UNIVERSITY OF TRENTO - Italy

Information Engineering and Computer Science Department

Time-Series Similarity Query Answering Using iSAX on Map Reduce

March 12, 2014

Ermias Andargie Walelgne

Advisor:

Prof. Themis Palpanas

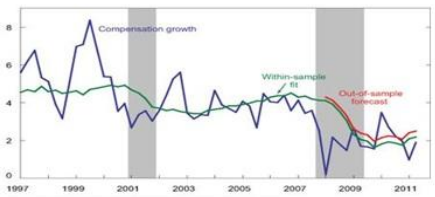
Time Series Data

- An organized discrete sequence of n real number values.

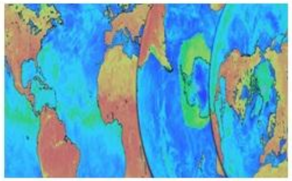
$$T = (t_1,\ldots,t_n), t_i \in R$$

- A contiguous recorded result of a certain phenomena, a measurement or, an observation collected at constant time interval.
- Applicable in various areas:

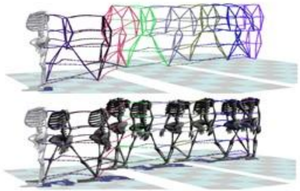
Finance and Economic



Hydrology



Motion detector



Anomaly Detection

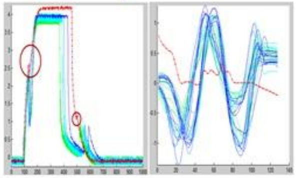
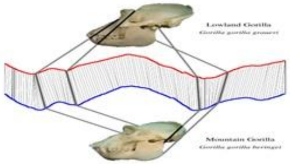
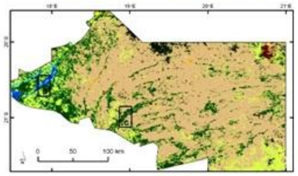


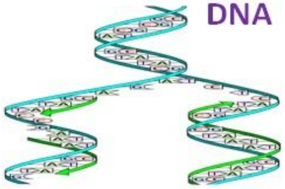
Image Processing



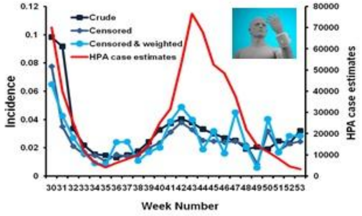
Remote sensor



DNA

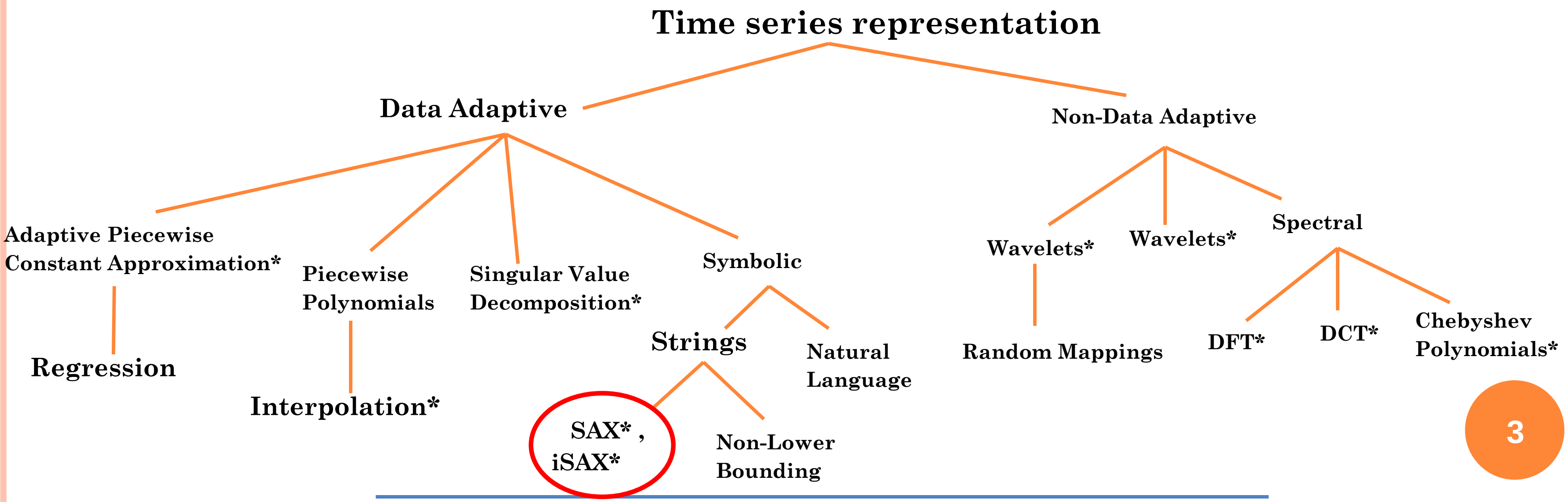


Medical Surveillance



Time series data representation and indexing

- Since time series data is high dimensional
- Processing the 'raw' time series requires:
  - High computation time and large memory space
- Dimensionality reduction and indexing methods



