Parallelizing BLIS-style Matrix Multiplication for TinyML on Ultra-Low-Power multicore RISC-V

Cristian Ramírez[1], Adrián Castelló[1] and Enrique S. Quintana-Ortí[1*]

[1]Universitat Politécnica de Valéncia

Abstract

We present our BLIS-like general matrix multiplication (GEMM) for the multicore RISC-V processor embedded in the GAP8 platform, and evaluate the parallel performance of this computational kernel in a deep learning (DL) scenario. Concretely, we leverage the IM2COL approach to transform the convolution layers into GEMM, and develop a parallelized algorithm tailored for this ultra-low-power processor, targeting DL inference with a MobileNet-v1 convolutional model. Our experimental results demonstrate that the GEMM algorithm achieves a significant speed-up compared to the sequential version. This work thus contributes to the development of high-performance and energy-efficient machine-learning applications for multicore RISC-V processors.

Introduction

Implementing deep learning (DL) on edge devices for Internet-of-Things (IoT) appliances is critical in order to enhance privacy. Moreover, moving computations from the cloud to IoT nodes, closer to the sensors, significantly reduces the amount of data sent over the network, thereby decreasing latency and power consumption [1, 2]. The wide variety of IoT applications, many of which employ DL-based technologies, has led to a diverse range of edge processor architectures, which include RISC-V ISA-based cores. This diversity together with the severe constraints of these devices on power consumption, memory capacity, and computational performance, emphasize the utmost need to carefully select the algorithms and optimize the software running on them.

```
1  void Gemm_B3C2A0( C[M][N], A[M][K], B[K][N] ){
2    for ( j_c = 0; j_c < N; j_c += N_c )
3      for ( p_c = 0; p_c < K; p_c += K_c ) {
4        //Pack B_c
5        B_c := B[p_c : p_c + K_c - 1][j_c : j_c + N_c - 1];
6        for ( i_c = 0; i_c < M; i_c += M_c ) {
7       //Pack C_c
8          C_c := C[i_c : i_c + M_c - 1][p_c : p_c + K_c - 1];
9          for ( p_r = 0; p_r < K_c; p_r += K_r )
10           for ( i_r = 0; i_r < M_c; i_r += M_r )
11             // Micro-kernel
12             for ( j_r = 0; j_r < N_c; j_r++ )
13               C_c[i_r : i_r + M_r - 1][j_r : j_r + N_r - 1]
14               += A_c[i_r : i_r + M_r - 1][p_r]*
15               B_c[p_r][j_r : j_r + N_r - 1];
16       }
17     }
18 }
```

Figure 1: GEMM *B3C2A0 variant algorithm.*

The general matrix multiplication (GEMM), $C = A \cdot B$, where A, B, C represent matrices of the appropriate dimension, is a critical operation for DL inference and training. Many high-performance implementations of GEMM in open-source and commercial libraries adhere to the GotoBLAS ideas [3] to formulate the algorithm as a collection of five nested loops around a key component, known as the *micro-kernel*, which performs a small-scale GEMM operation. In addition, they apply blocking to the matrix operands to improve the cache hit ratio. For this purpose, the loops in the GEMM operation are reordered in a specific manner, and the loop strides are set correspondingly. This results in different variants of the operation, which place blocks of A, B, C on specific levels of the memory hierarchy [4, 5].

Proposed Approach

We apply the techniques advocated by BLIS [6], rearranging the structure of the GEMM algorithm in order to accommodate the memory hierarchy of the GAP8 processor. This architecture is composed of 9 RISC-V cores (1 fabric controller, or FC, and an 8-core cluster), plus a number of scratch pad memories, asking the programmer to orchestrate the data movements between these memory levels as part of the algorithm.

Optimizing the IM2COL-based convolution for the GAP8 system results in the following contributions:

- We implement a BLIS-style GEMM algorithm that builds the GEMM micro-kernel upon the dot product, yielding an algorithm where (a block of) matrix A resides in the processor vector registers; see algorithm B3C2A0 in Figure 1. This organization thus takes advantage of specialized RISC-V extensions of instruction set architecture (ISA) for the GAP8 processor.
- We develop and validate an analytical performance model for GEMM on the GAP8 processor that delivers accurate estimations when integrating different micro-kernels into the algorithm.

