Using Data Transformations for Low-latency Time Series Analysis

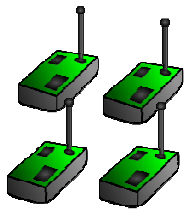
Henggang Cui (CMU)

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

PARALLEL DATA LABORATORY
Carnegie Mellon University

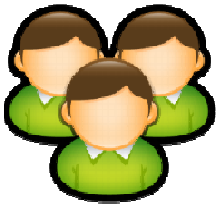
Carnegie Mellon
Parallel Data Laboratory

Time series data analytics



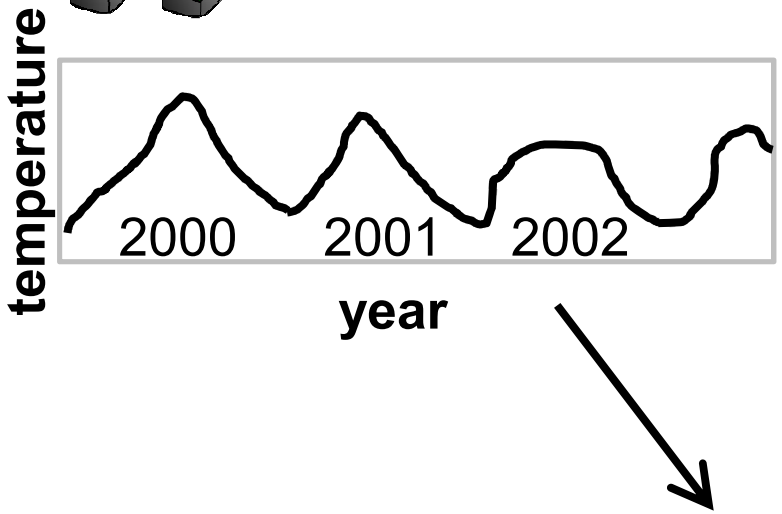
Time series data

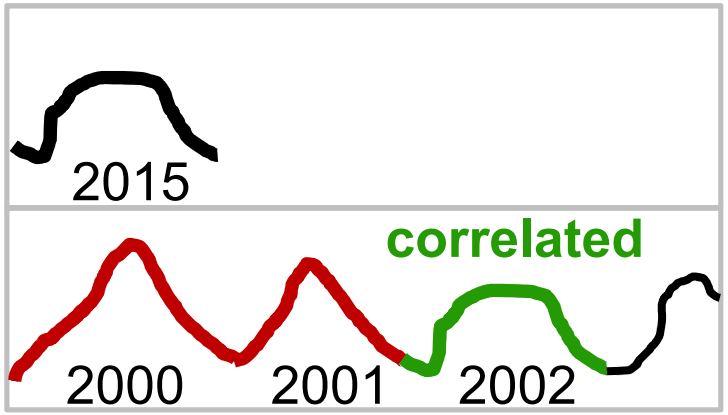
e.g., temperatures



Analytical queries

e.g., search for correlation





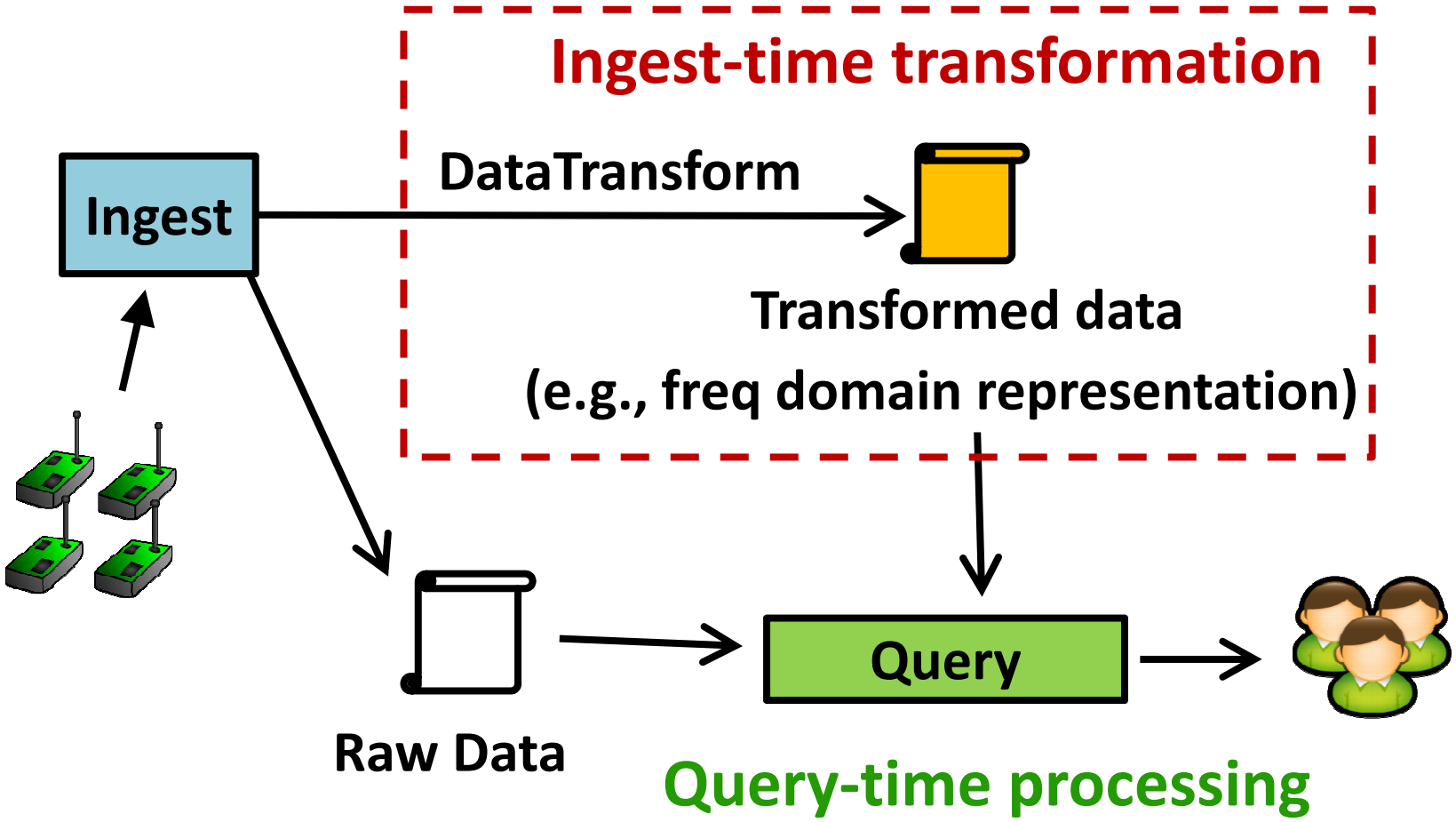
Time Series Database

Carnegie Mellon
Parallel Data Laboratory

Goals and previous approaches