Time series representation

- Data Adaptive
  - Adaptive Piecewise Constant Approximation*
    - Regression
  - Piecewise Polynomials
    - Interpolation*
  - Singular Value Decomposition*
  - Symbolic
    - Strings
      - SAX* , iSAX*
      - Non-Lower Bounding
    - Natural Language
- Non-Data Adaptive
  - Wavelets*
    - Random Mappings
  - Wavelets*
  - Spectral
    - DFT*
    - DCT*
    - Chebyshev Polynomials*

- **Motivation**
  - Indexing is a key for processing time series, but query answering even with an index can be slow
  - iSAX and its extension iSAX 2.0 indexes really massive dataset
  - MapReduce is a software framework for processing large dataset

- **Question:**
  - How we can answer iSAX based quires faster in distributive environment using MapReduce

Problem statement

- iSAX exact search is an expensive search - required an intensive computation and I/O cost
- Query answering takes more time as index size increases

Objective :

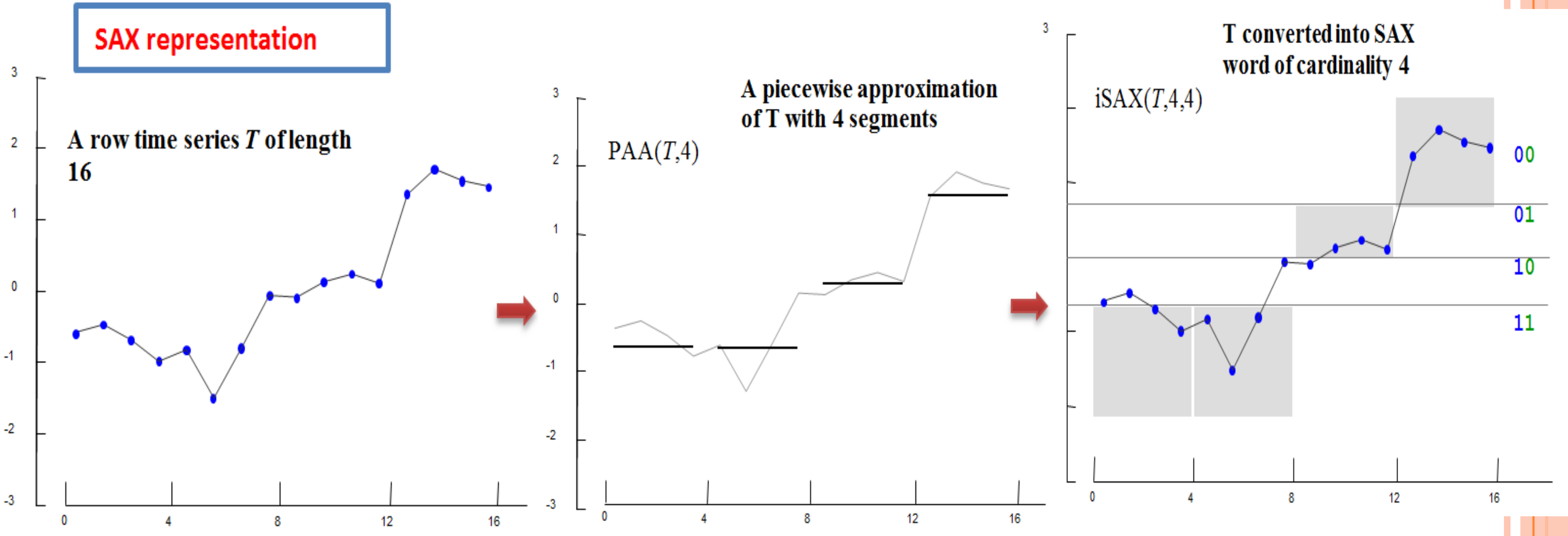
- Implement iSAX exact search in MapReduce
- Improve lower bound distance calculation method
- Analysis the execution time of simple, MapReduce, and KNN search
  - Using synthetic and real dataset
- Optimizing MapReduce query answering using other frameworks such as Memcached.

PRESENTATION OUTLINE

- Time series representation and indexing method
  - iSAX
- Introduction to MapReduce
- Implementation and methods
  - MapRedExactSearch
  - MaxCardMapRedExactSearch
  - K-Nearest Neighbor Search
- Experimental Results
- Summary and future directions
- Questions

