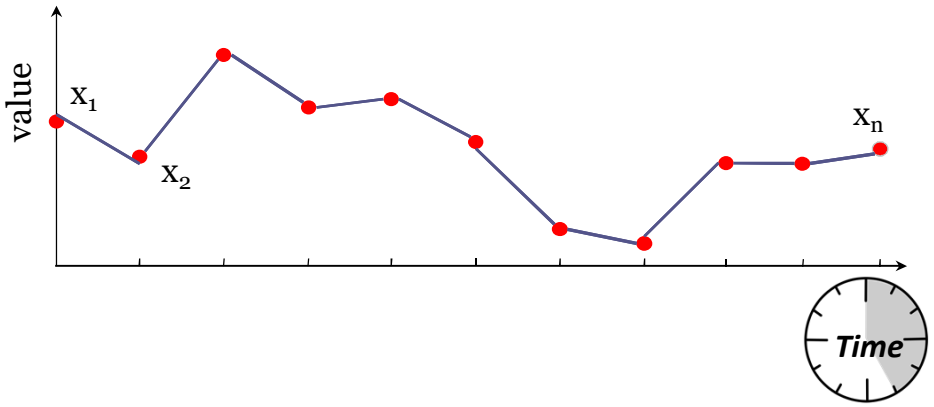
Parallel Data Series Indexing and Similarity Search on Modern Hardware

Panagiota Fatourou, Professor

University of Crete, Computer Science Department
Foundation of Research and Technology- Hellas
Greece

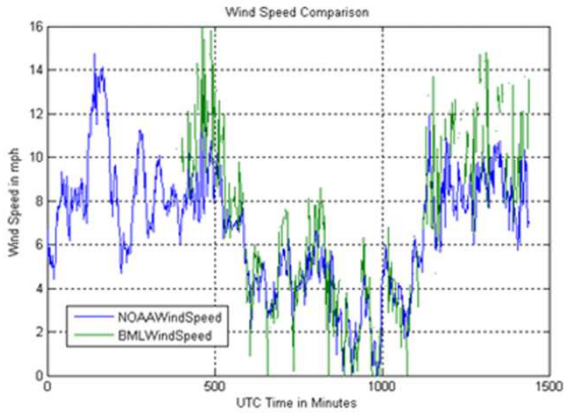
Data Series

- Sequence of points ordered along some dimension



Scientific Monitoring

meteorology, oceanography, volcanology, seismology, astronomy, finance, sociology, etc.



Wind speed

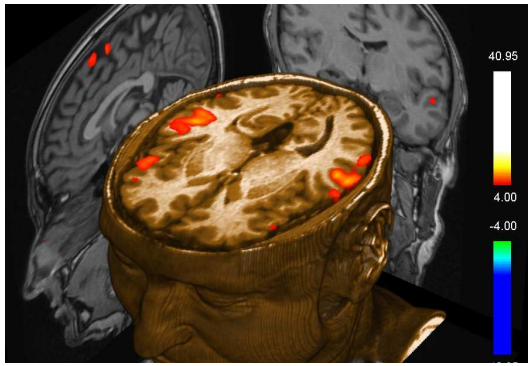
From ocean observing node project, http://bml.ucdavis.edu/boon/wind.html

Volcanic Activity Indicators

From British Geological Survey
https://www.bgs.ac.uk/geology-projects/volcanoes/

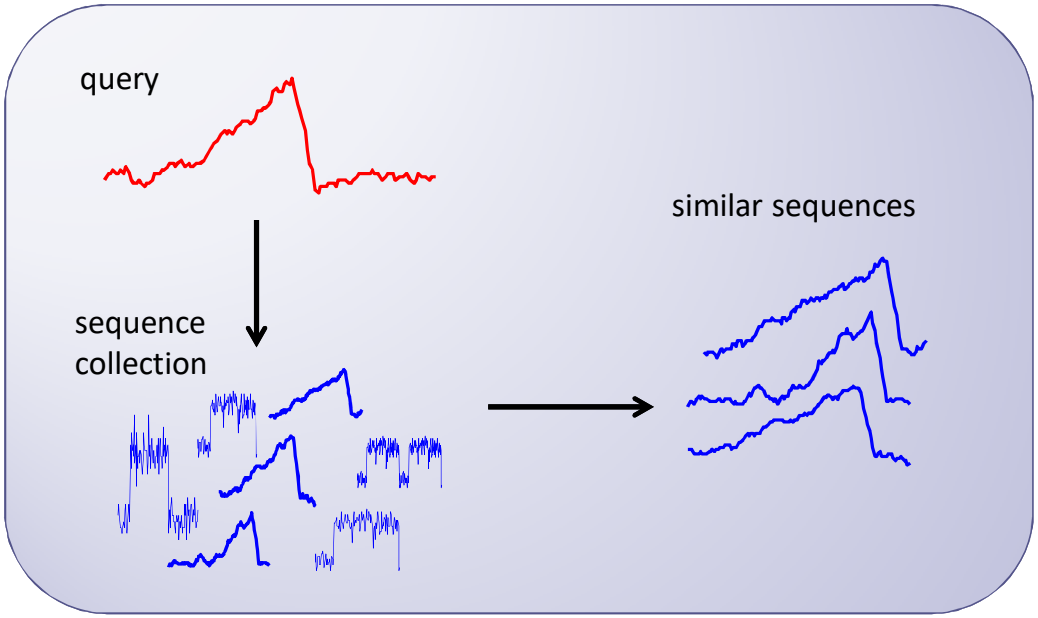
Neuroscience

- Functional Resonance Magnetic Imaging (fMRI) data
  - primary experimental tool of neuroscientists
  - reveal how different parts of brain respond to stimuli

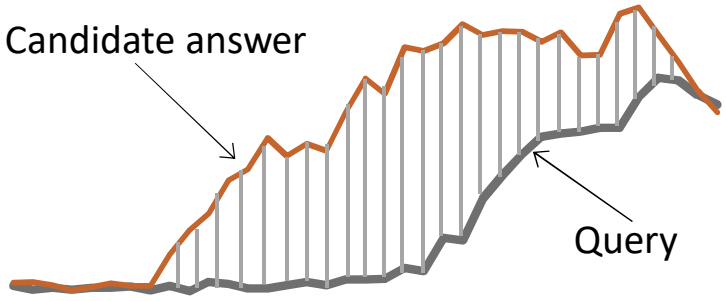


What do we want to do with data series?

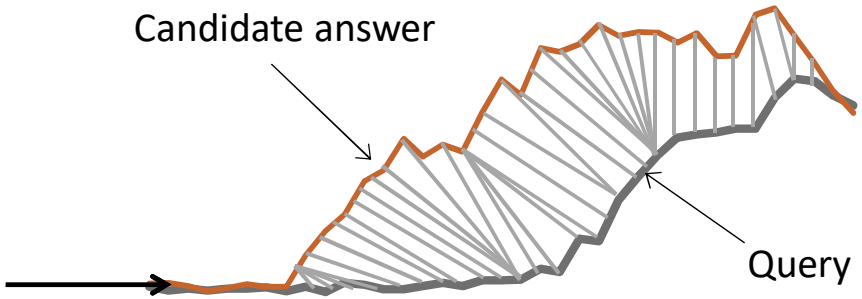
Similarity Search



Distance Metrics



Euclidean Distance



Dynamic Time Warping

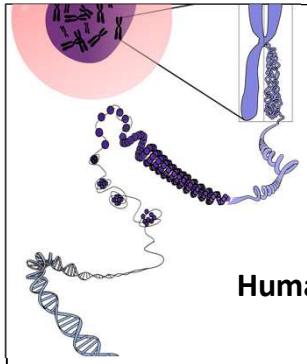
Challenge - Massive data Series collections

NASA's Solar Observatory

1.5 TB per day

Large Synoptic Survey Telescope

~30 TB per night



Human Genome project

130 TB

passenger aircrafts

20 TB per hour

data center and services monitoring

2B data series

4M points/sec

HARD, because of very high dimensionality: each data series has several hundreds to several thousands of points!

Contributions

- **ParIS+**, a disk-based concurrent data series index for modern hardware
  - Completely masks the CPU latency during index creation under I/O
  - Query answering up to an order of magnitude faster than previous approaches (~10 secs on 100 GB dataset)

  IEEE Trans. on Knowledge & Data Engineering 2021
  IEEE Big Data 2018

- **MESSI**, an in-memory data series index for modern hardware
  - Lower synchronization in index design
  - Novel algorithms for query processing
  - Similarity search at interactive speeds (~60msec on 100GB dataset)

  VLDB Journal 2021
  IEEE International Conference on Data Engineering (ICDE) 2020

Contributions

- **SING**, a data series similarity search accelerated by GPUs
  - CPU-GPU collaborative framework
  - Expands scalability of exact similarity search (~30msec on 100GB dataset)

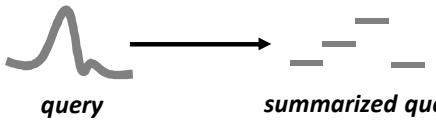
  IEEE International Conference on Data Engineering (ICDE) 2020

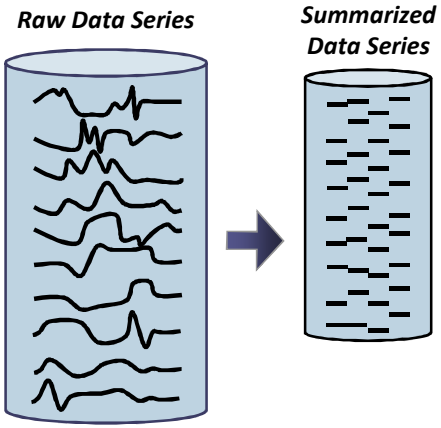
- **Odyssey**, a data series similarity search for multi-node clusters
  - Distributed data-series processing framework
    - good speedup and high scalability
    - takes advantage of full computational capacity of modern clusters
  - Efficient scheduling and load-balancing
  - Flexible partial replication scheme, which enables navigation through a fundamental trade-off between data scalability and good performance.