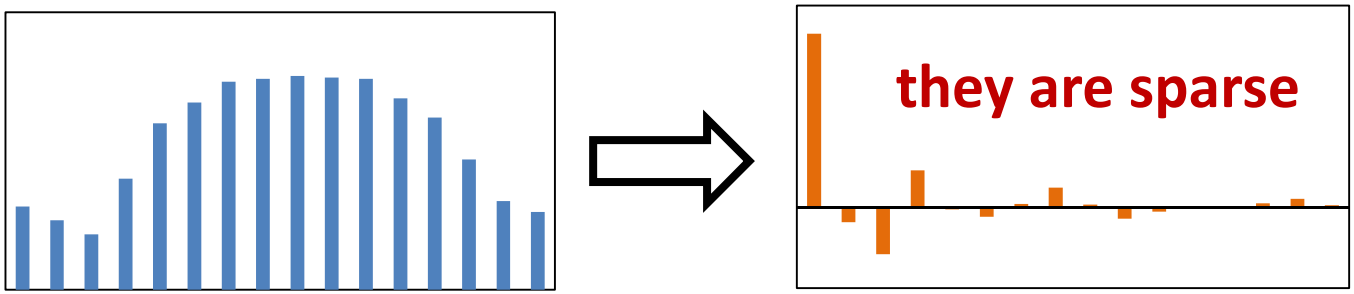
- Goals
  - Low latency queries
    - Sub-second latency for interactive queries
  - Handle large data size
    - Queries on both recent data and historical data
- Previous approaches
  - Archive data in compact forms [e.g., Cypress]
    - But, does not provide for low-latency interactive queries
  - 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



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



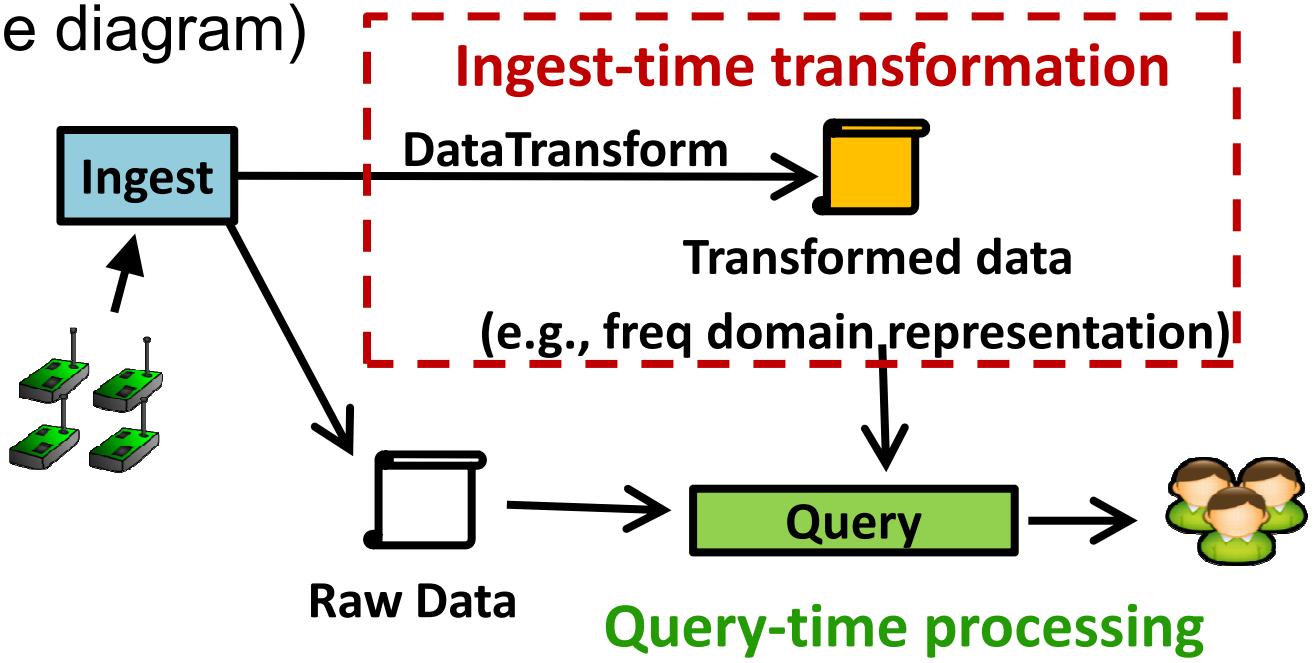
Temperatures **Wavelet coefficients**

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

Aperture

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

(same diagram)

