Energy-Efficient Time Series Analysis Using Transprecision Computing

Ivan Fernandez, Ricardo Quislant, Eladio Gutierrez, Oscar Plata
Dept. of Computer Architecture
University of Malaga, Spain
{ivanfv, quislant, eladio, oplata}@uma.es

***Abstract*—Time series analysis is a key step in monitoring and predicting events over time in domains such as epidemiology, genomics, medicine, seismology, speech recognition, and economics. *Matrix Profile* has been recently proposed as a promising technique to perform time series analysis. For each subsequence, the matrix profile provides the most similar neighbour in the time series. This computation requires a huge amount of floating-point (FP) operations, which are a major contributor ($\approx$ 50%) to the energy consumption in modern computing platforms. *Transprecision Computing* has recently emerged as a promising approach to improve energy efficiency and performance by tolerating some loss of precision in FP operations.**

In this work, we study how the matrix profile parallel algorithms benefit from transprecision computing using a recently proposed transprecision FPU. This FPU is intended to be integrated on embedded devices as part of RISC-V processors, FPGAs or ASICs to perform energy-efficient time series analysis. To this end, we propose an accuracy metric to compare the results with the double precision matrix profile. We use this metric to explore a wide range of exponent and mantissa combinations for a variety of datasets, as well as a mixed precision and a vectorized approach. Our analysis reveals that the energy consumption is reduced up to 3.3$\times$ compared with double precision approaches, while only slightly affecting the accuracy.

***Index Terms*—Time Series Analysis, Transprecision Computing, Floating-Point Unit, Vectorization**

I. Introduction

A time series is a chronologically ordered set of samples of a real-valued variable that can contain millions of observations. Time series analysis seeks extracting models in a large variety of domains [1] such as epidemiology, DNA analysis, economics, medicine, geophysics, speech recognition, etc. Particularly, *motif* [2] (similarity) and *discord* [3] (anomaly) discovery has become one of the most frequently used primitives in time series data mining [4]–[9]. It poses the problem of solving the all-pairs-similarity-search (also known as similarity join). Specifically, given a time series sliced in subsequences, retrieve the nearest motif and the furthest discord neighbour for every subsequence with respect to the rest ones. The state-of-the-art method for motif and discord discovery is *Matrix Profile* [10]. It solves the similarity join problem and allows time-manageable computation of very large time series datasets. In this work, we focus on this technique, which features the possibility of detecting similarities, anomalies, and predicting outcomes. It provides full joins without the need to specify a similarity threshold, which is a very challenging task in this domain. The matrix profile is another time series representing the minimum distance subsequence for each subsequence (motifs). Maximum values of the profile highlight the most dissimilar subsequences (discords).

We evaluate the latest implementations of matrix profile (SCRIMP [11] and SCAMP [12]) and find that a huge number of floating-point (FP) arithmetic operations are needed in order to analyze even short time series. In that sense, *transprecision computing* [13] has recently emerged as a promising approach to (1) improve energy efficiency, (2) provide better performance, (3) reduce area footprint, and (4) reduce memory bandwidth by tolerating some loss of accuracy in computed results. This paradigm reduces the number of bits for the exponent and the mantissa in FP operations in a flexible way, depending on the requirements of the application. It is well known that FP operations are a major contributor ($\approx$ 50%) [14] to the energy consumption in modern computing platforms.

Our goal in this work is to study the benefits and shortcomings of a transprecision approach in Matrix Profile, to attain energy-efficient and high-performance time series analysis for a wide range of applications. This opens up the opportunity to improve the detection of important events on mobile and embedded devices. Those devices can be used, for example, to prevent ecological disasters or medical issues (e.g., for early earthquake detection [15] or to predict a heart attack [16]).

To this end, we analyze matrix profile with real datasets using a transprecision emulation library, and evaluate energy savings of a real transprecision FPU. Our evaluation provides guidelines for implementations based on transprecision RISC-V processors, FPGAs or ASICs for embedded devices that aim to reduce energy consumption and area overheads.

The contributions of this work are the following:

- We develop two transprecision implementations of Matrix Profile (`SCRIMPff` and `SCAMPff`), using FlexFloat [17] to analyze the trade-off between precision and accuracy.
- We propose a new metric, called *Top-K Accuracy*, to measure the accuracy in the discovery of motifs and discords of a transformed time series (e.g., lower precision) with respect to the original time series.
- We provide a detailed analysis of `SCRIMPff` and `SCAMPff` in terms of energy when using *vectorization* and *mixed precision*, taking advantage of a proposed transprecision FPU. We find that energy efficiency can be improved up to 3.3$\times$ compared with double precision, while only slightly affecting the accuracy.

2643-3001/20/$31.00 ©2020 IEEE
DOI 10.1109/SBAC-PAD49847.2020.00022

II. Background

A. Time Series Analysis. The Matrix Profile

A *time series* T is a sequence of n data points t_i, $1 \leq i \leq n$, collected over time. Let be $T_{i,m}$ a subsequence of T, where i is the index of its first data point, T_i, and m is the number of data points in the subset, with $1 \leq i$, and $m \leq n$. In the literature, $T_{i,m}$ is also called a *window* of length m. Fig. 1 shows an example of a time series with two subsequences highlighted, $T_{i,m}$ and $T_{j,m}$.

