Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond

Zachary Zimmerman, Kaveh Kamgar, Nader Shakibay Senobari,
Brian Crites, Gareth Funning, Philip Brisk and Eamonn Keogh
University of California, Riverside
{zzimm001, kkamg001, 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. State-of-the-art algorithms for computing the Matrix Profile 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.

CCS CONCEPTS

• **Computer systems organization** → *Cloud computing*;
• **Theory of computation** → *Data structures and algorithms for data management;*

KEYWORDS

Time Series, Matrix Profile, SCAMP, Self-Join, AB-Join, Cloud Computing, Spot Instance, GPU, Tiling, Fault-Tolerance, Numerical Optimization, Seismology, Entomology.

SoCC '19, November 20–23, 2019, Santa Cruz, CA, USA

ACM 978-1-4503-6973-2/19/11…$15.00
https://doi.org/10.1145/3357223.3362721

ACM Reference format:
Zachary Zimmerman, Kaveh Kamgar, Nader Shakibay Senobari, Brian Crites, Gareth Funning, Philip Brisk, and Eamonn Keogh. 2019. Matrix Profile XIV: Scaling Time Series Motif Discovery with GPUs to Break a Quintillion Pairwise Comparisons a Day and Beyond. In *Proceedings of the ACM Symposium on Cloud Computing (SoCC'19), Santa Cruz, CA, USA, November 20-23. 2019*, 13 pages.
https://doi.org/10.1145/3357223.3362721

1 Introduction

Time series motifs are approximately repeated subsequences of a longer time series. As Figure 1 (and Figure 4) suggest, motifs 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 [41], 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].

A survey of the literature suggests that many medical, scientific and industrial laboratory analysts rarely 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]. "Minutes" may not seem impressively fast, until one recalls that many datasets in question take days or weeks to collect. For 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 to scale. Domains such as seismology and astronomy have a near-inexhaustible appetite for ever-larger datasets. For example, a recent paper reports that 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 larger) would elucidate further unexpected regularities.

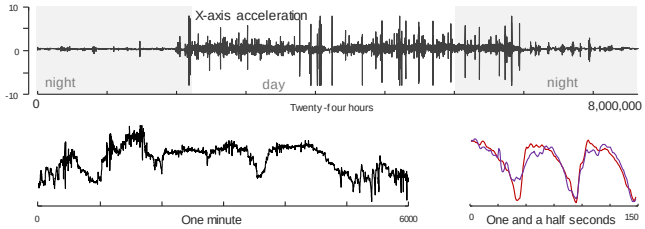

Figure 1: *top*) Twenty-four hours of time series from an accelerometer worn by a chicken. *bottom-left*) A zoom-in shows that the data is apparently void of structure; however, the top-1 motif (*bottom.right*) suggests that some behaviors *are* conserved. Inspection of video recorded in parallel suggests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

To meet the needs of domain experts, this paper presents a cloud-scale framework called SCAMP (SCAlable Matrix Profile) that 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 MP [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 MPs to be computed which are larger than previously considered.
3. We introduce novel numerical methods to increase performance and improve stability of the MP computation; this allows the use of single-precision floating-point calculations for some datasets, which allows our methods to be applied to larger datasets at a cheaper amortized cost.
4. We deployed a *fault-tolerant* framework that is compatible with "spot" instances [39], which major cloud providers (Amazon, Google, and Microsoft) offer at a major 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 [27].

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 MP calculations. In Section 4, we illustrate a few of the use cases for very large MPs through several case studies on challenging datasets. In Section 5, we provide a detailed empirical analysis of our ideas, before offering conclusions in Section 6.

2 Definitions and Assumptions

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

Key Assumption: Motif discovery under any reasonable definition is trivial if given the MP 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. Irrespective of the chosen definition, the MP alone is all that is needed to extract the motif in linear time and space [40][43]. Given this observation, this paper focuses exclusively on computing the MP 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 MP in-hand 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 MP here.

We 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 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 of 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, $D_i = [d_{i,1}, d_{i,2}, \ldots, d_{i,n-m+1}]$, such that $d_{i,j}$ $(1 \leq j \leq n\text{-}m+1)$ 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*. 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, in this paper we set $k=4$ and ignore the exclusion zone by setting $d_{i,j} = \infty$ $(i\text{-}m/4 \leq j \leq i+m/4)$. 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 (MP)* 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, $P = [\min(D_1), \min(D_2), \ldots, \min(D_{n-m+1})]$, 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 MP 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 *MP index*:

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

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

As presented, the MP is a *self-join*: for every subsequence in a time series T, we find its (non-trivial-match) nearest neighbor within the *same* time series. However, we can trivially generalize the MP to be an AB-join: for every subsequence in a time series A, record information about its nearest neighbor in time series B [44][41]. Note that A and B can be of different lengths, and generally, AB-join ≠ BA-join.

2.1 Observations on Precision

Several independent research groups have noted that for some time series retrieval tasks, 64-bit precision is unnecessarily precise [3][40]. Researchers have shown that reduced precision can be exploited to have significant performance benefits with minimal observable difference in quality of results [40][15]. This observation has been heavily exploited in deep learning [14][15]; however, it is rarely exploited for time series, except for allowing the use of Minimal Description Length to score and rank models [3], which is orthogonal to scalability considerations. Figure 4 shows an MP computed on some insect electrical penetration graph (EPG) data using 64-bit precision.

This plot suggests that the difference between MPs computed at 64 and 32-bit precision is so small it does not affect the motifs discovered, and is not visible unless we multiply the difference by a large constant; however, we must consider two caveats:

- The time series shown in Figure 4 is relatively short. To address ever longer time series, there is more potential for accumulated floating-point error to impact the result [18]. Even in this example we can see that the difference vector gets larger as we scan from left to right (Figure 4.*third-row*). We address this issue with our tiling scheme in Section 3.2.
- The information contained in the time series in Figure 4 is contained within a small range. This is true for some types of data, such as ECGs, accelerometer and gyroscope readings; however, there are also a handful of domains for which this is not true, such as seismology. A "great" earthquake has a magnitude of 8 or greater, but humans can feel earthquakes with magnitudes as low as 2.5, a difference of more than five orders of magnitude. Processing raw data with a large dynamic range is non-trivial (see Sections 3.2, 3.3, and 4.2).

Before proceeding, we note that this illustration offers another example of the utility of motif discovery. The time series in Figure 4 is a fraction of an entomologist's data archive [23]. The 2nd motif represents *ingestion of xylem sap behavior* [35], which is common and immediately recognizable by an entomologist; however, the 1st motif was unexpected: there is a "missed beat" during the xylem sap ingestion cycle.

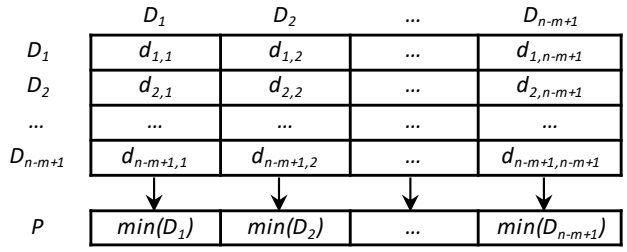

Figure 2: The relationship between the distance matrix, distance profile and MP. A distance profile is a column (also a row) of the distance matrix. The MP stores the minimum (off-diagonal) value of each distance matrix column; the MP Index stores the location of the minimum value within each column.

