Storage for Heterogeneous Data Stream

Rohan Karwa Amit Karyekar Rohit Joshi Drumil Jaswani

University of California
Los Angeles

CS219 Project,
December 2013

Table of contents

1 Introduction
- Heterogeneous Data Streams
- Problem Statement

2 Data Model
- Time Series Representation
- Symbolic Aggregation approXimation (SAX)

3 System Design
- Design Decisions
- Key Changes to BigTable
- Ideas borrowed from Azure
- Design

4 Basic Operations
- Read
- Write

5 Other functionality
- Compaction
- Old Data
- Failure Recovery

6 Experimental Analysis
- Query

7 Synchronization among Heterogeneous Data Streams
- Need for Synchronization
- Improvement

Data Streams

- What is Data Stream?
- Chronological Data
- Immutable Data
- Unbounded/Infinite Data
- Storage consideration Scalable
- Retrieving/Querying on Stored Data

Heterogeneous Data Streams

- Data Stream from different sources
- Different incoming rate
- Query Pattern Finding
- Storage consideration

Definition

- Designing a System for Efficient and Scalable Storage of Heterogeneous Data Stream
- Enable range data queries
- Support pattern finding

Symbolic Representation

Traditional Symbolic Representation Problem

- Paper [1] claims that none of the older techniques provides, lower bound guarantees

Contribution of the paper[1]

- Convert Time Series to Symbols with lower bound guarantee.

Lower Bound

Given

- Database represented by Symbols {a, b, c, d}
- Query Q which is similar to {b, c}

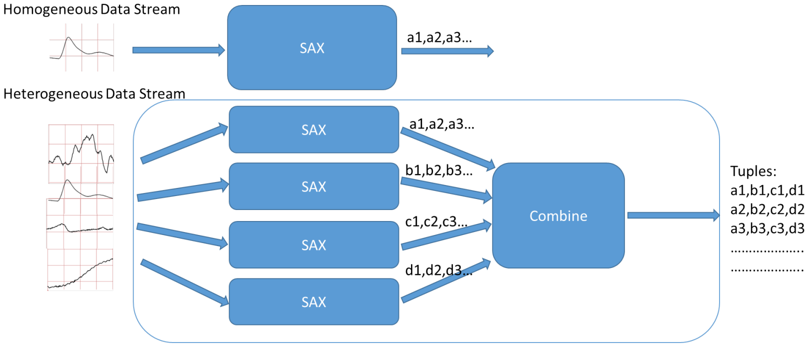
Aim

- Find all series which are similar to Q

Output with Lower Bound

- Possible: {a, b, c}, {b, c}, {b, c ,d}
- Never: {a, b}, {c} or {a, d}

Black Box


Homogeneous Data Stream
SAX
a1,a2,a3...
Heterogeneous Data Stream
SAX
SAX
SAX
SAX
a1,a2,a3...
b1,b2,b3...
c1,c2,c3...
d1,d2,d3...
Combine
Tuples:
a1,b1,c1,d1
a2,b2,c2,d2
a3,b3,c3,d3
....
....

Two Step Process

Step 1:

- Transform Time Series to Piecewise Aggregate Approximation (PAA) representation

Step 2:

- Symbolize the PAA representation into a discrete string

Step 1: Piecewise Aggregate Approximation (PAA)

Given N point Time Series

- C : C1 , C2 , C3 , C4 upto Cn

Convert to w dimensional vector space (w much less than n)

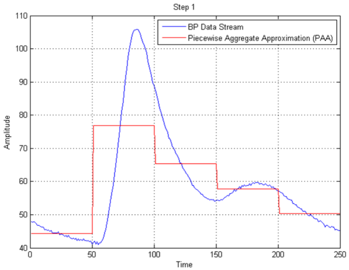
- C : C1 , C2 , C3 , C4 upto Cw

Where C= $\sum_{j=\frac{n(i-1)}{w}+1}^{\frac{n*i}{w}} Cj$

Step 2: Example

Given MIT Dataset (EEG,BP, ECG, Resp)