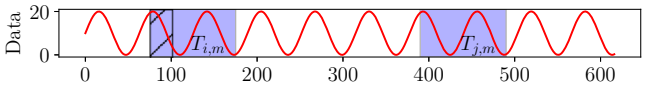

Fig. 1. Example of two subsequences, $T_{i,m}$ and $T_{j,m}$, of a given time series. When computing the matrix profile P, subsequences starting in the exclusion zone of $T_{i,m}$ are ignored for their high similarity.

One way to measure the similarity between two time series subsequences is to use the *z-normalized Euclidean distance* [11], $d_{i,j}$, which is calculated as follows:

$$d_{i,j} = \sqrt{2m\left(1 - \frac{Q_{i,j} - m\mu_i\mu_j}{m\sigma_i\sigma_j}\right)} \quad (1)$$

where $Q_{i,j}$ is the dot product of subsequences $T_{i,m}$ and $T_{j,m}$. μ_x and σ_x are the mean and the standard deviation of the data points in $T_{x,m}$, respectively.

We can use this distance measure to find the most similar subsequences out of all subsequences of a time series T. There are three steps to this procedure: (1) building a symmetric $(n-m+1) \times (n-m+1)$ matrix D, called *distance matrix*, with a window size m and a time series of length n. Each D cell, $d_{i,j}$, stores the distance between two subsequences, $T_{i,m}$ and $T_{j,m}$. Thus, each row (or column) of D stores the distances between a subsequence of T and every subsequence of T; (2) finding the two subsequences whose distance is minimum (i.e., most similar sequences) in each row (or column) of D. This can be computed by building the *matrix profile*, P, which is a vector of size $n - m + 1$. Each cell P_i in P stores the minimum value found in the i^{th} row (or column) of D; (3) finding the indices of the most similar subsequences of the matrix profile. This requires building another vector I –called *matrix profile index*– of the same size as that of P, where $I_i = j$ if $d_{i,j} = P_i$. In this way, P contains the minimum distances between subsequences of T, while I is the vector of "pointers" to the location of these subsequences. Fig. 2 depicts an example of the distance matrix (D), the matrix profile (P), and the matrix profile index (I) for the time series in Fig. 1.

Matrix Profile algorithms are designed to store only the matrix profile and the matrix profile index arrays, computing the minimum distances $d_{i,j}$ on the fly. Note that the neighboring subsequences of $T_{i,m}$ are highly similar to it (i.e., $d_{i,i+1} \approx 0$) due to the overlaps. Thus, we can exclude these subsequences from computing D to find similar subsequences other than the neighboring ones. This is done by defining an exclusion zone (diagonal stripped zone in Fig. 1) for each subsequence. In general, the exclusion zone for a subsequence $T_{i,m}$ of length m is $\frac{m}{4}$ [11].

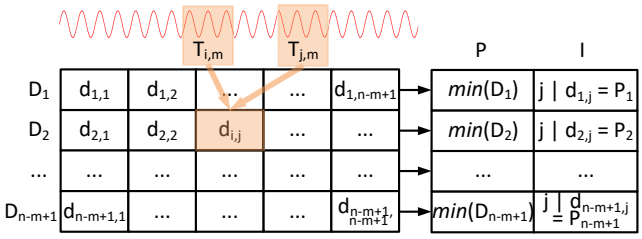

Fig. 2. Computation of the matrix profile P and the profile index I from the distance matrix D. P_i is the minimum distance of each row (or column) D_i. I_i is the index of the subsequence providing such minimum.

B. Transprecision Computing

Transprecision Computing [13] aims to boost energy efficiency and performance by exploiting numeric approximation in both hardware and software. It enables fine control over the precision of floating point arithmetic in space and time (where and when to use it). The key difference with approximate computing is that the first one guarantees the error, while the latter one provides uncertainty. This approach leads to significant energy savings and performance improvements without sacrificing overall quality of results.

Besides the IEEE-754 standard, transprecision computing allows the use of different exponent and mantissa bit combinations. Fig. 3 shows the types [14] that we use in this work for the energy evaluation. The bit count for each floating point datatype, along with the approximate range and smallest number that can be represented, are summarized in Table I. Note that, while binary16 and binary8 have the same range, the first one provides narrower steps between numbers and more precision.

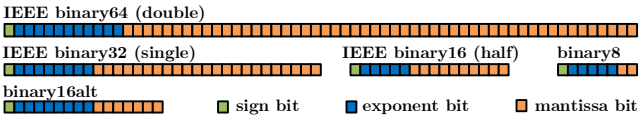

Fig. 3. Overview of the floating-point types used for energy evaluation.

One way to take advantage of transprecision computing, in terms of energy and performance, is the use of a transprecision Floating-Point Unit (FPU). The first silicon implementation of a 64-bit transprecision FPU can be found in [18], using 22nm technology node. This FPU supports the floating-point types depicted in Fig. 3, and the key idea behind it is to operate in scalars (64-bit) or in SIMD vectors of 2 elements (32-bit), 4 elements (16-bit) or 8 elements (8-bit). The authors evaluate the FPU using a RISC-V processor core (but can be used in other platforms, like FPGAs or ASICs) and obtain speedups up to 7.3× while reducing energy up to 7.94× with respect to *double* precision approaches. The source code of this FPU –known as FPnew– can be found in [19].

