DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Thesis type (Master's Thesis in Informatics, . . . )

# Thesis title

Author

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Thesis type (Master's Thesis in Informatics, ...)

# Thesis title

# Titel der Abschlussarbeit

| | |
|---|---|
| Author: | Author |
| Supervisor: | Supervisor |
| Advisor: | Advisor |
| Submission Date: | Submission date |

# Acknowledgments

# Abstract

Time series data mining has become essential for a large variety of disciplines ranging from seismology to medicine. With the use of matrix profiles, most target analyses, especially motif or discord discovery, have become trivial to a large extend. As the original formulations for calculating the matrix profile were relatively ineffective, substantial effort has been invested in computing them efficiently. In particular, the state-of-the-art algorithm SCAMP has convinced with its simplicity, parallelizability, and efficiency.

As FPGA acceleration promises a combination of high performance and competitive energy efficiency, we present a - to the best of our knowledge first - systolic array-based design, based on the SCAMP algorithm, to compute matrix profiles efficiently on FPGAs. Our design mapped to a concrete architecture using a High-Level Synthesis tool. This approach allows us to maintain a high level of abstraction, use streaming abstractions, and enable maintainability and portability across FPGA devices. Kernels synthesized from our design are shown to offer competitive performance in practice, scaling with both compute and memory resources. Finally, we outline possible optimizations as a basis for future work.

# Contents

# 1 Introduction

The understanding of the matrix profile, and its computation, is fundamental to the design of the underlying work. We briefly introduce the reader to concepts related to matrix profiles and their efficient computation, which are the basis for the formulations deployed on the Cerberas accelerator. Time series data mining, as data processing applications in general, greatly profits from streaming abstractions and the ability to express data transformations on a high level. We, therefore, employ a weak scaling approach to mapping the developed algorithm to a concrete hardware design running on the Cerebras accelerator. As the corresponding programming model is relatively novel and constitutes the basis of the introduced design and its subsequent implementation, the reader is introduced to fundamental concepts of the Accelerator and its abstractions. These concepts prove essential to address the complexities of the design while allowing the problem to be structured effectively. This chapter is split into two parts: first, the reader is introduced to the concept (and required notation) of a matrix profile and an algorithm (SCAMP) for its efficient computation; secondly, the chapter establishes fundamentals related to the target device, which make up the basic building blocks for the presented design and its implementation.

# 2 Theoretical Background and Related Works

Matrix profiling has emerged as a powerful technique for time series analysis, offering insights into underlying patterns, anomalies, and recurring motifs. This master thesis investigates the integration of the Cerebras Accelerator, a state-of-the-art hardware architecture designed for accelerating deep learning tasks, into the domain of matrix profiling. The Cerebras Accelerator's unique Wafer-Scale Engine (WSE) architecture, characterized by its massive scale and fine-grained parallelism, presents an intriguing opportunity to significantly enhance the efficiency of matrix profiling computations.

The research delves into the theoretical foundations of matrix profiling and explores the challenges associated with its computational demands. Leveraging the parallel processing capabilities of the Cerebras Accelerator, the study aims to optimize matrix profiling algorithms for enhanced speed and scalability. The thesis investigates how the WSE-2 architecture can be harnessed to efficiently perform essential matrix operations, such as sliding window calculations and motif discovery, crucial for time series analysis.

Furthermore, the research involves practical implementations and performance evaluations to assess the impact of Cerebras Accelerator on the overall efficiency of matrix profiling workflows. Comparative analyses against traditional hardware architectures and accelerators will provide insights into the unique advantages and potential limitations of employing Cerebras in this context.

The findings of this research not only contribute to the evolving field of time series analysis but also shed light on the adaptability and effectiveness of specialized accelerators, like the Cerebras Accelerator, in domains beyond their primary focus. Ultimately, this thesis seeks to bridge the gap between advanced hardware architectures and the demanding computational requirements of matrix profiling, offering a novel perspective on accelerating time series analysis for real-world applications.