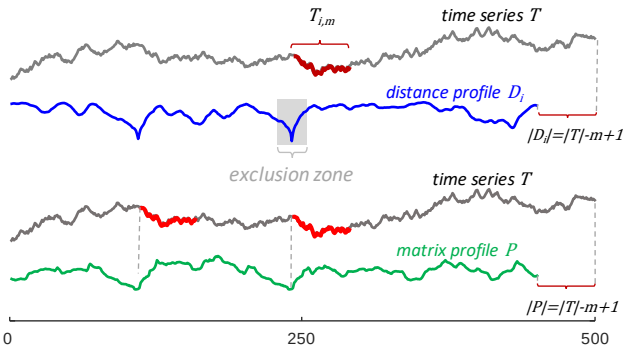

Figure 3: *top*) A distance profile D_i created from $T_{i,m}$ shows the distance between $T_{i,m}$ and all the subsequences in T. Values in the exclusion zone are ignored to avoid trivial matches. *bottom*) The MP P is the element-wise minimum of all the distance profiles. Note that the two lowest values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

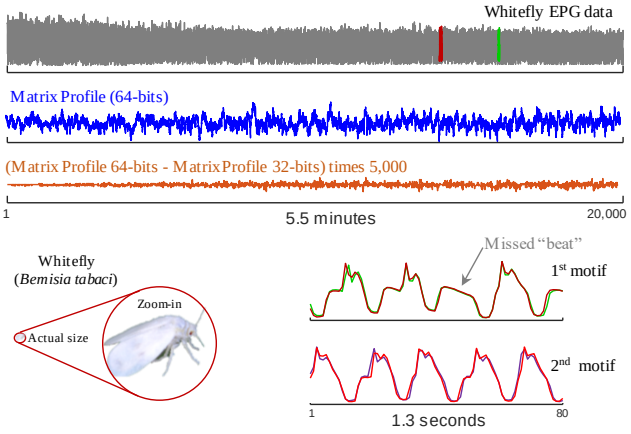

Figure 4: *top-row*) A snippet of whitefly insect EPG data. *second-row*) The MP computed with 64-bit precision. *third-row*) Because the 64-bit and 32-bit MPs are visually identical at this scale, we subtracted them, and multiplied the difference by 5,000. *bottom-row*) The whitefly is tiny, yet it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

If we had observed a *single* example, we could attribute it to chance or noise; however, motif discovery shows us that there are

at least two strongly conserved examples. This suggests that there exists some semantic meaning to this motif, which entomologists are currently exploring [23]. *To support scientific efforts to identify unexpected regularities in huge time-series archives, we introduce a GPU cloud system to compute large MPs.*

3 The SCAMP Framework

To compute large MPs, we introduce a framework that can be used by a ***cluster*** with a ***host*** and one or more ***workers***. A **host** can be a local machine, or a master server. A **worker** can be a CPU-based system or an accelerator (e.g., a GPU), following the host's direction. A **cluster** refers to the combination of a host and all of its associated workers. This can be the typical group of co-located nodes in a cloud, or 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$_{OPT}$ [41] is the current state of the art for computing MPs 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*.

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 a parallelogram-shaped tile along a 'meta'-diagonal, and maintains a local copy of the MP (i.e., the column-wise and row-wise minimum of the tile) in the shared memory. When a tile computation completes, each thread compares the thread-block-local copy of the MP with the MP stored in global memory; if a smaller value is found, the thread updates the global MP via an atomic access. SCAMP improves several aspects of GPU-STOMP$_{OPT}$, yielding a several-fold improvement in performance and allows efficient exploitation of newer GPU hardware. We explain these improvements in detail in the following sections.

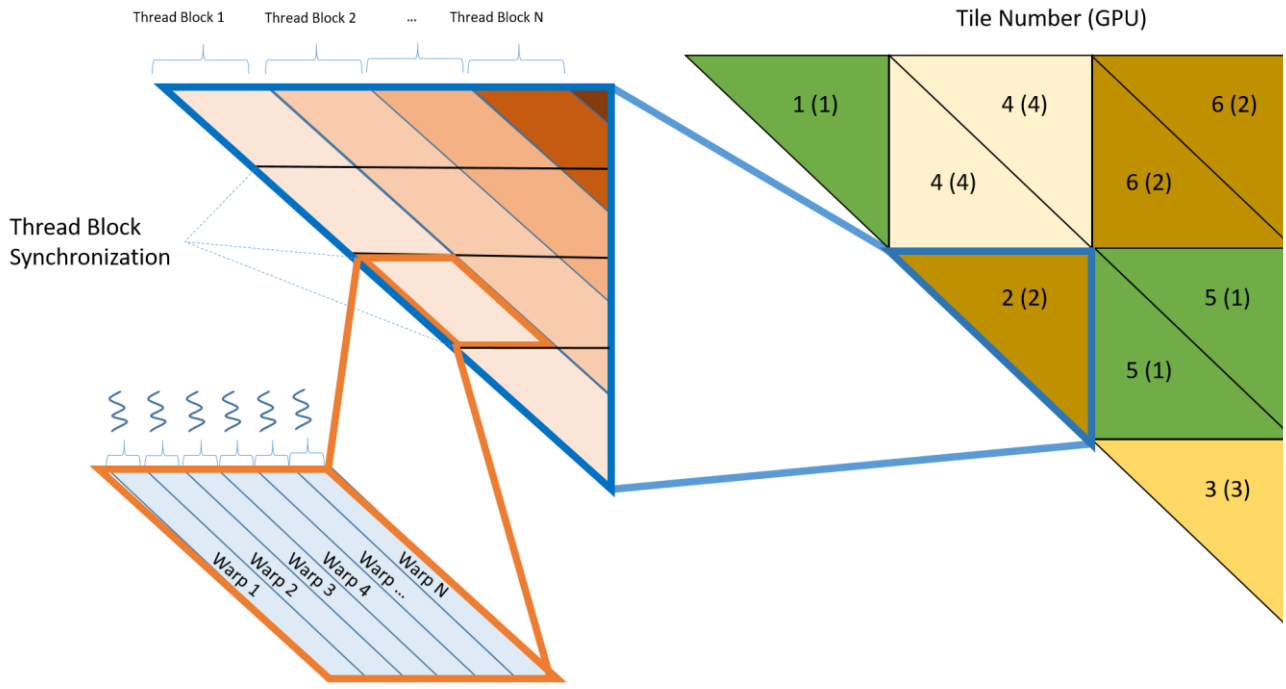

Figure 5: *left*) The GPU-STOMP$_{OPT}$ execution pattern, which is shared with the SCAMP_tile algorithm. *right*) The SCAMP tiling scheme using 4 GPUs. The illustration of the tiling scheme is for self-joins only; the lower triangular tile is computed with the same implementation, but with the inputs transposed.