TABLE I
FLOATING-POINT BIT COUNTS, RANGES AND SMALLEST NUMBERS

Type	Exponent	Mantissa	Range (≈)	Smallest (≈)
IEEE binary64	11	52	$-1.8 \cdot 10^{308} \leq x \leq 1.8 \cdot 10^{308}$	$2.2 \cdot 10^{-308}$
IEEE binary32	8	23	$-3.4 \cdot 10^{38} \leq x \leq 3.4 \cdot 10^{38}$	$1.4 \cdot 10^{-45}$
IEEE binary16	5	10	$-65,504 \leq x \leq 65,504$	$6 \cdot 10^{-8}$
binary16alt	8	7	$-3.4 \cdot 10^{38} \leq x \leq 3.4 \cdot 10^{38}$	$9.1 \cdot 10^{-41}$
binary8	5	2	$-65,504 \leq x \leq 65,504$	$1.5 \cdot 10^{-5}$

III. TRANSPRECISION MATRIX PROFILE

A. The SCRIMP Algorithm

SCRIMP [11] minimizes the computation by exploiting the fact that the dot product can be updated incrementally for subsequences in the diagonal of the distance matrix, D (see Fig. 2). Consequently, the dot product can be expressed as follows:

$$Q_{i,j} = Q_{i-1,j-1} - T_{i-1}T_{j-1} + T_{i+m-1}T_{j+m-1} \quad (2)$$

The baseline for our transprecision SCRIMP algorithm, `SCRIMPff` in Fig. 4, is a vectorized-parallel version presented in [20]. It first precomputes the mean and standard deviation of each time series subsequence (line 1), and initializes the matrix profile array (line 3). Then, the distances between pairs of subsequences are calculated following the diagonals of the distance matrix (lines 4-26). The for loop is fully parallelized, with each thread computing a random subset of diagonals provided by their indices in the $diag$ array in line 5.

For the first element of the diagonal, we need to compute the dot product of the first pair of subsequences (line 6) in parallel. The rest are updated following Eq. 2. For the proper vectorization of the dot product update, the algorithm separates the calculation of the diagonal in several steps: (1) the products in Eq. 2 are calculated in parallel for $vectFact$ elements of the diagonal (lines 14-15); (2) the previous dot product, q, is added to the element calculated in step i) (line 16); (3) the subsequent dot products are updated sequentially using the previous ones (lines 17-18) saving the last one in q for the next iteration of the diagonal (line 19); (4) distances are calculated in parallel (lines 20-21); and (5) the profile is updated in parallel as well (lines 22-25).

The vectorization factor, $vectFact$, is given by the transprecision datatype width with respect to that of double precision (line 2). We highlight the lines of code which are to be executed in the transprecision FPU. As discussed in Section III-C, the algorithm may work in either one precision configuration or a mixed-precision one, thus the green (lower precision) and red (higher precision) marks.

B. The SCAMP Alternative

Whereas following a similar computation scheme than `SCRIMPff`, `SCAMPff` replaces the sliding dot product with a mean-centered-sum-of-products in order to reduce the floating-point rounding errors and the number of operations required [12]. The following equations can be precomputed in $O(n-m+1)$ time, with $n-m+1=l$ being the length of the matrix profile vector:

$$df_i = \frac{T_{i+m-1} - T_{i-1}}{2}, \quad 0 < i < l \quad (3)$$

$$dg_i = T_{i+m-1} - \mu_i + T_{i-1} - \mu_{i-1}, \quad 0 < i < l \quad (4)$$

$$ssq_i = \begin{cases} \sum_{k=0}^{m-1}(T_k - \mu_0)^2, & i = 0 \\ ssq_{i-1} + (T_{i+m-1} - \mu_i + T_{i-1} - \mu_{i-1}) & \\ (T_{i+m-1} - T_{i-1}), & 0 < i < l \end{cases} \quad (5)$$

$$\sigma_i = \sqrt{ssq_i}, \quad 0 \leq i < l \quad (6)$$

Eqs. 3 and 4 are terms used in the covariance update of Eq. 7, and the standard deviation (L2-norm of subsequence $T_{i,m} - \mu_i$) calculated in Eqs. 5 and 6 is used for the Pearson correlation coefficient depicted by Eq. 8. Note the exclusion zone in the limits of Eq. 7 given by $\frac{m}{4}$.

$$\sigma_{i,j} = \begin{cases} \sum_{k=0}^{m-1}(T_k - \mu_0)(T_{k+j} - \mu_j), & i = 0, \frac{m}{4} < j < l \\ \sigma_{i-1,j-1} + df_i dg_j + df_j dg_i, & i > 0, \frac{m+4}{4} < j < l \end{cases} \quad (7)$$

$$P_{i,j} = \frac{\sigma_{i,j}}{\sigma_i \sigma_j} \quad (8)$$

$$D_{i,j} = \sqrt{2m(1 - P_{i,j})} \quad (9)$$

The matrix profile can be derived incrementally for each diagonal of the distance matrix, Eq. 7, from the calculation of the covariance of two subsequences of the first row (first piece in Eq. 7). The Pearson correlation coefficient in Eq. 8 can be computed in fewer operations and it is more robust than the Euclidean Distance used by SCRIMP. Eq. 9 calculates the distance from the Pearson coefficient in $O(1)$.

C. Mixed Precision

Transprecision computing can be applied to the entire algorithm by setting fixed exponent and mantissa widths for every floating point operation. However, we can change the precision to different parts of the code to find a trade-off between the accuracy of the results and energy efficiency. Mixed precision of double and single floating point operations has been successfully used in the past [21], [22] with a consequential gain in performance. In this paper, we propose applying mixed precision to the SCRIMP and SCAMP algorithms to harness the transprecision FPU.

