Scaling Time Series Motif Discovery with GPUs: Breaking the Quintillion Pairwise Comparisons a Day Barrier

Zachary Zimmerman, Kaveh Kamgar, Yan Zhu, Nader Shakibay Senobari, Brian Crites, Gareth Funning, Philip Brisk, and Eamonn Keogh
University of California, Riverside
{zzimm001, kkamg001, yzhu015, nshak006, bcrit001, gareth}@ucr.edu {philip, eamonn}@cs.ucr.edu

ABSTRACT

The discovery of conserved (repeated) patterns in time series is arguably the most important primitive in time series data mining. Called *time series motifs*, these primitive patterns are useful in their own right, and are also used as inputs into classification, clustering, segmentation, visualization, and anomaly detection algorithms. Recently the Matrix Profile has emerged as a promising representation to allow the efficient exact computation of the top-*k* motifs in a time series. The state-of-the-art algorithms for computing the Matrix Profile are STAMP and STOMP which are fast enough for *many* tasks. However, in a handful of domains, including astronomy and seismology, there is an insatiable appetite to consider ever larger datasets. In this work we show that with several novel insights we can push the motif discovery envelope using a novel scalable framework in conjunction with a deployment to commercial GPU clusters in the cloud. We demonstrate the utility of our ideas with detailed case studies in seismology, demonstrating that the efficiency of our algorithm allows us to *exhaustively* consider datasets that are currently only approximately searchable, allowing us to find subtle precursor earthquakes that had previously escaped attention, and other novel seismic regularities.

1. INTRODUCTION

Time series motifs are approximately repeated subsequences of a longer time series. As Figure 1 (and Figure 4) suggest, motifs can often reveal unexpected regularities in large datasets. In the last decade, time series motif discovery has become an increasingly important primitive for time series analytics, and is used in domains as diverse as seismology [4], astronomy, geology, ethology [44], neuroscience [22], medicine [13], consumer behavior [45], music [38] and sports analytics. In recent years, algorithmic advances (coupled with hardware improvements) have greatly expanded the purview of motif discovery. It has recently been shown that motif discovery is trivial given a data structure called the Matrix Profile (MP), and that the current state-of-the-art MP batch construction algorithm STOMP, can discover motifs efficiently enough for many users [44].

Moreover, an informal survey of the literature suggests that many medical, scientific and industrial labs, analysts rarely have to deal will datasets with more than a few million data points [24]. For such datasets, STAMP which is an *anytime* algorithm, can produce a high quality approximate MP in minutes, which approaches "interactive" time for most purposes [43]. Note that "minutes" may not seem impressively fast, until you recall that many of datasets in question take days or weeks to collect. As a concrete example, in Figure 1 the approximate motif discovery for this full-day chicken behavior dataset takes well under an hour. The biologist using this tool reports that "*this is fast enough for what I need.*" [24].

Nevertheless, we argue that there is an insatiable need for further scalability. Some domains, including seismology, astronomy and neuroscience have a near inexhaustible appetite to consider ever larger datasets. For example, a recent paper reports that by simply performing (approximate) motif search on larger datasets "*directly enabled the discovery of 597 new earthquakes near the Diablo Canyon nuclear power plant in California*" [22]. Undoubtedly, *exact* search of the same dataset (or a larger superset thereof) would yield even further subtle instances of unexpected regularities.

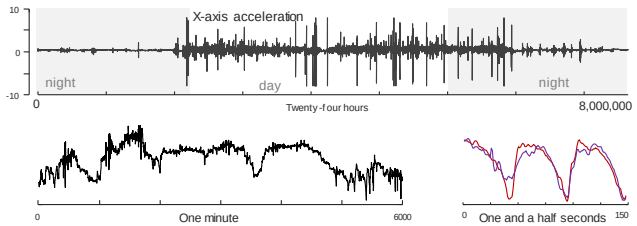

Figure 1: *top*) Twenty-four hours of time series from an accelerometer worn by a chicken (*Gallus domesticus*). *bottom-left*) A zoom-in shows that the data is apparently void of structure. However, the top-1 motif (*bottom.right*) suggests that some behaviors *are* highly conserved. Inspection of video recorded in parallel suggests this a *dustbathing* behavior [22].

Given this demand from domain experts, we have created a framework called SCAMP (SCAlable Matrix Profile) that greatly expands the purview of *exact* motif discovery. We summarize our major contributions below:

1. We provide a general distributed framework for the ultra-scalable computation of the Matrix Profile [43]. Both the performance and numerical stability are greatly improved via our method when dealing with long time series.
2. Our framework allows us to work with time series data which do not fit wholly into GPU memory, allowing Matrix Profiles to be computed which are larger than previously considered.
3. We introduce novel numerical methods to increase performance and improve stability of the Matrix Profile computation; this allows the use of single-precision floating-point calculations for many datasets, which in turn allows our methods to be applied to larger datasets at a cheaper amortized cost.
4. We have created a novel *fault-tolerant* framework that is compatible with the use of "spot" instances [47], which major cloud providers (Amazon, Google, and Microsoft) offer at a substantial discount, making motif discovery more affordable.
5. We provide a freely available open-source implementation of our framework which runs on Amazon Web Services in a cluster of instances equipped with Nvidia Tesla V100 GPUs, as well as optimized CPU code at [50].

The rest of this paper is organized as follows. In Section 2, we state our assumptions, introduce necessary definitions, and summarize related work. Section 3 has a description of our novel scalable framework and the improvements we made that allow us to further push the boundary of Matrix Profile calculations. In Section 4, we illustrate a few of the use cases for very large Matrix Profiles through several case studies on challenging datasets. In Section 5, we provide a detailed empirical analysis of our ideas, before offering conclusions and directions for related work in Section 6.

2. DEFINITIONS AND ASSUMPTIONS

We begin by stating our key assumption; it has been developed at length elsewhere [43][44], but we repeat it here for concreteness.

Key Assumption: *Motif discovery under any reasonable definition is trivial if given the Matrix Profile data structure.*

That is to say, there are a handful of definitions of time series motifs, top-*k* motifs, range motifs, biased motifs [9], contextual motifs [13] etc. No matter which definition is required, the Matrix Profile alone is all that is needed to extract the motif in linear time and space [40][43]. Given this observation, in this paper we focus on solely on computing the Matrix Profile as efficiently as possible; the reader can appreciate that this implicitly solves the task at hand. Our key assumption actually understates the case. Having the Matrix Profile computed is sufficient to solve many additional time series data mining tasks, including, discord discovery, chain discovery, snippet discovery, segmentation etc. [40][43]. For simplicity we ignore these additional uses of the Matrix Profile here.

We can now formally define the data type of interest, *time series*:

Definition 1: A time series T is a sequence of real-valued numbers t_i: $T = t_1, t_2, ..., t_n$ where n is the length of T.

We are typically interested not in *global*, but *local* properties of a time series. A local region of time series is called a *subsequence*:

Definition 2: A subsequence $T_{i,m}$ of a time series T is a continuous subset of the values from T of length m starting from position i. Formally, $T_{i,m} = t_i, t_{i+1},\ldots, t_{i+m-1}$, where $1 \leq i \leq n\text{-}m+1$.

Given a query subsequence $T_{i,m}$ and a time series T, we can compute the distance between $T_{i,m}$ and *all* the subsequences in T. We call this a *distance profile*:

Definition 3: A *distance profile* D_i corresponding to query $T_{i,m}$ and time series T is a vector of the Euclidean distances between a given query subsequence $T_{i,m}$ and each subsequence in time series T. Formally, $D_i = [d_{i,1}, d_{i,2}, ..., d_{i,n-m+1}]$, where $d_{i,j}$ $(1 \leq j \leq n\text{-}m+1)$ is the distance between $T_{i,m}$ and $T_{j,m}$.

We assume that the distance is measured by Euclidean distance between z-normalized subsequences [43][44]. Once we obtain D_i, we can extract the nearest neighbor of $T_{i,m}$ in T. Note that if the query $T_{i,m}$ is a subsequence of T, the i^{th} location of distance profile D_i is zero (i.e., $d_{i,i} = 0$) and close to zero just to the left and right of i. This is called a *trivial match* in the literature. We avoid such matches by ignoring an "exclusion" zone of length m/k before and after i, the location of the query, where $1 < k < m\text{-}1$.

What should the value of k be set to? In more than a dozen works considering hundreds of diverse datasets it has been shown to be inconsequential [9][43][44]. There is *one* possible case that would require more careful introspection. It is best explained by an analogy to *text* motifs in the presence of anadiplosis. Consider this line of wordplay from a Monty Python sketch ".. the very meaning of life itself*ish bastard*…". Here the string "self" belongs to *both* 'itself' and to 'self*ish*'. Something similar can happen with time series data. For example, in a motion captured ASL performance, the end of one signed word can overlap the beginning of the next word. In such a case the user needs to decide if he is willing to allow such overlapping by setting k to a smaller value.

However, given the relative unimportance of k, we simply set $d_{i,j}$ $(i\text{-}m/4 \leq j \leq i+m/4)$ to ∞, and the nearest neighbor of $T_{i,m}$ can thus be found by evaluating $\min(D_i)$.

We wish to find the nearest neighbor of every subsequence in T. The nearest neighbor information is stored in two meta time series, the *Matrix Profile* and the *Matrix Profile index*.

Definition 4: A *Matrix Profile* P of time series T is a vector of the Euclidean distances between every subsequence of T and its nearest neighbor in T. Formally, $P = [\min(D_1), \min(D_2), ..., \min(D_{n-m+1})]$, where D_i $(1 \leq i \leq n\text{-}m+1)$ is the distance profile D_i corresponding to query $T_{i,m}$ and time series T.

The i^{th} element in the Matrix Profile P tells us the Euclidean distance from subsequence $T_{i,m}$ to its nearest neighbor in time series T. However, it does not tell us the *location* of that nearest neighbor; this information is stored in the companion *Matrix Profile index*:

Definition 5: A *Matrix Profile index* I of time series T is a vector of integers: $I=[I_1, I_2, \ldots I_{n-m+1}]$, where $I_i=j$ if $d_{i,j} = \min(D_i)$.

Figure 2 illustrates the relationship between distance matrix, distance profile (Definition 3) and Matrix Profile (Definition 4). Each element of the distance matrix $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$ $(1 \leq i, j \leq n\text{-}m+1)$ of time series T.