In GPU-STOMP$_{OPT}$, each thread computes a diagonal of the distance matrix shown in Figure 2 by updating the dot product, *QT*,

3.2 Tiling Scheme

Rather than computing the entire distance matrix in one operation, we split it into tiles. Each tile independently computes an AB-join between two segments of the input time series. This allows the computation to scale to very large input sizes and distribute the work to many independent machines, as depicted in Figure 5.*right*. The host maintains information about its workers, such as the number and type of available GPUs, the memory capacity, and the CPU speed, to determine a tile width that can saturate its workers. The host generates tiles of this width and delegates them among the workers. For simplicity, this paper assumes that all of the workers are homogeneous (V100 GPUs) and that the most effective tile size (~1 million) fully saturates each worker during execution. This tile width is currently discovered empirically, but could be hard-coded once it is known a given system configuration.

3.2.1 Host Algorithm. The host executes the **SCAMP_host** algorithm, which employs multiple asynchronous workers, which could be threads or other nodes in a cluster (see Table 1). Line 1 determines the appropriate tile size for the problem instance and the relative tile execution order. Line 2 precomputes all necessary statistics of T needed to compute distances between subsequences. Lines 3-5 initialize a data structure containing all information necessary to compute the result for each tile in our problem instance, and insert the tile into a global work queue. Line 6, initializes asynchronous workers, who extract work from the queue. Line 7 retrieves and merges and the tile result and Line 8 outputs the result.

3.2.2 Tile Computation. All workers execute the **SCAMP_tile** algorithm to compute each tile's intermediate result (see Table 2), while unprocessed tiles remain in the global work queue. Line 2-6 extracts a tile from the work queue, along with its relevant information from the tile structure. Line 7 computes initial dot product values associated with the upper triangular tile. Line 8 executes an architecture-optimized kernel to compute the local MP and Index for that tile. Lines 9 and 10 compute the initial dot product values associated with the lower triangular tile and the result associated with that tile. The tile's computation similar to GPU-STOMP$_{OPT}$ [14][41], with additional optimizations, described in the rest of this section.

3.2.3 Optimizations. The host may run out of memory if tiles are sufficiently small and too many are pre-allocated; however, this can be overcome via optimization. For example, in a single node deployment, each worker, rather than the host, can construct the full tile upon its execution. In a distributed deployment, the maximum number of tiles in the queue can be limited, and more work can be added as each tile's processing completes. Further, it is possible to cache the *best-so-far* MPvalues as tiles computed by workers, enabling subsequent tiles to be initialized with more up-do-date MP values. These optimizations reduce the number of memory accesses during computation, but have been omitted from Table 1 and **Error! Reference source not found.** for simplicity of presentation.

Prior work established that the self-join problem exhibits symmetry in the distance matrix [41][44]; here, we note that the memory access pattern and the order of distance computations in SCAMP and GPU-STOMP$_{OPT}$ are similarly symmetric. The lower-triangular portion of the distance matrix (Figure 2) can be computed using the same subroutine as the upper-triangular portion simply by transposing the input. The SCAMP framework exploits this property to implement joins.

Table 1: The SCAMP_host Algorithm.

	Procedure ***SCAMP_host()***
	Input: *U*ser provided time series **T**, window length **w**, tile size **s**
	Output: Matrix Profile **P** and Matrix Profile Index **I**, of T
1	tiling ← **GetTiling**()
2	stats ← **PrecomputeTileStats**(T, w)
3	**for** row, col **in** tiling**:**
4	tile ← **CreateTile**(T, w, stats, row, col, s)
5	globalWorkQueue.**add**(tile)
6	**StartAsyncronousWorkers**()
7	P, I ← **WaitForWorkerResults**()
8	**return** P, I

Table 2: SCAMP Tile Computation

	Procedure ***SCAMP_tile()***
	Input: Thread safe work queue of tiles **workQueue**
1	**while** workQueue is not empty:
2	tile = workQueue.GetItem()
3	**if** tile **is** null:
4	**return**
5	A = tile.A, B = tile.B,
6	mp = tile.mp, mpi = tile.mpi, stats = tile.stats
7	QT ← **SlidingDotProducts**(A,B)
8	mp, mpi ← **DoTriangularTile**(A, B, stats, QT, mp, mpi)
9	QT ← **SlidingDotProducts** (B,A)
10	mp, mpi ← **DoTriangularTile** (B,A, stats, QT, mp, mpi)
11	**ReturnTileToHost**(mp, mpi)
12	**return**

3.2.4 Comparison to GPU-STOMP$_{OPT}$. Beyond the scope of the preceding discussion, SCAMP offers several distinct advantages over GPU-STOMP$_{OPT}$:

Extensibility: Since tiles are computed independently, SCAMP can provide different options for each tile's computation, which offers a pathway to run SCAMP on a heterogeneous compute infrastructure.

Numerical Stability: Each new tile introduces a 'reset' point for SCAMP's extrapolation. When a new tile computation begins, SCAMP directly computes high-precision initial dot products of the distance matrix at that row and column. This reduces the likelihood that rounding errors propagate along diagonals. In contrast, GPU-STOMP$_{OPT}$ extrapolates the diagonals of the distance matrix from a single initial value.

Fault-Tolerance: SCAMP_tile independently issues and completes processing for each tile; as a result, it is inherently preemptable, which increases the fault-tolerance of our framework. If a worker executing a tile "dies" or otherwise fails to complete its work, the host can simply reissue a new instance of the incomplete tile into the work queue. As mentioned in Section 1, many commercial cloud providers allow users to purchase spot instances at discounted prices. Spot instances are only useable by fault-tolerant applications because the cloud provider can kill the instance at any time. Thus, SCAMP provides a pathway for lower-cost cloud-based MP computation, which GPU-STOMP$_{OPT}$ cannot provide. SCAMP users can increase the number of compute resources purchased at a fixed cost point, which increases the size of the time series datasets they can process using SCAMP.

3.3 Numerical Optimization and Unrolling

To improve performance and numerical stability, SCAMP reorders GPU-STOMP$_{OPT}$'s floating-point computations and replaces its sliding dot product update (Equation 1) with a centered-sum-of-products formula (Equations 3-7). These transformations reduce each thread's demand for shared memory; at the same time, increasing the amount of shared memory allocated to each thread, allows each worker to compute four separate diagonals (Figure 6).

$$df_0 = 0;\ df_i = \frac{T_{i+m-1} - T_{i-1}}{2} \quad (3)$$

$$dg_0 = 0;\ dg_i = (T_{i+m-1} - \mu_i) + (T_{i-1} - \mu_{i-1}) \quad (4)$$

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

$$P_{i,j} = \overline{QT}_{i,j} * \frac{1}{\|T_{i,m} - \mu_i\|} * \frac{1}{\|T_{j.m} - \mu_j\|} \quad (6)$$

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

Equations 3 and 4 precompute 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 specific to self-joins and are a special case of a more general formula for an AB-join [27]. This new formula reduces the number of incorrectly rounded bits.

Equation 6 replaces the Euclidean distance used in previous MP computations [43][44][41] with the Pearson Correlation; Pearson Correlation can be computed incrementally using fewer computations than ED, and can be converted to z-normalized ED in O(1) by Equation 7. SCAMP also precomputes the inverse L2-norms in Equation 6 to eliminate redundant division operations from SCAMP's inner loop.

four diagonals), while ensuring the per-thread-block register and memory usage remains low enough to achieve 50% occupancy on a Tesla V100 GPU (see Ref. [11] for details). MP computation on the GPU is bound by shared memory *loads* not compute time. Unrolling permits SCAMP to use vectorized shared memory loads for dependencies, enabling consolidation of shared memory transactions.