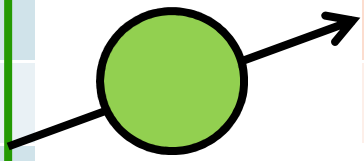


Ingest-time transformations

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

Temperature table

city	date	value
PGH	1/1/2000	35.2
PGH	1/2/2000	36.3
...	...	...
PGH	12/31/2000	...
...	...	...



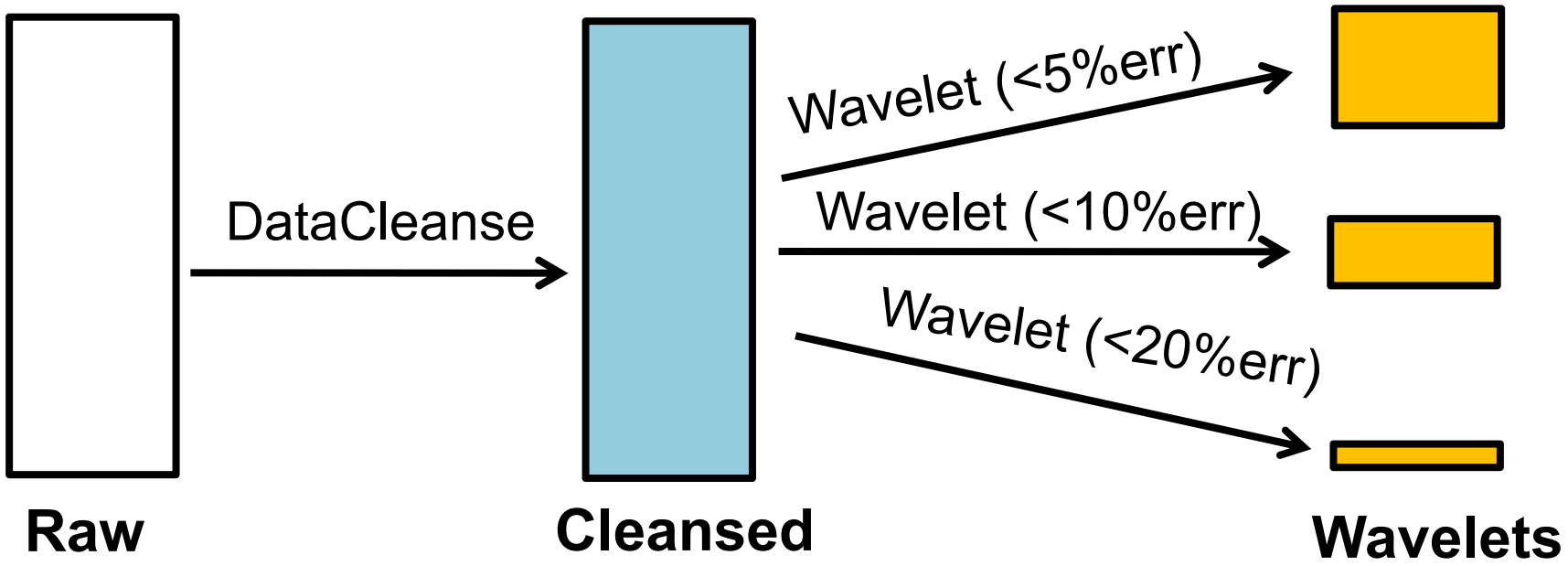
Wavelet table

city	date	wavelets
PGH	1/1/2000	<wavelets>
...	...	...