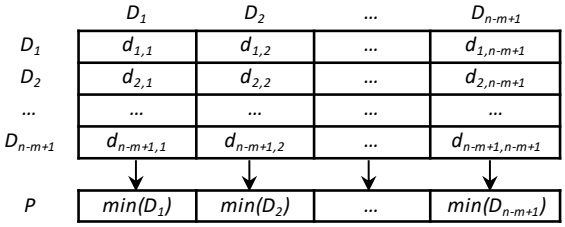

Figure 2: The relationship between the distance matrix, distance profile and Matrix Profile. A distance profile is a column (also a row) of the distance matrix. The Matrix Profile stores the minimum (off diagonal) value of each column of the distance matrix; the location of the minimum value within each column is stored in the companion Matrix Profile index.

Figure 3 shows a visual example of a distance profile and a Matrix Profile created from the same time series T.

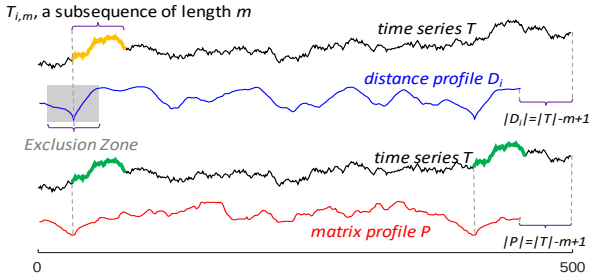

Figure 3: *top*) A distance profile D_i created from $T_{i,m}$ shows the distance between $T_{i,m}$ and all the subsequences in T. The values in the dark zone are ignored to avoid trivial matches. *bottom*) The Matrix Profile P is the element-wise minimum of all the distance profiles (D_i is one of them). Note that the two lowest values in P are at the location of the 1st motif in T).

Note that as we presented it, the Matrix Profile is a *self-join*: for every subsequence in a time series T, it records information about its (non-trivial-match) nearest neighbor in the *same* time series T.

However, we can trivially generalize it to be an AB-join; for every subsequence in a time series *A*, record information about its nearest neighbor in time series *B* [44][45]. Note that *A* and *B* can be of different lengths, and that in general, AB-join ≠ BA-join.

2.1 Observations on Precision

It has noted by several independent research groups that at least for some time series retrieval tasks, 64-bit precision is unnecessarily precise [3][40]. In recent years, and in various disciplines, researchers have shown that reduced precision computation can be exploited to have significant performance benefits with little to no observable difference in quality of results [40][15]. This observation has been heavily exploited in deep learning [15][48]; however, it is rarely exploited for time series, except for to allow the use of Minimal Description Length to score and rank models [3], which is orthogonal to scalability considerations. In Figure 4 we show a Matrix Profile computed on some insect electrical penetration graph (EPG) using 64-bit precision.

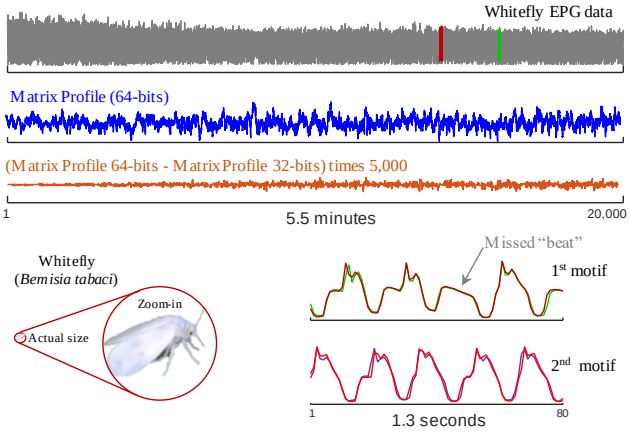

Figure 4: *top-row*) A snippet of whitefly insect EPG data. *second-row*) The Matrix Profile computed with 64-bit precision. *third-row*) Because the 64-bit and 32-bit Matrix Profiles are visually identical at this scale, we subtract them, and multiplied the difference by 5,000. *bottom-row*) The whitefly is a tiny insect, yet is produces extraordinary well conserved motifs.

This plot is very suggestive; There is a difference between the Matrix Profiles computed at 64 and 32-bit precision, but it is so small it does not affect the motifs discovered, and indeed is not even visible unless we multiple the difference by a huge constant. However, there are two caveats to consider.

- The time series shown in Figure 4 is relatively short. As we address ever longer time series, there is more potential for accumulated floating-point error making a significant difference [18]. Indeed, even in this example we can see that the difference vector gets larger as we scan left to right (Figure 4.*third-row*). We address this issue in Section 4.2
- The time series shown in Figure 4 also has the property that the information within it is contained within small range. This is true for some types of data, ECGs, accelerometer and gyroscope readings etc. However, there are handful of application domains for which this is not true, and seismology is one of them. A "great" earthquake has a magnitude of 8 or greater, but humans can feel earthquakes with a magnitude of just 2.5, a difference of more than five orders of magnitude. Processing raw data that spans such a range requires careful thought and very careful implementation, we address this in Sections 3.2, 3.3, and 4.2

Before moving on, we note that this illustration offers another example of the utility of motif discovery. The data in Figure 4 represents a tiny fraction of the entomologist's data archive [23]. The pattern represented by the 2nd motif is common and immediately recognizable by the entomologists as *ingestion of xylem sap behavior* [35]. However, the 1st motif was unexpected, there is a "missed beat" during the xylem sap ingestion cycle. If we had observed a *single* example of this, we might have attributed it to chance or noise. However, motif discovery shows us that there are at least two strongly conserved examples (in fact, there are several other examples). This suggests that there is some semantic meaning to this motif, which the entomologists are currently exploring [23].

In order to support the massive computations required to find unexpected regularities in these huge archives, we turn to the compute potential of GPUs and introduce our GPU framework for computing massive Matrix Profiles.

3. The SCAMP Framework

To compute large Matrix Profiles, we introduce a framework that can be used by a combination of a ***host*** and one or more ***workers***. A **host** could be a local machine, or a master server in a compute cluster. **Workers** follow the host's direction and can be other CPU-based systems in a cluster, or an accelerator such as a GPU. We use the term **cluster** to refer to a host and all of its associated workers. While a cluster can refer to the typical group of co-located nodes in a cloud, it can also refer to a single node with accelerators attached (e.g. a server equipped with several GPUs).

3.1 A Brief Overview of GPU-STOMP$_{OPT}$

GPU-STOMP$_{OPT}$ [45] is the current state of the art for computing Matrix Profiles on the GPU. The SCAMP algorithm can best be described in terms of a set of modifications and extensions to GPU-STOMP$_{OPT}$. Thus, for completeness, we include a brief description of the GPU-STOMP$_{OPT}$ algorithm below. The reader familiar with this material can skip to Section 3.2.

An illustration of the GPU-STOMP$_{OPT}$ algorithm is shown in Figure 5 (left). In GPU-STOMP$_{OPT}$, each thread computes a diagonal of the distance matrix shown in Figure 2 by updating the dot product, QT, at each point along the diagonal using Equation 1 and then computing the distance, $d_{i,j}$, via Equation 2.

$$QT_{i,j} = QT_{i-1,j-1} - t_{i-1}t_{j-1} + t_{i+m-1}t_{j+m-1} \quad (1)$$

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

Each GPU thread block computes the distances in parallelogram-shaped tiles along a 'meta'-diagonal. For each tile, the values needed to compute the distances are loaded into GPU shared memory. By loading the intermediate values into shared memory we achieve spatial and temporal reuse of local data. We keep a best-so-far cache of the Matrix Profile in shared memory as well; when a new distance is computed we update this cached value if its value is lower than the smallest distance computed thus far. After every tile's distances are computed, we check if any values in our thread-block-local copy of the Matrix Profile need to be pushed to the global Matrix Profile, and update via atomic access if necessary.

In SCAMP, we improve several aspects of GPU-STOMP$_{OPT}$, which leads to a several-fold improvement in the performance of the algorithm and allows us to exploit newer GPU hardware more efficiently. We explain these improvements in great detail in the following sections. For additional details on STOMP, GPU-STOMP and GPU-STOMP$_{OPT}$ please refer to [40][41][45].

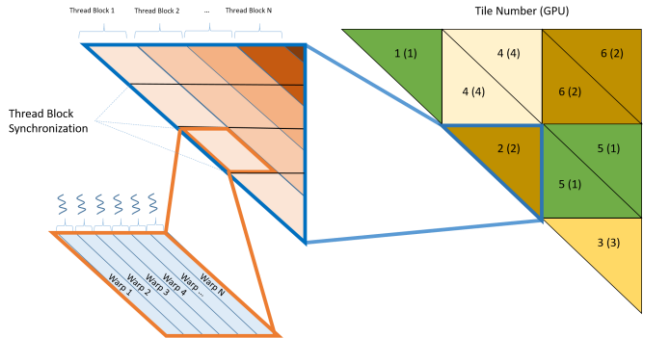

Figure 5: *left*) The GPU-STOMP$_{OPT}$ GPU execution pattern. This execution pattern is shared with the SCAMP_tile algorithm. *right*) The SCAMP tiling scheme using 4 GPUs. The illustration of the tiling scheme is for self-joins only. Note that while we only illustrate an upper triangular tile, the lower triangular tile is computed with the same implementation, but with the inputs transposed. For AB-joins we also need to compute the lower-triangular portion of the distance matrix.

3.2 Tiling Scheme

Instead of computing the entire distance matrix in one operation we can split the distance matrix into tiles, where each tile computes an AB-join between two segments of the input time series. Each tile is computed independently, which allows us to scale the computation to very large input sizes and distribute the work to many independent machines. Figure 5 (*right*) illustrates the intuition of our tiling scheme

The host machine maintains information about its workers their available hardware, such as the type of GPUs are available on the system, the memory capacity, and the CPU speed to determine a tile width that can saturate its workers and to delegate jobs among them. In the general case, this delegation can be complex and challenging to do effectively. For simplicity, in this paper we assume that all of the workers are the same and that the most effective tile size is one that fully saturates the worker during execution. This is currently discovered empirically, but once it is discovered for a given system it can be hardcoded. While our framework is compatible with heterogeneous infrastructure, for simplicity of presentation we assume a homogeneous compute cluster. We leave a detailed exploration of tile size and heterogeneous compute clusters to future work. For our use case, V100 GPUs become saturated around a tile width of one million, which is used as the default tile size for SCAMP and is the tile size used throughout the rest of this paper, unless otherwise noted. The host splits the distance matrix into square tiles whenever possible using the desired tile width and assigns each one to a worker; the following subsections describe the algorithms run on the host and the workers.

