and image systems with high throughput and low power consumption, compared to uniprocessor based architectures. As CMPs are emerging and evolving in increasing diversity and complexity, automated parallelization of digital signal processing (DSP) algorithms on CMPs becomes essential in embracing, employing and utilizing these architectures in high-performance signal and image systems.

We focus in particular on automatic parallelization of non-uniform local convolutions (NuCONV) because this computation is a key step linking the use of the fast Fourier transform (FFT) to the data sampled at unequally-spaced locations or with non-uniform sample density. The combination of the NuCONV with the FFT is referred to as the Non-uniform FFT (NuFFT) [1, 2, 3, 4, 5]. Our approach for automatic parallelization of non-uniform convolutions consists of three main components: (i) a unified model for CMP architectures; (ii) an algorithmic framework for data localization, and (iii) a systematic scheme for coupling the architectural model and algorithm framework with specific data sets and CMP features.

Diversity in CMP connectivity and capacity

Chip multiprocessing [7, 8, 9] represent a relatively new design and technology paradigm with respect to the limitations in design complexity and power consumption of current architectures. In this new paradigm, multiple cores are connected via an on-chip memory hierarchy (typically in the form of multi-level caches) and an on-chip network. This technology is presently used by almost all major processor manufacturers across different types of systems they produce, including personal computers built from multicores by Intel and AMD, game consoles such as the Sony-Toshiba-IBM Cell in Sony PlayStation 3 and Xenon processor in Xbox 360, the graphics processing units (GPUs) by NVIDIA and AMD, as well as in various embedded systems.

CMPs differ from one another in core-cache and cache-cache connectivity, even within those produced by the same chip manufacturer. For example, Intel quad-core Harpertown has a two-level cache hierarchy with private L1s and each pair sharing an L2; Intel quad-core Nehalem has a three-level cache hierarchy with private L1 and L2 caches, and an shared L3; and Intel Dunnington has private L1s, shared L2s for each pair of cores, and an L3 at the bottom level for all cores. CMPs differ also in core processing units (types, issue widths, register configurations, etc), on-chip cache capacities, off-chip bandwidths and data transfer protocols. Such diversity is expected to be even more pronounced in future CMPs.

Clearly, in this spectrum of CMPs, porting a parallel application from one machine to another is not an easy task, if one wants to achieve high performance. Manual parallelization and fine-grain performance tuning for each target CMP architecture becomes impractical in response to the fast pace of architectural advance on one side and the demand of timely adapting to and utilizing modern architectures on the other side. We characterize the regular CMP architectures at a high level by a connectivity-capacity hierarchy. In its simple form, a cache hierarchy can be represented using a tree, with the root standing for the off-chip memory, the leaves for the core processing units, and the intermediate nodes corresponding to the on-chip caches in between. In this representation, the nodes have capacity attributes and the edges have data transfer latency attributes. The tree structure is related to the graph for a more complex on-chip network as a spanning tree.

Non-uniform local convolution

In NuFFTs [1, 2, 3, 4, 5], a locally supported convolution kernel function h (and its Fourier dual H) is chosen for accurate and efficient translation of data, via direct evaluation, from unequally spaced locations to a Cartesian grid, or vice versa. For illustration purposes, we consider the case of translating *source* samples at unequally-spaced locations s_j to *target* data at t_i on a Cartesian grid that is determined together with the kernel function h:

$$u(t_i) = \sum_{\|s_j - t_i\|_\infty < W} h(t_i - s_j)\, q(s_j), \quad t_i, s_j \in R^3, \tag{1}$$

where W is the window size describing the convolution support along each dimension. Note that this operation is essentially a matrix-vector product with a sparse matrix. Under certain assumptions on the data q, the NuFFT may be viewed as permitting an arbitrary sample distribution. Equation (1) suggests two loops in a program that implements NuCONV, with the outer loop going over the target indices and the inner loop going over the source indices. We swap the positions of these loops for exploiting data locality and parallelism, because, for each source point, the number of the target points within the window and their locations are easy to determine, unlike the other way around.

Geometric tiling for data localization

In the irregularly sparse case, the index tiling scheme for dense matrix-vector products fails to serve the purpose of enhancing data locality and reuse. Instead, we cluster the source data into geometric cells on a Cartesian grid, which induces geometric tiling of the convolution matrix in the source indices [6]. This source clustering and tiling process visits each source data point only once, and it is highly parallelizable. Geometric tiling over the target space is simply a partitioning of the Cartesian indices along each dimension. Now, the source samples in a source cell update the same set of targets in the neighboring target cells. This data localization scheme is parameterized with the cell size on each of the source and target sides and the ordering in which the source cells are visited.