wavelet transform:
window size: 1 year,
error bound: 10%

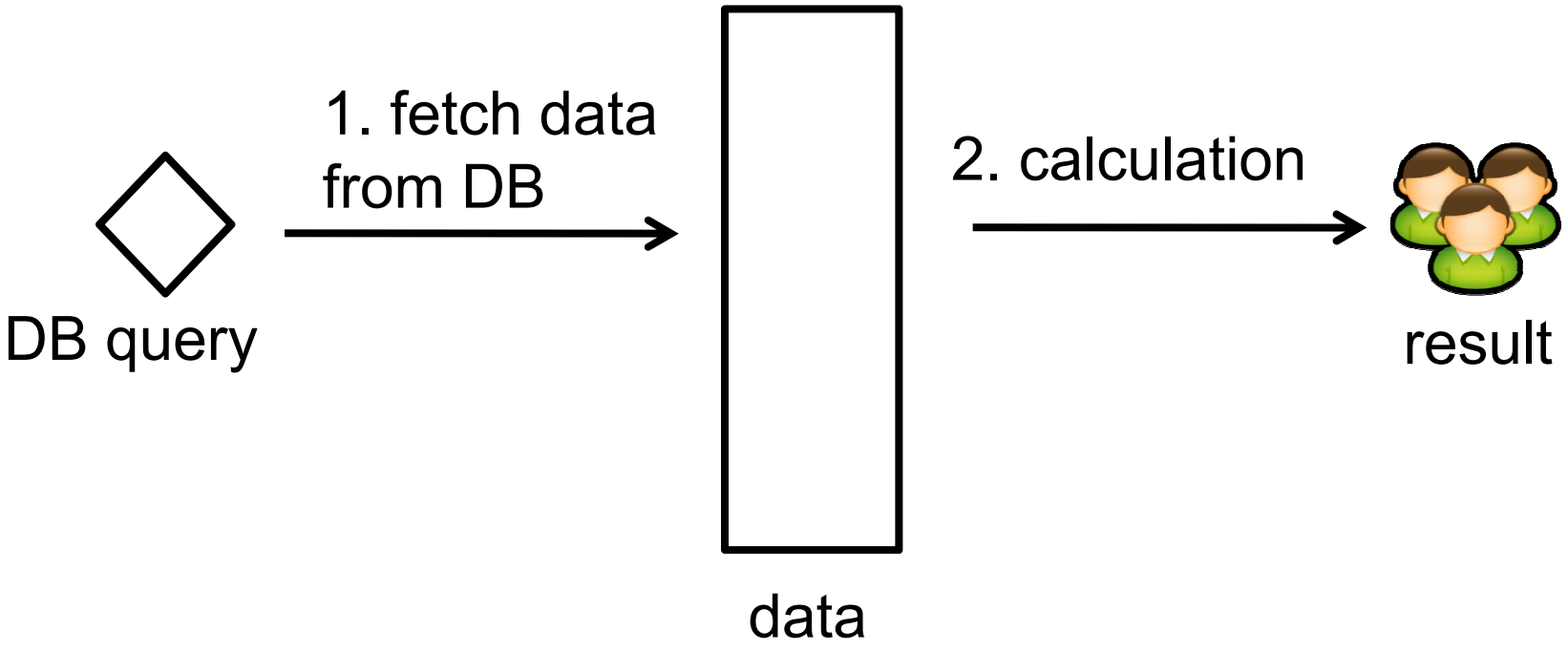
Ingest-time transformations

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



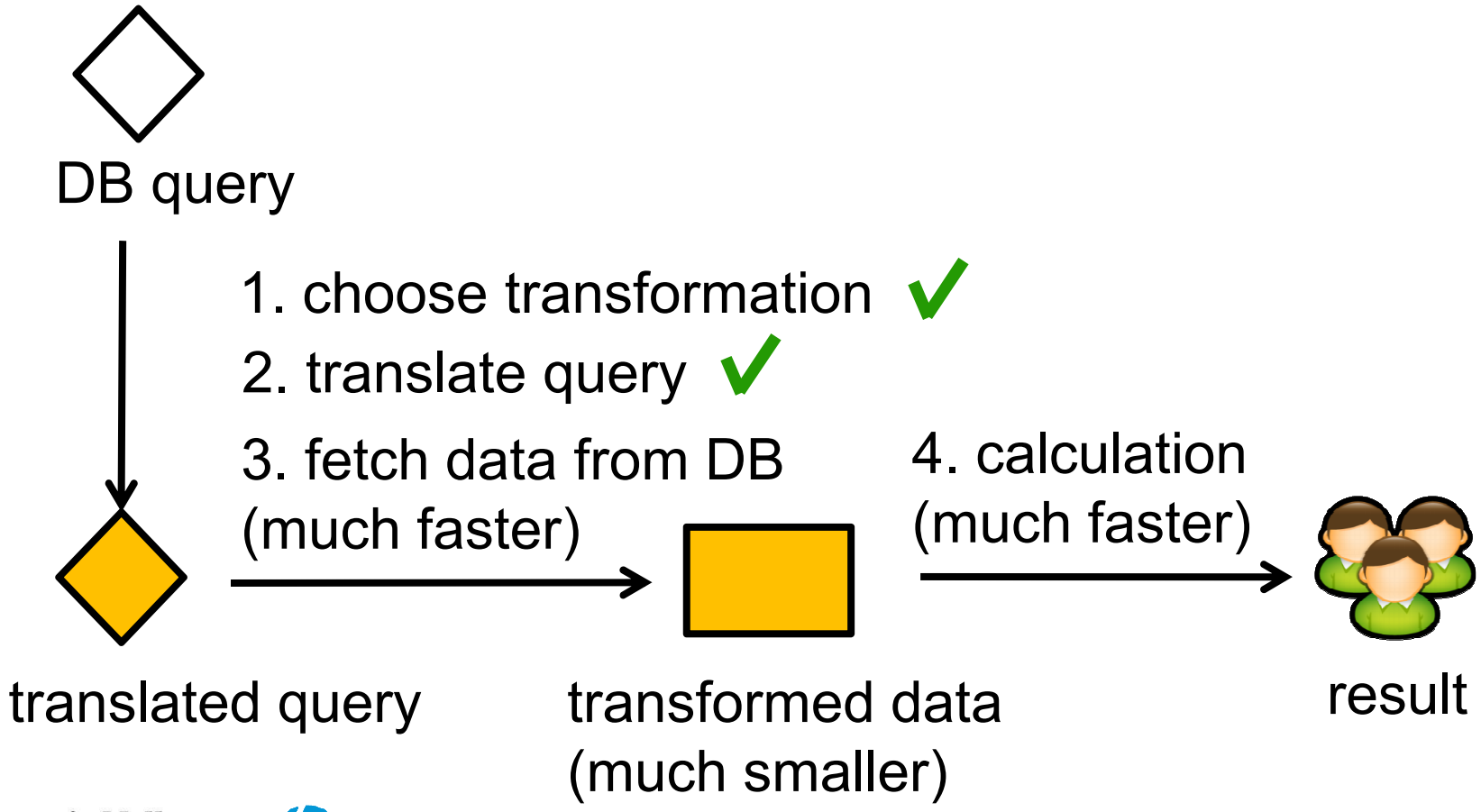
Query processing

- Workflow of analytical query tasks on raw data



Query processing

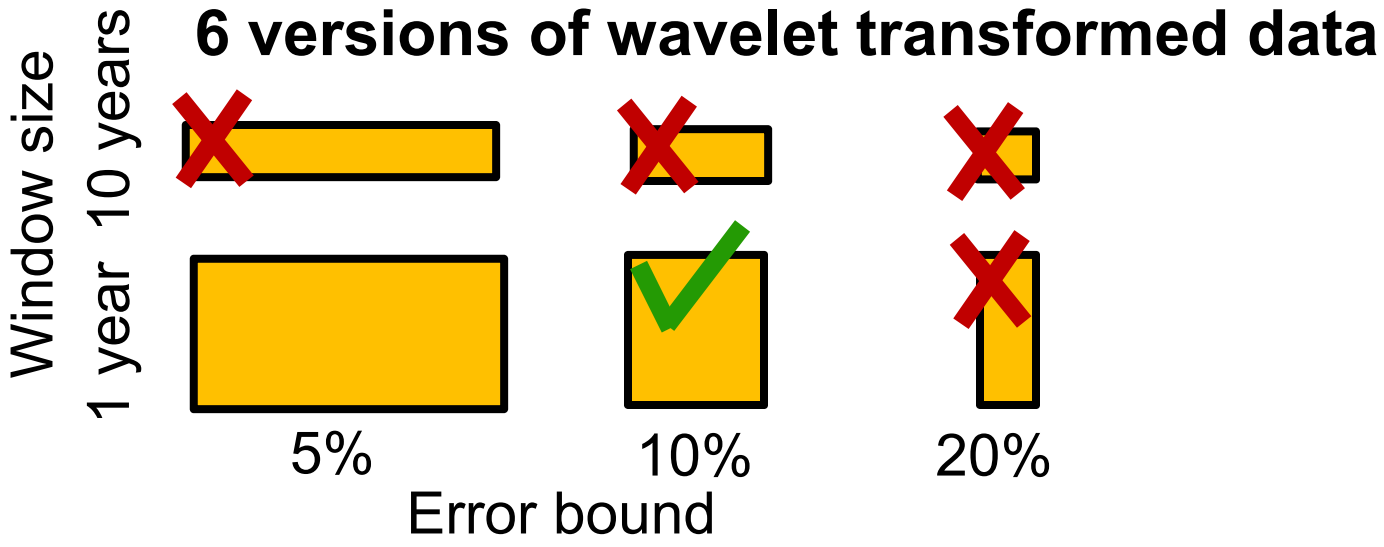