## 2.1 Matrix Profile

While the matrix profile is a reasonably intuitive concept, a comprehensive set of definitions and notations is required to introduce and explain the design and optimization of the kernels responsible for its actual computation. This section aims to introduce those fundamentals to the reader to express matrix profiles. The definitions are mainly in line with [Yeh+16] and [Zhu+16]. As the following subsections only briefly introduce the required concepts, the reader is referred to the references mentioned above for a more detailed explanation.

### 2.1.1 Definitions

We first introduce the core data structure associated with matrix profiles: *time series*.

**Definition 1** A *time series* $T$ of length $n \in \mathbb{N}$ is a *sequence* of real-valued numbers $t_i \in \mathbb{R}$:

$$T = t_1, t_2, \ldots, t_n$$

While time series data constitutes both input and output for matrix profile computation, we require a more granular data structure called a *subsequence* during the calculations. For a visual representation of a time series and a subsequence (depicted with a grey background), see Figure 2.1.

**Definition 2** A *subsequence* $T_{i,m}$ of a time series T is a continuous subset of the values from $T$ of length $m \in \mathbb{N}$ starting from position $1 \leq i \leq n$ - $m + 1$:

$$T_{i,m} = t_i, t_{i+1}, \ldots, t_{i+m-1}$$

*Note:* By this definition, every subsequence is itself a time series.

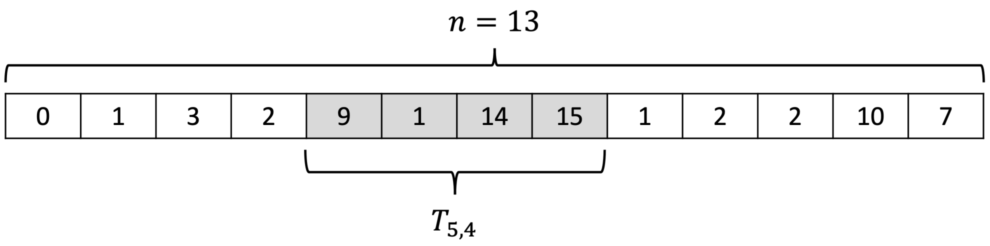

Figure 2.1: Time Series of length $n$ = 13 and Subsequence $T_{5,4}$

Calculating distances between subsequences is the core of the computation of matrix profiles. We define the (*z-normalized Euclidean*) *distance* between *time series*, and therefore *subsequences*, as follows:

**Definition 3** The *z-normalized Euclidean distance* between two time series $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_m$ of the same length $m \in \mathbb{N}$ is defined as

$$d(X,Y) = \sqrt{\sum_{i=1}^{m} (\bar{x}_i - \bar{y}_i)}$$

where $\bar{x}_i = \frac{x_i - \mu_X}{\sigma_X}$ and $\bar{y}_i = \frac{y_i - \mu_Y}{\sigma_Y}$. $\mu_Z and \sigma_Z$ denote the sample mean and standard deviation of a time series Z respectively. While this defenition is rather intuitive, we make use of the following reformation during the computation:

$$d(X,Y) = \sqrt{2m\left(1 - \frac{\sum_{i=1}^{m} x_i y_i - m\mu_X\mu_Y}{m\sigma_X\sigma_Y}\right)} = \sqrt{2m\,(1 - P\,(X,Y))} \tag{2.1}$$

with $P(X,Y)$ denoting the Pearson correlation coefficient between $X$ and $Y$:

$$P(X,Y) = \frac{\sum_i^m x_i y_i - m\mu_X\mu_Y}{m\sigma_X\sigma_Y} \tag{2.2}$$

Computing the distance between a single subsequence and every other possible subsequence of T results in the so-called *distance profile*. Thus, the distance profile can be utilized to find similar subsequences, i.e., subsequences with a small distance or subsequences that are substantially different, i.e., have a significant distance. In particular, the subsequence with the smallest distance is the closest match. If this distance is relatively small, the subsequence represents a motif, as it is similarly contained in T multiple times.

**Definition 4** A *distance profile $D_i$* of a time series $T$ is a vector of the Euclidean distances between a given query subsequence $T_i, m$ and each subsequence of length m in $T$. Formally,

$$D_i = (d_{i,1} d_{i,2} \ldots d_{i,n-m+1})$$

where $d_{i,j}\,(1 \leq i, j \leq n - m + 1)$ is the z-nromalized distance between $T_i, m$ and $T_j, m$.