Automatic data localization and task parallelization

Automatic parallelization of NuCONV with particular source-target data sets on a specific CMP involves finalizing the cell (or tile) size, tile traversing order, scheduling the computation among multicores and organizing data accesses for effective use of available on-chip caches. In other words, this process instantiates and attempts to optimize the mapping between the architectural model and the algorithmic tiling framework with specific input consisting of data sets, algorithmic parameters and architectural features. In particular, the geometric cells may be grouped together or split further, in adaptation to the target cache hierarchy. The mutual exclusion in updating target data is ensured by the zero-overlapping rule in concurrent distribution and by the maximal-overlapping rule in successive tiles during data traversal.

Preliminary evaluation

To evaluate our automated strategy, we performed two sets of experiments for the 2D case with 6K×6K source samples in the same spatial domain of a 50×50 Cartesian grid. The datum at each source sample updates 10×10 targets. The first set of experiments study the scalability of the adapted parallel convolution when increasing the number of cores in three target architectures we have (Harpertown, Nehalem, and Dunnington). These results are presented in Figure 1 and indicate that the performance scales well. Our second set of experiments investigates the importance of customizing the data localization and parallelization for convolution based on the target architecture. The plot in Figure 2 shows, for a 4-core configuration, three groups of bars, each of which corresponding to an architecture on which the code is executed. In each group, the three bars represent the performance of the three different versions adapted to different architectures. For example, the third group of bars corresponds to the performance numbers collected when running all the versions on the 4 cores of the Harpertown machine. In particular, the first, second, and third bars in that group correspond to the parallel convolution adapted to Dunnington, Nehalem, and Harpertown architectures, respectively. The bars in each group are normalized with respect to the version which is tailored for the target architecture. The results show that the version with customized data localization and parallelization for the target architecture has best performance on that machine.

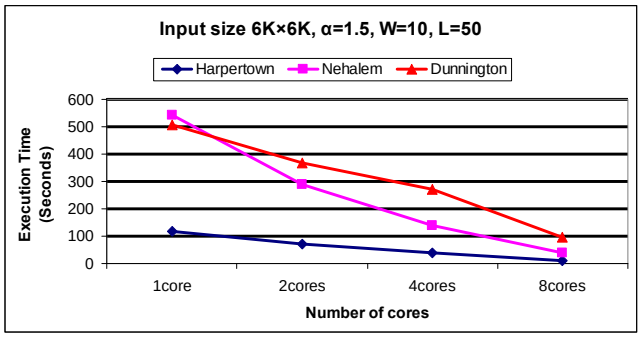

Figure 1: Execution time scalability results on machines of three different Intel multi-core architectures.

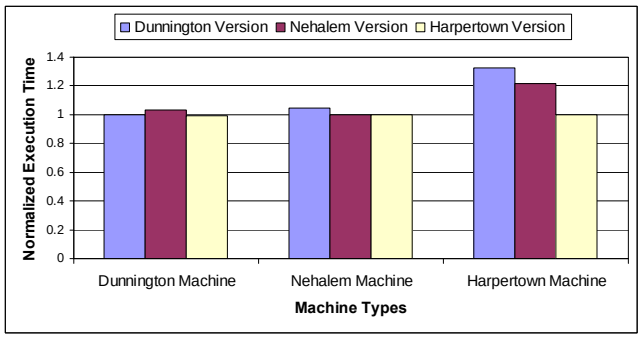

Figure 2: Results showing the importance of adapting parallel convolution to the target CMP architecture.

References

[1] A. Dutt and V. Rokhlin, "Fast Fourier transforms for nonequispaced data", *SIAM Journal on Scientific Computing*, 1993.

[2] D. Potts, "The Nonequispaced FFT: An indispensable algorithm for applied science", *ACM Computing Reviews Hot Topic*, 2008.

[3] D. Potts, G. Steidl, and A. Nieslony, "Fast convolution with radial kernels at nonequispaced knots", *Numer. Math.*, 2004.

[4] G. Beylkin, "On the fast Fourier transform of functions with singularities", *Applied and Computational Harmonic Analysis*, 1995.

[5] L. Greengard and J. Lee, "Accelerating the Nonuniform Fast Fourier transform", *SIAM REVIEW 46(3)*, 2004.

[6] T.S. Sorensen, T. Schaeffter, K.O. Noe, and M.S., Hansen, "Accelerating the nonequispaced fast Fourier transform on commodity graphics hardware", *IEEE Transactions on Medical Imaging*, 2008.

[7] K. Olukotun, L. Hammond, and J. Laudon, "Chip multiprocessor architecture: techniques to improve throughput and latency", 2007.

[8] P. Crowley, M. A. Franklin, J. Buhler and R.D. Chamberlain, "Impact of CMP design on high-performance embedded computing", *High Performance Embedded Computing workshop*, 2006.

[9] Y. Li, K. Skadron, Z. Hu and D. Brooks, "Performance, energy, and thermal considerations for SMT and CMP architectures", *HPCA-11*, 2005.