3.2.1 SCAMP_host

The host executes the **SCAMP_host** algorithm, shown in Table 1. SCAMP_host employs asynchronous streams, which allow the host to issue tasks to workers without blocking and without needing to explicitly manage the ordering of each operation and completion. The stream manages the dependencies of each event issued on it and enforces that the events within a stream execute sequentially.

Lines 1 and 2 determine the appropriate tile size for the problem instance and determines the tile ordering. For this work, the reader can assume that tiles are issued in order along the diagonal as shown in Figure 5, this allows us to use the host as a 'cache' for the current *best-so-far* Matrix Profile values. As additional tiles are computed, they can use the current information as their initial value, instead of having to start from scratch. This results in fewer memory reads/writes during computation.

Table 1: The SCAMP_host Algorithm

	Procedure *SCAMP_host()*
	Input: *U*ser provided time series T, window length w, workers
	Output: Matrix profile P and Matrix Profile index I, of time series T using window length w
1	n ← Length(T)
2	TileSize ← **GetTileSize**(), TileOrder ← **GetTileOrder**()
3	**for each** worker **do**:
4	mem[worker] = **AllocateStorage**(T, TileSize)
5	streams[worker] = **AllocateAsynchronousStream**(worker)
6	tile ← **PopTile**(TileOrder)
	// Start the first tile on each stream
7	**SCAMP_tile**(mem[worker], tile, stream[worker])
8	**done**
9	**while** TileOrder **not empty do**:
10	**for each** worker **do**:
	// Add operations to each stream
	// streams wait for previous ops before starting the next
	// Retrieve the data, then start the next tile if it exists
11	**RetrieveDataAsync**(mem[worker], tile, streams[worker])
12	**if** TileOrder **not empty**:
13	next_tile ←**PopTile(**TileOrder)
14	**SCAMP_tile**(mem[worker], tile, streams[worker])
15	**done**
	// Gather the data from each worker, then start then next tile on
	// the worker while the host merges the results from the
	// previous tile in parallel.
16	**for each** stream **in** streams:
17	mp, mpi = **SynchronizeAndRetrieveData**(stream)
18	**MergeResult**(P, I, mp, mpi)
19	**done**
20	**done**
21	**return** P, I

For each worker: line 4 allocates memory for all temporary arrays required to produce the result (the array length depends on the tile size); line 5 allocates an asynchronous stream for each worker; line 6 chooses the next tile to execute; and line 7 asynchronously issues a tile to each worker.

Lines 10-15 issue events to retrieve the result from each worker as it finishes its current tile, and issues the next tile to the worker when the data transfer finishes; this ensures that the steady state always has an in-flight tile executing on each worker. Once the data transfer is complete and the worker starts processing the next tile lines 16-19 merge the partial results into the Matrix Profile.

Note that there are some minor optimizations that can be performed to further improve tile ordering and reduce the overhead of tiling in certain cases. To improve the flow of this work, this optimizations are relegated to the supporting documentation at [50].

3.2.2 SCAMP_tile

Each worker executes the **SCAMP_tile** algorithm, shown in Table 2, to compute the intermediate result for that particular tile. The implementation of the SCAMP_tile algorithm depends on the worker's architecture. The host, which is aware of each worker's respective architecture, distributes the work accordingly.

Line 1 initializes the current Matrix Profile value to the *best-so-far* value computed for other tiles. Line 2 computes the initial dot product values associated with the upper triangular tile. Line 3 executes an architecture-optimized kernel to compute the Matrix Profile and Index for that tile. Line 4 computes the initial dot product values associated with the lower triangular tile and line 5 computes the result associated with that tile.

Prior work established that the self-join problem exhibits symmetry in the distance matrix [41][42]; however, it was less obvious that the memory access pattern and the order in which distances are computed in SCAMP and GPU-STOMP$_{OPT}$ also exhibit symmetry. In particular, the inputs can be transposed and our framework can generate the lower-triangular portion of the matrix as if it were the upper-triangular portion using the same implementation. We exploit this property to implement AB-joins in the SCAMP framework.

Table 2: SCAMP Tile Computation

Procedure *SCAMP_tile()*	
Input: Tile data **d**, time series **A**, time series section **B**	
Output: Matrix Profile **mp** and index **mpi** between section **A** and **B**	
1	mp ← d.best_so_far_mp, mpi ← d.best_so_far_index
	// Generate the initial dot products for the upper-right tile
2	QT ← **SlidingDotProducts**(A,B)
	// Compute the Matrix Profile for the upper-right tile
	// DoTriangularTile is implemented similarly to GPU-STOMP$_{opt}$
3	mp, mpi ← **DoTriangularTile**(A, B, d, QT, mp, mpi)
	// Generate the initial dot products for the lower-left tile
4	QT ← **SlidingDotProducts** (B,A)
	// Update the Matrix Profile using results from the lower-left tile
5	mp, mpi ← **DoTriangularTile** (B,A, d, QT, mp, mpi)
6	**return** mp, mpi

This approach has several distinct advantages GPU-STOMP$_{OPT}$:

- **Numerical Stability:** Every new tile introduces a 'reset' point for SCAMP's extrapolation. Each time a tile is started we directly compute the value of the initial dot products at that row and column of the distance matrix in high precision. This reduces the chance that we have large rounding errors and that when there *is* a rounding error, we do not propagate it as far. This contrasts with GPU-STOMP$_{OPT}$ where we extrapolate the diagonals of the distance matrix from a single initial value.
- **Preemptability:** Each tile is independently issued and completed. The SCAMP_tile algorithm is therefore preemptable, increasing the fault-tolerance of our framework. If a worker executing a tile "dies" or otherwise fails to complete its work, the host can just issue that work again without having to recompute work performed by machines other than the worker that failed. This benefit is extremely useful: as mentioned in Section 1, many commercial cloud providers allow for users to purchase spot instances at significantly reduced prices. These spot instances are only useful to fault-tolerant applications, as the cloud provider can kill the instance at any point. Using the fault-tolerance of SCAMP, we are able to use cloud resources at much cheaper rates than other applications which are not fault-tolerant. Enabling immense scalability requires significant compute resources, by reducing the cost of these compute resources we can afford more compute resources, thereby increasing the size of the datasets we can consider given a limited financial budget.
- **Extensibility:** Since each tile is independently computed, we can provide different options for each tile's computation. This allows for the potential of utilizing a heterogeneous infrastructure, with varying amounts of memory and compute power. Additionally a user could provide a separate, optimized implementation of the SCAMP_tile algorithm for each individual type of worker. Allowing for high performance even in a highly heterogeneous environment. There are other benefits to this extensibility, perhaps the user knows that a particular segment of the time series is important and wants to 'promote' motif discovery in that region. The user could customize the distance calculation for the tiles of specific interest, perhaps giving a higher weight to distances computed between the interesting regions.
- **Localized Matrix Profiles:** When each tile is computed by a separate device, the intermediate result must be stored somewhere to later be merged into the final Matrix Profile; however, these intermediate results represent the Matrix Profiles for pairs of localized regions in the data and could be immediately useful to the user. For example, when performing a join on one year of data, the tile size could be chosen to be one month. In this case, the intermediate results of the year-long data represent the AB-join Matrix Profiles for each pair of months. The utility of localized Matrix Profiles is illustrated in Section 5.2.

One disadvantage of SCAMP is that the subsequence window size cannot be larger than the tile size, and this is assumed in our implementation. If the window size was larger than the tile size, then each worker would have to process an overlapping region between tiles, resulting in a large amount of additional overhead and boundary checking, and would prevent SCAMP from partitioning the Matrix Profile computation into truly independent chunks. However, the subsequence length is almost always *much* smaller than the tile size, which is usually in the millions [43][44].

3.3 Numeric Changes for Speed and Stability

The numerical stability (and performance) of algorithms involving floating-point arithmetic can often be improved by carefully reordering the operations. With this in mind, SCAMP replaces GPU-STOMP$_{OPT}$'s sliding dot product equation update with a centered-sum-of-products formula, as shown in Equations 3-7.

$$df_i = \frac{T_{i+m} - T_i}{2} \tag{3}$$

$$dg_i = (T_{i+m} - \mu_{i+1.m}) + (T_i - \mu_{i,m}) \tag{4}$$

$$\overline{QT}_{i,j} = \overline{QT}_{i-1,j-1} + df_i dg_j + df_j dg_i \tag{5}$$

$$P_{i,j} = \overline{QT}_{i,j} * \frac{1}{\|T_{i..i+m}\|} * \frac{1}{\|T_{i..i+m}\|} \tag{6}$$

$$D_{i,j} = \sqrt{2m(1 - P_{i,j})} \tag{7}$$

Equations 3 and 4 are precomputations which compute the terms used in the sum-of-products update formula of Equation 5, and incorporate incremental mean centering into the update. Equations 3, 4, and 5 are specifically for self joins and are a special case of a more general formula for an AB-join (see Ref. [50] for a full derivation). Empirically, this new formula reduces the number of incorrectly rounded bits.

Additionally, we replace the Euclidean Distance (ED) used in previous Matrix Profile computations [43][44][45] with the Pearson Correlation (P) in Equation 6; P can be computed incrementally using fewer computations that ED, and can be converted to a normalized ED in linear time using Equation 7. As a further optimization, we precompute the inverse L2-norms in Equation 6 to eliminate redundant division operations from SCAMP's inner loop.

3.4 More optimizations for GPU-STOMP$_{OPT}$

The improved numerical stability that results from the changes reported in the previous section enabled us to explore the usage of reduced-precision floating-point operation without substantially sacrificing computational accuracy. Using lower-precision values improved cache performance and increases the number of concurrent floating-point operations that each GPU can execute. Thus, we implemented SCAMP using three different precision modes:

Double Precision (SCAMP DP): We store all intermediate shared memory values in double-precision floating-point and use double

precision floating-point instructions for all computation. This provides accurate results for all datasets that we considered, regardless of size, noise, ill-conditioned regions etc.

Single Precision (SCAMP SP): We store all intermediate data and perform all calculations in single precision. Using half the data width of double precision increases our memory and compute performance by approximately 2x. We found that this mode is adequate for highly regular datasets, such as ECG or accelerometer data, but may yield incorrect results for ill-conditioned data; a detailed analysis is deferred until Section 4.2.