SAX: **S**ymbolic **A**ggregate appro**X**imation

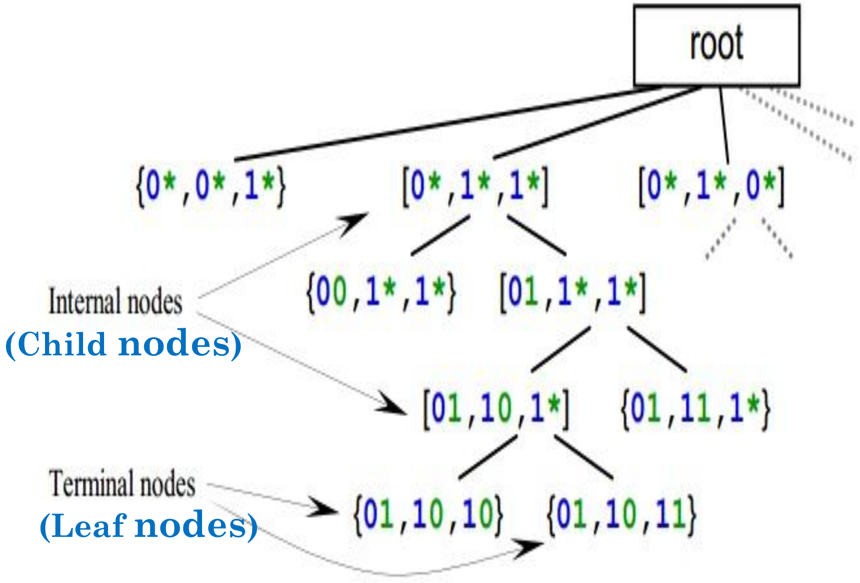
- Represent a time series *T* of length *n* into *w* segments

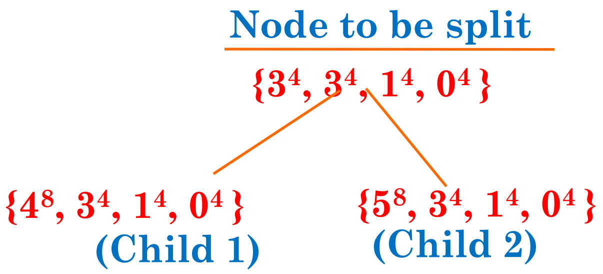


- iSAX word : **T^4 =*iSAX(T, 4, 4)* = {11,11,01,00}={3^4, 3^4, 1^4, 0^4 }**
- Comparing two iSAX words of different cardinality
- Comparing different cardinalities within a single word

iSAX indexing

- Hierarchical tree structure.
- **Leaf node** points to index file on the disk.
  - Example: iSAX word $\{7^8, 5^8, 3^8, 1^8\}$ can be mapped to 7.8_5.8_3.8_1.8.txt
- **Terminal nodes:** created when the number of time series in a leaf node exceeds leaf size




Node to be split

$\{3^4, 3^4, 1^4, 0^4\}$

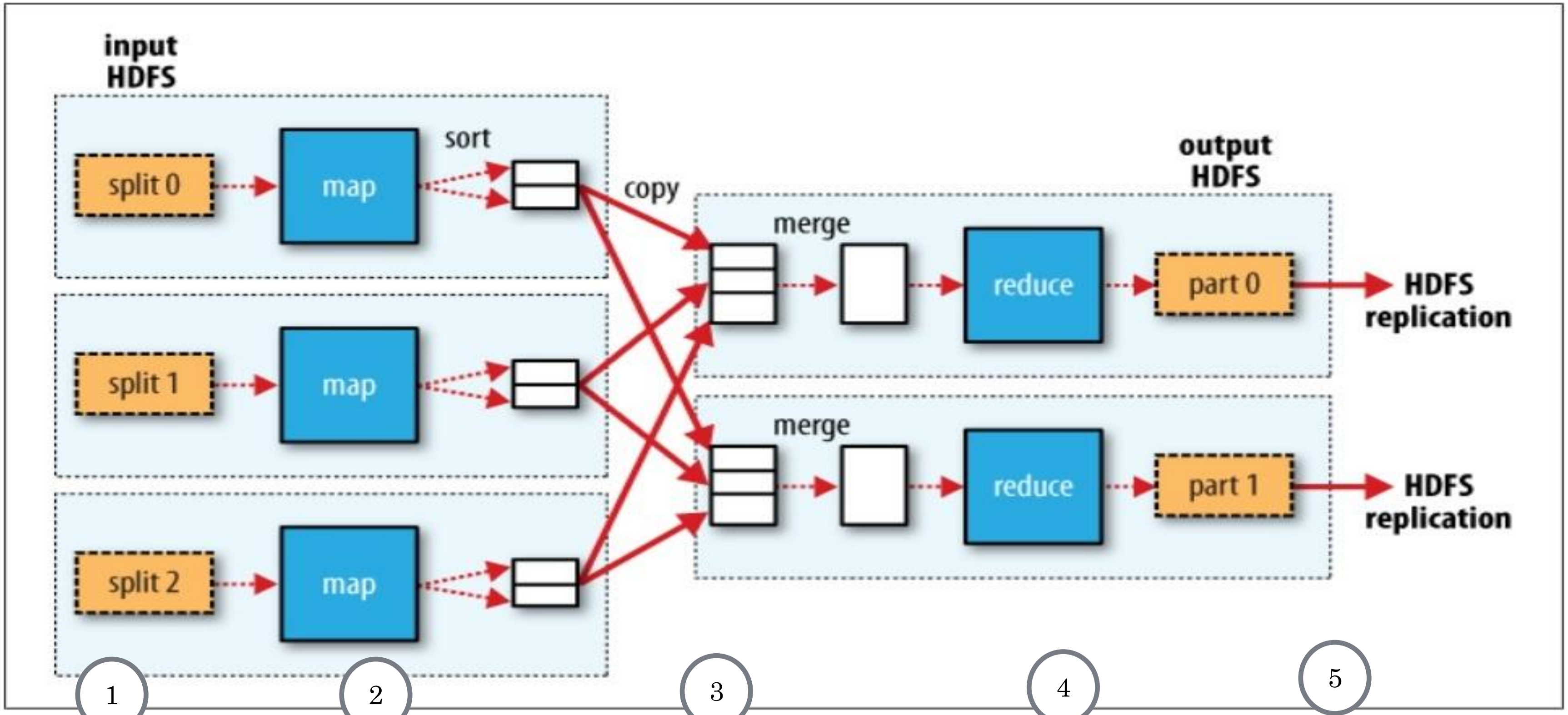
$\{4^8, 3^4, 1^4, 0^4\}$ (Child 1)