By computing the distance profile for each subsequence of $T$, we obtain the so-called *distance matrix*. Therefore, the distance matrix contains the distance between every subsequence and every other subsequence in $T$.

**Definition 5** A *distance matrix D* of a time series $T$ is the vector of all distance profiles $D_i\,(1 \leq i, j \leq n - m + 1)$:

$$D = \begin{pmatrix} D_1 \\ D_2 \\ \vdots \\ D_{n-m+1} \end{pmatrix} = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n-m+1} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n-m+1} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n-m+1,1} & d_{n-m+1,2} \cdots & d_{n-m+1,n-m+1} & \end{pmatrix}$$

where $d_{i,j}\,(1 \leq i, j \leq n - m + 1)$ once again denotes the z-normalized distance between subsequences $T_{i,m}$ and $T_{j,m}$. Finally, these definitions allow us to define the matrix profile, which can be expressed as the column-wise minima, henceforth referred to as aggregates, of the distance matrix. Therefore, $MP_i$ is the minimal distance between subsequence $T_{i,m}$ and any other subsequence in $T$. Note that due to the symmetry of Euclidean distances $(d_{i,j} = d_{j,i})$, the matrix profile also represents the vector row-wise minima.

However, the distance between any subsequence and itself is 0 ($d_{i,i}$ = 0). Furthermore, distances of adjacent subsequences are also relatively small. These matches are commonly referred to as (trivial matches) and are typically excluded, as they remain uninteresting for most use cases. The *exclusion zone* for a subsequence is defined as the set of indices that result in a trivial match. What is considered to be a trivial match depends on the application domain. Most commonly, an exclusion zone of $m/2$ is utilized, i.e., when computing the minima for subsequence $T_{i,m}$ , the subsequences $T_{i-m/4,m}, \ldots, T_{i+m/4,m}$ are ignored.

**Definition 6** The *matrix profile MP* of a time series $T$ is the vector corresponding to the column-wise minima of the distance matrix:

$$MP = \left(min_j\left(d_{1,j}\right) min_j\left(d_{2,j}\right) \ldots min_j\left(d_{n-m+1,j}\right)\right)$$

wherein $min_j\left(d_{1,j}\right)$ is the minimum of $D_i$ ignoring subsequences contained within the exclusion zone.

We are also interested in the index of the subsequence with the minimal distance, we introduce the *matrix profile index*, which represents the vector of indices corresponding to the entries in *MP*.

**Definition 7** The *matrix profile MP* of a time series $T$ is a vector of the corresponding indices of the matrix profile:

$$MPI = \left(argmin_j\left(d_{1,j}\right) argmin_j\left(d_{2,j}\right) \ldots argmin_j\left(d_{n-m+1,j}\right)\right)$$

In the case of several minima, the one with the smallest index is to be chosen.

A graphical representation of the matrix profile and matrix profile index can be found in Figure 2.2 Elements contained within the exclusion zone are depicted with a grey background.

### 2.1.2 Computation

While the definitions in Subsection 2.1.1 help understand the concept of matrix profiles, their straightforward implementation is relatively inefficient. The performance is inherently limited by the interinsic dot product operations of Equation 2.1 and Equation 2.2. In the following, a more efficient way of computing the matrix profile, an algorithm called SCAMP in accordance with [Zim+19], is introduced.