Mixed Precision (SCAMP MIX): This is similar to SCAMP SP, all intermediate values are stored in single precision and calculations are performed in single precision, except for Equation 5, which is computed using double-precision. Accumulating in double-precision along the diagonal provides the numerical stability of double-precision, while allowing the use of faster single-precision operations for computations whose numerical stability is less critical; however, this necessitates the introduction of several type casting operations can which increases pressure on special floating-point units (SFUs) which perform this conversion in the GPU. Some of this pressure can be alleviated by performing only the final add in Equation 5 in double-precision, keeping intermediate values in single precision only until the very end. This in turn prevents us from using floating-point multiply-add (FMA) instructions, as GPUs do not support mixed-precision FMA. We have found that SCAMP MIX provides accurate results for many datasets which can be represented in single precision.

We also experimented with half-precision (16-bit) floating-point operations, but found that incorrect motifs were identified for many of our data sets; consequently, we exclude half-precision from further consideration in this paper.

The new Matrix Profile computation in Eq. 3-7 reduces each thread's demand for shared memory. We increase the amount of shared memory allocated to each thread, allowing each thread to compute four separate diagonals. Specifically, we unroll the loop four times so that each thread computes sixteen new distances (four distances for each of four diagonals) per iteration, while ensuring the per-thread-block memory usage remains low enough to achieve 50% occupancy on a Tesla V100 GPU. For the interested reader, more information on the Volta GPU architecture is detailed in [52]; we have empirically observed that Matrix Profile computation is bound by shared memory *loads*, not compute time.

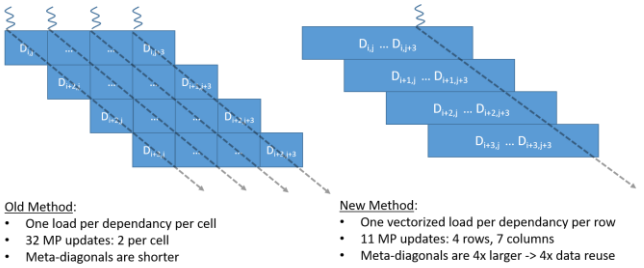

Figure 6: Illustration of one iteration of the innermost loop of our GPU implementation. Note that for self-joins, since we only compute half of the distance matrix, we must track both the Matrix Profile value for the columns and for the rows. For AB-joins we only need to look at the columns or the rows.

Further, this unrolling scheme facilitates the usage of vectorized shared memory loads for dependencies, which further reduces the pressure on the shared memory bottleneck and increases throughput by reducing the number of shared memory transactions executed. The FP32 and mixed-precision modes can execute two 128-bit loads per column dependency and one 128-bit load per row dependency, and all intermediate values computed during the Matrix Profile calculation can be stored in registers without spilling.

As the row or column distances are computed, we track the maximum per-row and per-column distances and update the corresponding Matrix Profile value in shared memory accordingly, resulting in row-wise and column-wise updates for each distance that is computed; this contrasts with GPU-STOMP$_{OPT}$ where every newly computed distance computed is compared to the Matrix Profile cache. Figure 6 illustrates our strategy for the innermost loop.

3.5 Putting it all Together: A GPU Cluster

We evaluate SCAMP, as described in the preceding subsections, on Amazon Web Services, which we selected as a being representative of commercially available cloud services. We illustrate our method in Figure 7. First, we partition a locally stored time series data set and partition it into equal-sized chunks ranging from 20 to 100 million elements. There is a tradeoff here between distributed overhead of starting new jobs, intermediate data size, and the risk of a job being preempted and losing work. We compress each chunk store it on the cloud (Amazon S3), where Amazon EC2 instances can readily download it. We use AWS batch to set up a job queue backed by a compute cluster of p3.16xlarge spot instances. As noted in Section 3.2, spot instances allow us to exploit the fault-tolerance of SCAMP, by running on machines that can be preempted without the risk of losing all of our progress, we can purchase this compute power more cheaply than if we needed dedicated resources, and this improves the price per performance of SCAMP. We issue an array batch job where each job computes the Matrix Profile for one tile. We issue one job at a time to each worker, and the tile size is specified to ensure that we achieve full saturation of the compute resources in each worker, maximizing the throughput of this pipeline without losing exorbitant progress if our instance gets preempted.

Each job first copies and decompresses the input segments that correspond to the row and column of its tile. Each tile has two inputs, one segment corresponding to the tile-row, and one segment corresponding to the tile-column, each job computes an AB-join on these inputs. Next, the job executes the SCAMP_host algorithm on the input, further subdividing its tile among its worker GPUs. Once SCAMP_host computes the Matrix Profile and index associated with the tile, it compresses the result and returns it to Amazon S3. The job is now complete and is dequeued. After all jobs terminate, we use another job to decompress and merge each tile's Matrix Profile into the final result. As long as the intermediate data doesn't get too large, this is a relatively simple step. In our one billion datapoint experiment, we merged the 196 GB of intermediate results in only 1 hour using only a single machine on AWS. This merging could be parallelized to take just a few minutes in a MapReduce [10] framework.

Intermediate output data volumes can be as large as tens or hundreds of gigabytes for input sizes as large as one billion elements. Small tile sizes produce too much localized information to reasonably store, even if it would otherwise be of interest to the user. SCAMP's space requirement is RN where R is the number of tile rows, and N is the length of the final Matrix Profile. If the tile size is 1, then $R = N$ and processing one billion elements would necessitate storage of the entire distance matrix comprising one quintillion values.

To put these numbers into context, if each intermediate value was eight bytes compressed on disk, the total storage requirement would be eight exabytes, the estimated aggregate storage capacity of Google's datacenters as recently as 2014 [42].

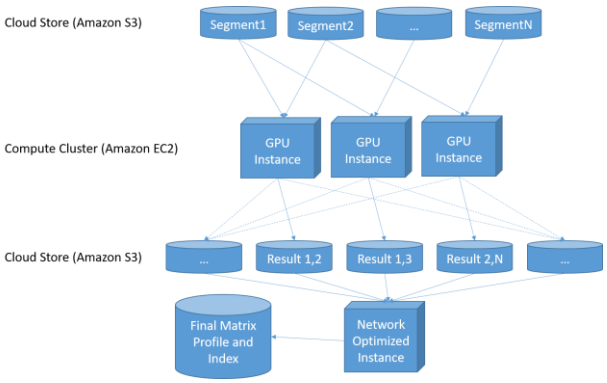

Figure 7: Illustration of how we distribute SCAMP in a cluster of GPU instances on AWS.

4. Empirical Evaluation

We begin by noting that all our experiments (including all the figures above) are reproducible. All code and data (and additional experiments omitted for brevity) are archived in perpetuity [50].

4.1 Performance Evaluation

4.1.1 *Comparison to GPU-STOMP$_{OPT}$ using V100*

We evaluate the performance of SCAMP against GPU-STOMP$_{OPT}$ and provide the results in Table 3. The first column shows the performance of GPU-STOMP$_{OPT}$ using the code from [45] on an Nvidia Tesla K80. The results here are very similar to those provided in [45] but vary slightly because we changed the timing of the experiment to be more precise.

The second column shows the timing results for the same code running on a single Nvidia Tesla V100 SXM2 on Amazon Elastic Compute Cloud. The large speedup here is due to two factors. First, the V100 has a much higher instruction throughput and memory bandwidth. Secondly, and even more importantly, on the Tesla K80 we are heavily bottlenecked by the instruction latency of the atomic updates to the Matrix Profile in shared memory. Shared memory atomics were emulated in software prior to the Maxwell architecture; post-Maxwell, they are implemented in hardware and included natively in the instruction set architecture (ISA) [49]. In some cases, emulated shared memory atomics could be slower than global memory atomics. Matrix Profile updates emerged as a performance bottleneck on the Tesla K80 due to these emulated atomics after all the other optimizations were applied.

The third column shows the timing results for SCAMP over column 1. The additional speedup is a result of the optimizations we made via our changes in Sections 3.3 and 3.4. The fourth and fifth columns show the additional speedup that using our two single precision modes in SCAMP provide. Recall that these results may not be identical to FP64 given the difference in numerical precision.

Table 3: SCAMP Runtime Evaluation

Algorithm	STOMP-GPU$_{OPT}$		SCAMP		
Architecture	K80	V100	V100	V100	V100
Precision	DP	DP	DP	MIX	SP
2^{18}	3.04s	0.34s (**8.9x**)	0.28s (**10.9x**)	0.27 (**11.3x**)	0.24s (**12.7x**)
2^{19}	11.4s	1.24s (**9.2x**)	0.68s (**16.8x**)	0.64 (**17.8x**)	0.57s (**20.1x**)
2^{20}	44.1s	4.81s (**9.2x**)	2.05s (**21.5x**)	1.82 (**24.2x**)	1.42s (**31.1x**)
2^{21}	174s	19.0s (**9.2x**)	6.99s (**24.9x**)	5.73 (**30.4x**)	4.38s (**39.8x**)
2^{22}	629s	69.2s (**9.1x**)	25.8s (**24.4x**)	21.5 (**29.3x**)	15.5s (**40.7x**)
2^{23}	2514s	277s (**9.1x**)	96.8s (**26.0x**)	77.8 (**32.3x**)	52.5s (**47.9x**)

4.1.2 *Ideal GPU Scalability*

Like prior algorithms that compute the Matrix Profile, SCAMP's ideal execution time is quickly and deterministically computable. Given the runtime of SCAMP on one GPU on a dataset of sufficient size to saturate compute performance we can construct an analytical model to estimate the execution time across *G* V100 GPUs on any data size under ideal assumptions (e.g., no communication overhead). We can use this model to answer an arguably more interesting question: Given G V100 GPUs, what size Matrix Profile can SCAMP compute given time T? Equation 8 answers the question, where N_o and T_o are initialization parameters provided by a trial run on a single V100 GPU. This equation is derived from the $O(n^2)$ computational complexity of SCAMP. We use this equation and the double precision SCAMP runtime for input size 2^{23} (Table 3) to construct Figure 8.

$$N = N_o \sqrt{\frac{TG}{T_o}} \quad (8)$$

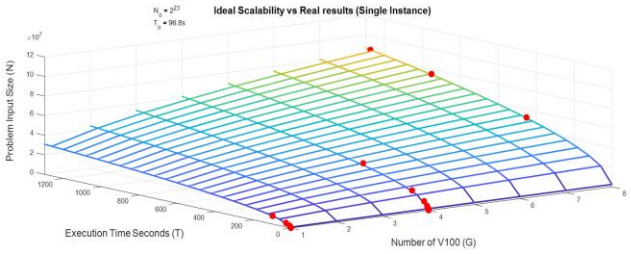

Figure 8: Equation 8 plotted using N_o and T_o from Table 3, the V100 double precision result for 2^{23}. Dots correspond to values measured during experiments throughout this paper. This is for a single non-preemptable instance equipped with GPUs. Equation 8 applies for multi-instance distributed work as well.