*Corresponding author: quintana@disca.upv

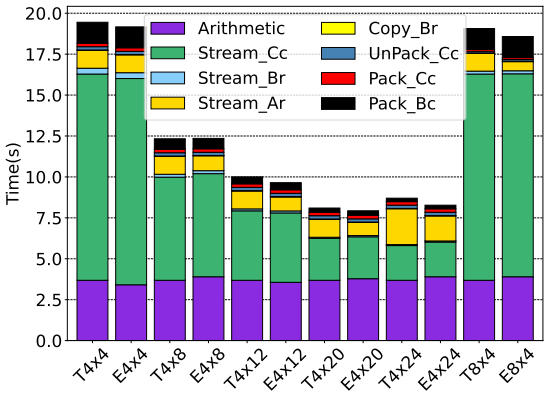

Figure 2: *Theoretical (T) and experimental (E) execution time for layer #12 of MobileNet-v1, for different micro-kernels, using a single cluster core in GAP8.*

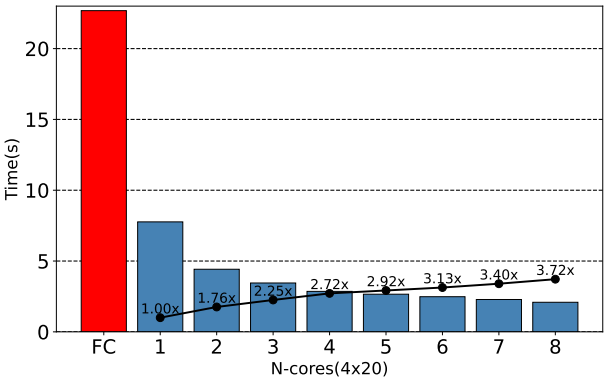

Figure 3: *Execution time and speed-up for layer #12 of MobileNet-v1, using the FC and the cluster cores in GAP8.*

- We assemble an instance of GEMM optimized for 8-bit integer (INT8) arithmetic that can be easily adapted for fixed-point arithmetic.
- We perform an experimental evaluation of the realization of GEMM on a single core (FC or cluster core) and optimize it for parallel execution on the RISC-V multicore processor using the best micro-kernel indicated by the model.

Experimental Results

We next evaluate the performance of our sequential and parallel implementations of GEMM on the GAP8 processor. For brevity, we discuss the results for layer #12 of the MobileNet-v1 model only. (Similar results were obtained for other layers of this model.) Applying the IM2COL transform to the target layer yields a GEMM of dimensions $m = 256$, $n = 784$, and $k = 2304$. To start with, we validate the analytical performance model, comparing the theoretical breakdown of execution time with the experimental values obtained from a real run. Figure 2 shows the time for the sequential implementation of the B3C2A0-based GEMM when integrating distinct micro-kernels. The bars labelled with the prefix "T" correspond to the theoretical time estimated by the analytical model while those labelled with the prefix "E" are from the real executions. The plot exposes that the model provides accurate estimations of the execution times, which can be useful for selecting the best micro-kernel for this RISC-V microprocessor. From this study, we can identify that the best micro-kernel for the target layer is 4×20.

In the parallel B3C2A0-based GEMM, we distribute the operations into two parts: On the one hand, the outermost loops of the algorithm up to the loop indexed by i_r in Figure 1 are executed on the FC. On the other hand, from that point inwards, the execution is dispatched to the cluster cores.

Figure 3 reports the overall execution time and speed-up for the execution of layer #12 of MobileNet-v1, using the FC core and the cluster cores in GAP8. The results expose the benefits of parallelizing the sequential code.

Conclusions

In this work, we have implemented and evaluated a parallel BLIS-like algorithm for the RISC-V-based GAP8 platform. In addition, we have compared our algorithm with a theoretical model that indicates the best micro-kernel size for a given GEMM scenario. Finally, we have evaluated the scalability of this solution when using the 8-core RISC-V cluster.

References

[1] K. Hazelwood et al. "Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective". In: *IEEE Int. Symp. HPC Architecture*. 2018, pp. 620–629.

[2] C. Wu et al. "Machine Learning at Facebook: Understanding Inference at the Edge". In: *IEEE Int. Symp. HPC Architecture*. 2019, pp. 331–344.

[3] Kazushige Goto and Robert A. van de Geijn. "Anatomy of High-performance Matrix Multiplication". In: *ACM Trans. Math. Softw.* 34.3 (May 2008), 12:1–12:25. ISSN: 0098-3500.

[4] Tyler M. Smith and Robert A. van de Geijn. "The MOMMS Family of Matrix Multiplication Algorithms". In: *CoRR* abs/1904.05717 (2019). arXiv: 1904.05717.

[5] A. Castelló, E. S. Quintana-Ortí, and F. D. Igual. "Anatomy of the BLIS Family of Algorithms for Matrix Multiplication". In: *30th Euromicro Int. Conf. Parallel, Distributed and Network-based Processing (PDP)*. 2022, pp. 92–99.

[6] Field G. Van Zee and Robert A. van de Geijn. "BLIS: A Framework for Rapidly Instantiating BLAS Functionality". In: *ACM Trans. Math. Softw.* 41.3 (2015), 14:1–14:33.