- Analytical query tasks using transformed data



Choosing transformation

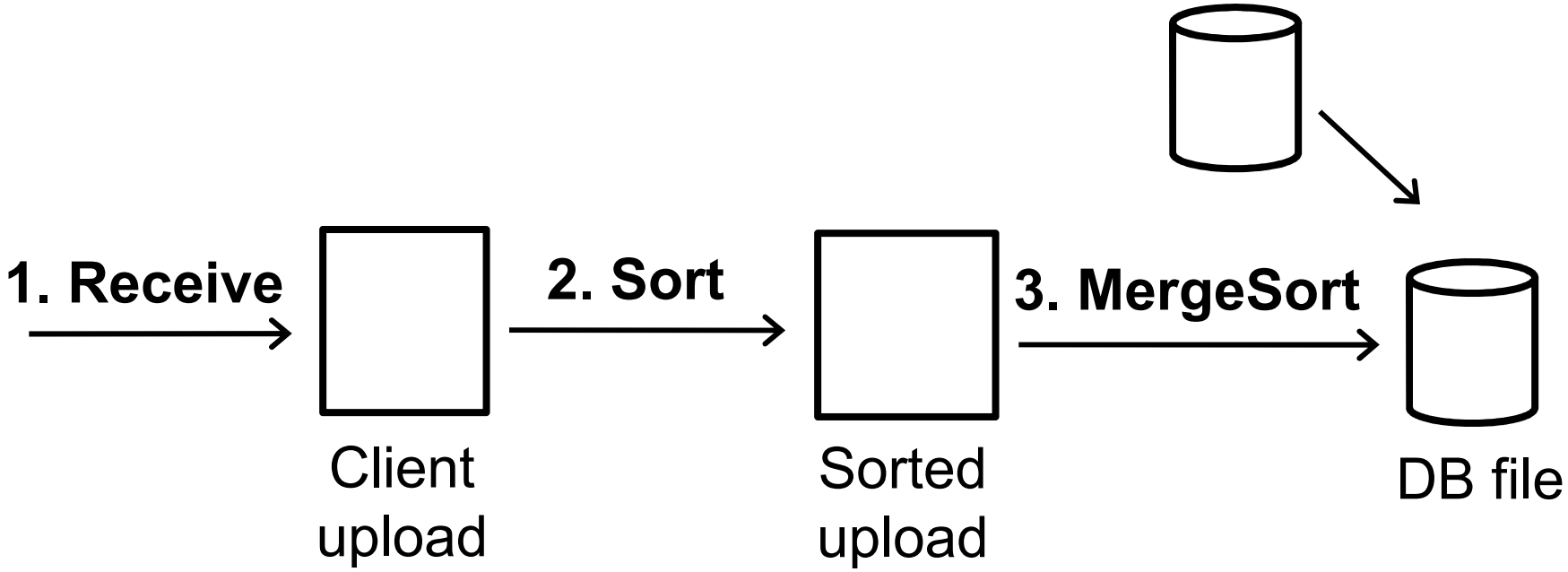
- Based on user-defined utility functions

```
If (not func == wavelet): utility = 0
If (window_size > 1 year): utility = 0
If (error_bound > 10%): utility = 0
utility = window_size * error_bound
```