$\{5^8, 3^4, 1^4, 0^4\}$ (Child 2)

For given (c = cardinality, w = word length) => c^w different iSAX words.

Leaf size (threshold): maximum time series a word (leaf node) can hold

MapReduce



1 Read input file and split it up across separate Map node

2 Map function runs and produce output for each map

3 Intermediate key-value pairs, which then input to Reducer

4 Reduce function generates an output for each Reduce node

5 Reads and aggregates the outputs from each node.

Exact search using MapReduce

- **Begin**: before starting the map tasks
  - Combined input file using CombineFileInputFormat
  - Each combined file is processed by different map task.
- **Mapper**:
  - Read all the time series from the combined file
  - Compute the minimum distance for each time series

Before Reducer

- Grouped Mapper output based on filename
- Ordered based on the distance.

- **Reducer:** processes each group separately.
  - Output the minimum from each group.
- **Result**:
  - Minimum distance of of all entries from the query.

Exact search using Maximum cardinality

- Lower bound distance:
  - Symbolic distance $\leq$actual distance
  - Used to prune the search space
  - Computed using the highest cardinality (8 bit)
- Used two separate Sequence files in HDFS
  1. Leaf Node with iSAX representation of each time series
  2. Leaf Node with raw time series

K - nearest neighbor search (KNN)

First MapRed Job output

KNNMapRedJob1

Outputs **k** closest entries from the query for each index file

KNNMapRedJob2

- **Mapper**:
  - Reads the records outputted by the first job
  - Changes the key of all the records to the same key

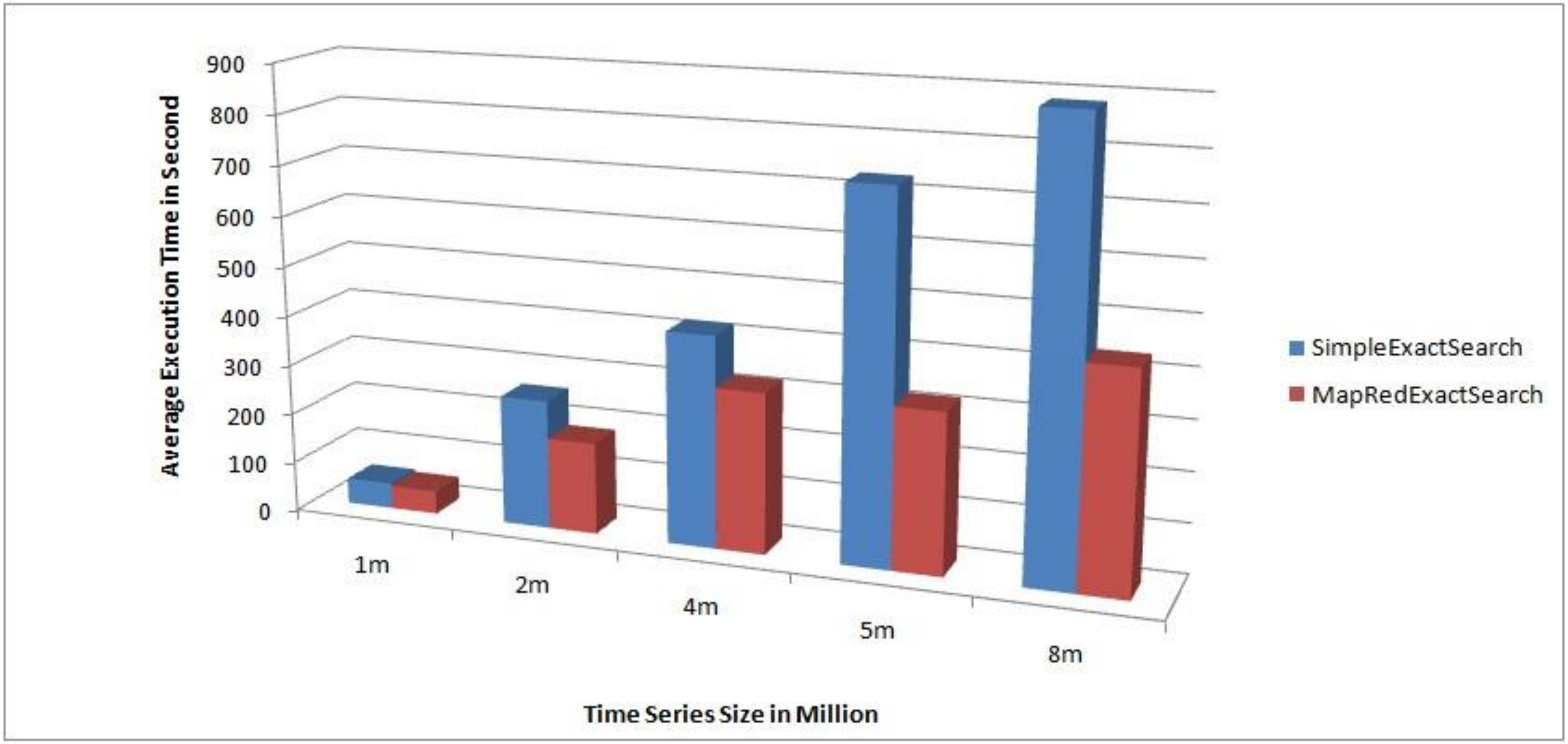
Before Reducer: Group all the out put together
Ordered according to the distance.

