RISC-V Vector Extension. A Case Study on Time Series Analysis

Jose Sanchez-Yun[1], Ivan Fernandez[2], Eladio Gutierrez[1], Ricardo Quislant[1] and Oscar Plata[1]

[1]Department of Computer Architecture, University of Málaga.
[2]Computer Sciences Department, Barcelona Supercomputing Center.

Abstract

Time series analysis is a topic of great interest, as it enables modeling, prediction, and understanding of sequential events across various domains. One of the most powerful tools in this field is Matrix Profile, which allows for scalable and accurate detection of anomalies and repetitive patterns. Specifically, SCAMP has emerged as one of the most efficient methods for computing the Matrix Profile due to its robustness, efficiency, and high parallelization capability. Vectorization is a powerful optimization technique for these algorithms, as it exploits SIMD instructions in modern CPUs to enhance computational efficiency. In this context, the RISC-V Vector Extension (RVV) introduces a flexible vector model with dynamic lengths. This enables algorithm optimization across different hardware platforms without requiring source code modifications. In this paper, we vectorize the SCAMP algorithm using the RISC-V Vector Extension and analyze the achieved speedup compared to the non-vectorized baseline. Additionally, we explore the benefits of dynamic vectors and compare them with alternative implementations. The results show speedup improvements of up to 94× over the sequential version of the algorithm using vectors of 8K bits and floating-point data of 64 bits.

Introduction

Time series analysis aims to study data samples organized sequentially over time. Its objective is to identify patterns (motifs), trends, seasonality, and anomalies (discords). This analysis provides valuable insights in various applications, including economics, medicine, meteorology, and epidemiology. The state-of-the-art approach for pattern and anomaly discovery is the Matrix Profile [1]. It offers a scalable and automated solution for time series analysis, providing a representation of the relationships between each subsequence within the time series. This enables efficient processing of large datasets while maintaining reasonable computation times.

One of the most efficient methods for computing the Matrix Profile is SCAMP [2]. This algorithm employs a highly parallel approach that fully exploits Euclidean distance computations. The performance of these algorithms depends largely on repetitive computations, including floating point calculations and profile comparison operations. Vectorization can optimize these computations by leveraging SIMD instructions, which allow executing the same operation on multiple data samples within a reduced clock cycle count.

RISC-V [3, 4] is an open-source ISA that has grown significantly since its inception due to its flexibility, scalability, and customization capabilities. It has gradually become a key component in global chip innovation. The vector extension (RVV) [5] introduces a set of vector registers and operations to the base ISA. Additionally, it supports vector length agnostic SIMD operations, where the hardware can be prompted to reveal the vector size. As a result, the same code runs on RISC-V processors with varying vector sizes without requiring modifications or recompilation.

In this paper, the SCAMP algorithm is vectorized using RISC-V Vector Extension (RVV) to exploit the CPU's SIMD operations and enhance the algorithm's performance. The experiments were conducted using the gem5 simulator [6], running multiple simulations while varying the vector register size across three time series of different lengths. The results show speedup improvements of up to 94× over the sequential algorithm using vectors of 8K bits and floating-point data of 64 bits.

SCAMP Vectorization

Among the many concepts of the RISC-V Vector Extension, three are key to this study: VLEN, the number of bits in the vector registers; VLMAX, the maximum number of elements that can be processed in a single operation, determined by the Selected Element Width (SEW) and VLEN; and VL, specifying the number of elements to be updated with results from a vector instruction, which must be less than or equal to VLMAX. SCAMP divides the time series into subsequences of a fixed length, determined by a sliding window. The algorithm searches for the most similar counterpart of a subsequence by computing the normalized Euclidean distance. This process constructs the Distance

Matrix, where each cell stores the Euclidean distance between two subsequences, and row and column indices represent their positions within the time series. The Distance Matrix is computed diagonally, since this way each element of the diagonal can be incrementally derived from the previous one, saving multiple computations to obtain the Euclidean distance. After computing all pairwise distances, the algorithm identifies the subsequence with the minimum distance for each subsequence. This is achieved by computing the minimum value through columns and rows and recording its index, resulting in the Matrix Profile and Matrix Profile Index arrays.