Each dot in Figure 8 corresponds to an experiment we performed, to illustrate how our analytical model holds up empirically. Note that the data for our distributed workload for the Cascadia Subduction Zone also aligns well with this plot; however, we did not include it due to space/readability constraints. The reader can find the more detailed figure at our supporting webpage [50].

An interesting conclusion we can draw from this is that the cost for a problem is constant if there is no distributed overhead. For example, to compute a join of 530 million in FP32, you can either use 8 V100s for 8 hours, or 64 V100s for 1 hour. The cost for these two options is the same if the cost per hour per V100 is the same.

4.1.3 *Distributed Performance on p3 spot instances*

We evaluated SCAMP's distributed capabilities on two very large earthquake datasets. Both datasets used forty V100 GPUs, each in a different configuration and both ran on an AWS EC2 Spot Instance fleet. A spot instance is a compute resource that AWS can preempt at any time (due to load, demand, etc.) so jobs running on a spot instance must be interruptible. When AWS preempts a SCAMP tile instance, we restart the tile which was being computed at the time, so some amount of work is lost. A spot instance fleet automatically provisions a consistent number of spot instances for the job queue. If one instance is preempted, AWS will provision another one for the fleet as long as there are instances available.

In Table 4 we see the results for the two large experiments done. For the Parkfield dataset, we ran on a five p3.16xlarge spot instance fleet, each of which is equipped with eight V100 GPUs, totaling forty across the fleet. Unfortunately, we found that the p3.16xlarge

instances were in high demand at the time we ran the experiment, so it took a long time to complete, as the jobs remained queued, during periods where AWS could not provide us with capacity to execute; however, we only paid for active GPU computation time.

Table 4: Summary of various distributed runs

Dataset	Parkfield 1B	Cascadia Subduction Zone
Size	1 Billion	1 Billion
Tile Size	~52M (1 month)	~ 25M (2 weeks)
Total GPU time	375.2 hours	375.3 hours
Spot Job Time	2.5 days	10hours 3min
Approximate Spot Cost	480 USD	620 USD
Intermediate Data Size	102.2 GB	196.4 GB

For the Cascadia subduction zone dataset, we used ten Amazon EC2 p3.8xlarge instances each equipped with four V100 GPUs. These instances were in lower demand and we were able to consistently maintain compute power (hence the shorter job time). The discrepancy in the cost of these two experiments is due to an increase in the spot price of these instances, which is driven by demand [51]. When using spot instances, we essentially scrape the 'bottom of the barrel' in terms of compute resources. We get the leftovers that were not paid for by customers who pay full price for non-preemptable instances. In other words, we are using resources that would have otherwise gone unused. When there is high demand for compute power, the value of the 'bottom-of-the-barrel' goes up. When there is low demand, the cloud provider is losing money and is willing to give that compute power to the highest bidder.

4.1.4 CPU Comparison

Table 5 compares an OpenMP-based multi-core CPU SCAMP implementation to our GPU optimized implementation. The CPU implementation uses a 72-core c5 18xlarge spot instance (Intel Skylake architecture). Both implementations compared in Table 5 employ double-precision (FP64) calculations.

As shown in Table 5, the 72 core c5.18xlarge instance saturates at an input size of 2^{21}, after which its runtime scales quadratically, as expected. At the time of writing, the c5.18xlarge has the same on-demand price on AWS as a p3.2xlarge which employs one V100 GPU. While it is difficult to compare runtimes across architectures, we *can* compare price per performance, which is shown in bold as a factor of improvement of the GPU over the CPU. In this case, the GPU is approximately one order of magnitude more cost-efficient. The price per performance for smaller input sizes is not a particularly good basis for comparison: we could have used a smaller instance type to achieve better price per performance on a CPU when small input data sizes do not saturate the 72 available cores on the c5 18xlarge spot instance.

Table 5: Optimized CPU SCAMP on a single AWS instance

Instance Type / Input Size	c5.18xlarge (72 cores) 3.06 USD/hr *Seconds*	p3.2xlarge (1 Tesla V100) 3.06 USD/hr *Sec/**speedup***
2^{18}	7	0.28 (**25x**)
2^{19}	14	0.68 (**20x**)
2^{20}	32	2.0 (**16x**)
2^{21}	76	7.0 (**11x**)
2^{22}	252	25.8 (**9.8x**)
2^{23}	933	96.8 (**9.6x**)

4.2 Precision Evaluation

Consider the three data snippets shown in Figure 9. Each has a constant region longer than the motif length *m* we are interested in. This is a source of numerical instability that is very common in many domains. However, as we will show, it is easy to fix.

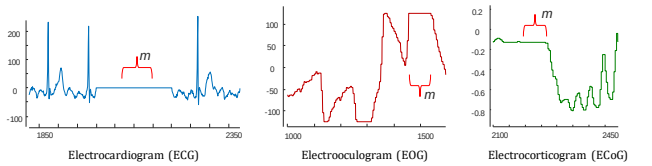

Figure 9: Three real time series from [9], each containing a constant region caused by different issue. *left*) An ECG (heart) with a disconnection artifact. *center*) An EOG (eye movement) with a hard-limit artifact. *right*) An ECoG (finger flexion) with constant region caused by low precision recording.

Such constant regions are surprisingly common, even in datasets that one might not expect to contain them. For example, just in the context of medical datasets, we see constant regions caused by:

- **Disconnection Artifacts**: We may see temporary disconnection of a monitoring lead, for example during a bed change.
- **Hard-Limit Artifacts**. Some devices have a minimum and/or maximum value defined by a physical limit of the device. As shown in Figure 9.*center*, if the true value exceeds that limit, a constant region will be reported.
- **Low Precision Artifact**: Many medical devices record at low-precision fixed-point; these seemingly short period constant values would *not* be constant at a higher precision.

The reason why constant regions matter is because we are interested in the similarity of *z-normalized* subsequences. Z-normalization requires dividing by the standard deviation, which is zero for a constant region. Moreover, there are often subsequences that are *almost* constant, and therefore would pass a bit-level test for "*at least two different values*", but would nevertheless result in a division by a number very close to zero.

We note that in the majority of cases these disconnection artifacts can be allowed to saturate to a Pearson Correlation of 1 or a z-normalized Euclidean Distance of 0 and removed later via a post processing step. In many domains these flat regions have little semantic meaning. However, sometimes these regions only *appear* to be flat in lower precision, but are actually are full of very small peaks, valleys, and interesting behavior when kept in high precision. If these small peaks and valleys are important, we should compute the Matrix Profile in double precision, as motifs are likely to be lost at lower precision in these cases.

Note that the majority of the instability here comes from the final distance calculation in Equation 3, where we must multiply by the inverse norm, which can get extremely large for these almost-flat regions. However, we should not introduce instability into the long running computation from this because Equation 4 does not utilize these inverse norms, making the potential for value explosion due to the multiplication of large numbers far less likely.

Figure 10 shows the relative error between the distance matrices computed with SCAMP DP vs SCAMP SP on a subset of the Parkfield data. The relative errors greater than 10 percent are shown in gray. At the top of the matrix, there is very little error, once the earthquake occurs, the error becomes catastrophic. As stated previously, this error occurs because FP32 cannot represent large numbers ($> 10^7$, in this case) with sufficient accuracy.

Note that there are two types of error occur the matrix. The first are scattered errors when the distance between the subsequences in the Parkfield earthquake and other arbitrary subsequences are computed. The second are poisoned diagonals, once a diagonal's error grows too large, it essentially becomes 'poisoned' as it can cause other distances between subsequence pairs along that diagonal to have large error as well.

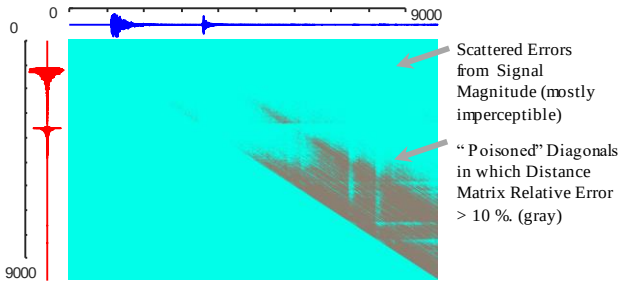

Figure 10: The Distance matrix for a small subset of data containing the Parkfield earthquake. We plot the relative error greater than 10% in the distance matrix for single precision vs double precision. This illustrates what can go wrong if the data representation is not expressive enough.

4.2.1 Comparison with Previous Update Method

In Figure 11, we compare SCAMP's update method with the previous method implemented in GPU-STOMP. We compute the result first in double precision, then plot the difference between the double and single precision results in Pearson Correlation of SCAMP and GPU-STOMP.

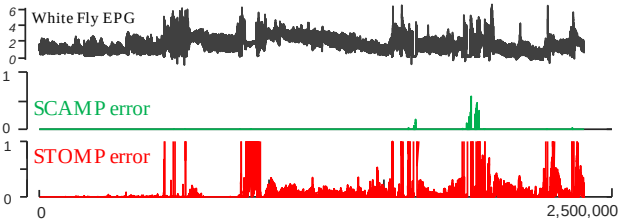

Figure 11: Single precision error comparison between GPU-STOMP$_{OPT}$ and SCAMP on White Fly EPG dataset. *top*) original data. *middle*) SCAMP absolute error. *bottom*) GPU-STOMP absolute error.

By comparing the bottom and middle of Figure 10, we can see how Equations 1 and 2, which compose the originally reported update method for GPU-STOMP, completely fail in single precision on this dataset. In the figure, we capped the error at 1 for GPU-STOMP, which is half of the range of Pearson Correlation. The actual values reported by GPU-STOMP in some cases were many times larger than the entire range of Pearson Correlation.

Notice that SCAMP only fails for the previously discussed disconnection artifacts. In such cases, the results for SCAMP can be cleaned up by a domain expert with very little effort, by simply not omitting small regions which where the signal was disconnected. On the other hand, GPU-STOMP cannot produce a meaningful result across almost the entire dataset.