  Proc. of the VLDB Endowment 2023

Previous Algorithms *[Zoumpatianos et al., VLDBJ'16]*

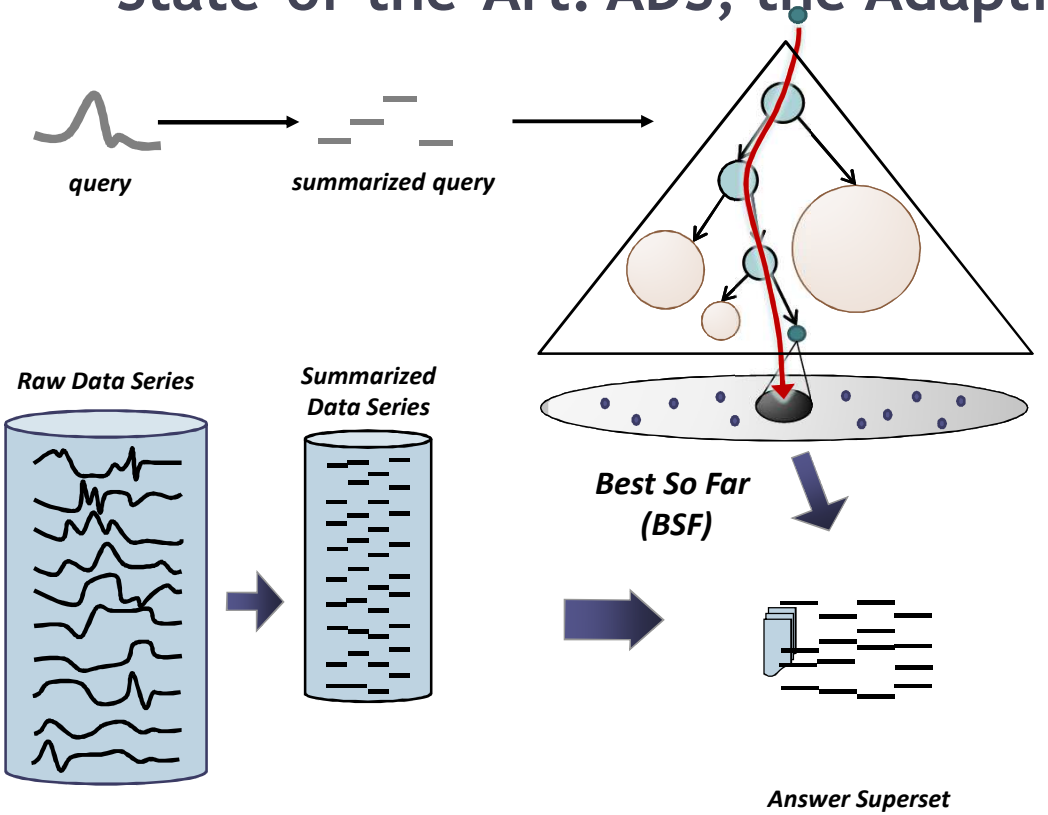
ADS, the Adaptive Data Series Index





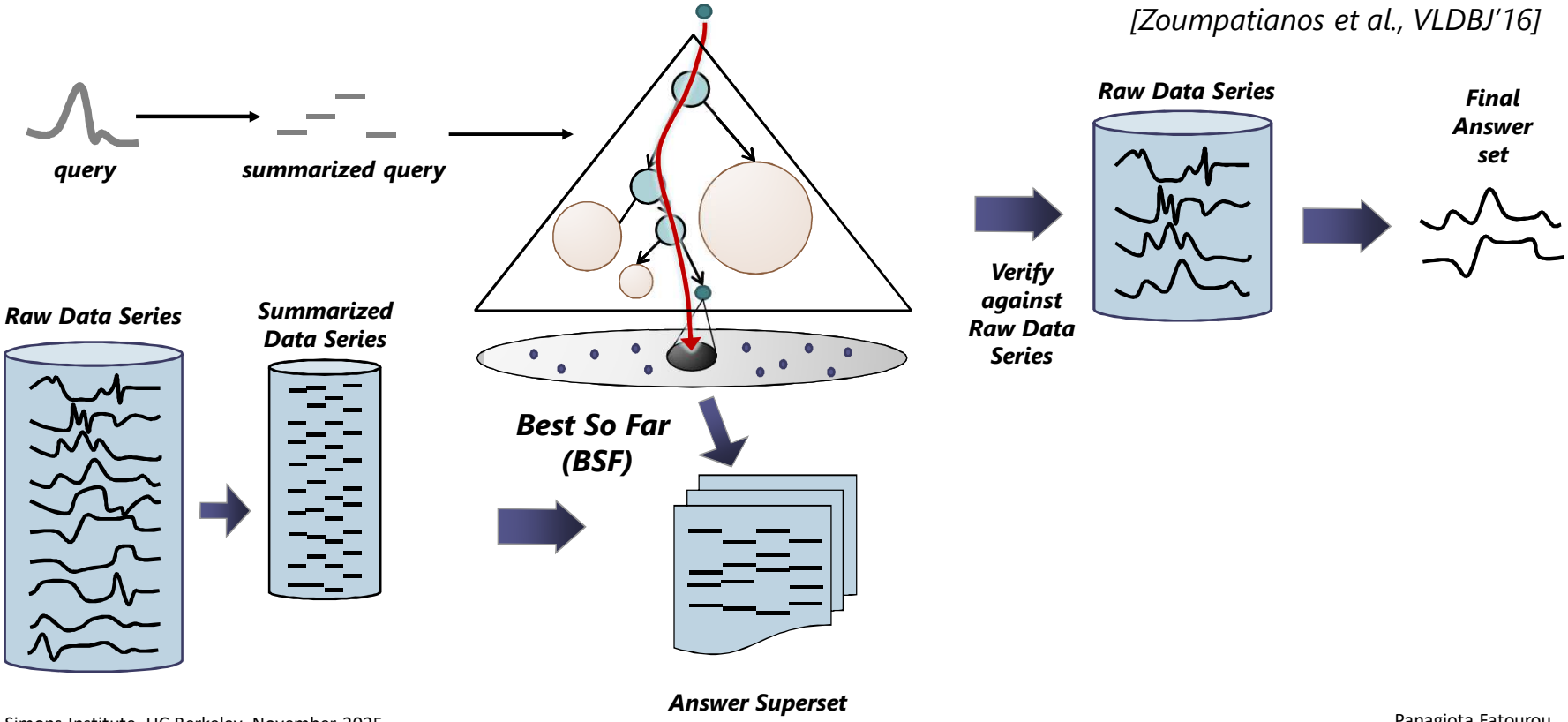
State-of-the-Art: ADS, the Adaptive Data Series Index

[Zoumpatianos et al., VLDBJ'16]



State-of-the-Art: ADS, the Adaptive Data Series Index

[Zoumpatianos et al., VLDBJ'16]



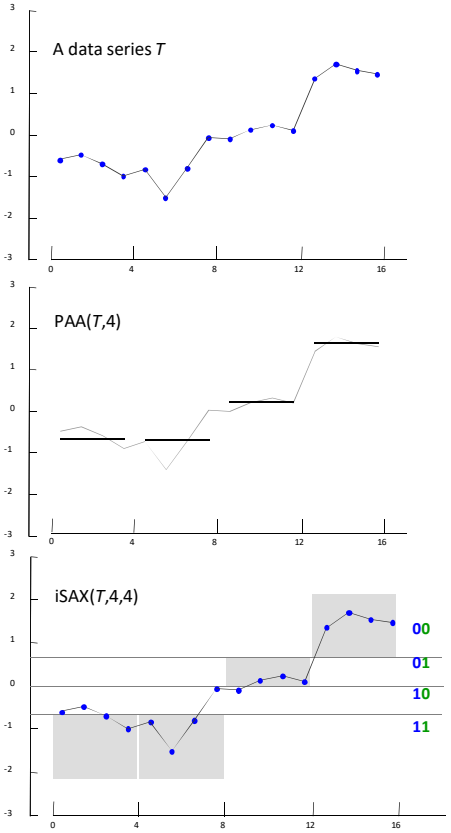
iSAX: Symbolic Aggregate approXimation

1. Represent data series *T* of length ***n*** with ***w*** segments using Piecewise Aggregate Approximation (PAA)

PAA(*T*,*w*) = $\overline{T} = \bar{t}_1, \ldots, \bar{t}_w$

where $\bar{t}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} T_j$

2. Discretize into a vector of symbols
   - Breakpoints map to small alphabet ***a*** of symbols



iSAX Representation

Lower Bound distance Calculation: Calculate distance between the iSAX summary of a data series and the query's iSAX Summary.