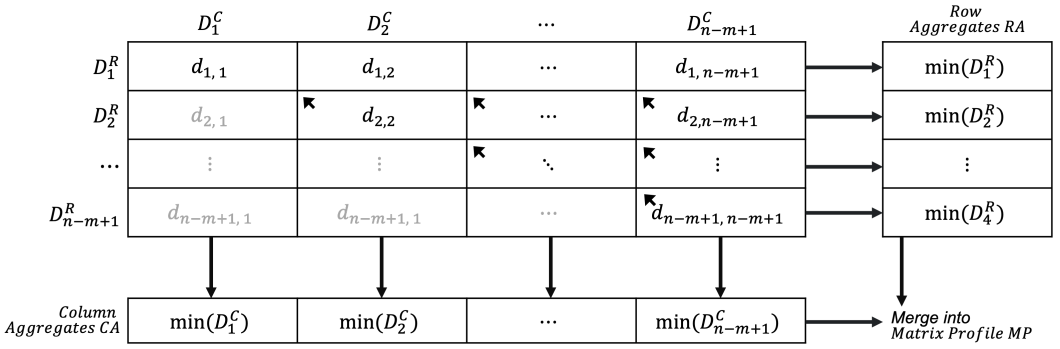

Figure 2.2: Matrix Profile $MP$ of a Time series $T$ as the column-wise minima of the Distance Matrix and the Matrix Profile Index $MPI$ as the vector of the corresponding indices. In this example, $d_{2,j}$ represents a column-wise minimum and is therefore integrated into the Matrix Profile.

The Pearson correlation $P_{i,j}$ between two subsequences $T_{i,m}$ and $T_{j,m}$ of a fixed time series $T = t_1, t_2, \ldots, t_n$ and a common subsequence length $m \in \mathbb{N}$, as explicitly formulated in Equation 2.2, can be computed as follows:

$$P_{i,j} = \overline{QT}_{i,j} * inv_i * inv_j \tag{2.3}$$

where,

$$\overline{QT}_{i,j} = \sum_{k=0}^{m-1} (t_{i+k} - \mu_i)(t_{j+k} - \mu_j) \tag{2.4}$$

and $inv_k$ denotes the inverse L2-Norm:

$$inv_k = \frac{1}{||T_{k,m} - \mu_k||} \tag{2.5}$$

SCAMP employes an optimization on $\overline{QT_{i,j}}$ by not implicitly calculation for all $i, j$ by using a centered-sum-of-products formula:

$$\overline{QT}_{i,j} = \overline{QT}_{i-1,j-1} + df_i \cdot dg_j + df_j \cdot dg_i \tag{2.6}$$

where,

$$df_k = \begin{cases} 0, & \text{if } k = 1 \\ \frac{t_{k+m-1} - t_{k-1}}{2}, & \text{if } 2 \leq k \leq n - m + 1 \end{cases} \tag{2.7}$$

and, with $\mu_i$ representing the sample mean of the subsequence $T_{i,m}$,

$$dg_k = \begin{cases} 0, & \text{if } k = 1 \\ t_{k+m-1} + (t_{k-1} - \mu_{k-1}), & \text{if } 2 \leq k \leq n - m + 1 \end{cases} \tag{2.8}$$

Equations 2.7 and 2.8 are used to precompute the terms used in Equation 2.6 and incorporate incremental mean centering into the update. In addition to $df$ and $dg$, the required L2-norm inverses are precomputed to avoid unnecessary recomputations.

As described in Equation 2.1, we can then convert the calculated Pearson correlation into Euclidean distance in $O(1)$ via:

$$d_{i,j} = \sqrt{2m\left(1 - P_{i,j}\right)} \tag{2.9}$$

Note, while the computation described in Equation 2.6 introduces a diagonal dependency between computations ($\overline{QT}_{i-1,j-1}$ is required to compute $\overline{QT}_{i,j}$), this dependency is circumvented at any point by explicitly calculating $\overline{QT}_{i-1,j-1}$ via the explicit dot product fomulation (Equation 2.4). The explicit caluclation is required for the first row, i.e., $\overline{QT}_{1,j}$ has to be calculated via the straightforward definition.