4.2.2 General Considerations for Precision

Table 6 presents an analysis of the effects of reducing precision on various datasets of different lengths. In each case we use a tile size of 1 million for SCAMP and allow GPU-STOMP to extrapolate the entire length of the input (not a parameter in GPU-STOMP). As previously noted, the tile size affects the amount of extrapolation SCAMP must perform on the data to produce the final result. For our experiment, we generate the Matrix Profile using both SCAMP and GPU-STOMP in each of our precision modes. Each entry in the table is the max absolute error found between the double precision Matrix Profile calculation and the other precision modes. For Table 6, we intentionally choose a window length longer than the longest flat artifact region in the data. This allows us to compare errors caused by the update formula only and not the inherent loss of information from an artifact that cannot be represented in lower precision. We have highlighted absolute errors greater than 0.01 in the Matrix Profile result in red as these results would probably not be considered accurate enough for most users.

Table 6: Maximum absolute error (Pearson Correlation) for various datasets/algorithms. Red values denote high error

Maximum absolute error	**Size (m)**	**SCAMP SP**	**SCAMP MIXED**	**STOMP SP**
Whitefly EPG	2.5M (1000)	$3.75*10^{-2}$	$1.36*10^{-2}$	$1.89*10^{1}$
ECG	8.4M (100)	$3.14*10^{-4}$	$2.20*10^{-4}$	$2.07*10^{-3}$
Earthquake	1.7M (200)	$6.35*10^{-1}$	$6.35*10^{-1}$	$3.17*10^{3}$
Power Demand	10M (4000)	$4.85*10^{-2}$	$2.06*10^{-2}$	$2.22*10^{-1}$
Chicken	9M (1000)	$4.92*10^{-2}$	$7.71*10^{-3}$	$2.27*10^{1}$
99.9 percentile absolute error	**Size (m)**	**SCAMP SP**	**SCAMP MIXED**	**STOMP SP**
Whitefly EPG	2.5M (1000)	$3.00*10^{-3}$	$2.08*10^{-3}$	$1.55*10^{1}$
ECG	8.4M (100)	$4.40*10^{-5}$	$2.59*10^{-5}$	$4.02*10^{-4}$
Earthquake	1.7M (200)	$6.08*10^{-1}$	$6.08*10^{-1}$	$1.94*10^{3}$
Power Demand	10M (4000)	$8.52*10^{-3}$	$3.48*10^{-3}$	$1.29*10^{-1}$
Chicken	9M (1000)	$1.96*10^{-3}$	$7.25*10^{-4}$	$1.70*10^{1}$

This experiment shows that while SCAMP is 3 or more orders of magnitude more accurate than STOMP on these datasets, SCAMP suffers a substantial loss in accuracy when using 32-bit data representations; however, this loss comes with the benefit of improved performance. If a user's dataset and application are can tolerate the loss of accuracy, there is much to be gained in terms of efficiency. SCAMP SP has about 50% more error on average than SCAMP MIX, while STOMP SP struggles to find meaningful results. Empirically, we observe that SCAMP SP works well on data that is highly regular with a small min-max range. ECG would be an example of this kind of data. With some additional effort it may be possible to produce better results in for more types of data, but we leave this task for future work.

Additionally, note that both SCAMP MIX and SCAMP SP completely fail on the Earthquake dataset in Table 6. The reason for this is that the large earthquake's signal has a magnitude of greater 10^7, which is outside the range of the values single precision can represent to the required accuracy.

5. Case Studies in Seismology

As Figure 1 and Figure 4 suggest, motifs are of interest in many domains. However, we confine our case studies to seismology, as it is a domain with obvious and direct importance for humankind.

In geophysics, seismic data are a primary source of information about Earth's interior structure and processes. We define seismic data as any recorded motion (i.e. displacement, velocity, acceleration) measured using seismic instruments at the Earth's surface. We naturally think of this motion can be caused by earthquakes or volcanic activity, however it can also be created by thunderstorms, wind, ocean waves, nuclear tests, landslides, the movements of glaciers, or even, on rare occasions human activity (i.e. 100,000 soccer fans celebrating a goal). Seismic data are surprisingly versatile, but one of the most important applications is to detect and locate seismic events (earthquakes). Detected and located seismic events can be used for studying earthquake source processes and source physics, fault behavior and interactions, for determining Earth's velocity structure, and in general helping to constrain seismic hazard [12]. Along with the improvement of seismic data instruments, reductions in cost, improvements in networking, data management and repositories, have resulted in a

power law increase in seismic data volume [19]. Probing this huge volume of data is an ongoing challenge in seismology.

Performing query search for seismic data has been shown to increase the detectability of seismic events by one order of magnitude [29][36]. However, this method requires a priori known queries (often referred to as '*waveform templates*' in seismology) as input. Although waveforms of events in a local earthquake catalog can be used, this relies on suitable events being present in the catalog. To identify suitable queries, [6] developed an 'autocorrelation' motif discovery method, but this was limited to one hour of waveform data at a time as the method is computationally expensive in terms of both memory and time [6][7][34]. Being restricted to one hour of data limited this otherwise potentially powerful method, as the seismologists wish to consider much larger time scales. For example, aftershocks of large earthquakes can occur over a time period of days to months [29], swarms (volcanic and non volcanic) can take weeks to months [17], and repeating earthquakes can have recurrence intervals on the order of months to years [25].

Some other recent studies developed a fast motif discovery by "fingerprinting" – converting seismic time series to small and dense proxies, or "fingerprints" and then performing Locality-Sensitive Hashing (LSH) on them [4][32] LSH is a fast approximate nearest neighbor search method that reduces the similarity search dimensions. Although the LSH method sped up the similarity search process tremendously (i.e. ~143 times faster than traditional autocorrelation for one week of continuous data), it also produces false positive results (e.g. 12 events for one week of continuous data with 24 catalog events; [32]) and, more importantly, *false negatives* (e.g. 3 events out of 24 catalog events; [32]). In addition, LSH requires the careful selection of a number of tuning parameters that strongly influence the success of the search, and whose values may vary for different regions, data sets and applications. The tuning parameter selection process requires visual inspection, and validation against the results of other methods.

In contrast, SCAMP can exactly search datasets that are can only be searched approximately using current methods. To show this, we consider the milestone of 1,000,000,000 data points. One billion data points is equivalent to ~579 days (~1.5 years) of seismic data with a 20 Hz sample rate. Below, in two examples, we show how moving from motif discovery for hours of data to years of data is a potential game changer in seismic data mining.

5.1 Foreshocks and aftershocks of Parkfield

The town of Parkfield, located on the San Andreas fault in central California, experienced four magnitude ~6 earthquakes in the 20th Century, in 1901, 1922, 1934 and 1966 [25][29][36]. Based on the quasi-periodic nature of the events, a repeat event was predicted to occur between 1985 and 1993 [25], a prediction that spurred a major project, the 'Parkfield Earthquake Prediction Experiment', an attempt to capture the earthquake and all associated phenomena with the best available instrumentation. Although it occurred over a decade 'late' in 2004, the most recent Parkfield earthquake was recorded in extraordinary detail by the Parkfield High Resolution Seismic Network (HRSN), a dense array of borehole seismometers (preferred due to their low levels of noise). In addition, the thousands of aftershocks that followed the earthquake, as well as any possible foreshocks or other event precursors were also recorded at similarly high quality.

In order to investigate *i*) whether the HRSN data contain information on any aftershocks that were not included in the earthquake catalog, and *ii*) whether there was any change in behavior of the seismicity before the mainshock; we ran SCAMP on 580 days (1,002,240,008 points) of 20 Hz horizontal component seismic data (from 28 November 2003 to 9 July 2005) from the HRSN station EADB, centered on the 2004 Parkfield event time (i.e. 28 September 2004). We set the query length at 100 samples, equivalent to 5 seconds of data. We band-pass filtered the data between 2 and 8 Hz, a frequency range suitable for detecting both local and low frequency earthquakes (LFEs), a class of events that typically have low signal to noise ratios. Figure 12 shows a zoom-in of two sections of the waveform and their corresponding Matrix Profile.

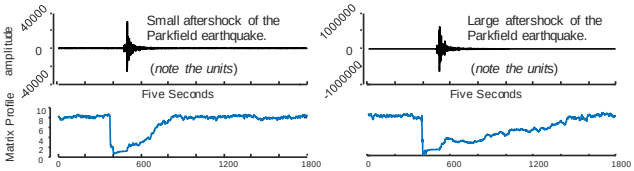

Figure 12: Examples of a waveform snippet (*top*) and corresponding MP shape (*bottom*) for aftershocks of the Parkfield earthquake. *left*) a small aftershock. *right*) a larger aftershock with a waveform amplitude that is three orders of magnitude larger.

Examining the motifs for aftershocks of the Parkfield earthquake, we notice that they have a very characteristic shape (a much higher resolution figure archived in Ref. [50] shows this more clearly). Note that the MP drops abruptly as the query window starts to capture the beginning of the earthquake waveforms and then gradually increases back to the background noise level. The duration of this gradual increase is longer for the larger event (Figure 12.*right*), consistent with the empirical relationships of signal duration with event magnitude [21][8].

This Matrix Profile shape indicates that the two waveforms being compared have similar shapes at their beginnings, and dissimilar shapes at their ends. The first arrivals (first motions) of seismic waves have polarities (either up or down) that reflect both the mechanism of the earthquakes that generated them and their location relative to the station. An abrupt initial drop in the MP, therefore indicates that the two waveforms have the same first motion polarity. The next few seconds of arrivals to the station include later arriving seismic phases that include reflections, refractions and reverberations of seismic waves – collectively referred to as the seismic 'coda' – these are much more sensitive to differences in earthquake location, and therefore much less similar between pairs of events [1]. Even so, while the MP for the coda is, as we would expect, higher than for the first motions, it remains lower than the background noise (Figure 12), and therefore remains indicative of an earthquake.

From this observation we can propose two important applications the MP results for seismology: *i*) The abrupt initial drop of MP can be used to select the primary phase arrival of seismic events, which is an ongoing challenge in seismology [26][33] (*ii*) The length of the MP valley from the sudden drop to its recovery can help to measure the coda length, and it is been shown that the length of coda correlates with the magnitude of earthquakes [8][21].

To show the power of the MP for identifying earthquakes we performed a simple event-detection experiment. Note that as discussed above, here we are using a Matrix Profile containing the Pearson correlation coefficient (which we denote as MPCC). The Pearson correlation is the common metric used in seismological studies [31][25][37], rather than the Euclidean distance, to which it can be trivially converted (and vice versa). The MPCC has the advantage of being bounded in the range [-1,1].

We count the number of MPCC peaks when they are separated by at least 100 samples (5 seconds); this prevents overcounting the same earthquake when multiple peaks are present for one event. Additionally, long traces of seismograph data often contain repeated patterns corresponding to a special types of sensor noise. Fortunately, these are easy to filter, as they create near perfect motifs. We thus count the number of MPCC peaks above 0.9 but below 0.99.