Real Distance Calculation: Calculate real (Euclidean) distance between the query series and a data series.

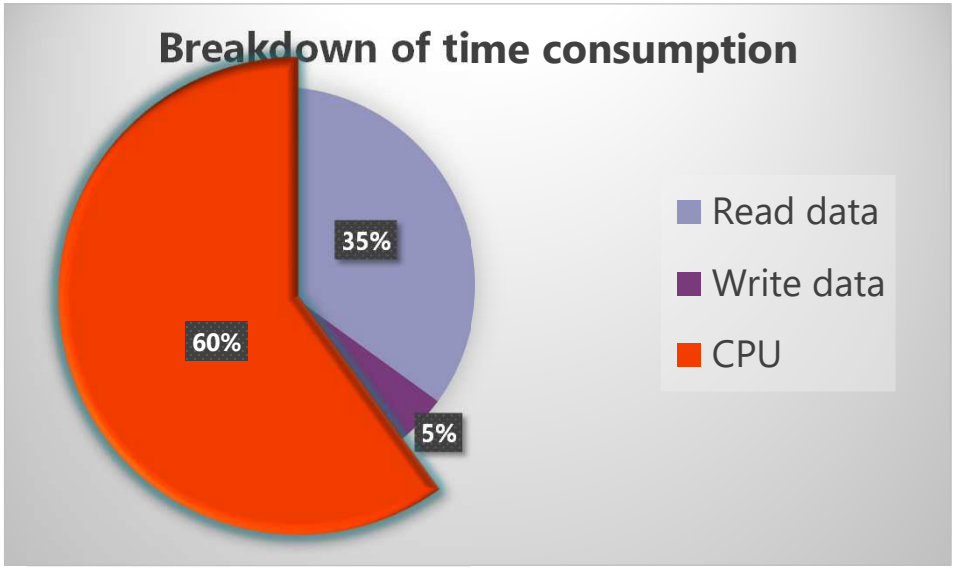
Lower-Bound Property of iSAX summaries: If the lower bound distance between a query Q and a data series DS is higher than a value v, then the real distance between Q and DS is also higher than v.

ParIS+ Index

B. Peng *(Chinese Academy of Sciences), P. Fatourou and* ***T. Palpanas*** *(Université Paris Cité)*

IEEE Trans. on Knowledge & Data Engineering 2021
IEEE Big Data 2018

ADS Index Creation



~60% of time spent in CPU: potential for improvement!

ParIS+: Parallel Indexing of Disk-Based Data Series Collections

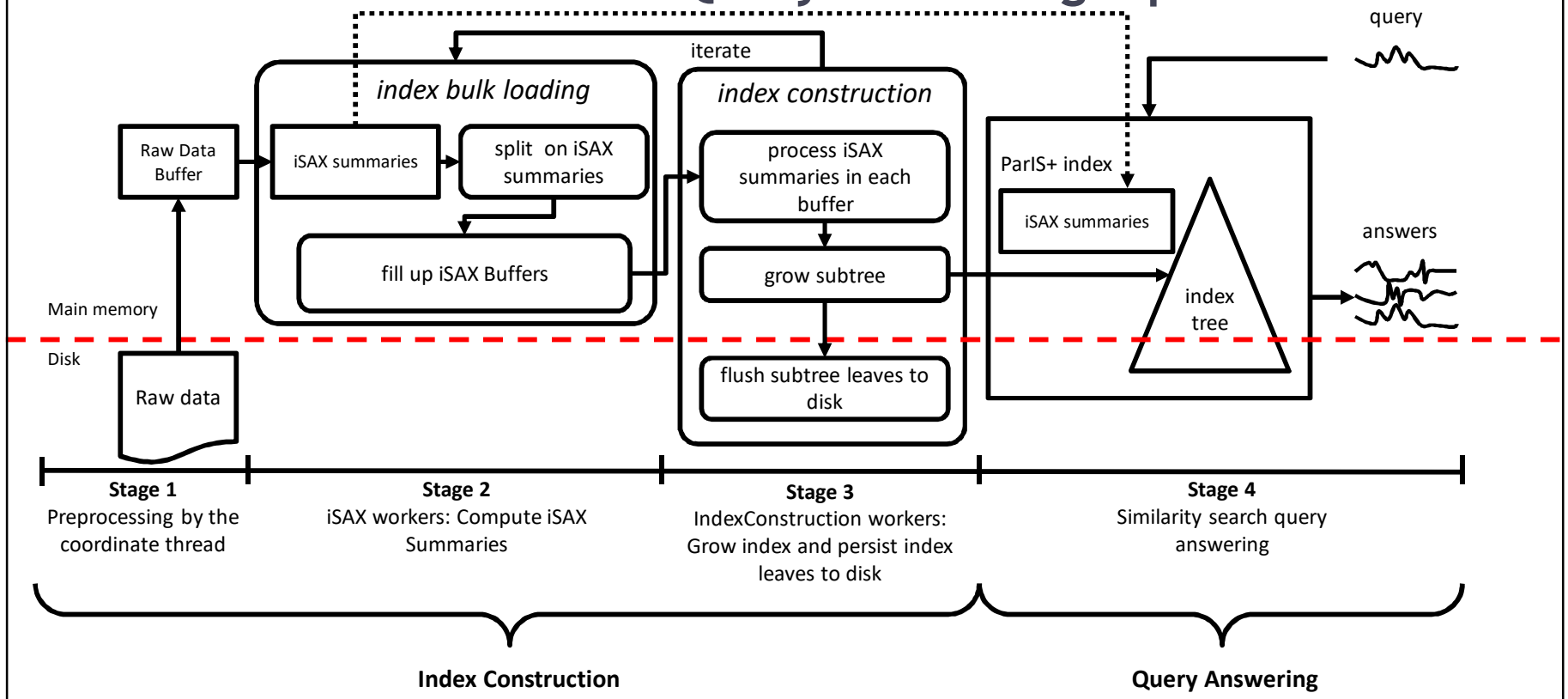
Main Challenges

- Have threads performing computation during I/O.
- Ensure locality in a) processing the data, b) creating the tree index, and c) traversing the index during query answering.
- Perform most of the computation in an embarrassingly parallel way to ensure good speedup.
- Reduce synchronization cost as much as possible during index construction and query answering.