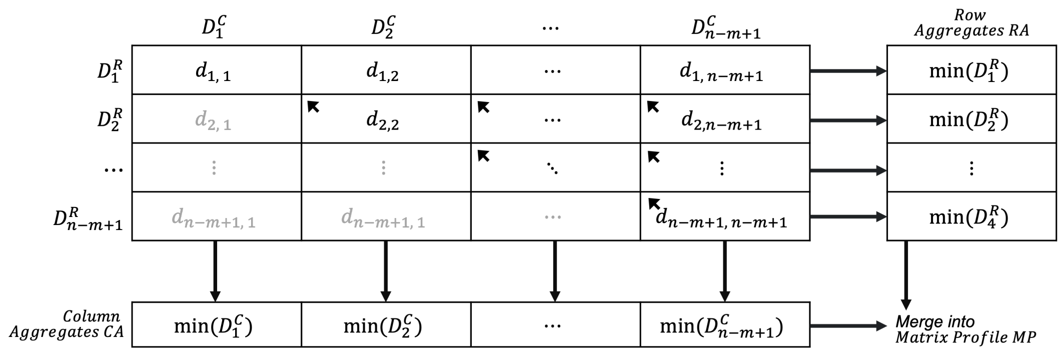

Figure 2.3: Computation performed by the SCAMP algorithm. In particular, only values above (and including) the main diagonal are computed. The diagonal dependency introduced through the updated formulation is visualized through upward-pointing arrows.

SCAMP does not compute the entire matrix but rather only elements above the main diagonal. This can be done due to the symmetric nature of the problem. We, therefore, store row- and column-wise aggregates, i.e., the minima of the considered values. These aggregates are merged subsequently to obtain the resulting matrix profile, as depicted in Figure 2.3.

## 2.2 Cerebras Wafer Scale Engine

The field of artificial intelligence (AI) has witnessed an exponential growth in recent years, catalyzed by advancements in hardware architectures tailored for accelerating AI workloads. Among these architectures, the Cerebras Wafer Scale Engine (WSE-2) stands out as a revolutionary platform designed to address the computational demands of modern AI models. This introduction provides an overview of the Cerebras WSE, its significance in AI research, and its potential implications for various domains.

The Cerebras WSE-2 is a groundbreaking computing platform developed by Cerebras Systems, designed to tackle the challenges of training and inference in deep learning models. Unlike traditional chip architectures composed of multiple individual chips interconnected through complex networks, the WSE-2 adopts a radically different approach by integrating all computational elements onto a single wafer. This wafer-scale integration eliminates the need for complex interconnects, minimizing latency and maximizing throughput, thus offering unparalleled performance for AI workloads.

**Key Features:**

- **Wafer-scale Integration:** The most distinctive feature of the Cerebras WSE-2 is its wafer-scale integration, where all compute, memory, and communication resources are densely packed onto a single silicon wafer. This design eliminates the communication overheads associated with traditional multi-chip systems, enabling efficient parallel processing of AI tasks.
- **Large-scale Array of Compute Cores:** The WSE-2 comprises thousands of processing cores, each optimized for executing neural network computations. These cores operate in parallel, leveraging the massive parallelism inherent in deep learning algorithms to accelerate model training and inference.
- **On-chip Memory Hierarchy:** To facilitate efficient data access and manipulation, the WSE-2 incorporates a sophisticated on-chip memory hierarchy, including fast local memory and caches accessible by all compute cores. This design minimizes data movement overheads, further enhancing performance and energy efficiency.
- **High-bandwidth Interconnect:** Despite its monolithic design, the WSE-2 features a high-bandwidth interconnect that enables efficient communication between compute cores and memory units. This interconnect architecture ensures seamless data exchange, enabling scalable performance for both small and large-scale AI models.

At the heart of the Cerebras CS-2 is the second-generation Wafer-Scale Engine (WSE-2). the WSE-2 is a massive parallel processor built from a single 300mm wafer. It offers 850,000 cores, optimized for sparse linear algebra with support for FP16, FP32, and INT16 data types. It contains 40GB of onboard SRAM divided between its cores, with 220Pb/s of interconnect bandwidth and 20PB/s of memory bandwidth, ensuring that each core can access its local memory in a single clock cycle.

The device allows programmers to interact with the cores using a lower-level code that targets the WSE's microarchitecture directly using a domain-specific programming language called the Cerebras Software Language, or CSL. In CSL, the cores on the WSE-2 are referred to as Processing Elements (PEs). The programmer can write code that targets every PE of the wafer such that compute and memory are optimally utilized. The programmer communicates to the device via a set of runtime APIs executed on one or more host computers.