Rather than using only double and single precision, we define a high and a low precision that can be set to any

```
1:  μ, σ ← computeMeanDev(T, m);                          ▷ precalculation of statistics
2:  vectFact ← 8/sizeof(datatype);                        ▷ 8 bytes (double) by the size of the transprecision datatype
3:  P ← ∞;                                                ▷ matrix profile array initialization
4:  for idx ← tid * numDiag to (tid + 1) * numDiag − 1 do   ▷ parallel for
5:      i ← 0; j ← diag_idx;                              ▷ diag array contains the diagonals assigned to each thread
6:      q ← dotProduct(T_{i,m}, T_{j,m});                 ▷ compute dot product for the first diagonal element (SIMD)
7:      d ← dist(m, q, μ_i, σ_i, μ_j, σ_j);               ▷ compute distance for the first element of diagonal
8:      if d < P_i then                                   ▷ update matrix profile and index arrays
9:          P_i ← d; I_i ← j;
10:     if d < P_j then                                   ▷ distance matrix is symmetric
11:         P_j ← d; I_j ← i;
12:     i ← i + 1;
13:     for j ← diag_idx to size(P) do
14:         for k ← 0 to vectFact − 1 do                  ▷ compute new dot product elements in parallel (SIMD)
15:             qs_k ← t_{i+m−1+k} t_{j+m−1+k} − t_{i−1+k} t_{j−1+k};
16:         qs_0 ← qs_0 + q                               ▷ update with the first dot product of the diagonal
17:         for k ← 1 to vectFact − 1 do                  ▷ update remaining dot products (sequentially)
18:             qs_k ← qs_k + qs_{k−1};
19:         q ← qs_{vectFact−1};                          ▷ store last dot product for next iteration
20:         for k ← 0 to vectFact − 1 do                  ▷ compute distances in parallel (SIMD)
21:             ds_k ← dist(m, qs_k, μ_{i+k}, σ_{i+k}, μ_{j+k}, σ_{j+k});
22:             if ds_k < P_{i+k} then                    ▷ update matrix profile and index arrays
23:                 P_{i+k} ← ds_k; I_{i+k} ← j + k;
24:             if ds_k < P_{j+k} then
25:                 P_{j+k} ← ds_k; I_{j+k} ← i + k;
26:         i ← i + vectFact;
```

Fig. 4. SCRIMP Flexfloat (`SCRIMPff`) algorithm (transprecision operations highlighted).

possible exponent and mantissa configuration allowed by the transprecision FPU. For the algorithms involved in this study, we store the time series codified with high precision and both the matrix profile and the precalculated statistics with low precision. For `SCRIMPff`, Fig. 4 shows the lines of code that works with high precision highlighted in red, and those which work with low precision highlighted in green. The dot product calculations use high precision as they may require a larger numeric range. Distance calculation as well as calculations with means and standard deviations are performed with less precision. For `SCAMPff`, we compute the covariance in Eq. 7 at high precision, which may have a large numeric range depending on the series. The correlation in Eq. 8, which varies between -1 and 1, is computed at low precision.

D. Top-K Accuracy Metric

Time series motifs [2] and discords [3] have been used for more than 15 years in the field of data mining for their capacity to find time series subsequences with special significance. In this section, we define these special subsequences and propose a metric to measure the accuracy in the detection of motifs and discords from two time series.

The definitions of the motif and the Top-K motifs of a time series are presented in the remainder of this subsection.

Definition 1: The motif M_1 of a time series T is the unordered pair of subsequences $\{T_{i,m}, T_{j,m}\}$ which is the most similar among all possible pairs:

$$M_1 = \{T_{i,m}, T_{j,m}\} \iff dist(T_{i,m}, T_{j,m}) \leq dist(T_{u,m}, T_{v,m})$$

$$\forall i, j, u, v;\ i \neq j, u \neq v.$$

Definition 2: The Top-K motifs $M_{1,K}$ of a time series T is the set of the first K motifs:

$$M_{1,K} = \begin{cases} M_K \cup M_{1,K-1}, & K > 1 \\ M_1, & K = 1 \end{cases}$$

being M_K the motif (M_1) of the time series $T \setminus M_{1,K-1}, \forall K > 1$.

We can define the discord and the Top-K discords of a time series in a similar manner:

Definition 3: The discord D_1 of a time series T is the unordered pair of subsequences $\{T_{i,m}, T_{j,m}\}$ which is the most dissimilar among all possible pairs:

$$D_1 = \{T_{i,m}, T_{j,m}\} \iff dist(T_{i,m}, T_{j,m}) \geq dist(T_{u,m}, T_{v,m})$$

$$\forall i, j, u, v;\ i \neq j, u \neq v.$$

Definition 4: The Top-K discords $D_{1,K}$ of a time series T is the set of the first K discords:

$$D_{1,K} = \begin{cases} D_K \cup D_{1,K-1}, & K > 1 \\ D_1, & K = 1 \end{cases}$$

being D_K the discord (D_1) of the time series $T \setminus D_{1,K-1}, \forall K > 1$.