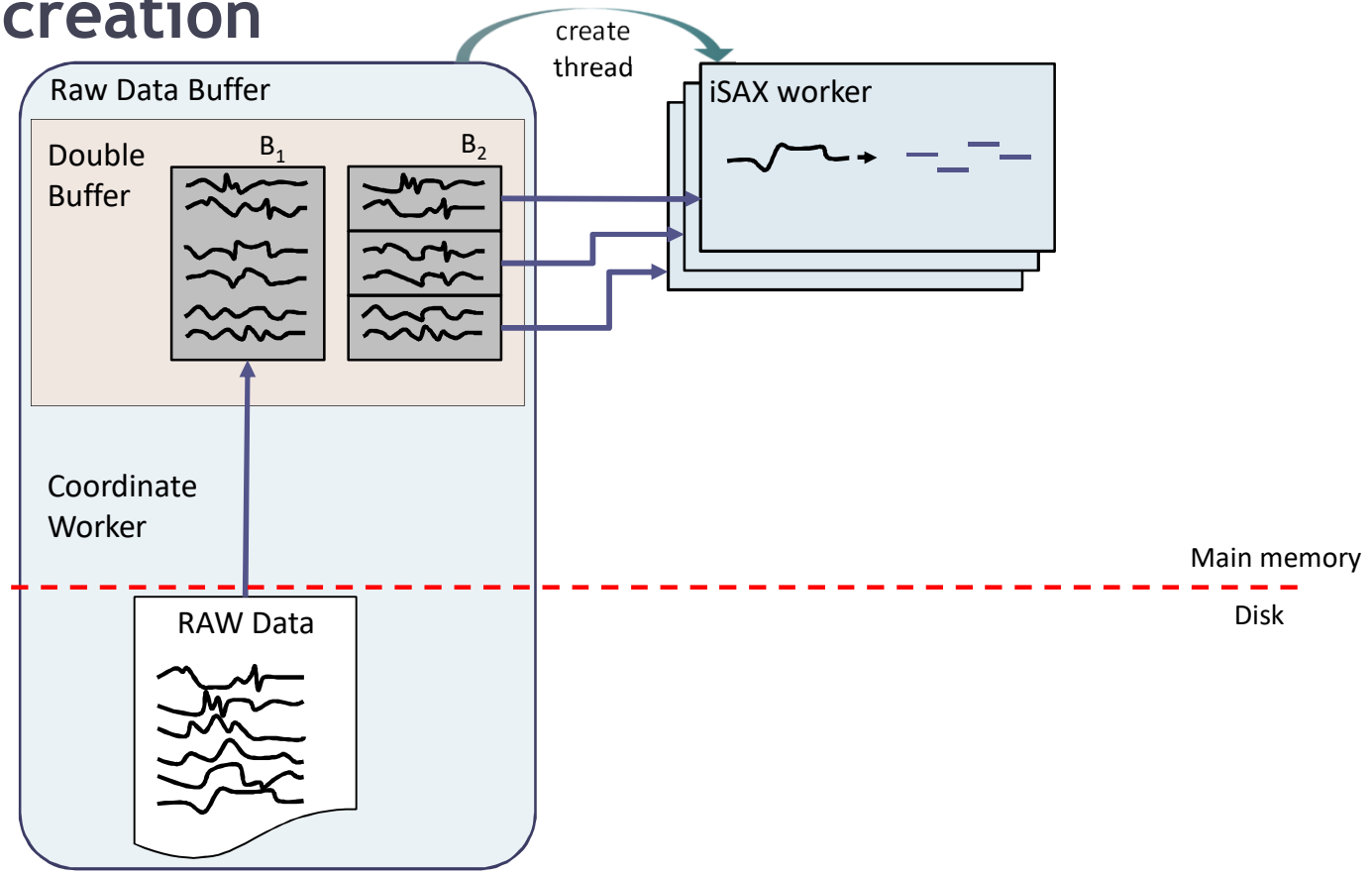
Achievements of ParIS+

- Completely masks the CPU latency during index creation
- Query answering up to 8x faster than previous approaches

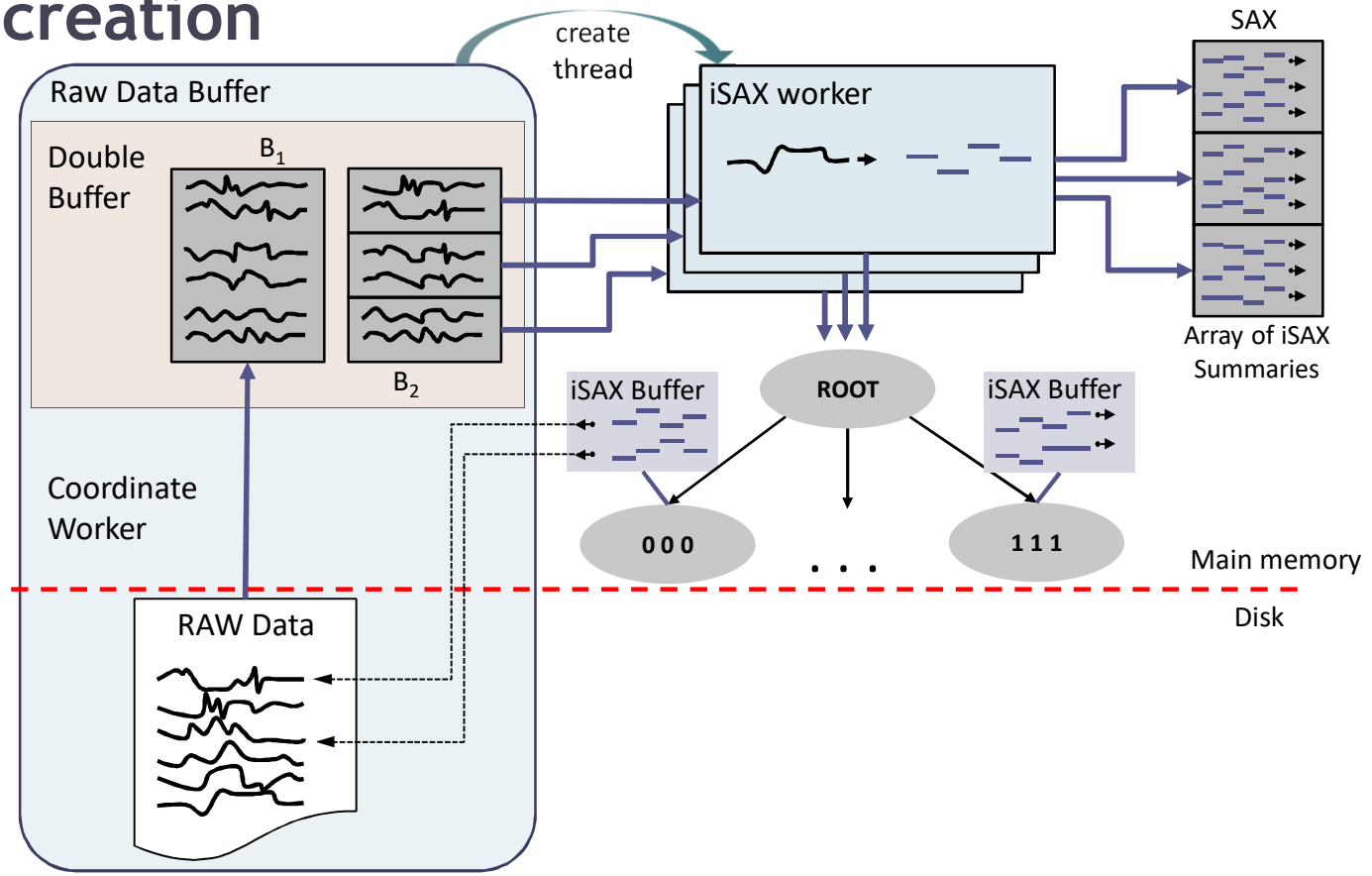
Index Creation and Query Answering Pipeline



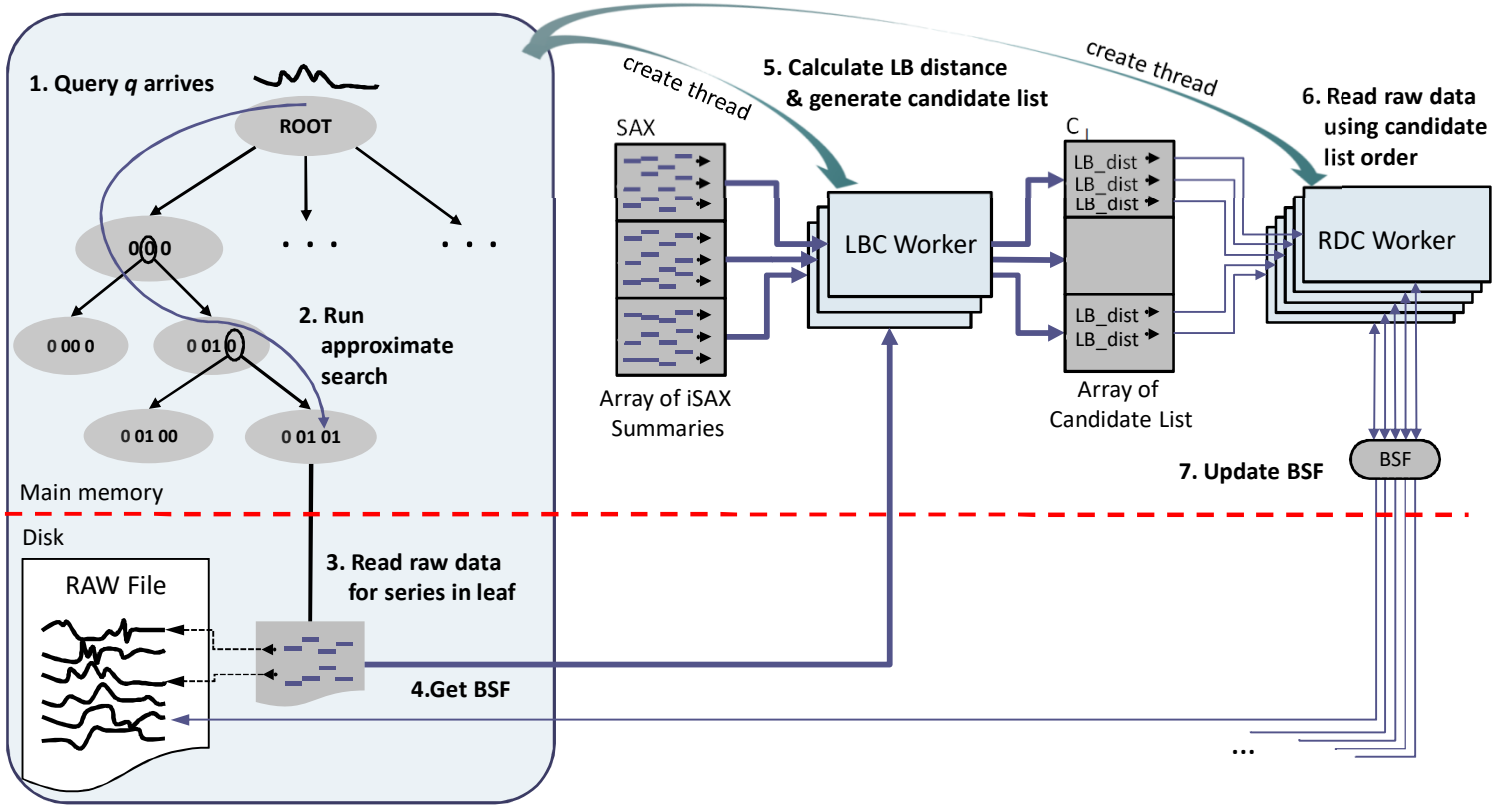
Index creation



Index creation



ParIS+ exact query answering



Experimental Setup

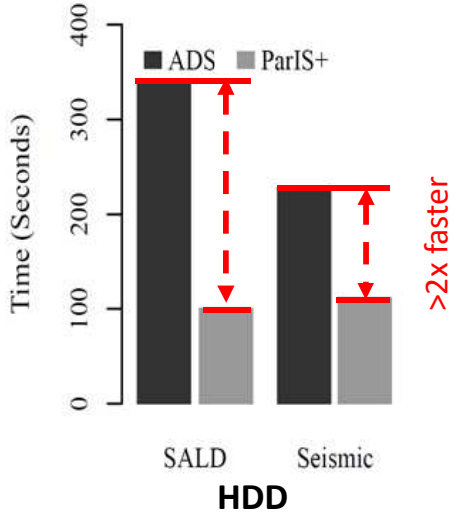
Machine: 2x Intel Xeon E5-2650 v4, 12 cores (24 hyper-threads)