SCAMP tracks the maximum per-row and per-column distances and updates the corresponding MP value in shared memory when an improvement occurs, resulting in a single update per row. In contrast, GPU-STOMP$_{OPT}$ compares every newly computed distance to the MP cache.

3.4 Floating-point Precision Options

We evaluated SCAMP under two precision modes:

SCAMP$_{DP}$ performs all computation and stores all intermediate shared memory values in double-precision. SCAMP DP generated accurate results for all datasets that we tested, regardless of size, noise, ill-conditioned regions, etc.

SCAMP$_{SP}$ performs all computation and stores all intermediate shared memory values in single-precision, which increasing performance and memory utilization by ~2x. SCAMP$_{SP}$ was adequate for highly regular datasets, such as ECG or accelerometer data, but may yield incorrect results for ill-conditioned data (see Section 4.2 for a detailed analysis). Using vectorized shared memory loads, SCAMP$_{SP}$ executes two 128-bit loads per column dependency and one 128-bit load per row dependency. This enabled all intermediate values to be stored in registers without spilling.

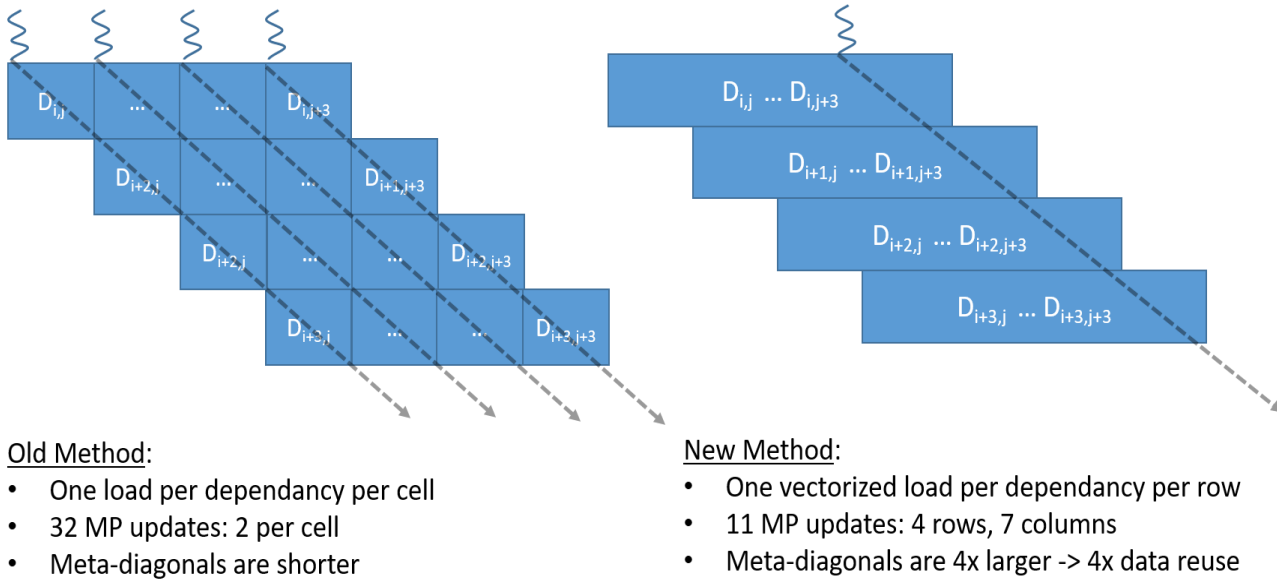

Figure 6: One iteration of the innermost loop of GPU-STOMP$_{OPT}$ (left) and SCAMP (right). Self-joins require only half of the distance matrix, but we must track both the MP value for the columns and for the rows. AB-joins only require the columns or the rows.

compute 16 new distances per iteration (four distances for each of

We tested SCAMP using half-precision (16-bit) floating-point operations but found that SCAMP identified incorrect motifs for many data sets; we do not consider half-precision any further.

3.5 Multi-Node AWS Deployment

We deployed SCAMP on Amazon Web Services (AWS), as representative commercially available cloud platform (see Figure 7). We first partition our time series data set into equal-sized chunks ranging from 20 to 100 million elements. There is a tradeoff here between the overhead of initiating new jobs, intermediate data size, and the risk of a job being preempted and losing work. We compress each chunk and store it on the cloud (Amazon S3), where it can be read by worker nodes. There is existing work on array stores, [54], that might be leveraged in providing access to the input array among worker nodes, but for simplicity we defer a study on these methods to future work.

We use AWS batch to set up a job queue backed by a compute cluster of p3.16xlarge spot instances. We issue an array batch job in which each job computes the MP for one tile. We issue one job per worker, and the tile size is specified to ensure full saturation of each worker's compute resources. This maximizes throughput of the processing pipeline without risking exorbitant progress loss if Amazon preempts a worker.

Each worker first copies and decompresses its input segments corresponding to the row and column of its tile. Each tile has two inputs: a segment corresponding to the tile-row, and another corresponding to the tile-column; each job computes an AB-join on the inputs. Next, the worker executes **SCAMP_host** on the input, further subdividing the tile among its GPUs. Once the worker computes the MP and index associated with the tile, the result is compressed and written back to Amazon S3.

Each job dequeues after it terminates. After all jobs terminate, another job decompresses and merges each tile's MP into the final result; as long as intermediate data growth is limited, this is relatively simple. In a 1 billion datapoint experiment, we merged 196 GB of intermediate results in ~1 hour using one AWS machine. The merging step could be further parallelized using a framework such as MapReduce [10].

Intermediate output data volumes can grow to tens or hundreds of gigabytes for input sizes up to 1 billion elements. Small tile sizes produce too much local information to reasonably store. SCAMP's space requirement is O(*RN*) where *R* is the number of tile rows, and *N* is the length of the final MP. If the tile size is 1, then $R = N$ and processing one billion elements necessitates storing the distance matrix (~1 quintillion values). If each intermediate value is eight bytes compressed on disk, the total storage requirement would be ~8 exabytes, the estimated aggregate storage capacity of Google's datacenters in 2014 [42].

4 Experimental Evaluation

All experiments reported here are reproducible. All code and data (and additional experiments omitted for brevity) are archived in perpetuity [27].

4.1 Performance Comparison

4.1.1 Comparison to GPU-STOMP$_{OPT}$. Table 3 reports the result of a direct comparison of SCAMP to GPU-STOMP$_{OPT}$ using random walk datasets of various lengths. The first column reports the performance of GPU-STOMP$_{OPT}$ using the code from Ref. [41] on an Nvidia Tesla K80 GPU. The results here are similar, but vary slightly due to a change in the timing of the experiment to improve precision.