Having $MI_{1,K}$ and $DI_{1,K}$ the Top-K sets of unordered pair of indices $\{i, j\}$ of the unordered pair of subsequences in $M_{1,K}$ and $D_{1,K}$, respectively, we define the Top-K Motif/Discord Accuracy in the following manner:

Definition 5: The Top-K Motif (Discord) Accuracy AM (AD) of a time series T with respect to another time series TT is the number of coincidences among the unordered pairs in $MI^{T}_{1,K}$ and $MI^{TT}_{1,K}$ ($DI^{T}_{1,K}$ and $DI^{TT}_{1,K}$):

$$AM^{T \to TT}_{1,K} = |MI^{T}_{1,K} \cap MI^{TT}_{1,K}|$$

$$AD^{T \to TT}_{1,K} = |DI^{T}_{1,K} \cap DI^{TT}_{1,K}|$$

Def. 5 is pointless when T and TT are time series from different applications. However, it can be useful when we want to know the degree of coincidence of the matrix profile of a time series and that of the same time series codified with less precision or affected by a transform operator. We can get to know if, after a transformation of the original time series, the matrix profile ends up unveiling the same motifs and discords, or a significant subset of them.

IV. Experimental Evaluation

A. Methodology

We use an Intel Xeon Phi 7210 "Knights Landing" many-core processor with 64 cores and 256 threads for each experiment. We compute the SCRIMP and SCAMP matrix profiles with double and single floating point precisions, and we use the FlexFloat library [17] to explore precision configurations lower than single precision. This library has a high impact in performance so we constrained the time series length[1]. We use a transprecision FPU [18], proposed for RISC-V processors and suitable for FPGAs and ASICs, for our energy estimations, which are based on the FlexFloat operation breakdown statistics and the energy per operation numbers given in [18].

Table II summarizes the parameters of the time series that we use for the experiments. *Audio* corresponds to the song *London Bridge is Falling Down* [23] converted into Mel-frequency Cepstral Coefficients which are commonly used in speech recognition [24]. *ECG* is an electrocardiogram signal from the European ST-T Database [25]. We select the 180000 first samples of the V4 electrode from ECG 0103. *Power* is a time series of fridge-freezer power consumption numbers collected over a whole year in a set of UK households [23]. We analyze the first 180000 samples of the series. *Seismology* is a time series of seismic data collected by a seismograph in a geologically active region of the Long Valley Caldera, California [23]. *Human Activity* comprises a time series with information of the optical flow of an actor performing activities from picking up an object to talking on a mobile phone [23]. Last, *Penguin Behavior* is a time series of penguin magnetometer telemetry [26]. We scale some time series by a factor to keep all the series in a comparable value range.

[1]SCRIMP and SCAMP perform value extrapolation based on a previous result along the diagonals (dot product and covariance, respectively). In such scenarios, it is possible that accumulated errors provide misleading results for large time series. However, as production implementations of SCRIMP and SCAMP are based on a tiled approach, where the dot product or covariance is calculated from scratch for each 128K-512K elements of the diagonal, our evaluation and conclusions are also valid for time series of larger sizes.

TABLE II
Time series dataset parameterization

Time series	n	m	Max	Min	Scale
Audio	20234	200 (2s)	6.69	-56.48	1
ECG	180000	500 (2s)	2.6	0.32	1
Power	180000	1325 (8h)	14.0	0	0.1
Seismology	180000	50 (2.5s)	6.96	-1.86	0.01
Human Activity	7997	120 (12s)	2.51	-1.9	1
Penguin Behavior	109842	800	0.52	-0.21	1

B. Results

We use the Top-K Accuracy metric of Def. 5 to evaluate our proposals, comparing the transprecision matrix profile of the time series with respect to its double precision counterpart for all the experiments. The value of K will depend on the number of significant events of a given time series, which will be eventually determined by a domain expert. One way of setting it is defining a profile threshold for both motifs and discords. In this case, we consider K=100 in order to provide a consistent metric across the datasets.

Figs. 5 and 6 present the Top-100 motif/discord accuracy w.r.t. double for a wide range of configurations of exponent and mantissa in `SCRIMPff` and `SCAMPff`, respectively. In most cases, single precision provides 100% accuracy with respect to double precision (see the points tagged with Single). We notice that most of the plots follow a cube-like shape where the accuracy decreases dramatically after a given combination of exponent and mantissa. This may occur whether one or both of the following scenarios appear: (1) the range of the exponent has been exceeded; (2) the precision provided by the mantissa is not enough for the calculations.

Comparing both algorithms, we can observe that SCAMP is more robust and presents a better numeric stability than SCRIMP for all the datasets. This fact can be clearly noticed for *Penguin Behavior*, where a slight decrease in the length of the mantissa makes SCRIMP fail in detecting events, while SCAMP provides more margin in this reduction.

1) Mixed Precision.: Table III shows the mixed precision results for `SCRIMPff` and `SCAMPff`. We obtain better results with `SCAMPff` for most datasets, as expected. It is worth noting the case of Penguin Behavior, where `SCRIMPff` is not able to detect any of the events while `SCAMPff` finds near 100% of them with the (8/23, 5/10) configuration.

Overall, our evaluation shows that the mixed-precision configurations provide better accuracy results than using only