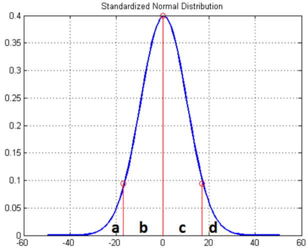
- Data Stream = BP, Data Points = 250 pts/ sec
- Window Size = 1 sec, Pane Size = 0.20 sec (5 Panes per Window)



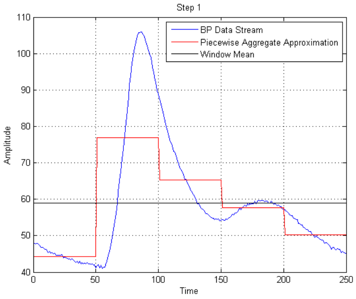
Step 2: Discretization

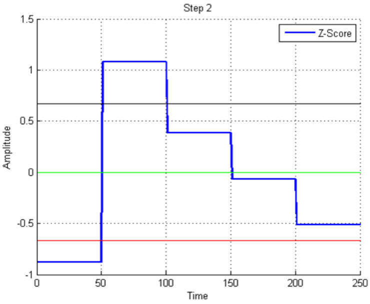
Let a, b, c, d be the symbols
Desirable:All the symbols to be used with equal probability
How ?

- Step 2.1 : Z-Score computation
- Step 2.2 : Symbol Assignment



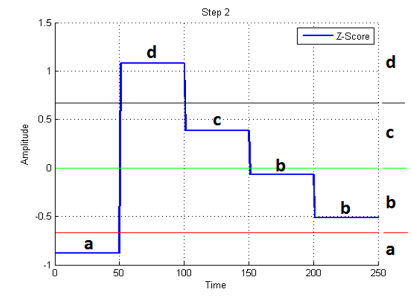
Step2: Example Z score Computation

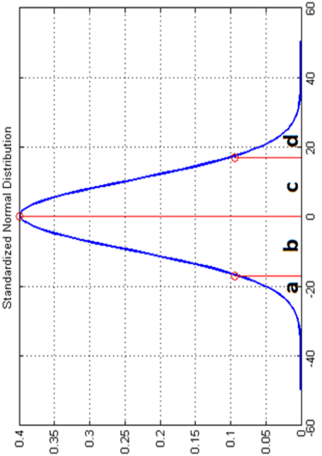




$$\text{Step1}: \text{Std Deviation} = 16.6024$$

Step2: Example - Symbol Assignment





Inspiration

- Inspired from Google Big Table and Azure Storage
- Exploiting the fact that the data is immutable
- Input stream data is in chronological order
- Design supports range queries as a first class operation
- Support for parallel query processing

Key Changes to BigTable

- SSTable (Key:Value Store) to BSTStore (Range Store)
- Simplified Write, TabletServer need not manage write
- No requirement for Memtable
- New Component: Write Master
  - Responsible for writing content
  - Enables Run Time monitor
  - Source for stream processing (d-stream, time stream)

Idea borrowed from Azure

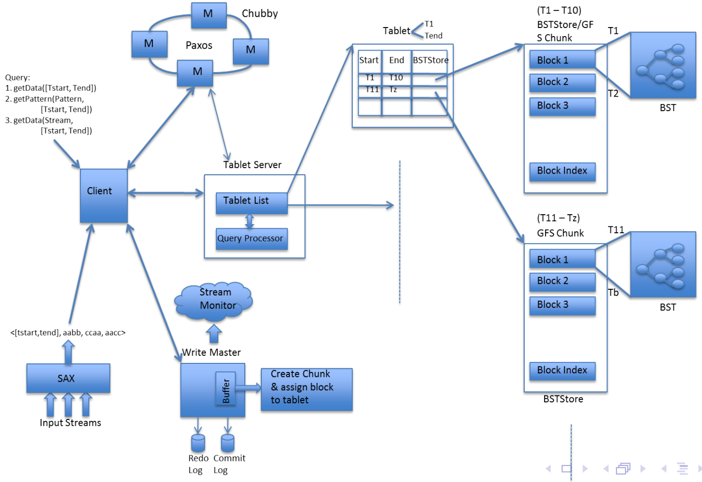
Important idea borrowed from Azure

- Notion of Sealing the nodes/chunks, making it immutable
- At given time only one unsealed tablet (read performance improvement)