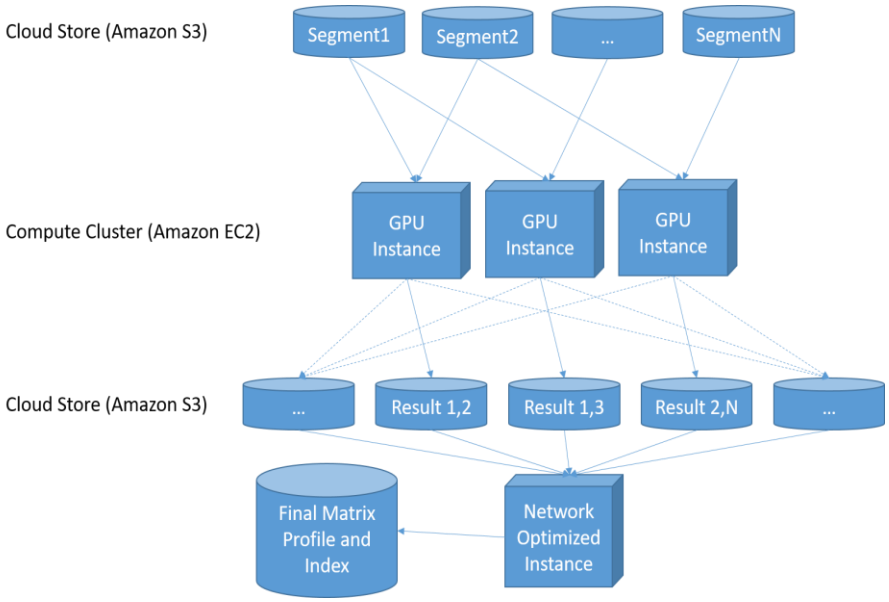

Figure 7: Illustration of how to distribute SCAMP in a cluster of GPU instances on AWS.

Table 3: SCAMP Runtime Evaluation

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

The second column reports the execution time of the same code (still GPU-STOMP$_{OPT}$) running on a single Nvidia Tesla V100 SXM2 on Amazon EC2. The reported speedup is due to the V100's higher instruction throughput compared to the K80, which is bottlenecked by the latency of atomic updates to shared memory. Nvidia implemented shared memory atomics in hardware and included them in their instruction set architecture (ISA) starting with the Maxwell GPU family [30]; they are no longer a performance bottleneck on newer GPU architectures. The third and fourth columns report the execution time and speedups (relative to Column 1) of SCAMP$_{DP}$ and SCAMP$_{SP}$ running on the V100 GPU. The reported speedups are due to the optimizations described in Sections 3.2 and 3.4 (SCAMP$_{DP}$) and the conversion from double to single precision (SCAMP$_{SP}$); SCAMP$_{SP}$ does not always produce the same result as SCAMP$_{DP}$.

4.1.2 Scalability. Figure 8 depicts an analytical performance model for SCAMP's execution time under ideal conditions. Given the runtime of SCAMP (T_o) on one GPU on a dataset of a size (N_o) which sufficiently saturates compute performance, we construct an analytical model (Equation 8) to estimate SCAMP's execution time across G GPUs on a time series of length N under ideal assumptions (e.g., no communication overhead).

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

N_o and T_o are initialization parameters provided by one trial run on a single V100 GPU. We use this equation and the SCAMP$_{DP}$ runtime for input size 2^{23} (Table 3) to construct the model:

Each data point in Figure 8 corresponds to an experiment we ran, which demonstrates that the empirical model is highly accurate. The data for our distributed workloads in the next section also align well was this plot but were not included due to space and readability constraints. More detail is available on our supporting webpage [27]. Under this model, the cost of a problem remains constant if there is no distributed overhead. For example, to compute a join of 530 million using double-precision, one can either use 8 GPUs for 8 hours, or 64 GPUs for 1 hour. The cost is identical as long as there is no difference in the cost per hour for GPU compute time.

4.1.3 Distributed Performance: p3 spot instances. Next, we evaluate SCAMP's performance on two very large earthquake datasets. Both experiments ran on 40 V100 GPUs, each in a different configuration, on an AWS EC2 spot instance fleet. A spot instance fleet automatically provisions a consistent number of spot instances for the job queue. If one instance is preempted, AWS provisions another for the fleet as long as there are available instances. A spot instance user accesses compute resources not sold to customers who pay full price for non-preemptable instances. Spot instance prices increase when demand is high; when demand is low, the provider loses money, but mitigates losses by selling preemptable access to the highest bidder.

The Parkfield dataset ran on a five p3.16xlarge spot instance fleet, where each instance is equipped with eight V100 GPUs. The p3.16xlarge instances were in high demand at the time of the experiment: many jobs remained queued at times that AWS could not provide capacity to execute; we were only charged for active GPU compute time. The Cascadia Subduction Zone dataset ran on ten Amazon EC2 p3.8xlarge instances each equipped with four V100 GPUs. These instances were in lower demand than those used for the Parkfield data set experiments, allowing faster job completion time with less queuing overhead. The spot price of Amazon spot instances is dynamic and demand-driven [39], and we were charged a higher spot price. Table 4 reports the results of these experiments.

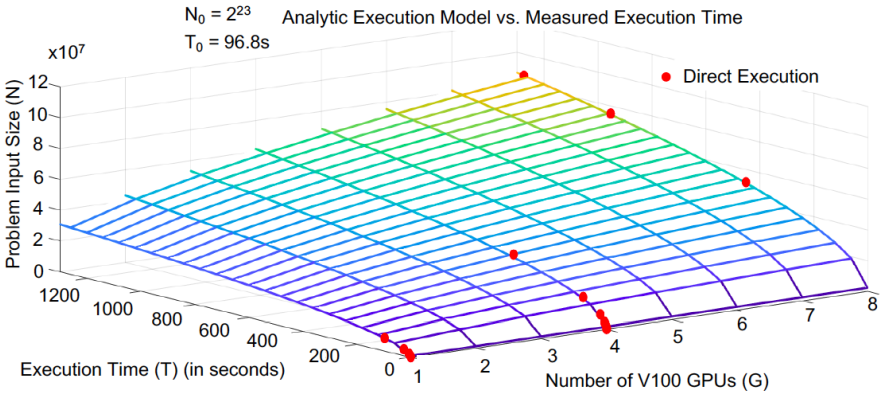

Figure 8: Equation 8 plotted using N_o and T_o from Table 3, the V100 double precision result for a dataset with 2^{23} data points. Dots correspond to values measured during experiments reported in this paper. Results are for a single non-preemptable instance equipped with G GPUs. Equation 8 also generalizes to multi-instance distributed workloads.

Table 4: Summary of various distributed runs on AWS spot instances

Dataset	Parkfield	Cascadia
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

Table 5: Optimized CPU and GPU SCAMP$_{DP}$ cost on a single AWS spot 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**)

4.1.4 CPU Comparison. Table 5 compares the performance of our GPU implementation of SCAMP$_{DP}$ to a CPU implementation running on a 72-core c5 18xlarge spot instance (Intel Skylake CPU). The CPU implementation saturates performance 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 cross-architecture performance, we can and do 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 an imperfect basis for comparison: we could have used a smaller spot instance type to achieve better price per performance on a CPU when small input data sizes fail to saturate the 72 available cores on the c5 18xlarge instance.

4.2 Precision Evaluation

Consider the three data snippets shown in Figure 9. Each has a constant region longer than the chosen motif length *m*. Constant regions are a source of numerical instability. Many scientists are interested in the similarity of *z-normalized* subsequences. Z-normalization divides each data point by the standard deviation of the entire subsequence. For a constant region, the standard deviation is 0. Near-constant subsequences are also problematic, because they pass a bit-level test for two distinct values but result in division by a number very close to 0.