To vectorize the SCAMP algorithm, the diagonals of the Distance Matrix are packed together in vectors of size VLEN, processing VL elements at a time. For most of the computation, VL is set to VLMAX to maximize hardware utilization. If the Matrix Profile size is not a multiple of VLMAX, the last iterations leave empty slots in the vector register. In other architectures, such as x86 with AVX extensions, the vector instructions always process VLMAX elements. When there are not enough elements to fill the register, padding must be used [7]. In contrast, RVV allows VL to be dynamically adjusted (VL $\leq$ VLMAX). When processing the last elements of the Matrix Profile, VL is set precisely to match the remaining elements. This approach simplifies programming, as there is no need to manually pad the vector register and application arrays. It also enhances code portability and scalability, avoiding modifications or recompilation when running on RISC-V processors with different VLEN values.

Experimental Evaluation

The experiments aim to achieve two main objectives. First, to assess the performance improvement resulting from vectorization. For this purpose, SCAMP is executed using RVV and compared against its baseline sequential version. Second, the impact of VLEN on execution time is analyzed by running tests with four different VLEN sizes. The benchmarks include three time series of varying lengths, all based on real data, namely, penguin, audio and human, which contain 109842, 20334, and 7997 elements, respectively.

The experiments are conducted using the gem5 simulator [6] on a 16-core RISC-V processor running at 2 GHz. The system has 32 GiB of single-channel DDR4 RAM and a three-level cache with 32 KiB, 64 KiB, and 16 MiB at the L1, L2, and L3 levels, respectively.

Figure 1 shows the execution time for each time series in both the sequential and vectorized versions, with different VLEN values. Even with a low VLEN of 256, the average speedup across the three time series is 3.15× compared to the sequential version.

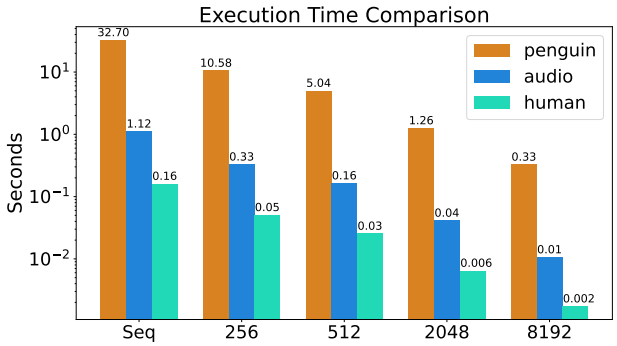

Figure 1: *SCAMP execution time comparison (log scale): Sequential vs. vectorized with varying VLEN.*

With VLEN set to 8192, the average speedup reaches 93.81×, demonstrating the significant performance gains achievable through vectorization in these algorithms. Additionally, the execution time reduction follows an almost linear trend as VLEN increases. This is expected, as larger VLEN values proportionally enhance data processing capacity per clock cycle.

Conclusion

This paper presents a vectorized implementation of the SCAMP algorithm for time series analysis using the RISC-V Vector Extension (RVV). The study examines the benefits of RVV and evaluates its efficiency by comparing execution times against the non-vectorized version varying the vector size of the underlying hardware. The results show speedup improvements of up to 93.81× compared to the sequential version, highlighting the importance of vectorization in these algorithms.

References

[1] *The UCR Matrix Profile Page*. `https://www.cs.ucr.edu/~eamonn/MatrixProfile.html` [Accessed: 14/03/2024].

[2] Zachary Zimmerman et al. "Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond". In: *Symp. on Cloud Computing*. SoCC '19. 2019, pp. 74–86.

[3] Andrew Waterman et al. *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. Tech. rep. UCB/EECS-2011-62. Dept. EECS, Univ. California, Berkeley, 2011, pp. 97–116.

[4] K. Asanović and D. A. Patterson. *Instruction sets should be free: The case for RISC-V*. Tech. rep. UCB/EECS-2014-146. Dept. EECS, Univ. California, Berkeley, 2014.

[5] RISC-V Unprivileged Horizontal Committee. *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture*. 2024.

[6] Nathan Binkert et al. "The gem5 simulator". In: *SIGARCH Comput. Archit. News* 39.2 (Aug. 2011), pp. 1–7.

[7] R. Quislant et al. "Time series analysis acceleration with advanced vectorization extensions". In: *J Supercomput* 79 (2023), pp. 10178–10207.