- **Reducer**:
  - Output the k first records.
- **Finally:**
  - Update the existing closest neighbor's list with k smallest distance

Experimental setup

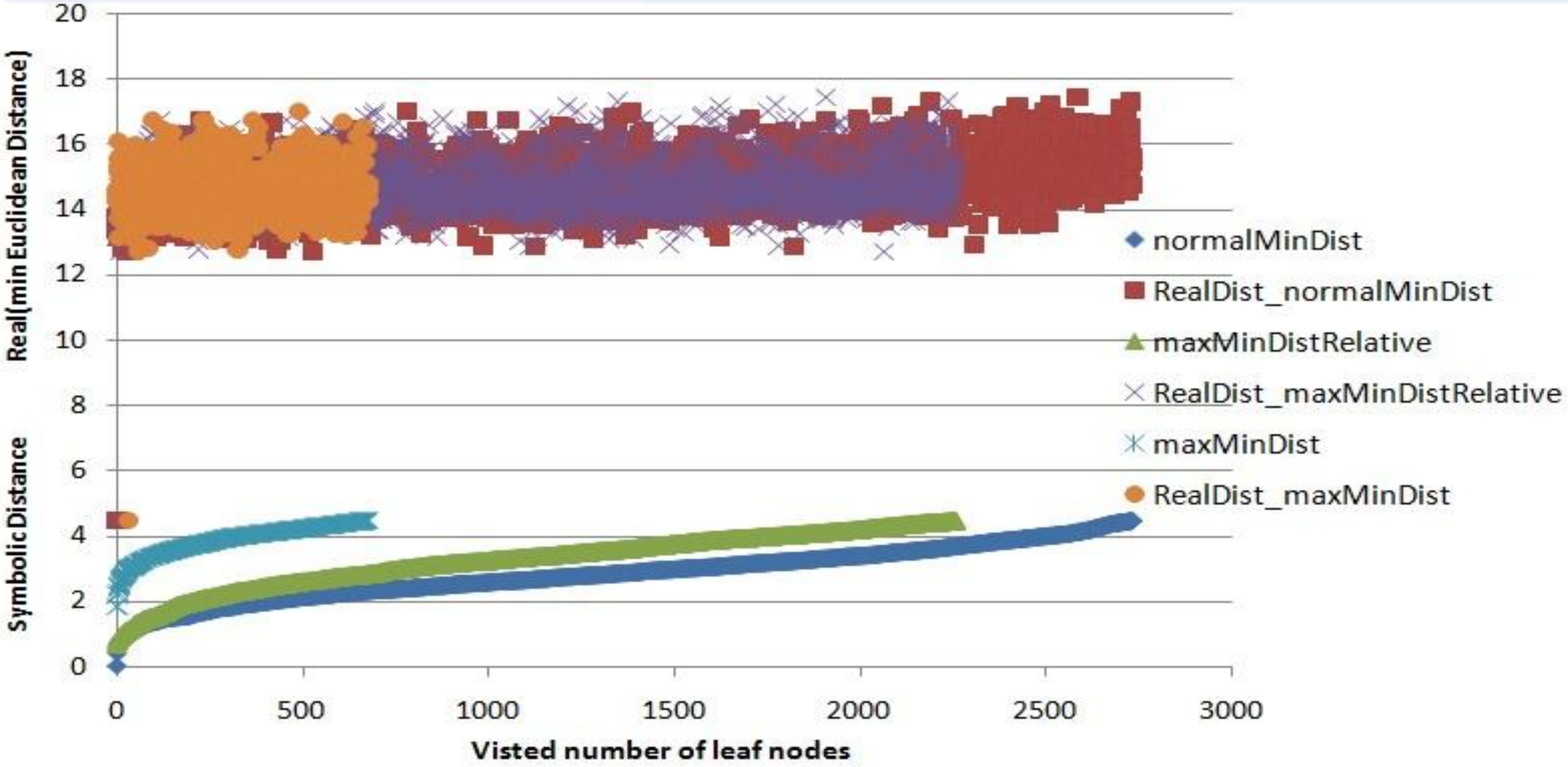
- Experiments are conducted on single node pseudo distributed mode
- Configured:
  - On Intel 64 bit Core i5-2430M CPU @ 2.4GHz
  - Memory size 4GB and Ubuntu 12.04 LTS
  - Hadoop version 1.0.3
  - Java as Programming language
- Data sets:
  - Randomly generated with length 128,
    - Base cardinality=2, Word length=8, leaf size = 100,1000,&10000
    - Time series size1, 2, 4, 5 and 8 million
  - Homo.sapiens.NCBI36 42 DNA chromosome 5 and 11
- Results are averages over 5 runs for each query
- Average execution time measured in seconds