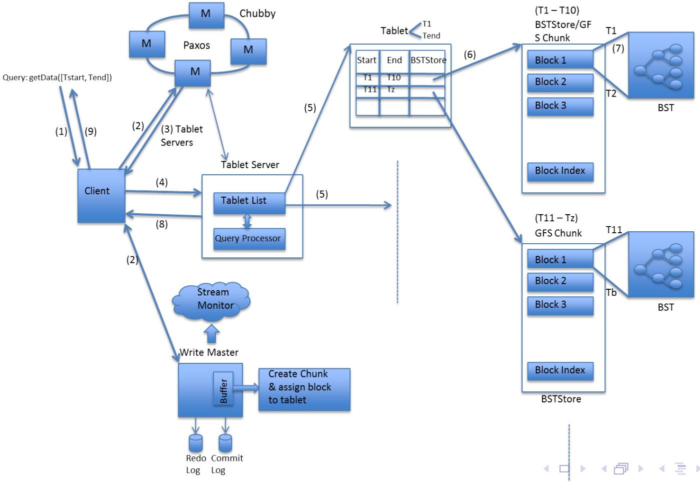
Input Data to the System

- As per SAX output, can be tuple in any format

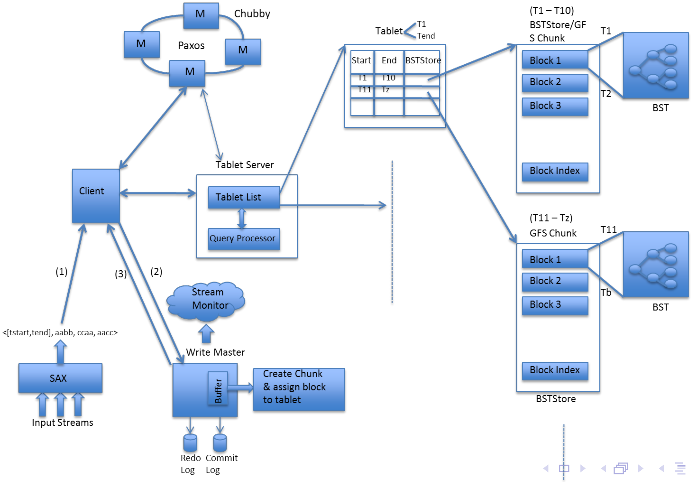
Design


M
Chubby
M
Paxos
M
M
Query:
1. getData([Tstart, Tend])
2. getPattern(Pattern,
[Tstart, Tend])
3. getData(Stream,
[Tstart, Tend])
Client
Tablet Server
Tablet List
Query Processor
Tablet
T1
Tend
Start
End
BSTStore
T1
T10
T11
Tz
(T1 – T10)
BSTStore/GF
S Chunk
T1
Block 1
Block 2
Block 3
T2
BST
Block Index
(T11 – Tz)
GFS Chunk
T11
Block 1
Block 2
Block 3
Tb
BST
Block Index
BSTStore
Stream
Monitor
Write Master
Buffer
Create Chunk
& assign block
to tablet
Redo
Log
Commit
Log
<[tstart,tend], aabb, ccaa, aacc>
SAX
Input Streams

Read


M
Chubby
M
Paxos
M
M
Query: getData([Tstart, Tend])
(1)
(9)
(2)
(3) Tablet
Servers
Tablet Server
Client
(4)
Tablet List
(5)
(8)
Query Processor
(2)
Stream
Monitor
Write Master
Buffer
Create Chunk
& assign block
to tablet
Redo
Log
Commit
Log
Tablet
T1
Tend
Start
End
BSTStore
T1
T10
T11
Tz
(5)
(6)
(T1 – T10)
BSTStore/GF
S Chunk
T1
(7)
Block 1
Block 2
T2
Block 3
BST
Block Index
(T11 – Tz)
GFS Chunk
T11
Block 1
Block 2
Tb
Block 3
BST
Block Index
BSTStore