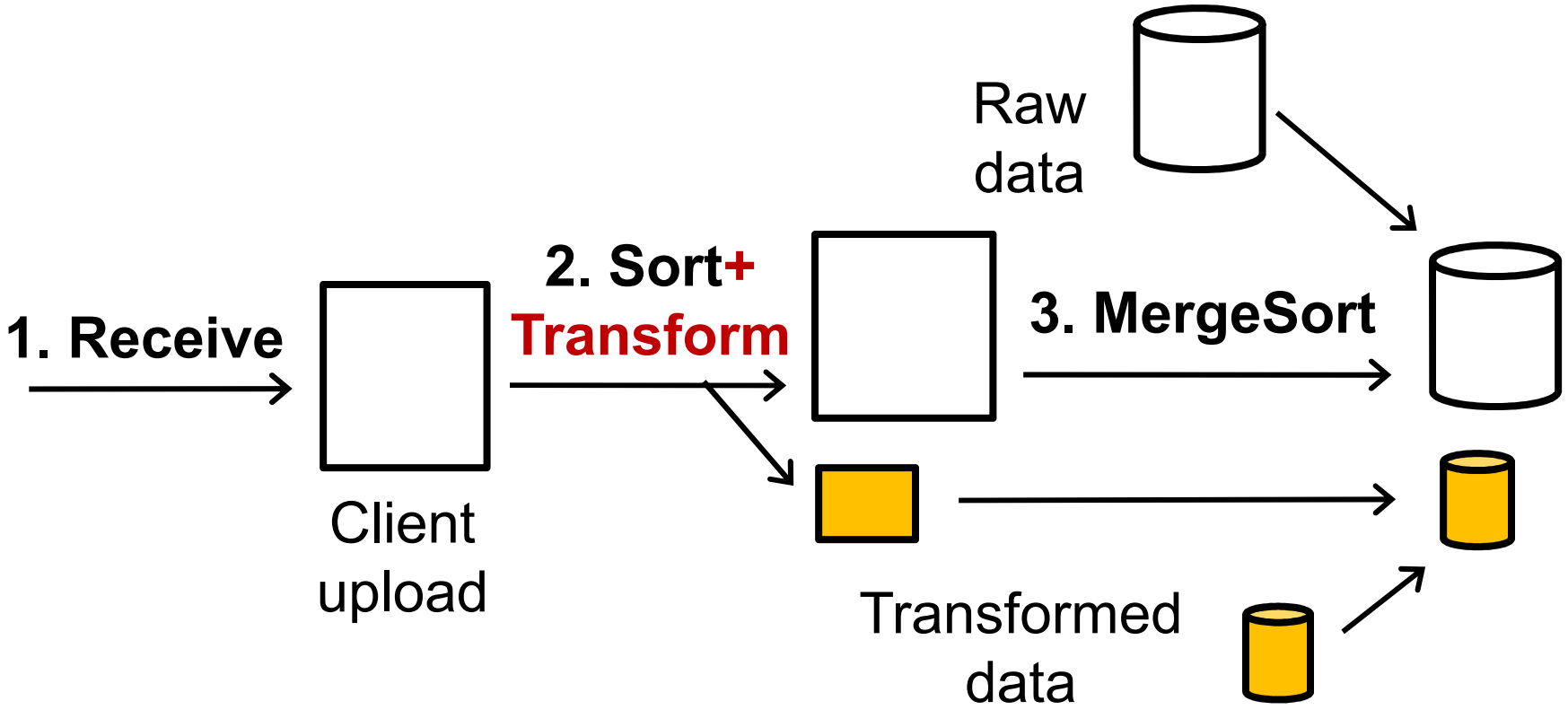
Aperture implementation

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



Aperture implementation

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