Figure 13 shows the number of MPCC motifs per day for 580 days of EADB data. Although we targeted the Parkfield earthquake aftershocks, by looking at the number of MPCC motifs per day, we detected other nearby earthquakes and their aftershocks as well, notably the 2003 M_w 6.5 San Simeon event and two other moderate (M_w4.0–4.5) earthquakes nearby.

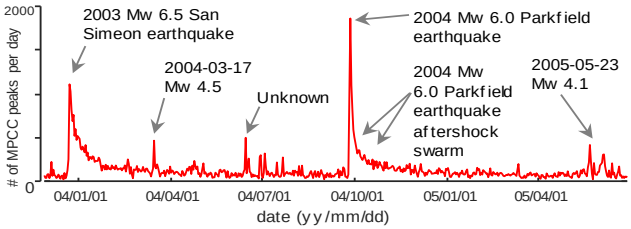

Figure 13: Daily numbers of discovered motifs for 580 days of data centered on the Parkfield earthquake (04/09/28), measured on the horizontal component of station EADB, located ~10 km from the epicenter. Motifs are selected based on the peak values of the MP correlation coefficient (MPCC).

A series of motif peaks in the lead-up to the Parkfield mainshock (around 04/07/01) may represent previously undetected foreshock activity (i.e. there are no corresponding events in the regional earthquake catalog), and merit further investigation.

In Figure 14 we compare the total number of motifs with $0.9 \leq$ MPCC < 0.99 over the first 90 days of the Parkfield aftershock sequence with the number of catalog aftershocks retrieved from the Northern California Earthquake Data Center (NCEDC), we find ~16 times more detections for the former.

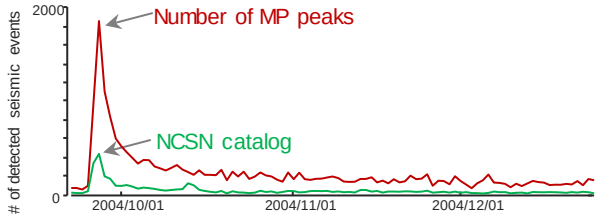

Figure 14: Comparison between the number of events in the USGS NCSN Catalog (green line) and the number of motifs detected using SCAMP (red line) for the Parkfield earthquake aftershock sequence. For the catalog events we considered all events in a box with length ~200 km centered on the Parkfield mainshock epicenter. The start of seismicity in this plot is 4 days before the Parkfield earthquake

Note that some small faction of these MP thresholding-based detections might be station artifacts; however, our visual inspection suggests that these account for less than 5% of the events.

We also fit the Omori-Utsu aftershock rate equation [16] to the detected and catalogued aftershocks of the Parkfield earthquake. Figure 15 shows that the number of motifs per day fit the Omori-Utsu law almost perfectly.

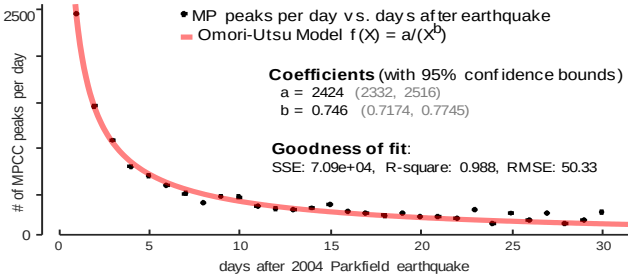

Figure 15: A fit of an Omori-Utsu relationship (i.e. the law that describes aftershock rate behavior) to the number of motifs per day for the first 30 days after the Parkfield mainshock. The R-squared of 0.988 indicates a very good fit and shows how the number of motifs can describe the expected aftershock behavior almost perfectly.

The values retrieved from the Omori-Utsu law can provide information about the physics of the mainshock [16] and also even can be used for forecasting large aftershocks [28].

Here we only present a small portion of the results that *could* be extracted from the MP for 580 days of data. The threshold of 0.9 that we use is relatively high and can be set to lower values for detecting (LFEs). In the next section we discuss LFEs and their importance in greater detail. There also some motifs in the months before the Parkfield earthquake that could represent foreshocks and/or precursory seismicity, and should be investigated in greater detail (Figure 13). Note that these applications are viable when the continuous data time series is long enough that there is a high probability a near neighbor exists for each seismic event. Our experience suggests that 580 days of seismic data for an area like Parkfield that has a high seismicity rate (i.e. tens of thousands of events in that period, Figure 13), would essentially guarantee this.

5.2 Detecting subtle seismic motifs

The Cascadia subduction zone, where the Juan de Fuca plate subducts beneath the North American plate, is a tectonic province extending from coastal Northern California north to Vancouver Island. Since the discovery of the episodic slow slip on the subduction interface in this region and the discovery of a non-volcanic tremor (NVT) that accompanies these slow slips, there is an ongoing effort to better understand how these phenomena affect earthquakes on the subduction zone, particularly because it has the potential to produce great earthquakes i.e. magnitude ~9 [2][20].

It has been shown that the NVT likely consists of a swarm of low frequency content seismic events that are called 'low frequency earthquakes'. For a better understanding the relationship of slow slip and tremor, the precise determination of the source of an NVT is an important task. As NVT waveforms typically have non-impulsive arrivals, meaning that it is difficult to pick the first seismic wave arrival times, the location process can have large uncertainties meaning that detecting and locating individual LFEs is often a better alternative.

To test the SCAMP's ability to detect LFEs, and more generally to explore the seismicity of the southern Cascadia subduction zone, we ran SCAMP on 579 days of data (start date 2006/03/01) for the vertical component of station I02A, located near Mapleton, OR. We band pass filter these data at 2–8 Hz and resample them to 20 Hz. We set the query length this time to 200 (10 seconds), based on the length of LFE templates used in previous studies. Figure 16 shows the motif density over time for this experiment.

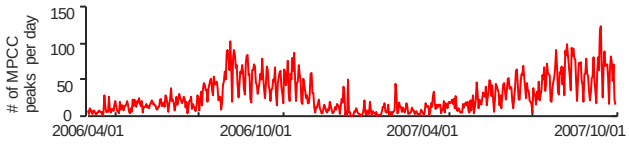

Figure 16: Discovered motifs for 579 days of seismic data recorded on the vertical channel of station I02A, located near Mapleton, OR. The number of discovered motifs based on MPCC thresholding method shows two six-month periods were detected motifs gradually increase, that start in mid 2006 and mid 2007. We estimate that >90% of these discovered motifs are low frequency earthquakes (see Figure 17).

By examining at the time series of daily motif detections, we observe that in 2006 the number of motifs starts to increase around August and decreases in November. The number of motifs again starts to increase in June 2007 and starts to decrease around October (Figure 16). In order to classify some of these motifs as LFEs, we visually inspect some of the days with the highest number of detected motifs. We observe that most of these detections are LFEs based on their shapes, durations and frequency contents. We also observe that the MP shape around the motifs for regular earthquakes are different from those of the LFEs (Figure 17). The valley shape for the motifs corresponding to regular earthquakes has the same characteristics that we observe for earthquakes at Parkfield (i.e. a sudden drop and gradual increase back to the MP values for background noise, Figure 17.*left*). However, the shapes of the MP around the LFEs are almost flat and stay low until the end of the LFE signal (Figure 17.*right*).

By visual inspection, we conclude that a large portion of the detected motifs are LFEs and our result indicates two Episodic Tremor and Slip (ETS) events – one that starts in mid-2006 and one that start in mid-2007 (Figure 16). Our results broadly agree with those reported in [5] for the southern Cascadia region. They reported one main ETS event in August-September 2006 (that they name 'E11') and another in July-August 2007 ('E26'). However our detection shows a more gradual increase in detected LFEs, rather than a sharp onset, as reported in that earlier study.

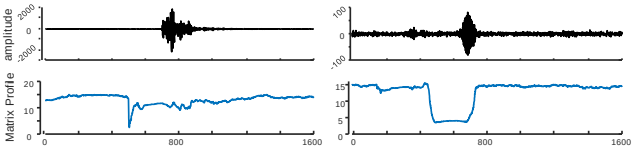

Figure 17: *left*) An example of an earthquake waveform snippet (top) and MP shape (bottom) in the vicinity of a discovered motif for a 'regular' earthquake (i.e. an earthquake that contains high frequency content). Note that the MP drops abruptly when the tail of the query (length 200 samples) includes the beginning of the event and goes back to the background level after the query head passes the whole seismic signal. *right*) Another earthquake; based on the shape, duration and frequency content of the waveform, we suspect this is an LFE. In this case, the MP shape around the motif stays flat until the end of seismic event.

More investigation is required to confirm that the ~90% of 'LFE-like' motifs discovered in the Cascadia data set are indeed LFEs. This investigation should include inspections of data from multiple stations and determination of event locations.

One open question is whether the sources of these LFEs are permanent 'asperities' (source regions on the fault) which repeatedly slip during each successive slow slip event, or temporary sources that repeat only during that specific slow slip event [53]. One novel feature of SCAMP is that it produces intermediate partial AB-join MPs (section 3.2.2) that can be used here to answer this question. Here using these partial AB-join MP we search for motifs between two subsets of our seismic data from Cascadia: one (A) containing data from the peak motif discovery period in 2006 (100 days from 11/07/2006 to 19/10/2006) and the other (B) from the corresponding period in 2007 (84 days from 08/07/2007 to 30/09/2007). By comparing the AB-join MP and the MP from all 579 days for the A period, we observe that the motif density only decreases by 20%. This implies that 80% of the LFEs in the 2006 slow slip event recurred in the 2007 slow slip event.

On the basis of these results, we suggest that the majority of LFE sources are likely to have been stable at least during the 2006 and 2007 slow slip events.

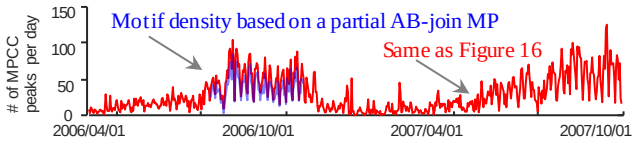

Figure 18: Comparison of self-join and AB-join for seismic data from Cascadia. Red line shows the self-join results from Fig 16 zoomed in for the time window of 11/07/2006 to 19/10/2006. Blue line is the motif density based on a partial AB-join MP. The B time period contains data from 08/07/2007 to 30/09/2007. The motif discovery is based on thresholding of MPCC values between 0.9 and 0.99. The shapes of the motif density time series for both cases are similar but the AB-join case contains 20% fewer motifs. This implies that 80% of the motifs discovered in the A period (the 2006 slow slip event) have similar events in the B period (the 2007 slow slip event).