Write


M
Chubby
M
Paxos
M
M
Tablet
T1
Tend
Start
End
BSTStore
T1
T10
T11
Tz
(T1 – T10)
BSTStore/GFS Chunk
T1
Block 1
Block 2
Block 3
T2
BST
Block Index
Client
Tablet Server
Tablet List
Query Processor
(T11 – Tz)
GFS Chunk
T11
Block 1
Block 2
Block 3
Tb
BST
Block Index
BSTStore
(1)
(3)
(2)
Stream Monitor
<[tstart,tend], aabb, ccaa, aacc>
Write Master
SAX
Buffer
Create Chunk & assign block to tablet
Input Streams
Redo Log
Commit Log

Write

Steps for Push operation:

- Get Tablet server with latest Tablets
- Update corresponding Tablet/Add new tablet
- Update chubby and Tablet server with new Range

Compaction

- Create new chunk from two consecutive chunks
- Update tablet, tablet server and chubby
- Delete old chunks

Old Data

- Infrequent occurrence
- How?
  - Find the appropriate chunk
  - Break the chuck
- Tradeoff between space and efficiency

Failure Recovery

- Similar to Big Table
- If a write master fails, new write master is elected and it reads the redo and commit logs of the old write master.

Query

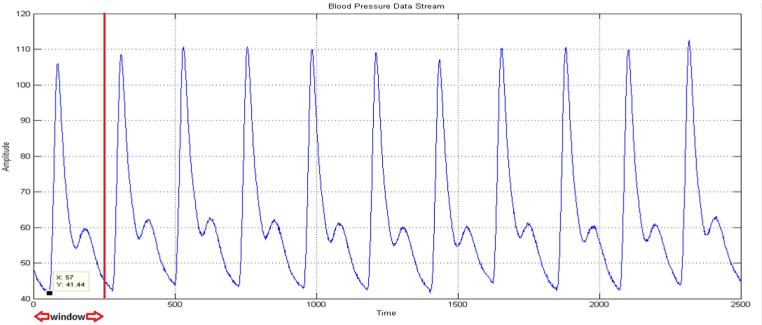
Given MIT Dataset (four data streams EEG,BP, ECG, Resp)

- Data Points = 250 per second
- Window Size = 1 sec
- Pane Size = 0.2 sec (5 Panes per Window)
- Symbols = a, b, c, d