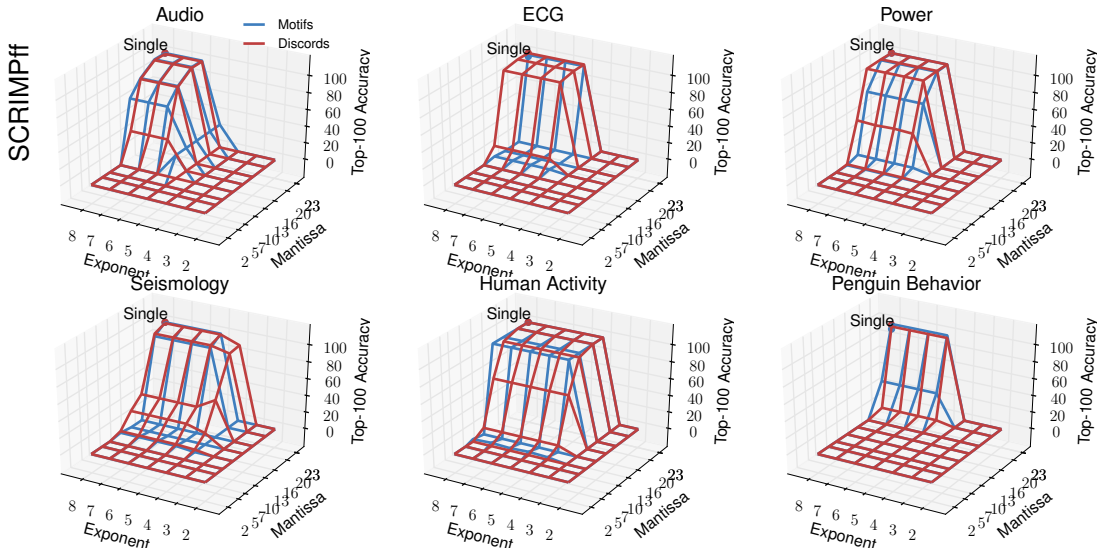

Fig. 5. SCRIMPff Top-100 accuracy results with respect to double.

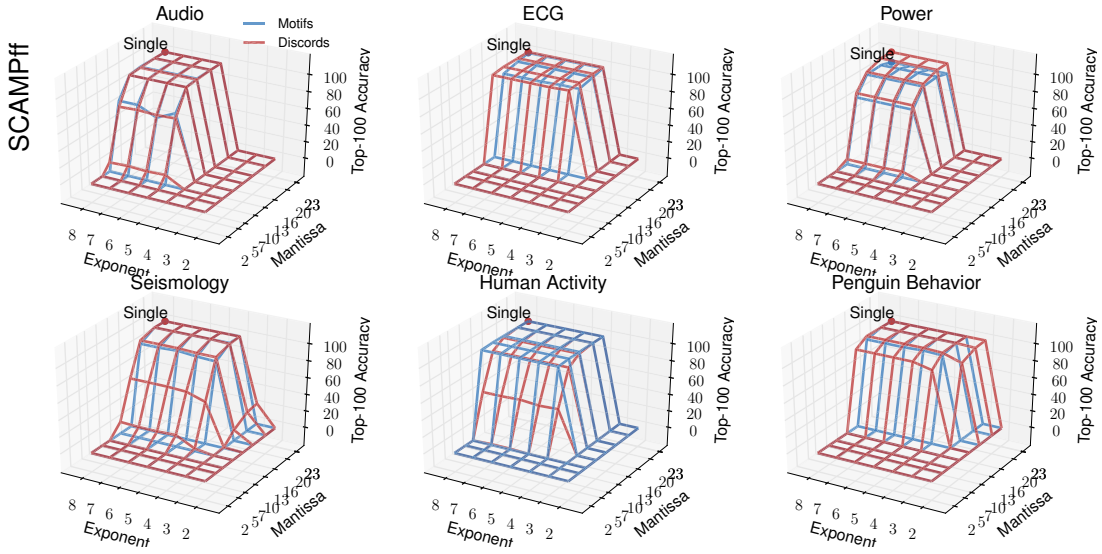

Fig. 6. SCAMPff Top-100 accuracy results with respect to double.

one reduced precision configuration throughout the code. The experiments suggest that a mixed configuration of (8/23, 5/10) can be used for the majority of the applications analyzed in this work. In this sense, we can observe that SCAMP results for Audio present 70% accuracy for a 5/10 configuration (see Fig. 6), whereas the accuracy increases to 95% using (8/23, 5/10) mixed precision (see Table III). The rest of the series present a similar behavior with SCAMP: 0% accuracy for ECG with the 5/10 configuration, while up to 99% discord accuracy with the (8/23, 5/10)-mixed-precision configuration; 0% motif and 7% discord accuracy for Power with the 5/10 configuration, whereas 81% motif and 65% discord accuracy with mixed precision; and so on. We find that SCRIMP presents a similar accuracy pattern.

We observe that the (8/23, 5/2)-mixed-precision configuration does not yield meaningful results for any of the analyzed datasets, since the low bit count for the mantissa (only two bits) does not provide enough resolution for the correlation calculations. It can be noticed that detecting discords (anomalies) is more accurate than detecting motifs (similarities). This can be due to the fact that similarities are low values of the profile which require more precision than discords, that typically are higher ones. This fact takes more relevance in very monotonic time series, where most of the subsequences are similar to each other (e.g., the beats in an electrocardiogram – ECG).