All results presented here were obtained by simple post-processing of the MP produced by SCAMP, and possibilities for further refinement in analysis and interpretation remain open. The versatility and precision that SCAMP provides suggests that SCAMP has a rich future in seismic data mining – a discipline that traditionally has suffered from false negatives.

6. Discussion and Conclusion

We introduced novel algorithms and optimizations that can exploit modern GPU hardware to allow significant progress in the size of datasets that can be *exactly* searched for motifs. To the best of our knowledge, this work is the first time any research effort has reported performing a quintillion *exact* pairwise comparisons on a single dataset[1]. Likewise, this is the first work to do *exact* motif search on over a year (1.59 years) of continuous earthquake data.

We believe that this work addresses a pent-up need that is near universal in data analytics. For example, [32] note that in the context of motif discovery in seismology, "*scalability bottlenecks prevented seismologists from making use of the decades of data at their disposal*." [32]. Likewise, in neuroscience, [22] argue for the importance of "(motifs) *of neural activity in understanding how information is encoded*". That work resorted to approximations and downsampling to make motif discovery tenable [22], however this work *exactly* searches datasets that are orders of magnitude larger that they were able to consider.

We have made all code freely available to the community to confirm, extend and most importantly, *exploit* our work.

[1] For the pedant. Since the pairwise distances are symmetric and the distance to self is always zero, to find motifs in a billion length time series, we only needed to compute four hundred ninety-nine quadrillion, nine hundred ninety-nine trillion, nine hundred ninety-nine billion, five hundred million comparisons.

7. REFERENCES

[1] K. Aki and B. Chouet. Origin of coda waves: source, attenuation, and scattering effects. *Journal of Geophysical Research*, 80(23): 3322-3342, 1975.

[2] B. Atwater, A. Nelson, J. Clague, G. Carver, D. Yamaguchi, P. Bobrowsky, and H. Kelsey. Summary of coastal geologic evidence for past great earthquakes at the Cascadia subduction zone. *Earthquake spectra*, 11(1): 1-18, 1995.

[3] N. Begum, B. Hu, T. Rakthanmanon, and E. J. Keogh. Towards a minimum description length based stopping criterion for semi-supervised time series classification. *IRI*, 333-340, 2013.

[4] K. J. Bergen and G. C. Beroza. Detecting earthquakes over a seismic network using single-station similarity measures. *Geophysical Journal International*, 213(3): 1984-1998, 2018.

[5] D. Boyarko, Det al. (2015). Automated detection and location of tectonic tremor along the entire Cascadia margin from 2005 to 2011. Earth and Planetary Science Letters, 430, 160-170.

[6] J. Brown, G Beroza, & D. Shelly (2008). An autocorrelation method to detect low frequency earthquakes within tremor. Geophysical Research Letters, 35(16).

[7] R. Butler, T. Lay, K. Creager, P. Earl, K. Fischer, J. Gaherty, and J. Tromp. The Global Seismographic Network surpasses its design goal. *Eos, Transactions American Geophysical Union*, 85(23): 225-229, 2004.

[8] B. Castello, M. Olivieri, and G. Selvaggi. Local and duration magnitude determination for the Italian earthquake catalog, 1981–2002. *Bulletin of the Seismological Society of America*, 97(1B): 128-139, 2007.

[9] H. A. Dau and E. J. Keogh. Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. *KDD*, 125-134, 2017.

[10] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1): 107-113, 2008.

[11] C. W. Ebeling and S. Stein. Seismological identification and characterization of a large hurricane. *Bulletin of the seismological society of America*, 101(1): 399-403, 2011.

[12] E. H. Field, et al. Uniform California earthquake rupture forecast, version 3 (UCERF3)—The time-independent model. *Bulletin of the Seismological Society of America*, 104(3): 1122-1180, 2014.

[13] I. Fox, L. Ang, M. Jaiswal, R. Pop-Busui, and J. Wiens. Contextual motifs: Increasing the utility of motifs using contextual data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 155– 164, 2017.

[14] L. Gualtieri, S. J. Camargo, S. Pascale, F. M. Pons, and G. Ekström. The persistent signature of tropical cyclones in ambient seismic noise. *Earth and Planetary Science Letters*, 484: 287-294, 2018.

[15] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning*, pages. 1737-1746. JMLR, 2015.

[16] S. Hainzl and D. Marsan. (2008). Dependence of the Omori‐Utsu law parameters on main shock magnitude: Observations and modeling. *Journal of Geophysical Research: Solid Earth*, 113(B10), 2008.

[17] D. Hill, W. Ellsworth, M. Johnston, J. Langbein, D. Oppenheimer, A. Pitt, and S. McNutt. The 1989 earthquake swarm beneath Mammoth Mountain, California: An initial look at the 4 May through 30 September activity. *Bulletin of the Seismological Society of America*, 80(2): 325-339, 1990.

[18] N. M. Ho and W. F. Wong. Exploiting half precision arithmetic in Nvidia GPUs. *HPEC*, 1-7, 2017.

[19] A. Hutko, M. Bahavar, C. Trabant, R. Weekly, M. Fossen, and T. Ahern. Data products at the IRIS‐DMC: Growth and usage. *Seismological Research Letters*, 88(3): 892-903, 2017.

[20] R. Hyndman and K. Wang. The rupture zone of Cascadia great earthquakes from current deformation and the thermal regime. *Journal of Geophysical Research: Solid Earth*, 100(B11): 22133-22154, 1995.

[21] F. Klein. (2002). User's guide to HYPOINVERSE-2000, a Fortran program to solve for earthquake locations and magnitudes. *US Geological Survey*, 02-171(1.0), 2002.

[22] I. Kolb, G. T. Franzesi, M. Wang, S. B. Kodandaramaiah, C. R. Forest, E. S. Boyden, and A. C. Singer. Evidence for long-timescale patterns of synaptic inputs in CA1 of awake behaving mice. *Journal of* Neuroscience, 1519-17, 2017.

[23] K. Mauck. (2018) Personal communication

[24] A. Murillo (2018). Personal Communication.

[25] R. Nadeau, W. Foxall, and T. McEvilly. Clustering and periodic recurrence of microearthquakes on the San Andreas fault at Parkfield, California. *Science*, 267: 503-7, 1995.

[26] S. E. J. Nippress, A. Rietbrock, and A. E. Heath. Optimized automatic pickers: application to the ANCORP data set. *Geophysical Journal International*, 181(2): 911-925, 2010.

[27] T. Ogita, S. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6): 1955-1988, 2005.

[28] T. Omi, Y. Ogata, Y. Hirata, and K. Aihara. Forecasting large aftershocks within one day after the main shock. *Scientific reports*, 3: 2218, 2013.

[29] Z. Peng and P. Zhao. Migration of early aftershocks following the 2004 Parkfield earthquake. *Nature Geoscience*, 2(12): 877, 2009.

[30] T. Perol, M. Gharbi, and M. Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2): e1700578, 2018.

[31] G. Poupinet, W. L. Ellsworth, and J. Frechet. Monitoring velocity variations in the crust using earthquake doublets: An application to the Calaveras Fault, California. *Journal of Geophysical Research: Solid Earth*, 89(B7): 5719-31, 1984.

[32] K. Rong, C. Yoon, K. Bergen, H. Elezabi, P. Bailis, P. Levis, and G. Beroza. Locality sensitive hashing for earthquake detection: a case study of scaling data-driven science. In *Proceedings of the Very Large Database Endowment.*To Appear.

[33] Z. Ross and Y. Ben-Zion. Automatic picking of direct P, S seismic phases and fault zone head waves. *Geophysical Journal International*, 199(1): 368-381, 2014.

[34] A. Royer and M. Bostock. A comparative study of low frequency earthquake templates in northern Cascadia. *Earth and Planetary Science Letters*, 402: 247-256, 2014.

[35] W. Sandanayaka, Y. Jia, and J. G. Charles. EPG technique as a tool to reveal host plant acceptance by xylem sap-feeding insects. *Journal of Applied Entomology*, 137: 519–529, 2013.

[36] D. P. Schaff and F. Waldhauser. One magnitude unit reduction in detection threshold by cross correlation applied to Parkfield (California) and China seismicity. *Bulletin of the Seismological Society of America*, 100(6): 3224-3238, 2010.

[37] D. P. Schaff and F. Waldhauser. Waveform cross-correlation-based differential travel-time measurements at the Northern California Seismic Network. *Bulletin of the Seismological Society of America*, 95(6): 2446-2461, 2005.

[38] D. Silva, C-C M. Yeh, G. Batista, E. Keogh: SiMPle: Assessing Music Similarity Using Subsequences Joins. ISMIR 2016: 23-29.

[39] D. W. van Liere. The significance of fowls' bathing in dust. *Animal Welfare*, 1:187–202, 1992.

[40] R. D. Vatavu. Small gestures go a long way: how many bits per gesture do recognizers actually need? In *DIS '12*, pages 328-337. ACM, 2012.

[41] K. Wang and A. M. Tréhu. Invited review paper: Some outstanding issues in the study of great megathrust earthquakes—The Cascadia example. *Journal of Geodynamics*, 98: 1-18, 2016.

[42] What-if. https://what-if.xkcd.com/63/.

[43] 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 *ICDM*, pages 1317-1322. IEEE, 2016.

[44] 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 *ICDM*, pages 739-748. IEEE, 2016.

[45] Y. Zhu, Z. Zimmerman, N. S. Senobari, C. C. M. Yeh, G. Funning, A. Mueen, and E. Keogh. Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. *Knowledge and Information Systems*, 1-34, 2018.

[46] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen and Gustavo Batista (2015). The UCR Time Series Classification Archive. URL www.cs.ucr.edu/~eamonn/time_series_data/

[47] https://aws.amazon.com/ec2/spot/

[48] Han, S., Mao, H. and Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. ICLR, 2016

[49] https://devblogs.nvidia.com/gpu-pro-tip-fast-histograms-using-shared-atomics-maxwell/

[50] TODO: Project website

[51] https://aws.amazon.com/ec2/spot/pricing/

[52] Nvidia Tesla V100 Whitepaper: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[53] Sweet, J. R., Creager, K. C., & Houston, H. (2014). A family of repeating low-frequency earthquakes at the downdip edge of tremor and slip. *Geochemistry, Geophysics, Geosystems*, *15*(9), 3713-3721.