Query Q over

- Data Stream = BP
- Sequence = dcbbbdbba

Query Example

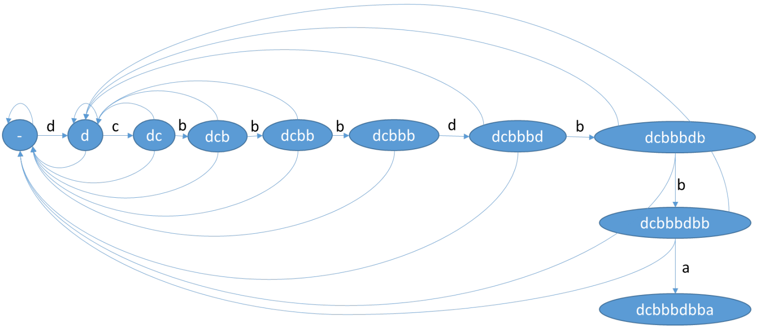


adcbb bdbba dcbbb dbbad cbbad bbadc bbbdb badcb abdbb adbba

Figure: Query:dcbbbdbba

String Matching KMP

State Diagram



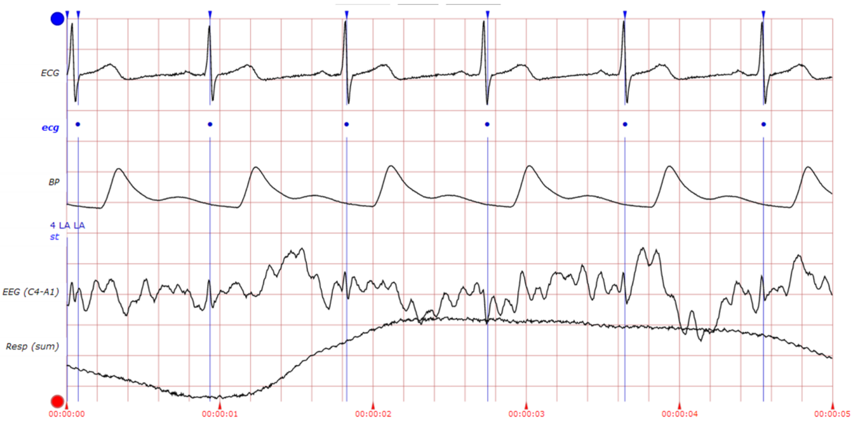
Drawback of Naive Approach

- Under the naive approach, each data stream is assumed to be independent.
- As a result, we might not be able to capture some important dependency among streams.
- Hence, we plan to add synchronization among the heterogeneous data streams.

Drawback of Naive Approach

- Suppose you have two data streams: EEG and BP. BP is having a minor fluctuation in window W while EEG is having major fluctuation in the same time window.
- Naive approach with constant window pane size will treat the two streams as independent and hence will be unable to capture any dependency.
- However under our proposed model, EEG will govern the window pane length and will be able to capture the corresponding change in BP.

Drawback of Naive Approach



Improvement

- Introduce notion of model for on the fly learning and prediction of the window pane length and fluctuation.
- Keep the window pane length adaptable as per the fluctuations in the heterogeneous data stream.
- Govern the data summarization and approximation as per the data stream having maximum fluctuations.

Description

- Synchronization module will work closely with the summarization module to pass data to the persistence layer

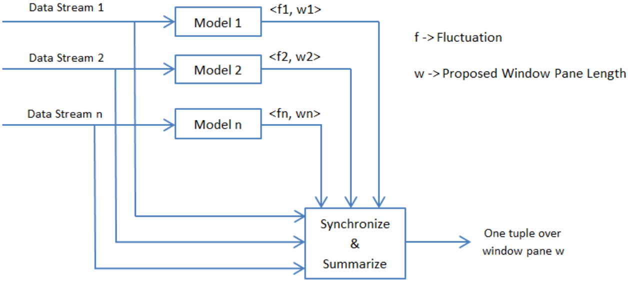


Figure: Synchronization Model

Main Idea

- Maintain n models, one for each data stream.
- For each data stream, the model will:
  - Quantify the fluctuations (using mean and standard deviation).
  - Propose window pane length based on the fluctuations.

Window Pane Length Computation

- In model, say over time T, we have N data points.
- On N data points, try to fit window panes of different length starting from 1 up to length W.
- For each window pane length, calculate the error introduced due to window approximation.
- Finalize W as the maximum value below the permissible error rate.
- Calculate Wfinal by scaling W to T by formula: (W/N) * T
- Maintain Wfinal as the proposed window pane length for the input data stream.

Summarization

- Summarization module takes feedback from each model regarding the window pane length.
- From these feedback, summarization module chooses model with maximum fluctuations. From the chosen model, the proposed window pane length is retrieved.
- Content from all data streams is individually averaged as per retrieved window pane length.
- A single tuple is created with all the averaged values to achieve synchronization.

Conclusion

- We have successfully designed an efficient data storage and retrieval system for heterogeneous data streams.
- Through our experimental analysis, we successfully verified that our system was capable to efficiently handle pattern queries.
- By adding synchronization among the data streams, we can capture the dependency relationship that may exist among the heterogeneous data streams.

References

1 Jessica Lin, Eamonn Keogh, Stefano Lonardi, Bill Chiu, A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, DMKD' 03, June 13, 2003

2 Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, Bigtable: A Distributed Storage System for Structured Data, OSDI 06

3 Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, Leonidas Rigas, Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency, SOSP '11, October 23-26, 2011

4 https://bitbucket.org/rohankarwa/sax

Thank You

Questions?