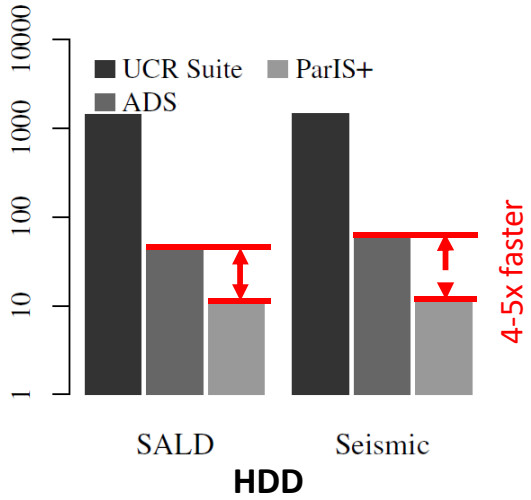
Datasets: Random Dataset (50GB-200GB; series length: 256)
SALD Dataset (100GB; series length: 128): neuroscience MRI
Seismic Dataset (100GB; series length: 256): seismic signals

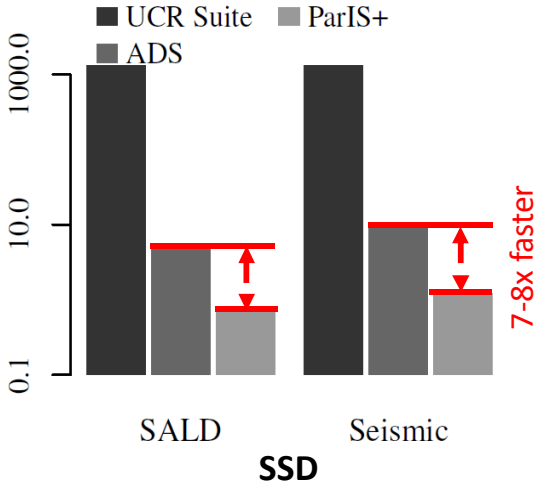
Algorithms: UCR Suite: serial scan method
ADS: disk-based index
ParIS+

Experiments

Time performance of indexing and exact query answering on real datasets







index creation: ParIS+ is >2x times faster than ADS

query answering HDD: ParIS+ is 4-5x faster than ADS

query answering SSD: ParIS+ is 7-8x faster than ADS

MESSI Index

B. Peng *(Chinese Academy of Sciences), P. Fatourou and* ***T. Palpanas*** *(Université Paris Cité)*

VLDB Journal 2021
IEEE International Conference on Data Engineering (ICDE) 2020

Motivation

Airbus stores PB of data series, describing the behavior over time of various aircraft components

- vibrations of bearings in engines
- way pilots maneuver the plane through fly-by-wire system.

Analytics algorithms often operate on data subsets, which fit in memory:

- data relevant to landings from pilots

Astrophysics and neuroscience

- different subsets of data need to be analyzed

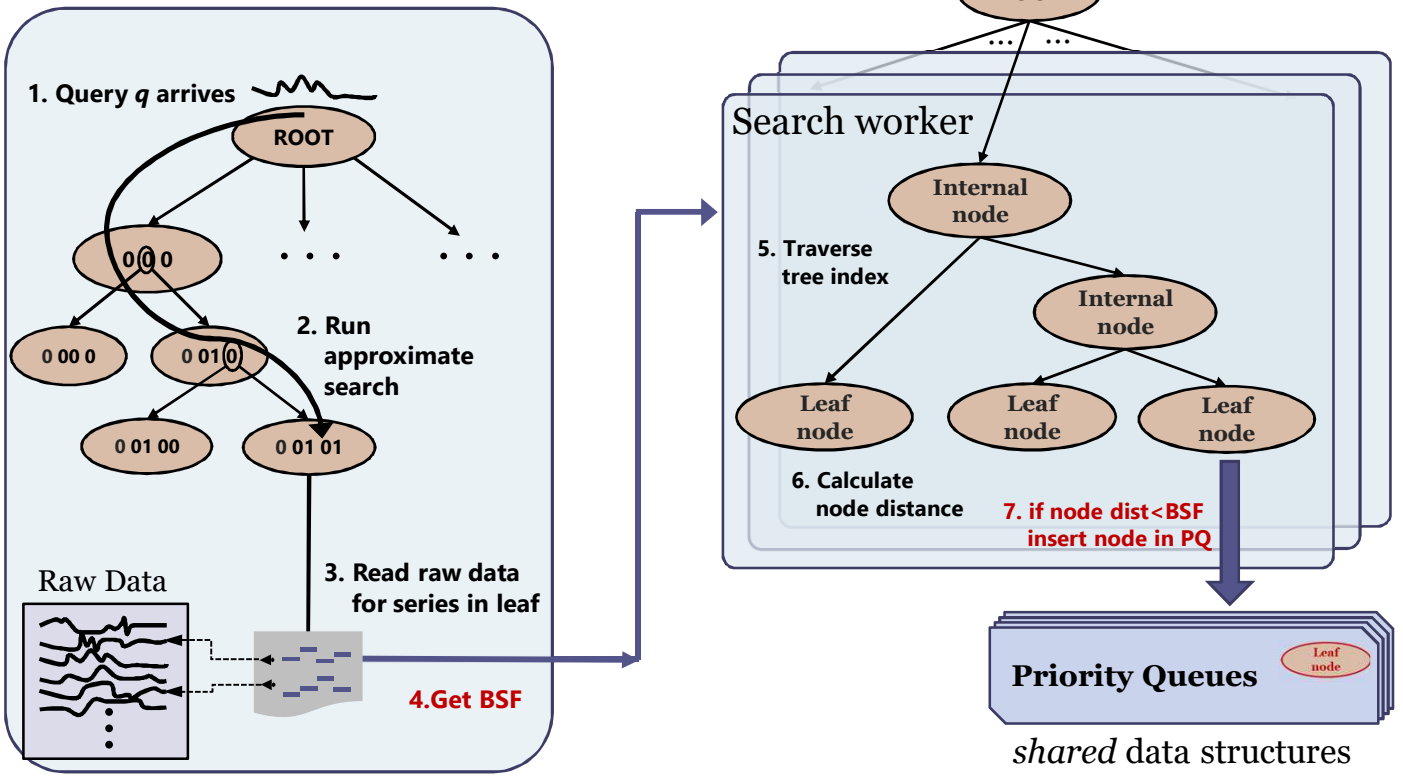
Main Challenges

- No CPU cost can be hidden under I/O.
  - More careful design choices and coordination of parallel workers.
  - More subtle design for index construction and new algorithms for answering queries.
- Adaptations of alternative solutions, which have proven to perform the best in other settings, were not optimal.
- More complex concurrency patterns in data structures.
  - Concurrent priority queues for storing data series that cannot be pruned, and for processing them in order.
- To achieve load balancing, we had to come up with a scheme where all priority queues had about the same number of elements.
  - We experimented with several designs for reducing the synchronization cost among different workers that access the priority queues and for achieving load balancing.