Although this device is primarily motivated towards AI workloads, This thesis explores the viability of applying the available hardware and compute power to other diverse workloads like Matrix Profiling.

### 2.2.1 Processing Elements

The 850,000 PEs on the WSE-2 are structured in a 2D grid. Each PE contains a general-purpose compute element (CE), a fabric router, and 48kB of local SRAM memory with single-cycle read/ write access latency. PE-to-PE communication latency is also one cycle. The PE contains a network router with links to the CE and to the routers of the four nearest PEs in the north, south, east, and west directions. Communication is integrated into the instruction set, at single 32-bit word granularity, and is accordingly as fast as arithmetic.

The CE's instruction set supports FP32, FP16, and INT16 data types. The Cerebras ISA supports optimized vector operations for processing tensors as well as general-purpose control instructions. A CE can execute vector instructions that perform up to eight operations per clock for FP16 operands.

Each PE is additionaly clocked at 850MHz. Each program is allowed to allocate a certain number of PEs for executing the workload, henceforth, we will refer to this as the resource rectangle.

## 2.3 Related Works

# 3 Design

The previous chapter has introduced the reader to the concept of matrix profiles and their efficient computation and key concepts of development on Cerebras Wafer Scale Engine. The following chapter presents the top-level architecture as a combination of a Driver Application and an Accelerated Kernel by utilizing this background. The presented design constitutes a tiled version of the algorithm described in Subsection 2.1.2 and can, therefore, decouple the problem size from the concrete kernel implementation. Given the hardware capabilities of the WSE-2, a Tiled version of the Kernel is introduced in the chapter and explores the various types of tiling solutions that can be incorporated.

We then explore the various implications of resource allocation on the WSE-2 on the memory bandwidth, computation time and startup time.

We finally look at various scheduling approaches for large and small time series.

## 3.1 Architecture

The program is divided into 3 parts.

### 3.1.1 CSL Kernel

```
var row_iters: u8 = math_lib.min(n_x - exclusion_lower, n_y);

    for (@range(u8, row_iters)) | row | {
            var diag_max: u8 = math_lib.min(n_x - exclusion_upper + 1, n_x - row);

            var cur_dg_b: f32 = dg_b[row];
            var cur_df_b: f32 = df_b[row];
            var cur_norm_b: f32 = norm_b[row];
            for (@range(u8, exclusion_lower, diag_max, 1)) | diag | {
                    var col: u8 = diag + row;
                    // Calculate corr
```

```
                var corr: f32 = cov[diag] * norm_a[col] * cur_norm_b;
                if (corr > 0.99999946 and corr < 1.0) {
                        corr = math_lib.ceil(corr);
                } else if (corr > 1.0) {
                        corr = math_lib.floor(corr);
                }

                if (math_lib.isNaN(corr)) {
                        // Smallest value possible is -1 so set to -2.
                        corr = -2.0;
                }
                // Update cov
                var coeff: f32 = df_a[col] * cur_dg_b + dg_a[col] * cur_df_b;
                cov[diag] = cov[diag] + coeff;

                // Update profile
                update_profile(corr, P_a, P_i_a, row, col);
                update_profile(corr, P_b, P_i_b, col, row);
        }
}
```

#### 3.1.2 Host Driver

#### 3.1.3 Driver

## 3.2 Kernel Design

The main SCAMP algorithm.

## 3.3 Tiling

- SCAMP implements tiling.
- Talk about algorithm to compute tile positions.

### 3.3.1 Trapezoidal

- Area = b * h

**Algorithm 1** SCAMP Driver

```
    Input : Time Series $T$ of length $n \in \mathbb{N}$ and subsequence length $m \in \mathbb{N}$
    Output : Matrix Profile $MP$ and Matrix Profile Index $MPI$
1: $df, dg, inv \leftarrow PreComputeStatistics(T, m);$
2: $QT_{init} \leftarrow PreComputeInitialQTRow(T, m);$
3: $rowAggregates, columnAggregates \leftarrow (-\inf, -1);$
4: for $iteration \leftarrow 0$ to $\lceil \frac{n-m+1}{1} - 1 \rceil$ do
5:     $iteration_i \leftarrow MatrixProfileKernel(QT_{init}$,df,dg,inv$);$
6:     $UpdateAggregates(rowAggregates, columnAggregates, iteration_i);$
7: end for
8: $PostCompute(rowAggregates, columnAggregates);$
9: return $MP, MPI;$
```

- Same as square.
- Need modification to create large number of extra tiles with no elements.