MapRedExactSearch compared to simple exact search



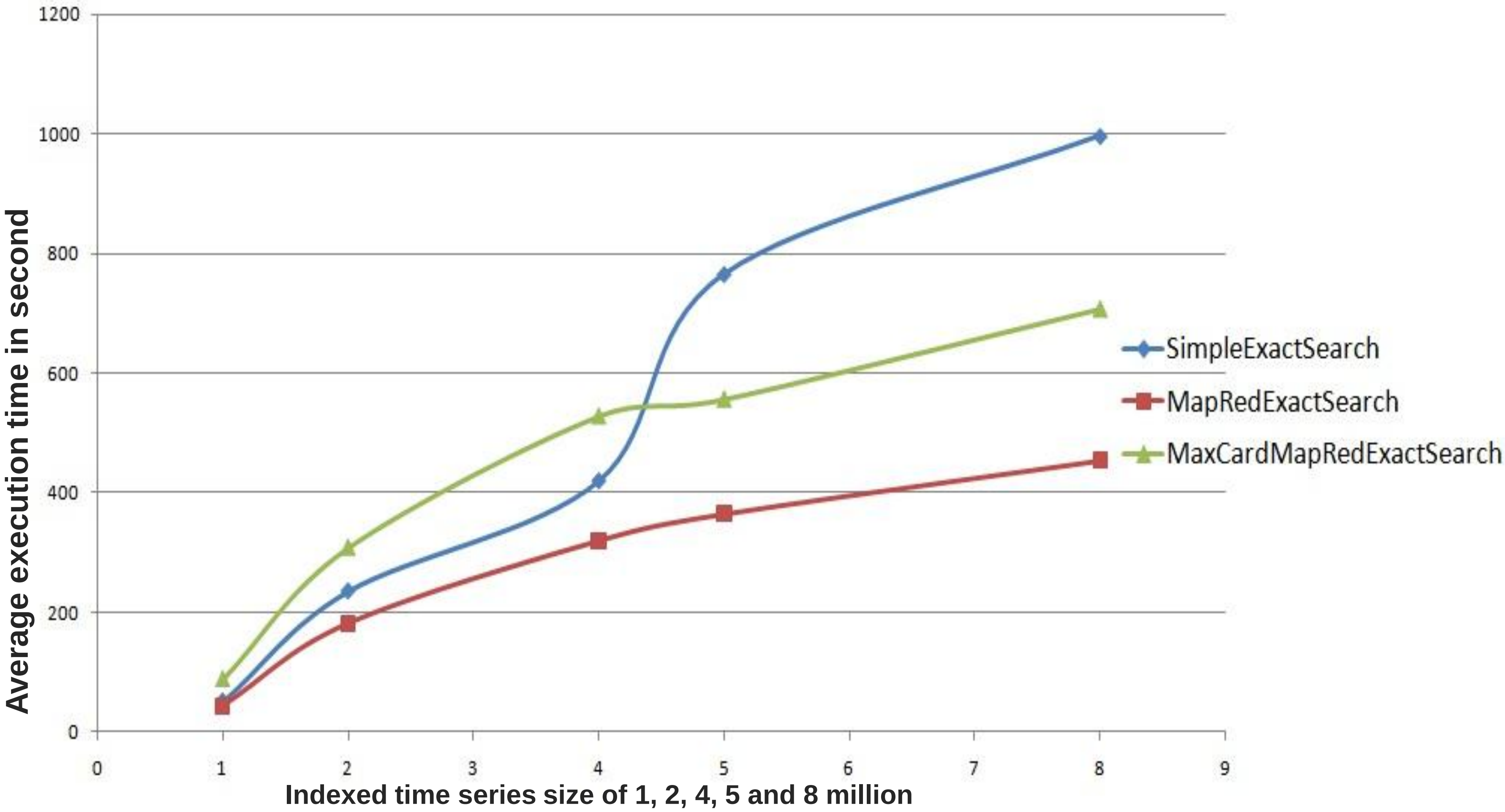
- Figure: Different average execution time of Simple and MapReduce implementation for indexed size 1m, 2m, 4m, 5m and 8m time series leaf size=10000