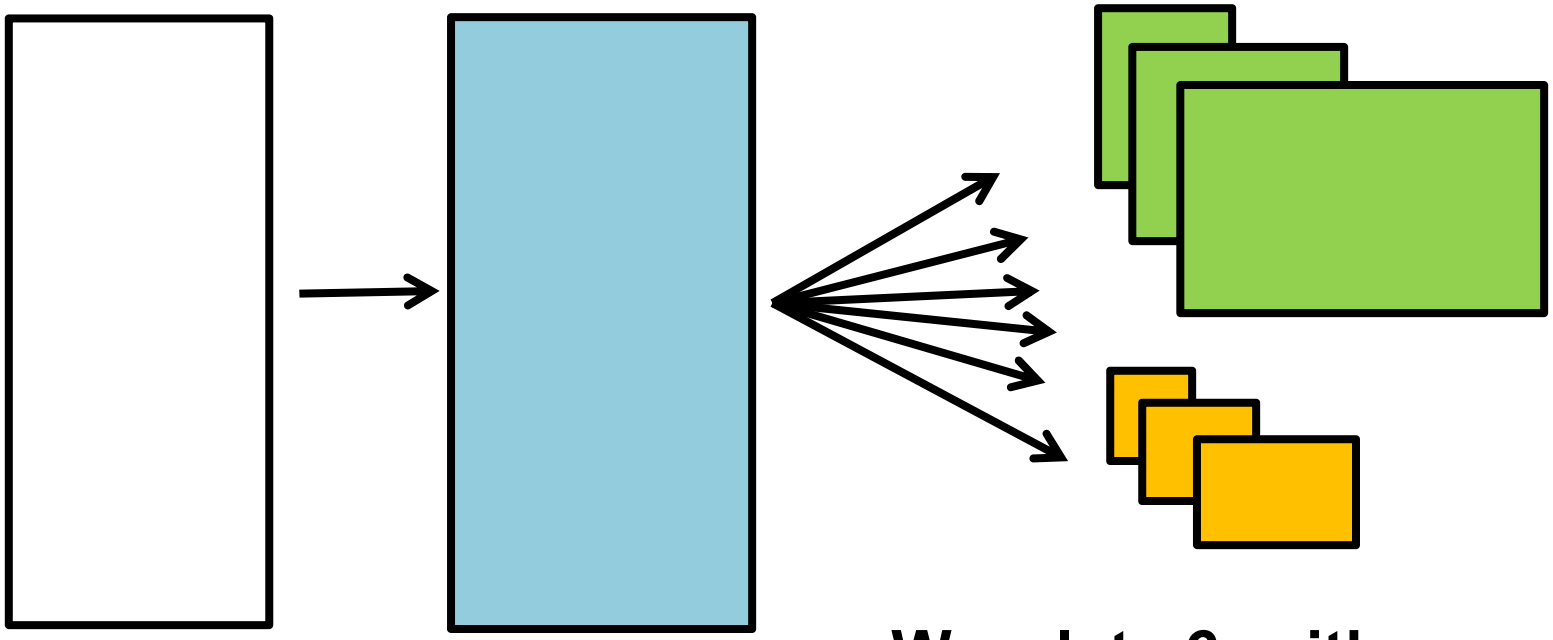
Evaluation setup

- 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



Raw

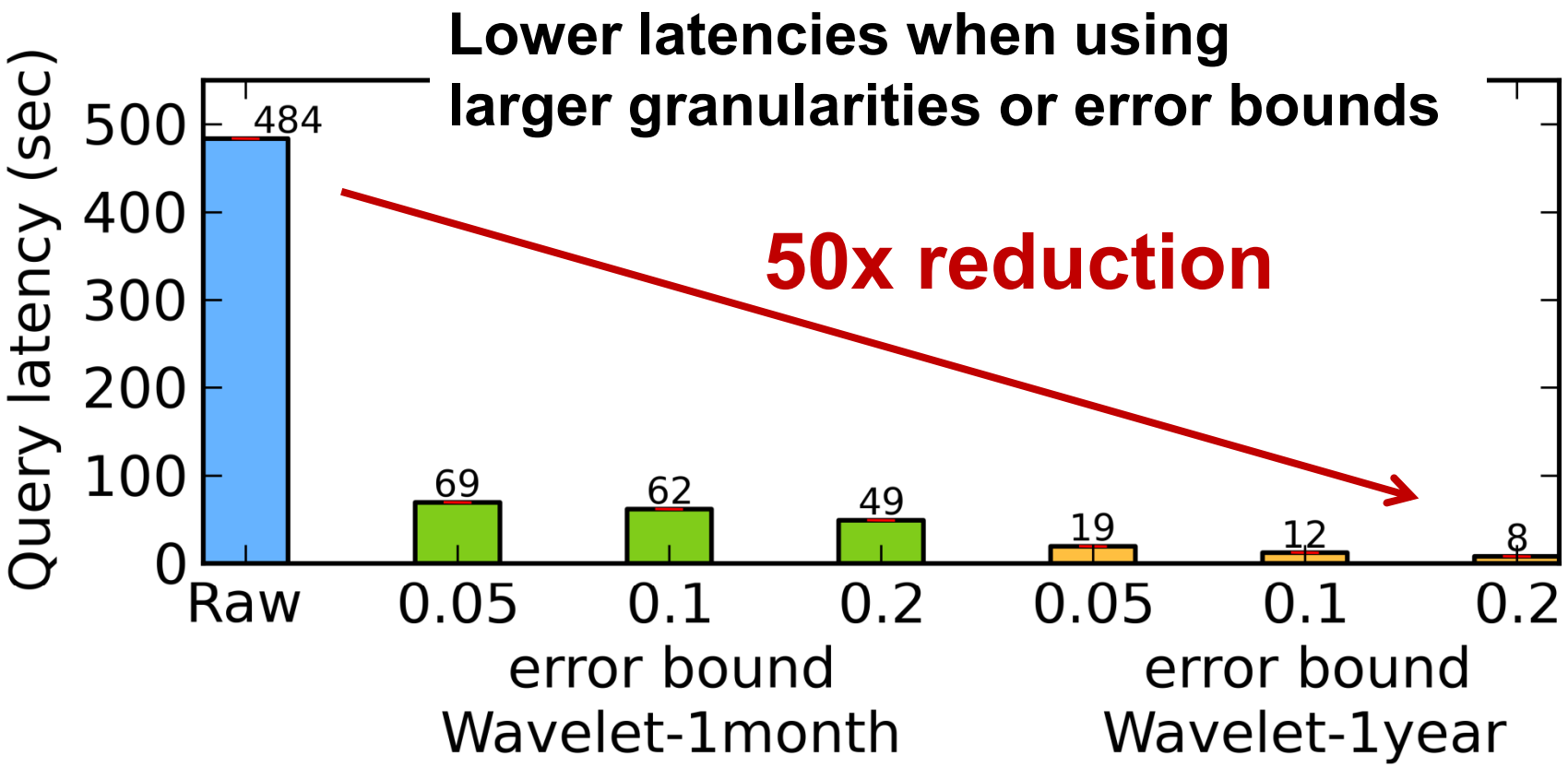
Cleansed

Wavelet x6, with

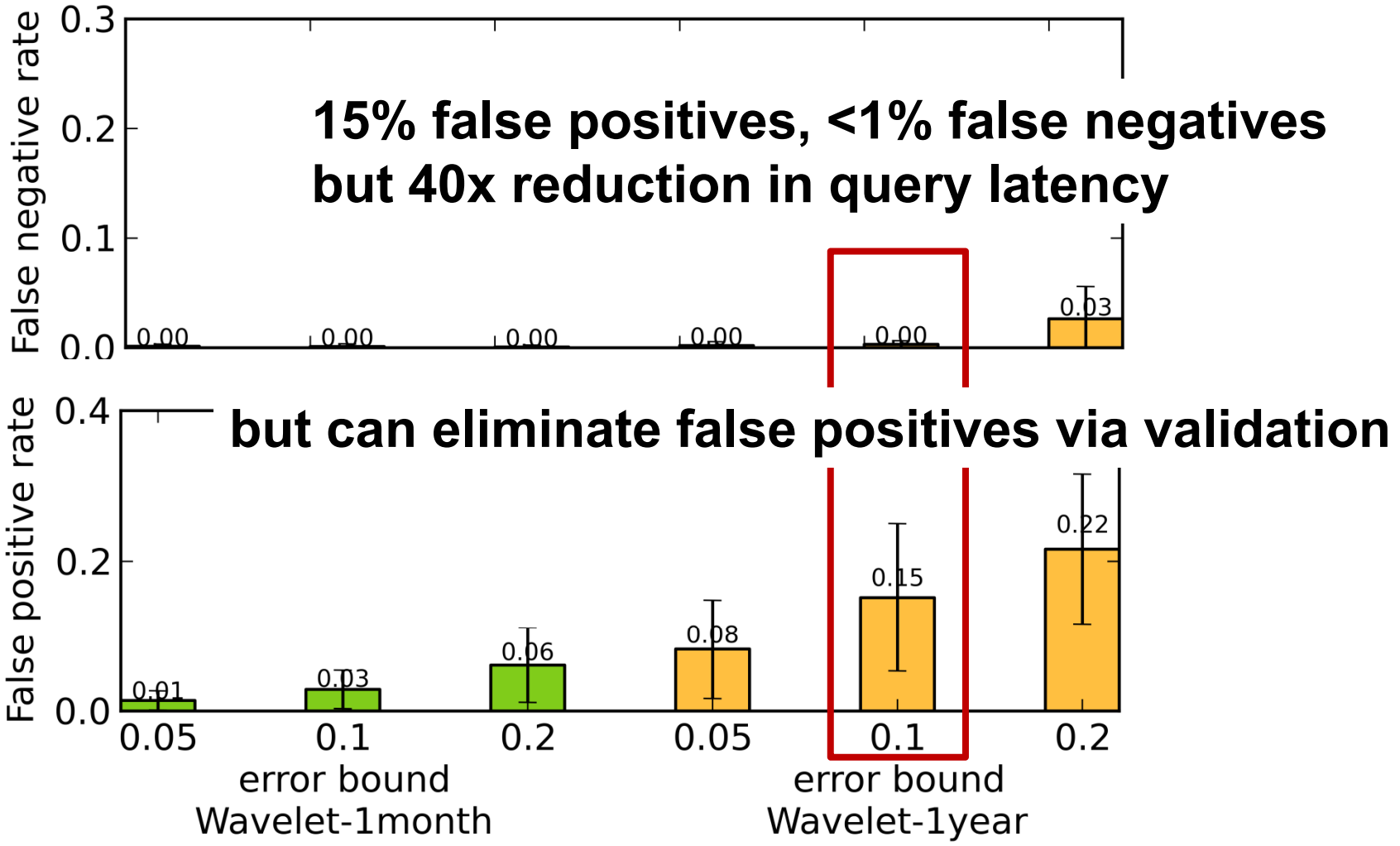
- window size: 1 month and 1 year
- error bound: 5%, 10%, and 20%

- **Only 4% overhead on ingestion throughput**

Query latencies



Errors of query results



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