The number of detected events and its significance must be eventually determined by a domain expert. Thus, we can think that presenting a time series subsequence to a domain expert,

TABLE III
MIXED PRECISION TOP-100 ACCURACY RESULTS

T	High Exp/Man	Low Exp/Man	SCRIMPff Accuracy Mot/Disc	SCRIMPff Accuracy ±10 Mot/Disc	SCAMPff Accuracy Mot/Disc	SCAMPff Accuracy ±10 Mot/Disc
Audio	8/23	8/7	14/9	16/31	54/86	100/97
	”	5/10	38/0	99/0	95/99	100/100
	”	5/2	0/0	0/0	1/0	1/0
ECG	8/23	8/7	0/1	0/1	0/51	0/56
	”	5/10	10/57	10/60	25/99	30/100
	”	5/2	0/0	0/0	0/0	0/0
Power	8/23	8/7	47/31	67/95	39/25	78/99
	”	5/10	68/92	96/100	81/65	100/100
	”	5/2	0/0	0/0	0/0	0/0
Seis.	8/23	8/7	3/17	55/21	0/3	0/6
	”	5/10	7/68	86/70	12/40	15/45
	”	5/2	0/0	0/0	0/0	0/0
Hum.	8/23	8/7	72/24	80/63	91/84	99/92
	”	5/10	100/85	100/97	100/98	100/99
	”	5/2	0/3	0/4	0/0	0/1
Peng.	8/23	8/7	0/0	0/0	15/89	85/98
	”	5/10	0/0	0/0	81/99	100/99
	”	5/2	0/0	0/0	0/0	0/0

in its context, and moved slightly to the left/right might end up with the expert coming to the same conclusion as if we did not move the subsequence. For that reason we introduce the concept of Accuracy ±10 (see Table III), which is the accuracy calculated for the motif/discord indices ranging in a ±10 interval. Using this metric the accuracy peaks 100% for most of the datasets.

Mixed precision configurations are aimed at balancing the trade-off between accuracy and performance (time/energy) of the Matrix Profile algorithms. We can reach high peaks of accuracy with the (8/23, 5/10) configuration, while reducing time and energy consumption as described in the following section.

2) Energy.: Fig. 7 presents the normalized energy consumption of a) SCRIMP with respect to SCRIMP double, b) SCAMP with respect to SCAMP double and c) SCAMP with respect to SCRIMP with different precisions, respectively. As the energy is calculated with respect to the number of FPU operations performed by the algorithms, the proportion holds independently of the time series. We can observe a 60% energy reduction when using single precision instead of double for both algorithms. Furthermore, we can not only expect a reduction in time for this configuration due to an improved used of the memory hierarchy, but also because of SIMD capabilities, allowing two single precision elements computed at a time. According to our experiments, single precision provides the same accuracy than double in the majority of cases (see Fig. 5 and Fig. 6).

One way to reduce the energy consumption even further is the use of mixed precision. Our SCRIMPff and SCAMPff mixed precision configurations can yield up to 50% and 25% energy reduction over the single precision application respectively. The savings for SCRIMPff are more pronounced since there are more operations computed in low precision. The dot product is computed in high precision but the distance in Eq. 1 is calculated in low precision. However, the distance

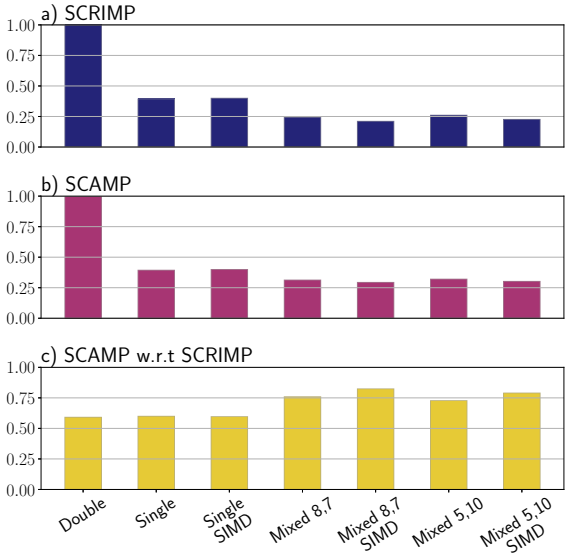

Fig. 7. Normalized FPU energy.

in SCAMPff is given by the Pearson correlation coefficient in Eq. 8 which entails fewer operations. We confirm this fact when comparing SCAMP with respect to SCRIMP in plot c) of Fig. 7, where SCAMP reduces the energy consumption between 18% and 40% for the same given precision. There are also a significant number of comparisons computed in low precision in both algorithms, although the energy cost of this operation is not as high as that of multiplications or sums, so the savings are not so high either. On the other hand, the SIMD support of the FPU yields roughly the same energy numbers but opens the opportunity to improve the performance even more (up to 4 operations at a time).

V. RELATED WORK

Multiple techniques for time series motif and discord discovery can be found in the literature: probabilistic solu-

tions [2], spatio-temporal models [27], indexing [28], symbolic representation [29] or Euclidean distances [10]. A survey of time series motif discovery can be found in [8]. Related to transprecision computing, we can find the project of this paradigm in [13], a software library for transprecision emulation [17] and hardware implementations [14], [18], [30].

The authors of Matrix Profile provide a precision evaluation in [12]. They explain that constant regions are a source of numerical instability due to zero standard deviation. Additionally, they provide a comparison between STOMP [31] and SCAMP, reporting maximum absolute error for each execution. However, this work (1) does not compare SCRIMP over SCAMP, although they compare against GPU-STOMPopt which is based on the same diagonal approach as SCRIMP but without the anytime property; (2) only considers standard IEEE floating point representations (double and single precision), and (3) does not discuss whether motifs and discords are conserved in a measurable manner. To the best of our knowledge, there are no previous works based on transprecision computing for time series analysis, and particularly for Matrix Profile.

VI. Conclusions and Future Work

This work studies the benefits from using a transprecision approach for time series analysis, and particularly from a transprecision FPU, defining the exact needed precisions according to the requirements of each part of the algorithm. We develop `SCRIMPff` and `SCAMPff` implementations that will help the community to design energy-efficient time series analysis solutions based on transprecision RISC-V processors, FPGAs or ASICs while minimizing requirements. Our analysis reveals that, for a variety of applications, the energy consumption of the Matrix Profile algorithms is reduced up to 3.3× compared with double precision, while obtaining accurate results.