Maximum cardinality and lower bound



- Bound is calculated using
  - Maximum Cardinality=256
  - Maximum Cardinality selected local to each leaf node, and
  - Symbolic representation of the leaf node

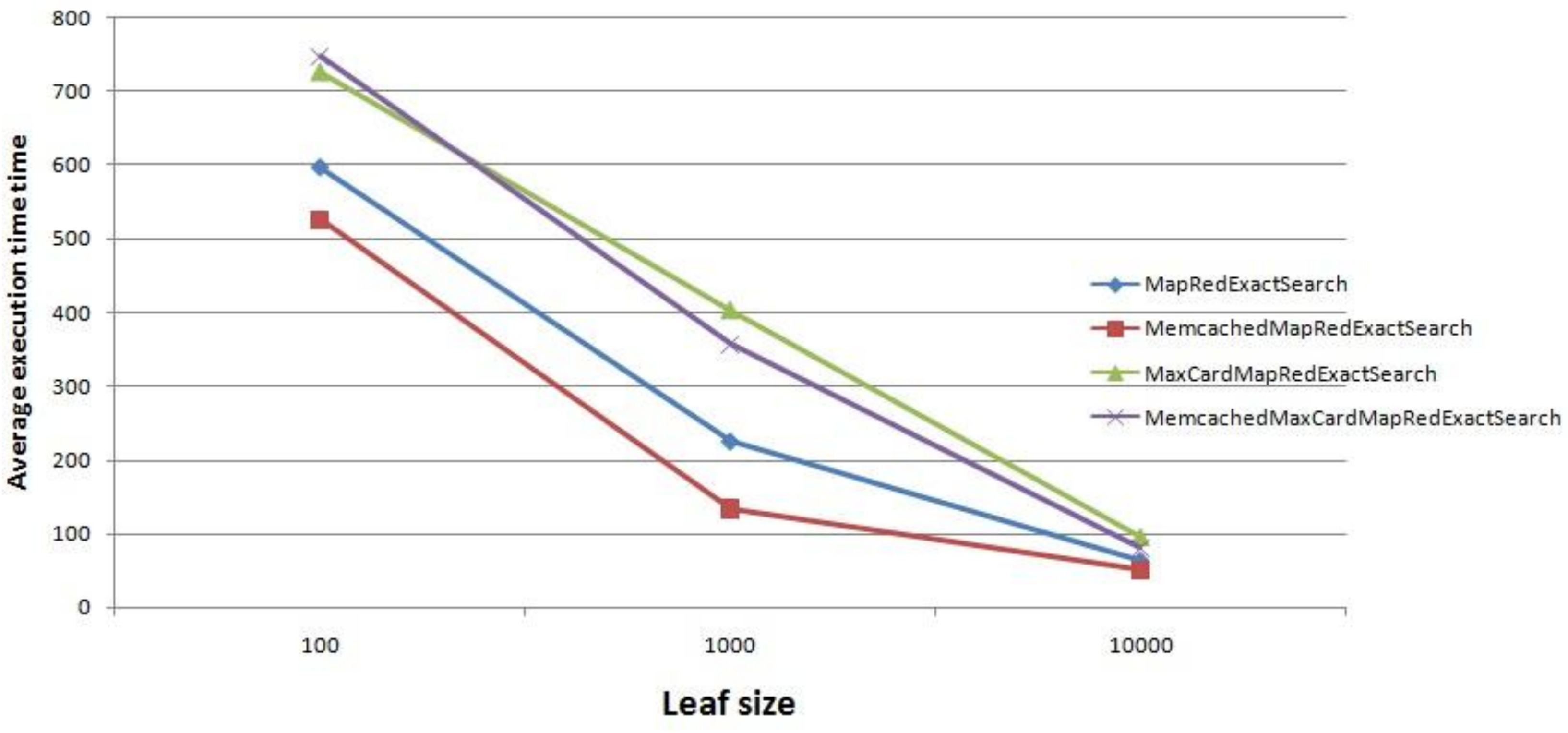
Simple ,MapRedExactSearch, and MaxCardMapRedExactSearch



- Average execution time of searching queries in seconds tested for
  - **Simple search**,
  - **MapReduce with out using Maximum cardinality** and
  - **MapReduce using Maximum cardinality**