Constant regions are common. For example, in medical datasets, we have observed constant regions caused by:

Disconnection Artifacts: These may occur due to disconnection of a monitoring lead, e.g., during a bed change.

Hard-Limit Artifacts: Some devices have a minimum and/or maximum threshold defined by a physical limit of the technology. If the true value exceeds the limit for a period of time, a constant value occurs for the duration (Figure 9.*center*).

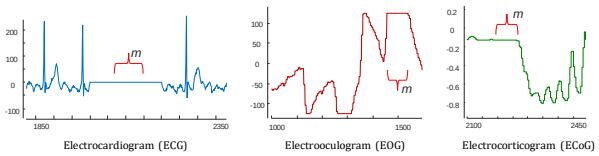

Figure 9: Three time series containing a constant region caused by different issue [9]. *left*) An ECG (heart) with a disconnection artifact. *center*) An EOG (eye movement) with a hard-limit artifact. *right*) An ECoG (finger flexion) with constant region caused by low precision recording.

Low Precision Artifacts: Many devices record at low-precision fixed-point; observed constant values may *not* be constant at a higher precision.

In most cases, disconnection artifacts saturate to a Pearson Correlation of 1 or a z-normalized Euclidean Distance of 0, and are removed later via a post processing step. If small peaks and valleys are important in a low-precision artifact scenario, the MP can be computed and stored in double-precision.

4.2.1 Comparison with Previous Update Method. Figure 10, compares SCAMP's update method (Equations 3-7) with the prior method implemented in GPU-STOMP$_{OPT}$. We compute the result first in double precision, then plot the absolute error in computed Pearson Correlation between the double and single precision for both SCAMP and GPU-STOMP$_{OPT}$.

The bottom and middle of Figure 10 elucidate how Equations 1 and 2 (GPU-STOMP$_{OPT}$'s update method), completely fail in single precision on this dataset. We capped the error at 1 for GPU-STOMP$_{OPT}$, which is half of the range of Pearson Correlation. The actual values reported by GPU-STOMP$_{OPT}$ were many times larger than the entire range of Pearson Correlation.

In contrast, SCAMP only exhibits error in constant regions that arise due to disconnection artifacts. Here, a domain expert can easily clean up SCAMP's results with minimal effort by omitting these regions from consideration when analyzing the output of SCAMP. In contrast, GPU-STOMP$_{OPT}$ fails to produce a meaningful result across almost most of the dataset.

4.2.2 General Considerations for Precision. Next, we analyze the effect of reducing precision on various datasets of different lengths. We use a tile size of 1 million for SCAMP while GPU-STOMP$_{OPT}$ computes across the entire length of the input in one go, as it does not perform tiling. We generate the MP using SCAMP$_{DP}$, SCAMP$_{SP}$ and GPU-STOMP$_{OPT}$ with single and double precision. We used a window length longer than the longest flat artifact region in the data, to allow us to isolate errors caused by the update formula from the inherent loss of information from artifacts that cannot be represented in lower precision.

Table 6 presents the results of the experiment. Altogether SCAMP was three or more orders of magnitude more accurate than STOMP on these datasets. Each entry in Table 6 is the maximum absolute error found between the double and single-precision MP calculations. We highlight absolute errors that exceed 0.01 in red to emphasize that a domain scientist would not consider these results sufficiently accurate to use or report.

SCAMP$_{SP}$ suffers a substantial accuracy loss compared to SCAMP$_{DP}$ but achieves much higher performance. If a user's dataset and application can tolerate the loss of accuracy, there is much to be gained in terms of efficiency. We observe that

SCAMP$_{SP}$ works well on data that is highly regular with a small min-max range, exemplified by ECG data.

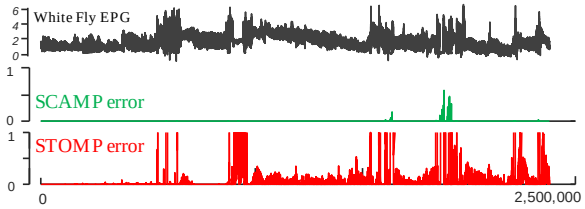

Figure 10: Single precision error comparison between GPU-STOMP$_{OPT}$ and SCAMP on White Fly EPG dataset. *top*) original data. *middle*) SCAMP absolute error. *bottom*) GPU-STOMP absolute error.

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

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

SCAMP$_{SP}$ completely fails on the Earthquake dataset in Table 6. This is because the large earthquake's signal has a magnitude greater than 10^7, which cannot be represented precisely by single-precision floats. It may be possible to reduce the error of SCAMP$_{SP}$ for more types of data, but we leave this task for future work.

5 Case Studies in Seismology

Figures 1 and 4 suggest that motifs are important to many domains. Due to space limitations, we limit our case studies reported in this paper to seismic data, which provide information about Earth's interior structure and processes. We define seismic data to be any recorded motion (e.g., displacement, velocity, acceleration) measured using seismic instruments at the Earth's surface. Detected and located seismic events (i.e. earthquakes) can be used for studying earthquake source processes and source physics, fault behavior and interactions, for determining Earth's velocity structure, and to constrain seismic hazard [12]. Many of these applications benefit from detection of smaller events, which can be missed due to insensitive detection algorithms, or human analyst error [48]. Improvements to seismic data instruments, networking and data management, and reductions in cost, have resulted in a power law increase in seismic data volume [19]. Probing this huge volume of data is an ongoing challenge.

Performing query searches for seismic data can 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. While an 'autocorrelation' motif discovery method can identify suitable queries, it is expensive computationally in terms of memory and time [6][34]. The analysis in [6] was restricted to one hour of data, which limited the number of discoverable motifs.

Other studies have performed motif discovery by converting seismic time series to small and dense proxies, and computing a Locality-Sensitive Hash (LSH) [4][7][32], an approximate and reduced-dimension nearest neighbor search. This approach was ~143x faster than autocorrelation for one week of continuous data, but produced false positive and false negative results [7]. In addition, LSH requires the careful selection of multiple, data set-specific tuning parameters, a process that requires visual inspection and validation against the results of other methods.

In contrast, SCAMP can exactly search datasets that can only be searched approximately using current methods. We consider the milestone of one billion data points (~579 days, ~1.5 years) of seismic data with a 20 Hz sample rate. In two examples, we demonstrate how and why transitioning motif discovery timescales from hours of data to years of data is a potential game changer for the field of seismic data mining.

5.1 Detecting Foreshocks and Aftershocks

The town of Parkfield, located on the San Andreas fault in central California, experienced four magnitude ~6 earthquakes in the 20th Century: 1901, 1922, 1934 and 1966 [45]. A repeat event was predicted to occur between 1985 and 1993, spurring the 'Parkfield Earthquake Prediction Experiment', which tried to capture the earthquake with the best available instrumentation. The actual event (the 'mainshock') occurred 'late' in 2004, and was recorded in extraordinary detail by the low-noise, borehole seismometers of the Parkfield High Resolution Seismic Network (HRSN) [45][47]. Many of these earthquakes were detected and cataloged in real-time at the Northern California Earthquake Data Center (NCEDC) by an automated procedure, and quality checked for false positives by human analysts. We use this catalog as a reference. To investigate *i*) whether the HRSN data contain information on any aftershocks that were not included in the NCEDC catalog, and *ii*) whether there was any change in behavior before the mainshock, we ran SCAMP on 580 days (1,002,240,008 points) of data from Parkfield. We use 20 Hz horizontal component seismic data (from 28-11-03 to 9-7-05) from the HRSN station VCAB, centered on the 2004 Parkfield mainshock time (i.e. 28-9-04). We set the query length at 100 samples (5 seconds). We band-pass filtered the data between 2 and 8 Hz, a frequency range that can detect low signal-to-noise ratio earthquakes.