### 3.3.2 Square

- Area = w x h
- Currently implemented in SCAMP
- Edge cases are rectangles.
- Diagonal tiles are upper triangles

### 3.3.3 Triangles

- Area = 1/2 * b * h
- Currently implemented in SCAMP, can also be implemneted on the cerebras interface
- More jobs to be spread across limited number of PEs.
- More iterations on larger time series.

**Algorithm 2** Tiled Kernel

```
    Input : Time series $T_a$, $T_b$, $df$, $dg$, $inv$ and $args$ for current Tile.
    Output : Row- and Column-Wise Aggregates for the current Tile
 1: for $ProcessingElement \leftarrow 0$ to $w * h$ do
 2:     $rowAggregates_{ProcessingElement} \leftarrow (-\text{inf}, -1);$
 3:     $columnAggregates_{ProcessingElement} \leftarrow (-\text{inf}, -1);$
 4:     if $args.full\_tile$ is 0 then                    ▷ Compute only top triangle
 5:         $QT \leftarrow computeQT(T_a, T_b);$
 6:         for $row \leftarrow 0$ to $min(args.n_x - args.exclusion\_lower, args.n_y)$ do
 7:             for $diag \leftarrow args.exclusion\_lower$ to $min(args.n_x - args.exclusion\_upper +$
    $1, args.n_x - row)$ do;
 8:                 $col \leftarrow row + diag;$
 9:                 $correlation \leftarrow QT_{diag} \cdot inv_{row} \cdot inv_{col};$
10:                 if $correlation > rowAggregate_{ProcessingElement,row}.value$ then
11:                     $rowAggregate_{ProcessingElement,row}.value \leftarrow (correlation, column);$
12:                 end if
13:                 if $correlation > rowAggregate_{ProcessingElement,column}.value$ then
14:                     $rowAggregate_{ProcessingElement,column}.value \leftarrow (correlation, row);$
15:                 end if
16:             end for
17:         end for
18:     end if
19: end for
20: $rowAggregates \leftarrow (-\text{inf}, -1);$
21: $columnAggregates \leftarrow (-\text{inf}, -1);$
22: for $ProcessingElement \leftarrow 0$ to $w * h$ do
23:     $Merge(rowAggregates, rowAggregates_{ProcessingElement});$
24:     $Merge(columnAggregates, columnAggregate_{ProcessingElement});$
25: end forreturn $rowAggregates, columnAggregates;$
```

## 3.4 Resource Allocation

### 3.4.1 Rectangle

### 3.4.2 Square

## 3.5 Scheduling

### 3.5.1 One Shot

### 3.5.2 Iterative

# 4 Implementation

## 4.1 WSE Programming Environment

### 4.1.1 CSL

### 4.1.2 Python Driver

# 5 Experiments and Results

## 5.1 Model

### 5.1.1 Single Tile

#### Execution Time

#### Memory

## 5.2 Performance

#### CPU & GPU Baseline

## 5.3 Error

# List of Figures

# List of Tables

# Bibliography

[Yeh+16] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets." In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 1317–1322. DOI: `10.1109/ICDM.2016.0179`.

[Zhu+16] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh. "Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins." In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 739–748. DOI: `10.1109/ICDM.2016.0085`.

[Zim+19] Z. Zimmerman, K. Kamgar, N. S. Senobari, B. Crites, G. Funning, P. Brisk, and E. Keogh. "Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond." In: *Proceedings of the ACM Symposium on Cloud Computing*. SoCC '19. Santa Cruz, CA, USA: Association for Computing Machinery, 2019, pp. 74–86. ISBN: 9781450369732. DOI: `10.1145/3357223.3362721`.