Using Data Transformations for Low-latency Time Series Analysis

Henggang Cui (CMU)

Kimberly Keeton, Indrajit Roy, Krishnamurthy Viswanathan (HP Labs)
Gregory R. Ganger (CMU)

PARALLEL DATA LABORATORY
Carnegie Mellon University
Time series data analytics

- Time series data e.g., temperatures
- Analytical queries e.g., search for correlation

Time Series Database

2000 2001 2002

Temperature

Year

2000 2001 2002

2015

Correlated
Goals and previous approaches

• Goals
  • Low latency queries
    – Sub-second latency for interactive queries
  • Handle large data size
    – Queries on both recent data and historical data

• Previous approaches
  • Archive data in compact forms [e.g., Cypress]
    – But, does not provide for low-latency interactive queries
  • Keep/use only recent data [e.g., Scuba]
    – But, excludes use of historical data
  • Provide approximate results via sampling [e.g., BlinkDB]
    – But, not for recent data or complex analytics
Data transformations at ingest

Ingest

Ingest-time transformation

DataTransform

Transformed data
(e.g., freq domain representation)

Query-time processing

Raw Data

Query

People

Ingest

DataTransform

Transformed data

(e.g., freq domain representation)
Data transformation examples

- Wavelet transform for temperature data
  - Frequency domain representation
  - Can calculate correlations directly from it
  - Data ranges can be approximated with few coefficients
    - Much smaller than raw data w/ small amount of error

- Other transformations implemented
  - Count-min sketch, ARMA model fitting, FFT, etc.

![Temperatures vs. Wavelet coefficients](image)
Aperture

• Timeseries DB with ingest-time transformation
  • User defined ingest-time transformations
  • Answer queries using transformed data

(same diagram)
Ingest-time transformations

• Transform ingested data for every *window*
  • Window is a range of rows
• Generate and store windows of transformed data

<table>
<thead>
<tr>
<th>Temperature table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>city</strong></td>
</tr>
<tr>
<td>PGH</td>
</tr>
<tr>
<td>PGH</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>PGH</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wavelet table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>city</strong></td>
</tr>
<tr>
<td>PGH</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

wavelet transform:  window size: 1 year, error bound: 10%
Ingest-time transformations

- Chained transformations
  - E.g., data cleansing then wavelets
- Multiple versions of transformed data
  - E.g., with different error bounds or window sizes

![Diagram showing data cleansing process with wavelet transformations]

Raw → DataCleanse → Cleansed → Wavelets (with error bounds)
Query processing

• Workflow of analytical query tasks on raw data

1. fetch data from DB
2. calculation

DB query

result

data
Query processing

• Analytical query tasks using transformed data

DB query

1. choose transformation ✓
2. translate query ✓
3. fetch data from DB (much faster)
4. calculation (much faster)

translated query translated query

transformed data (much smaller) result
Choosing transformation

- Based on user-defined utility functions

\[
\text{If } (\text{not } \text{func} == \text{wavelet}): \text{utility} = 0 \\
\text{If } (\text{window}\_\text{size} > 1 \text{ year}): \text{utility} = 0 \\
\text{If } (\text{error}\_\text{bound} > 10\%): \text{utility} = 0 \\
\text{utility} = \text{window}\_\text{size} \times \text{error}\_\text{bound}
\]
Aperture implementation

- Implemented on top of LazyBase [Cipar et al., Eurosys’12]

1. Receive
   - Client upload
2. Sort
   - Sorted upload
3. MergeSort
   - DB file
Aperture implementation

- Implemented on top of LazyBase [Cipar et al., Eurosys’12]
- Extend Sort stage to Sort+Transform
Evaluation setup

• Hardware information
  • HP ProLiant DL580 machines, each with
    – 60 Xeon E7-4890 @2.80GHz cores
    – 1.5 TB RAM
    – Running Fedora 19
  • One machine for the results in this talk
  • Multiple machines for scalability experiments
Evaluation: correlation search

• Dataset
  • Climatic data from National Climatic Data Center
  • Daily temperature, dew point, and wind speed from hundreds of stations since 1930s

• Table schemas
  • \{metric, station, date, value\}
  • 350 million rows in total

• Testing query task
  • Given 10 years of temperature data in station X
  • Find all data series with correlation > 0.8
Data transformations

- Only 4% overhead on ingestion throughput

Wavelet x6, with
- window size: 1 month and 1 year
- error bound: 5%, 10%, and 20%
Query latencies

Lower latencies when using larger granularities or error bounds

50x reduction
Errors of query results

15% false positives, <1% false negatives but 40x reduction in query latency

but can eliminate false positives via validation
Other use cases

• Event occurrence monitoring
  • Transform: event logs → count-min sketches
    – Bloom filter-like summaries
  • Query use: determine frequency of an event type

• Anomaly detection
  • Transform: observations → list of anomalies
  • Query use: find anomalies in a time range
Conclusions

• Aperture: ingest-time transformation for efficient time series data analytics
  • User-defined transformations when data ingested
  • Answer analytical queries using transformed data

• Many real-world use cases
  • 1~4 orders of magnitude reduction in query latency
  • Less than 20% query error
  • Less than 10% ingestion overhead