Figure 11 shows a zoom-in of two sections of the waveform and their corresponding MPs. The motifs for aftershocks of the Parkfield earthquake have a very characteristic shape. The MP drops abruptly as the query window begins to capture the beginning of the earthquake waveforms, followed by a gradual increase back to the background noise level, indicating that the two waveforms

being compared have similar shapes at their beginnings, and dissimilar shapes at their ends.

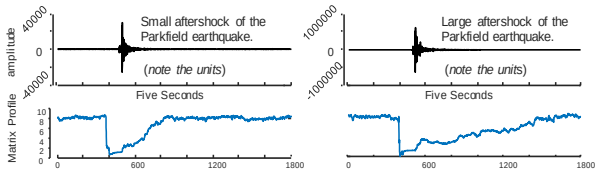

Figure 11: Examples of a waveform snippet (*top*) and corresponding MP shape (*bottom*) for aftershocks of the Parkfield earthquake. *left*) a small aftershock. *right*) a larger aftershock with a waveform amplitude that is three orders of magnitude larger.

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. The initial drop in the MP indicates the waveforms have the same first motion polarity. The next few seconds of arrivals to the station include reflections, refractions and reverberations of seismic waves – collectively referred to as the seismic 'coda' – which are much more sensitive to differences in earthquake location, and therefore much less similar between pairs of events [1]. The duration of the gradual increase in the MP is longer for the larger event (Figure 11.*right*), consistent with the empirical relationships of signal duration (and coda length) with event magnitude [21][8]. We propose two important applications of MP results to seismology: *ii*) The abrupt initial drop of the MP can select the first motions 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, which correlates with earthquake magnitude [8][21].

Next, we performed an event-detection experiment using a MP containing the Pearson Correlation Coefficient (MPCC, for short). Pearson correlation is bounded in the range [-1,+1], can be trivially converted to Euclidean Distance, and is widely used in seismology studies [31][25][37]. We count the number of MPCC peaks separated by at least 100 samples (5 seconds) to prevent overcounting the same earthquake when multiple peaks are present for one event. Long traces of seismograph data often contain repeated patterns corresponding to special types of sensor noise; these are easy to filter, as they create near perfect motifs. We count the number of MPCC peaks in the range [0.90, 0.99].

Figure 12 shows the number of MPCC motifs per day for our 580 days of VCAB data. Although we targeted the Parkfield earthquake, we detected other nearby earthquakes and their aftershocks, notably the 2003 M_w 6.5 San Simeon event, and two other moderate (M_w 4.0–4.5) earthquakes nearby. A series of motif peaks in the lead-up to the Parkfield mainshock (around 04/07/01) do not correspond to events in the regional earthquake catalog, and may represent previously undetected foreshock activity; we have reported them to collaborators in seismology to investigate.

Figure 13 compares the total number of motifs in the MPCC range [0.9, 0.99] over the first 90 days of the Parkfield aftershock sequence with the number of catalog aftershocks reported in the NCEDC catalog. This analysis reports ~16x more detections than those reported by the NCEDC. Some of these thresholding-based detections may be station artifacts, but visual inspection suggests that they account for less than 5% of the events.

We also fit the Omori-Utsu aftershock rate equation [46] to the detected and catalogued aftershocks of the Parkfield earthquake. Figure 14 shows that the number of motifs per day fit the Omori-Utsu law almost perfectly. Values retrieved from the Omori-Utsu rate equation can provide information about the physics of the mainshock [16] and also even can be used for forecasting large aftershocks [28].

5.2 Detecting Subtle Seismic Motifs

Low frequency earthquakes (LFEs) are seismic events that occur deep in the crust and typically have very low signal-to-noise ratio signals. LFE recurrence is a proxy for movements at the roots of fault zones, and may be useful in short-term earthquake forecasting [51][52][53]. LFEs have been observed in the Cascadia subduction zone, where the Juan de Fuca plate subducts beneath the North American plate, from coastal Northern California to Vancouver Island. This 'megathrust' fault has the potential to produce great (magnitude ~9) earthquakes [2], motivating LFE detection in this region. Their low signal-to-noise ratios make detecting them challenging and time consuming (e.g., requiring sophisticated methods and visual inspection; [49][[50][6]).

In order to see if we can detect these novel events in this region, 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 to 200 (10 seconds), based on the length of LFE templates used in previous studies [49].

Figure 15 shows the motif density over time for this experiment. The number of motifs starts to increase around August 2006 and decrease in November 2006, and again increase in June 2007 and start to decrease around October 2007. We visually inspected some of these motifs (in both time and frequency domain) and classified them in four categories: i) regular earthquakes (less frequent, Figure 16. left.) ii) weather or human related signals (frequent), iii) Station artifact (less frequent), iv) LFE-like signals (frequent, Figure 16.right). Confirming a signal to be LFE is not easy, typically requiring detection at several stations and visual inspection of its frequency spectrum. In Figure 16 we show a discovered motif that was confirmed as a true LFE in [49]. Note that the MP for the LFE is not as low as regular earthquake but much lower than the background noise (Figure 16).

In general, we detect fewer than 150 motifs per day in this dataset. This means that in order to discover LFEs a seismologist needs to inspect fewer than 150 sub-windows per day of data, a task that would take minutes to perform. In contrast, the traditional visual inspection method for detecting LFEs (e.g., brute force checking [50]) requires inspection of thousands of sub-windows (e.g., 17280 sub windows with a 5 second skip), potentially taking hours for each day of seismic data. Running SCAMP before searching for these subtle and important motifs could potentially provide a large time savings for seismologists and make their discovery much easier in this domain.

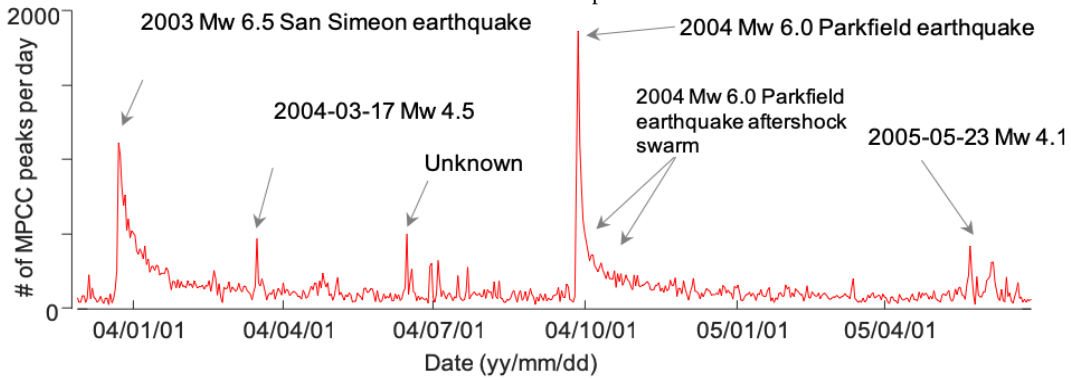

Figure 12: Daily number of discovered motifs for 580 days of data centered on the Parkfield earthquake (04/09/28), measured on the horizontal component of station VCAB, located ~10 km from the epicenter. Motifs are selected based on the peak MPCC values.