Achievements of MESSI

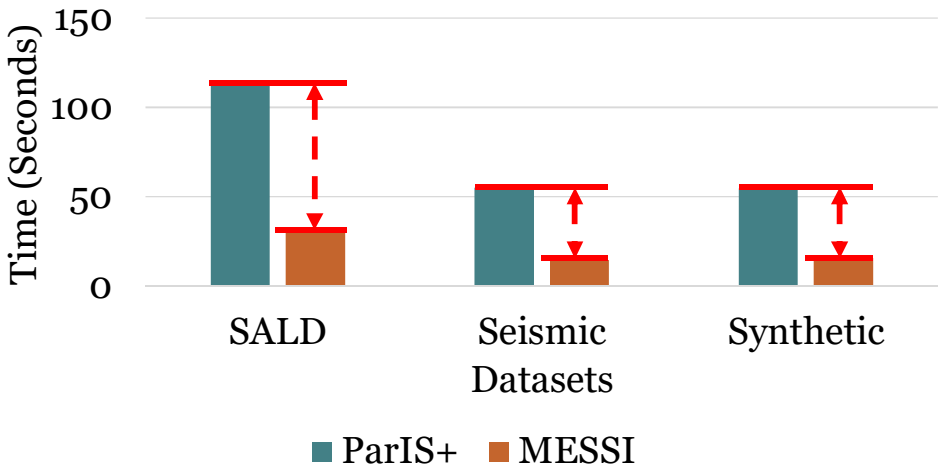
- Up to 4x faster in index construction time when compared to (in-memory) ParIS+
- Novel query answering scheme
  - ✓ up to 11x better performance than ParIS+.
  - ✓ Up to 100x faster than the state-of-the-art serial scan algorithm
- Similarity search at interactive speeds (50msec on 100GB dataset)

MESSI Query answering



Experiments

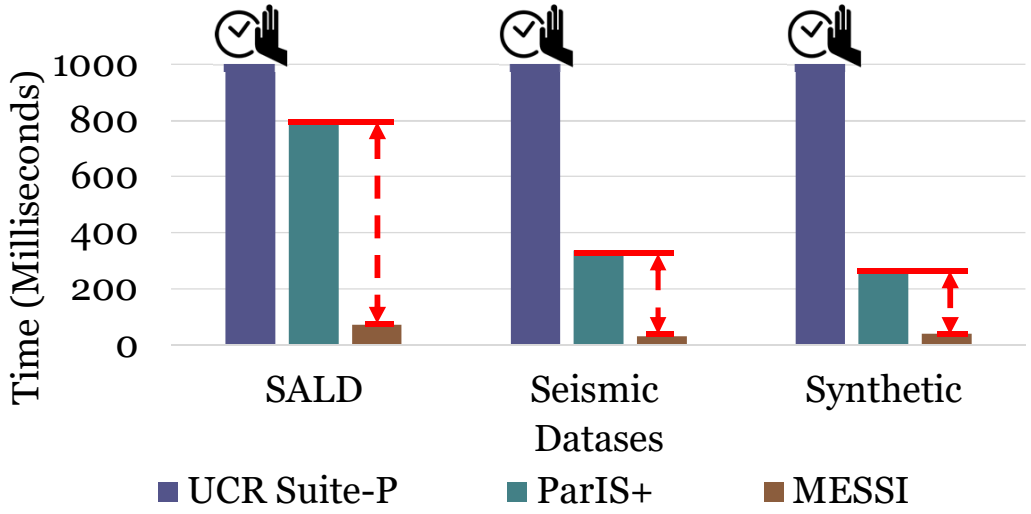
Time performance of index creation (100GB datasets)



MESSI up to 4x faster than **ParIS+ (in-memory)** for index creation

Experiments

Time performance of exact query answering



MESSI up to 11x faster than **ParIS+ (in-memory)** for query answering

SING Index

B. Peng *(Chinese Academy of Sciences), P. Fatourou and* ***T. Palpanas*** *(Université Paris Cité)*

IEEE International Conference on Data Engineering (ICDE) 2020

SING: Sequence Indexing Using GPUs

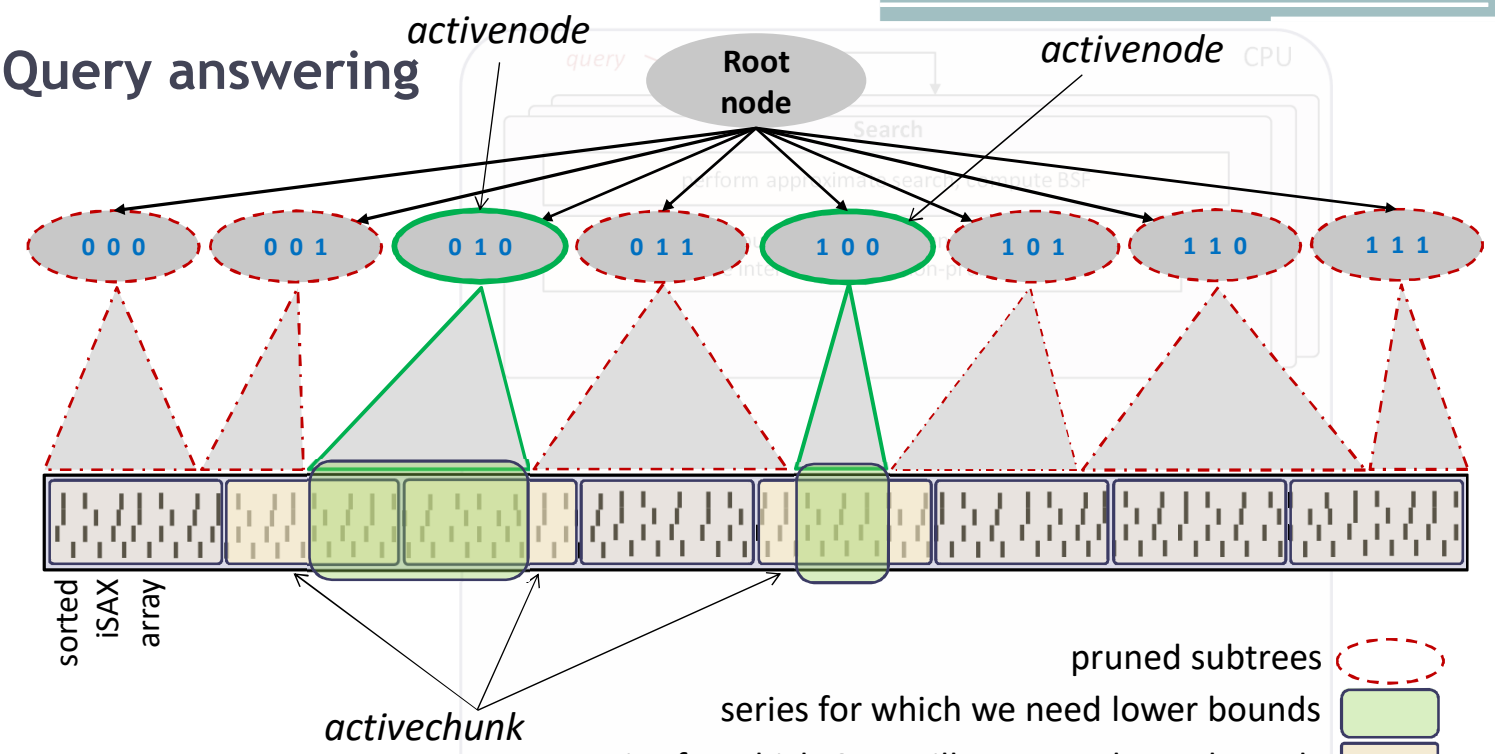
Challenges

- Limited GPU memory size (12GB of RAM)
  - much smaller than raw data
- Slow interconnect speeds (PCI-Express 3.0 x16 delivers 10GB/sec)
  - Moving raw data needed by individual queries is prohibitively expensive
  - Even moving small ad hoc subsets of data required by queries incurs a prohibitively high cost (0.4% of a 100 GB dataset requires > 40msec).
  - Raw data cannot be processed in the GPU.
- Non-sophisticated Streaming Processors (GPU cores)
  - Not suited for supporting complex data structures/branching required in tree-like indexing.

SING Main Ideas & Contributions

- Execute an in-order traversal of the root subtrees and store i-SAX summaries in an array in order
- Store this array in GPU memory
  - ✓ Serves large data series collections using limited GPU memory
  - ✓ Performs mostly consecutive in-memory accesses
- New pruning strategy that ignores entire root subtrees
  - ✓ Reduces the number of lower-bound distance computations that the GPU should execute.