An interesting future work would be the evaluation of the transprecision analysis of time series in complete implementations of RISC-V processors and FPGA-based devices, analyzing the improvements of both performance and energy by using this approach in a complete system.

References

[1] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*, 4th ed. Springer-Verlag, 2017.

[2] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic Discovery of Time Series Motifs," in *KDD*, 2003.

[3] E. Keogh, J. Lin, S.-H. Lee, and H. Van Herle, "Finding the Most Unusual Time Series Subsequence: Algorithms and Applications," *Knowl. Inf. Syst.*, 2006.

[4] A. McGovern, D. H. Rosendahl, R. A. Brown, and K. K. Droegemeier, "Identifying predictive multi-dimensional time series motifs: An application to severe weather prediction," *Data Mining and Knowledge Discovery*, 2011.

[5] C. Cassisi, M. Aliotta, A. Cannata, P. Montalto, D. Patanè, A. Pulvirenti, and L. Spampinato, "Motif discovery on seismic amplitude time series: The case study of mt etna 2011 eruptive activity," *Pure Appl. Geophys.*, 2013.

[6] B. Szigeti, A. Deogade, and B. Webb, "Searching for motifs in the behaviour of larval Drosophila melanogaster and Caenorhabditis elegans reveals continuity between behavioural states," *Journal of The Royal Society Interface*, 2015.

[7] P. Garrard, V. Nemes, D. Nikolic, and A. Barney, "Motif discovery in speech: Application to monitoring alzheimer's disease," *Current Alzheimer Research*, 2017.

[8] S. Torkamani and V. Lohweg, "Survey on time series motif discovery," *Wiley Interdis. Rev.: Data Mining and Knowledge Discovery*, 2017.

[9] E. Cartwright, M. Crane, and H. J. Ruskin, "Financial Time Series: Motif Discovery and Analysis Using VALMOD," in *ICCS*, 2019.

[10] 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 *KDD*, 2016.

[11] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh, "Matrix Profile XI: SCRIMP++: Time series motif discovery at interactive speeds," in *KDD*, 2018.

[12] 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 *SoCC*, 2019.

[13] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," in *DATE*, 2018.

[14] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *DATE*, 2018.

[15] R. M. Allen, Q. Kong, and R. Martin-Short, "The MyShake Platform: A Global Vision for Earthquake Early Warning," *Pure and Appl. Geophy.*, 2020.

[16] K. H. C. Li, F. A. White, T. Tipoe, T. Liu, M. C. Wong, A. Jesuthasan, A. Baranchuk, G. Tse, and B. P. Yan, "The current state of mobile phone apps for monitoring heart rate, heart rate variability, and atrial fibrillation: Narrative review," *JMIR Mhealth Uhealth*, 2019.

[17] G. Tagliavini, A. Marongiu, and L. Benini, "Flexfloat: A software library for transprecision computing," *IEEE TCAD*, 2018.

[18] S. Mach, F. Schuiki, F. Zaruba, and L. Benini, "A 0.80 pJ/flop, 1.24 Tflop/sW 8-to-64 bit Transprecision Floating-Point Unit for a 64 bit RISC-V Processor in 22nm FD-SOI," in *VLSI-SOC*, 2019.

[19] "FPnew source code," https://github.com/pulp-platform/fpnew, accessed 25 June 2020.

[20] I. Fernandez, A. Villegas, E. Gutierrez, and O. Plata, "Accelerating time series motif discovery in the Intel Xeon Phi KNL processor," *The Journal of Supercomputing*, 2019.

[21] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov, "Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy," *ACM Trans. Math. Softw.*, 2008.

[22] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, and et al., "Design, implementation and testing of extended and mixed precision BLAS," *ACM Trans. Math. Softw.*, 2002.

[23] C. M. Yeh, H. V. Herle, and E. Keogh, "Matrix Profile III: The Matrix Profile Allows Visualization of Salient Subsequences in Massive Time Series," in *KDD*, 2016.

[24] A. V. Oppenheim and R. W. Schafer, "From frequency to quefrency: a history of the cepstrum," *IEEE Signal Processing Magazine*, 2004.

[25] A. Taddei, G. Distante, M. Emdin, P. Pisani, G. B. Moody, C. Zeelenberg, and C. Marchesi, "The European ST-T database: standard for evaluating systems for the analysis of ST-T changes in ambulatory electrocardiography," *European Heart Journal*, 1992.

[26] "Penguin Data," www.cs.ucr.edu/~eamonn/MatrixProfile.html, accessed 25 June 2020.

[27] A. McGovern, D. H. Rosendahl, R. A. Brown, and K. K. Droegemeier, "Identifying predictive multi-dimensional time series motifs: An application to severe weather prediction," *Data Mining and Knowledge Discovery*, 2011.

[28] B. K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary $\mathcal{L}_p$ norms," in *Int'l. Conf. on Very Large Data Bases*, 2000.

[29] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *KDD*, 2007.

[30] G. Stazi, F. Silvestri, A. Mastrandrea, M. Olivieri, and F. Menichelli, "Synthesis time reconfigurable floating point unit for transprecision computing," in *APPLEPIES*, 2018.

[31] Y. Zhu, Z. Zimmerman, N. S. Senobari, 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 *ICDM*, 2016.