Storing and Querying Big Energy Time Series

Christian Thomsen (chr@cs.aau.dk)

joint work with Søren Kejser Jensen and Torben Bach Pedersen
The challenge

- Wind turbines and solar panels have a lot of sensors that can deliver data values several times per second
  - A modern wind turbine has up to 15,000 streams
- This generates a lot of data
  - 10 reads/second, 4 bytes, 15000 streams ➔ ~50 GiB per day from one wind turbine
The current situation

• The available information is currently not exploited or stored

• Many monitoring solutions consider few (~100) sensor streams and store only a single value for every $x$ minutes (e.g., the average)
  – $x$ is typically $\frac{1}{2}$, 1, 2, 5, or 10

• Important things might not be seen
Example of ”missing the point” :-)
What we want to do...

- Store and use all available sensor data
- Support efficient aggregate queries on historical data
- Support analysis of data while it is being ingested
  - Detect underperformance and other problems immediately
  - Enable predictive maintenance
  - For example, detect and fix a problem before the wind turbine breaks
How we want to do it

- Time-series can contain millions of points
- An efficient way to store and process them is to represent them by models
- We use a model-based approach for the time-series data
- A (user-defined) error can be allowed
  - For example 5%, 1%, or 0%
Simple example – linear models
ModelarDB

• We have developed ModelarDB which uses models to store time series data
• Time series-specific functionality implemented in a system-agnostic library
• We have implemented some model types and the user can optionally add more. ModelarDB will automatically pick the best
• Query processing and storage from existing systems (we have used Spark & Cassandra)
Storage requirements for a real-world data set from the energy domain

<table>
<thead>
<tr>
<th>Storage Method</th>
<th>Size in GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV files</td>
<td>617.52 GB</td>
</tr>
<tr>
<td>PostgreSQL 10.1</td>
<td>782.87 GB</td>
</tr>
<tr>
<td>InfluxDB 1.4.2</td>
<td>4.34 – 4.44 GB</td>
</tr>
<tr>
<td>Apache Parquet files</td>
<td>106.94 GB</td>
</tr>
<tr>
<td>Apache ORC files</td>
<td>13.50 GB</td>
</tr>
<tr>
<td>Apache Cassandra 3.9</td>
<td>111.89 GB</td>
</tr>
<tr>
<td>ModelarDB</td>
<td>2.43 – 2.86 GB</td>
</tr>
</tbody>
</table>
Performance example for large aggregate queries
Performance summary

- ModelarDB provides support for fast ingestion, good compression, and fast large aggregate queries.
- ModelarDB remains competitive for small aggregate and point/range queries.
- Other systems are good for one of these, but not both.
Conclusion and future work

- ModelarDB provides novel model-based compression within an error bound
- Good performance
- Integrated with Spark and Cassandra
- Future directions
  - Indexing to increase query performance further
  - Represent correlated streams together