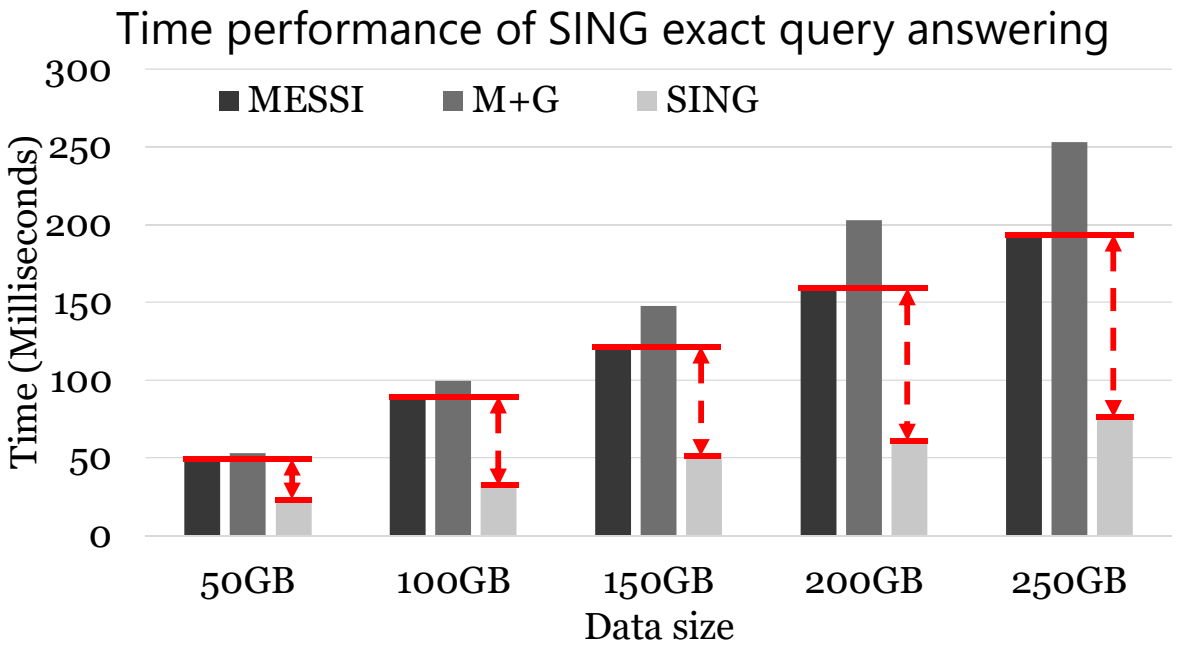
SING Query answering



SING Main Ideas & Contributions

- Novel way to compute lower bound distances
  - ✓ Avoids the use of a dictionary
  - ✓ Employ a simple polynomial function
- Effectively divide the workload among the GPU and CPU cores, and orchestrates their parallel execution.
  - ✓ CPU workers start processing candidate answers without waiting for the GPU computation to complete.
  - ✓ Split the work into chunks
  - ✓ GPU streams the result of the work on each chunk to the CPU threads
  - ✓ CPU completes the computation for corresponding data series

Experiments



2x faster than MESSI (~30msec on 100GB datasets)

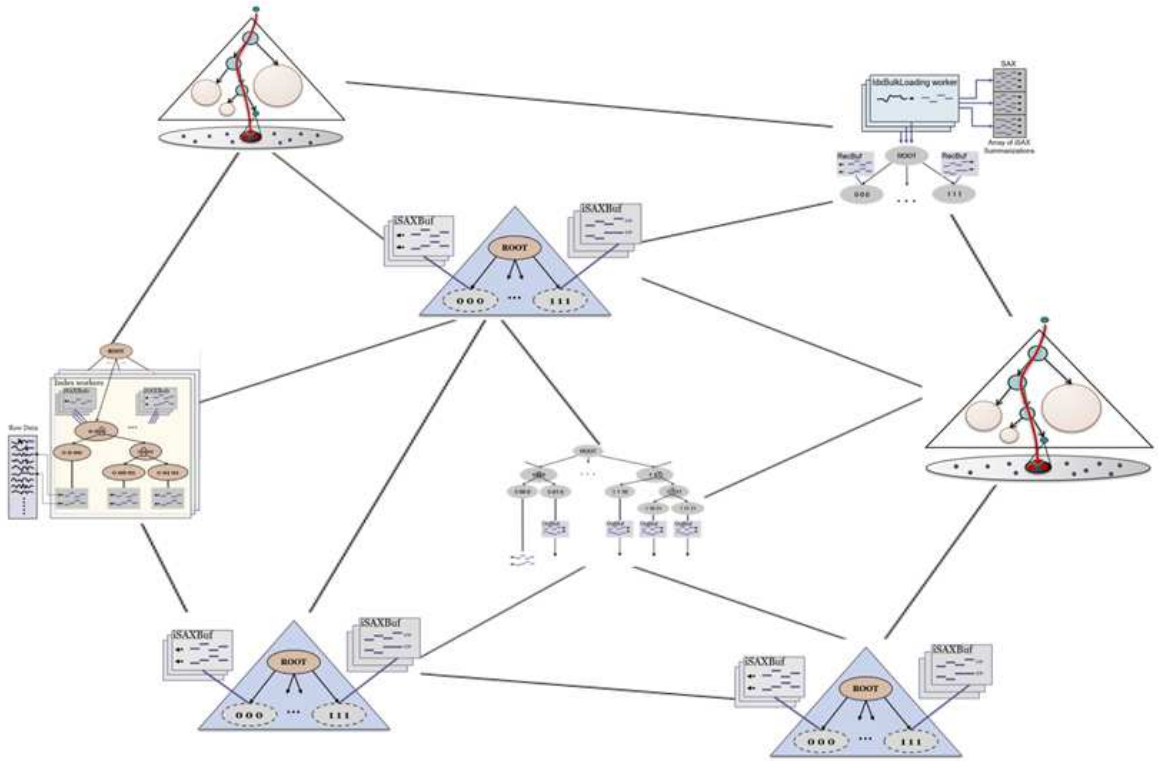
SING becomes increasingly faster than **MESSI** as dataset size grows

Odyssey: A Journey in the Land of Distributed Data Series Processing

M. Chatzakis, P. Fatourou, E. Kosmas, T. Palpanas, B. Peng

Proceedings of the VLDB Endowment, Vol. 16, No. 5, August 2023

Techniques and Methodology



An arbitrarily large batch of queries is provided in the system as input.

Goals

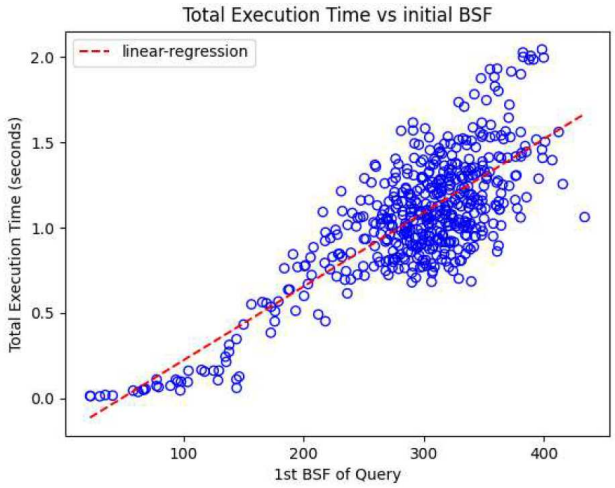
- Minimize makespan
- Scalability

Odyssey - Challenges

- Ensuring scalability.
  - Increasing the available hardware resources should decrease the time cost or enable to process additional data in a proportional way
- Query scheduling
  - come up with mechanisms for estimating the execution time of queries
- Load-balancing
  - we need to replicate data in order to make such a mechanism viable
  - moving big volumes of data series around is prohibitively expensive
- Fundamental trade-off between data scalability and good performance
- Maintain good parallelization properties for efficient execution in multi-core CPUs inside each system node
- Achieve high pruning power during query answering
- Data Partitioning

Our Contribution - Odyssey: A Distributed Data Series Indexing & Processing Framework

- Odyssey is the first solution that exploits parallelization both inside and across system nodes.
- **Scheduling:** We develop an algorithm for assigning queries to nodes, which tries to balance workload across nodes by computing an estimation of the execution time of each query.
  - We performed a query analysis that showed that similarity search queries, for which the initial BSF is high, tend to also have high execution times.