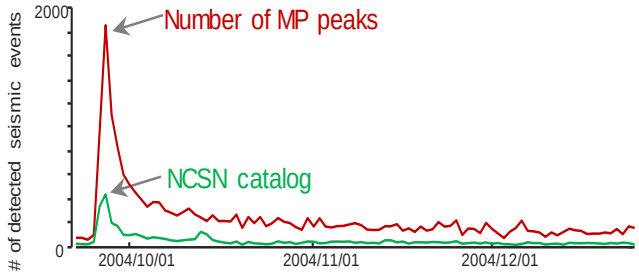

Figure 13: The number of events in the USGS NCSN Catalog (green line) and the number of motifs detected using SCAMP (red line) for the Parkfield earthquake aftershock sequence. For the catalog events we considered all events in a box with length ~200 km centered on the Parkfield mainshock epicenter. The start of seismicity in this plot is 4 days prior to the Parkfield earthquake

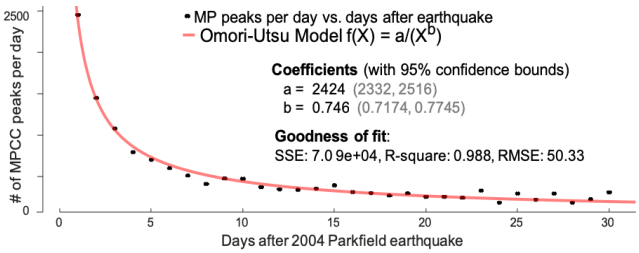

Figure 14: A fit of an Omori-Utsu relationship [46] (i.e. the law that describes aftershock rate behavior) to the number of motifs per day for the first 30 days after the Parkfield mainshock. The R-squared of 0.988 indicates a very good fit and shows how the number of motifs can describe the expected aftershock behavior almost perfectly.

These results were obtained by post-processing an MP produced by SCAMP; possibilities for further refinement remain open. These results show that SCAMP can detect LFEs, and has the potential to more generally explore the seismicity of the southern Cascadia subduction zone and other similar regions. We believe that SCAMP has a rich future in seismic data mining – a discipline that traditionally suffers from false negatives – and other domains that produce time series.

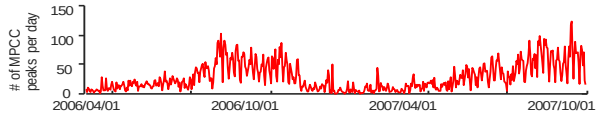

Figure 15: Discovered motifs for 579 days of seismic data recorded on the vertical channel of station I02A, located near Mapleton, OR. The number of discovered motifs based on MPCC thresholding method shows two six-month periods were detected motifs gradually increase, that start in mid-2006 and mid-2007. We believe many of these motifs are low frequency earthquakes (see Figure 16).

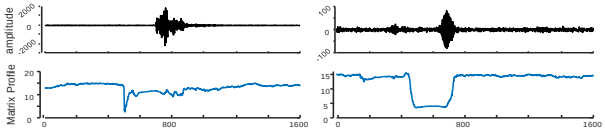

Figure 16: *left*) An example of an earthquake waveform snippet (top) and MP shape (bottom) in the vicinity of a discovered motif for a 'regular' earthquake. *right)* A waveform snippet and corresponding MP from a confirmed LFE (identified by [49]).

6 Conclusion

SCAMP *exactly* searches for motifs in time series at the data-center scale. 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 time series dataset. Likewise, we believe this to be the first work to do *exact* motif search on more than one year (1.59 years to be precise) of continuous earthquake data. All code has been made freely available to the general public [27], whom we invite to confirm, extend, and *exploit* our efforts.

ACKNOWLEDGMENTS

This work was supported in part by NSF Awards #1161997, #1528181, and #1763795.

REFERENCES

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

[2] B. Atwater, et al. 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] Yoon, C. E, et al. Earthquake detection through computationally efficient similarity search. *Science advances*, *1*(11): e1501057, 2015.

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

[9] H. Dau and E. 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] Nvidia Tesla V100 Whitepaper: http://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[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* (2017), pages 155– 164.

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

[15] S. Gupta, et al. Deep learning with limited numerical precision. In *Proceedings of the 32nd ICML*, pages. 1737-1746. JMLR, 2015.

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

[17] D. Hill, et al. The 1989 earthquake swarm beneath Mammoth Mountain, California: An initial look at the 4 May through 30 September activity. *Seismological Society of America*, 80(2): 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*, B11: 22133, 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 et al. Evidence for long-timescale patterns of synaptic inputs in CA1 of awake behaving mice. 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] SCAMP Supporting Webpage: https://sites.google.com/view/2019scamp

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

[31] G. Poupinet, et al. Monitoring velocity variations in the crust using earthquake doublets: An application to the Calaveras Fault, California. *Geophysical Research: Solid Earth*, 89(B7): 1984.

[32] K. Rong, et al. Locality sensitive hashing for earthquake detection: a case study of scaling data-driven science. In *VLDB 2017*.

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

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

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

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

[41] Y. Zhu, et al. Exploiting a novel algorithm and GPUs to break the ten quadrillion pairwise comparisons barrier for time series motifs and joins. *KAIS* 1-34, 2018.

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

[43] C. C. M. Yeh, et al. 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, et al. 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] Bakun, W. H., et al. The Parkfield, California, earthquake prediction Experiment. *Science*, *229*(4714): 619-624, 1984.

[46] Utsu, T., & Ogata, Y. The centenary of the Omori formula for a decay law of aftershock activity. *Journal of Physics of the Earth*, *43*(1): 1-33, 1995.

[47] "HRSN (2014), High Resolution Seismic Network. UC Berkeley Seismological Laboratory. Dataset. doi:10.7932/HRSN."

[48] Brodsky, E. E. (2019). The importance of studying small earthquakes. Science, 364(6442), 736-737.

[49] Boyarko, D. C., & Brudzinski, M. R. (2010). Spatial and temporal patterns of nonvolcanic tremor along the southern Cascadia subduction zone. Journal of Geophysical Research: Solid Earth, 115(B8).

[50] Shelly, D. R. (2010). Migrating tremors illuminate complex deformation beneath the seismogenic San Andreas fault. Nature, 463(7281), 648.

[51] Shelly, D. R., G. C. Beroza, and S. Ide (2007). Non-volcanic tremor and low-frequency earthquake swarms, Nature 446, no. 7133, 305.

[52] Obara, K., & Kato, A. (2016). Connecting slow earthquakes to huge earthquakes. Science, 353(6296), 253-257.

[53] Rubinstein, J. L., Shelly, D. R., & Ellsworth, W. L. (2009). Non-volcanic tremor: A window into the roots of fault zones. In New Frontiers in Integrated Solid Earth Sciences (pp. 287-314). Springer, Dordrech

[54] The TileDB Array Data Storage Manager, VLDB'16. https://tiledb.io/