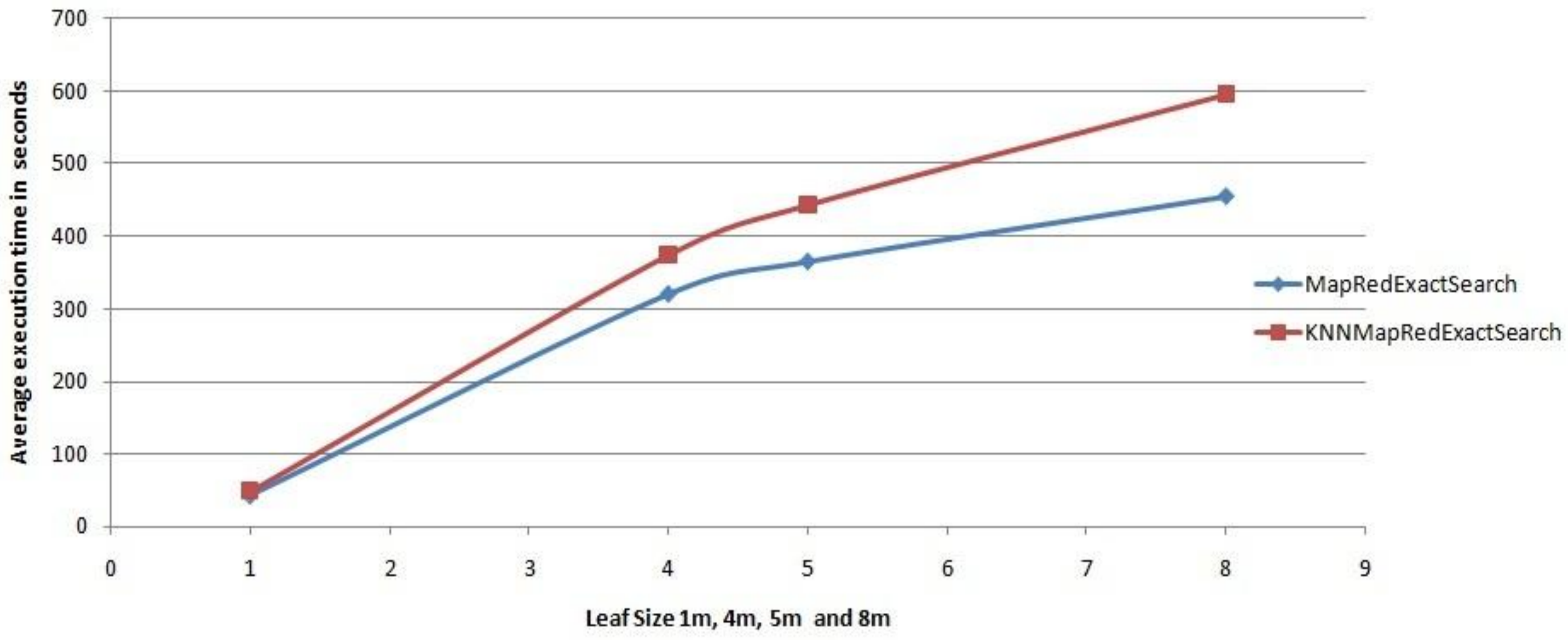
Substituting the Reducer with Memcached option

- **Memcached** : distributed memory-based object catching system
- Often used to hold small objects in RAM for fast possessing



- Average execution time of 1million time series under leaf size 100,1000 and 10000 using **memcached**

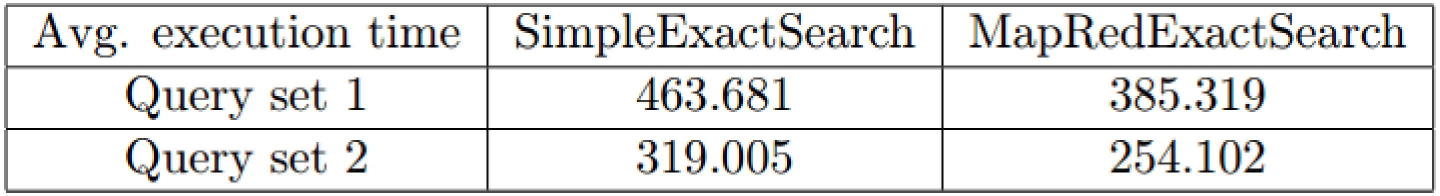
K –Nearest Neighbor compared with MapRedExactSearch



- Average execution time of K-NN and MapRedExactSearch for time series size 1,4,5 and 8 million, where **k=4**

Genome data

- Each genome data converted into time series and indexed with base cardinality =2, word length=8, time series length=128, leaf size =10000
- Generated 5 different queries by randomly changing two DNA symbols of the sequence



Avg. execution time	SimpleExactSearch	MapRedExactSearch
Query set 1	463.681	385.319
Query set 2	319.005	254.102

Summary

- On this thesis:
  - MapReduce to answer iSAX time series query with small average execution time than simple search
  - Highest cardinality for computing lower bound minimize the number of leafs visited
    - Has computation cost
  - MapReduce implementation using highest cardinality get advantage over simple search for large dataset
  - MapRedExactSearch algorithm has very fast execution time than the other approaches
  - Applicable for K-Nearest Neighbor search

Future directions

- Running all algorithms using real cluster of multiple nodes.
- More research on memcached implementation
- Carful consideration of MapReduce job configuration is crucial.
    - Example: On the split size and number of Reducers
- Supporting with other distributive frameworks such as ActiveMQ

Thank you!

Any Questions ?