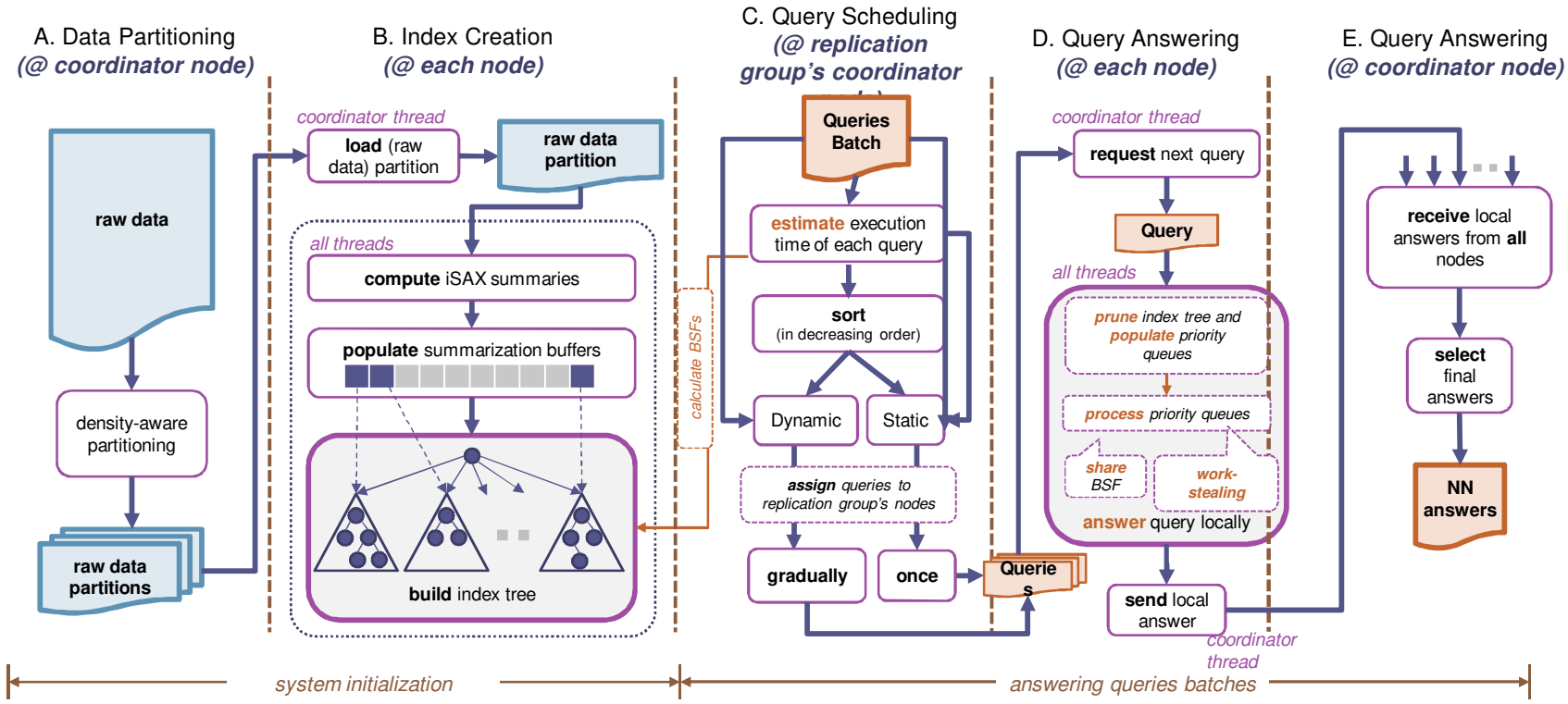


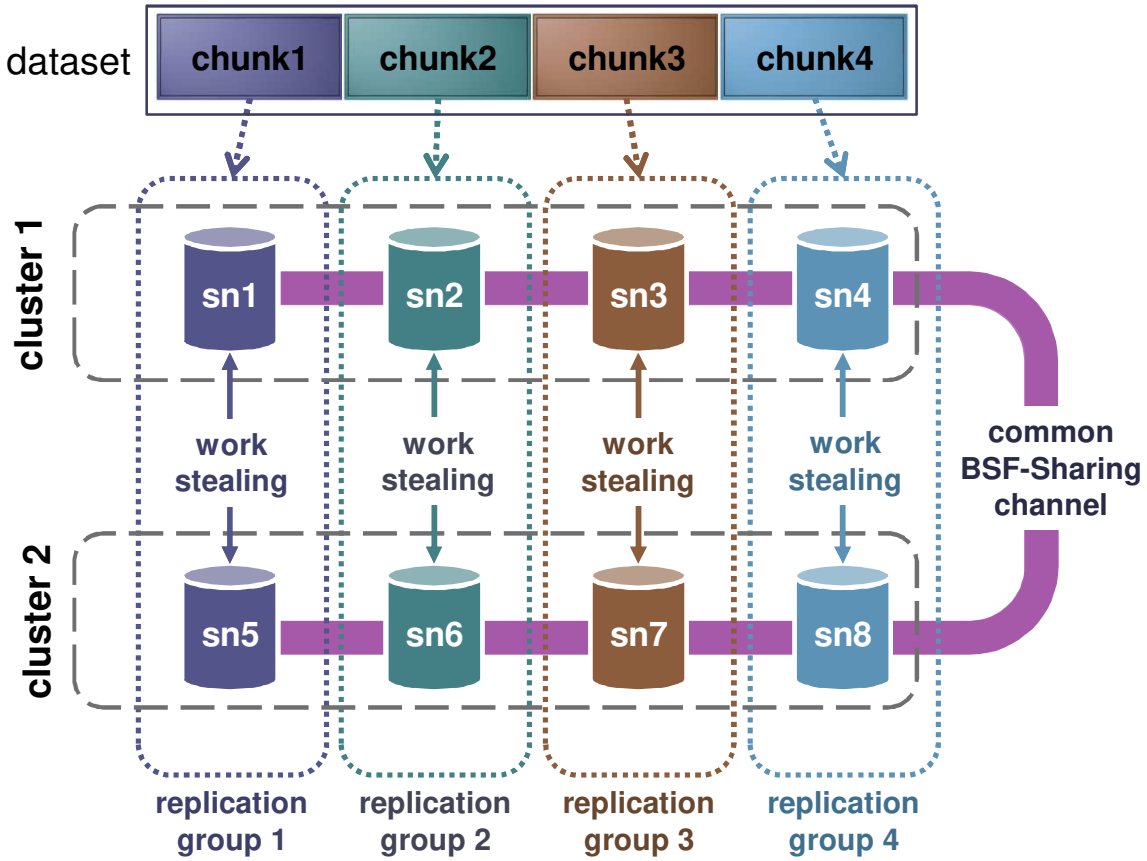
Linear regression for Seismic queries prediction

Our Contribution - Odyssey: A Distributed Data Series Indexing & Processing Framework

- **Work-Stealing:** We present an exact search algorithm that supports work-stealing between nodes that share (parts of) the index.
- **Flexible replication:** Odyssey supports different replication degrees among the nodes, allowing users to navigate the entire spectrum of solutions, trading space for speed.
- **Partitioning:** We propose a density-aware data partitioning method that can efficiently partition data in a way that improves work balancing.
- **Scalability and Performance:** Experiments show that Odyssey exhibits an almost linear scale-up, and up to 6.6x times faster exact query answering times than competitors.

Techniques and Methodology




dataset
chunk1
chunk2
chunk3
chunk4
cluster 1
cluster 2
sn1
sn2
sn3
sn4
sn5
sn6
sn7
sn8
work stealing
work stealing
work stealing
work stealing
common BSF-Sharing channel
replication group 1
replication group 2
replication group 3
replication group 4

Data Series Indexing: Conclusions

- **ParIS+:** parallel on-disk data series index
  - Similarity search up to 15x faster than serial scan (10sec on 100GB dataset)
- **MESSI**: parallel in-memory data series index
  - Similarity search at interactive speeds (50msec on 100GB dataset)
- **SING**: parallel in-memory data series index, accelerated by utilizing GPUs
  - Expands scalability of exact similarity search
- **ODYSSEY**: distributed data series processing
  - almost linear scale-up, and up to 6.6x times faster exact query answering times than competitors

Current & Future Research Directions of Interest

- First **Lock-Free** Index and Exact Similarity Search Algorithm

  [P. Fatourou, E. Kosmas, Th. Palpanas, G. Paterakis: FreSh: A Lock-Free Data Series Index, SRDS 2023]

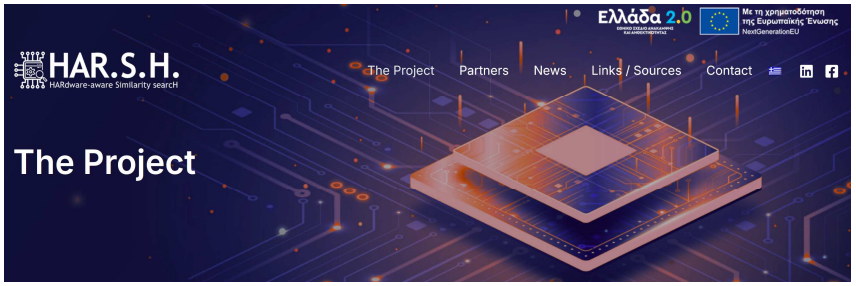
- New Concurrent Priority Queues optimized for best performance in the data series setting

  [O. Grimes, A. Hassan, P. Fatourou, R. Palmieri, PIPQ: Strict Insert-Optimized Concurrent Priority Queue, DISC 2025]

Directions for further Research

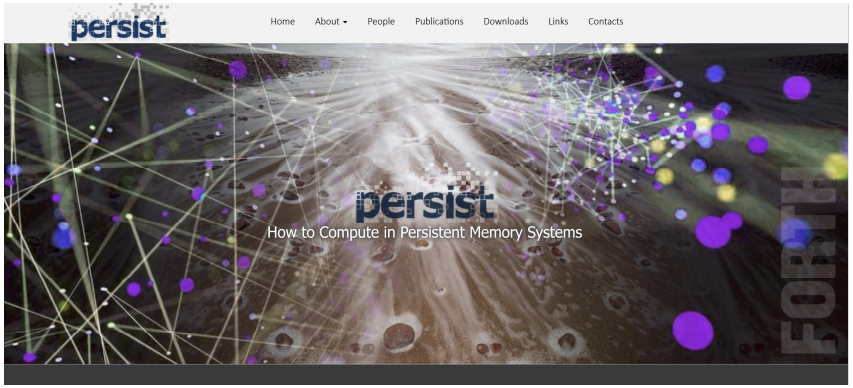
- Copying with dynamic data (batches of updates, streaming)
- Hardware-aware, hardware-supported and hardware-accelerated data series processing
- Supporting more complex or other types of queries (e.g. predicates, approximate queries, subsequence queries, etc.)
- Copying with data series of variable size

Thank you!



https://harsh-project.gr/

https://persist-project.gr/



faturu@csd.uoc.gr
www.ics.forth.gr/~faturu/