DB&A: AN OPEN SOURCE WEB SERVICE FOR METER DATA MANAGEMENT

Søren Aa. Mikkelsen and Rune H. Jacobsen
{smik.rhj}@eng.au.dk
Department of Engineering
Aarhus University, Denmark

Anders F. Terkelsen
aft@haugstad-terkelsen.dk
Haugstad & Terkelsen ApS
Denmark
OUTLINE

- Introduction & scope
- Data model
- Common analytics for MDM
- Our evaluation results
  - Case study: SmartHG project
  - Experimental work
- Discussion
- Conclusion & future work
INTRODUCTION (1/2)

- Increasing number of initiatives dealing with Smart Grid
  - Core: Meter data
- There exist many commercial solutions
  - Established companies: IBM, Oracle, Microsoft
  - New start-ups: Energyworx, Virdata, Waylay.io
- No open solution for deploying value-added energy management services
  - Have to “reinvent” MDM service for their purpose
INTRODUCTION (2/2)

Our proposal

- Open Web service as a reference implementation for Meter Data Management (MDM)
  - Fit for use in multi-architecture
  - Address specific data latency scope
  - Considers service composability
  - Considers five fault scenarios
- Used in an EU project called the *SmartHG project* for over 8 months
- Evaluated in an experimental setup
CONTEXT: THE SMARTHG PROJECT

Ecosystem of energy-aware web services

Two goals:

1. Minimise the energy usage and costs for each individual home
2. Support the Distribution System Operator (DSO) in optimising the operation of the distribution grid.
CONSIDERED ARCHITECTURES

System architectures

- Cloud-based
- HEMS-based
- Hybrid-based
Data latency = time a sensor acquires a measurement until it is stored in the MDM

**DLL 0**: Firm real time – Often require closed loop controller

**DLL 1**: Soft real time – Degrades system’s QoS.

**DLL 2 and 3**: Low real time requirements

Source: Courtesy of Accenture
RELATIONAL MODEL OF METER DATA

Consumption tree level

\[ MD_{M_i} \]

\[ MD_{M_{i+1}} \]

\[ MD_{M_{i+2}} \]

\[ MD_{M_{i+3}} \]

\[ MD_{M_{i+4}} \]

\[ MD_{M_{i+5}} \]
**DATA MODEL FOR RESIDENTIAL HOMES**

**Fault scenarios**
1. Meters may be attached to different appliances during its life-cycle
2. Meters can break
3. Meters are able to send data in burst-mode
4. Meters may be installed wrongly
5. Meters may sent faulty data
THREE COMMON ANALYTICS FOR MDMS

- Filtering
- Condensation
- Virtual Metering
DATA CONDENSATION

- Condensation – specialisation of aggregation
- Common services prefer even sampled time series
- Services have different granularity requirements
- However, automatic data condensation will make fault detection harder

Algorithm 1: Condensation of energy data.

```plaintext
Input : M, t_s, t_e, Δt, k
Output: I with granularity of Δt between t_s and t_e
M_k ← FetchMeasurementList (M, k, t_s, t_e);
T ← CreateTimestampGenerator (t_s, t_e, Δt);
t = NextTimestamp (T);
foreach m_i, m_{i+1} in CreatePair (M_k) do
    while TimeOf (m_i) > t do
        t ← NextTimestamp (T);
    end
    while TimeOf (m_i) ≤ t ≤ TimeOf (m_{i+1}) do
        I ← InterpolateAppend (I, t, m_i, m_{i+1});
        t ← NextTimestamp (T);
    end
end
```

M: Measurement set  Δt: Granularity

k: Meter port  t_e: End time

AARHUS UNIVERSITY
SCIENCE AND TECHNOLOGY

SØREN AAGAARD MIKKELSEN
DB&A: AN OPEN SOURCE WEB SERVICE FOR METER DATA MANAGEMENT
VIRTUAL ENERGY PORT CONCEPT

- Situation: Only when current measurements are available at submeter level
- Uncertainty: Largest loads are often inductive
- Aim: Higher accuracy

Constraint in calculating the virtual energy measurements:

\[
\frac{E(t+\Delta t) - E(t)}{\Delta t} \geq I_{RMS}(\Delta t) \times V_{RMS}(\Delta t) \times \text{pf}(\Delta t)
\]
IMPLEMENTATION

- RESTful web application
  - Django 1.7.4
  - Data model implemented as an app
  - Use of PostgreSQL database for cloud application and SQLite for HEMS
  - Django REST Framework
    - Serialisation and deserialization
    - User permission policies
    - Throttling the rate of requests
    - Pagination of responses

Available at https://github.com/dbservice/dbservice/
SMARTHG: DEPLOYMENT EXPERIENCE

- Deployed from February to September 2015
- Experience from field trial
  - Continuously roll-out of metering equipment → System must adapt
  - Malfunctioning → Replacement
  - Being shut off unintentionally or removed
  - Service maintenance

Obtained over 80 million measurements from the test bed.

Table 2. Attributes obtained from the test bed.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Unit</th>
<th>$T_s$ [s]</th>
<th>Meter level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentSummationDelivered</td>
<td>mWh</td>
<td>30-60</td>
<td>Main and sub</td>
</tr>
<tr>
<td>InstantaneousDemand</td>
<td>mW</td>
<td>20-40</td>
<td>Main and sub</td>
</tr>
<tr>
<td>MeterLineFrequency</td>
<td>mHz</td>
<td>10-120</td>
<td>Main and sub</td>
</tr>
<tr>
<td>MeterCurrentRMS</td>
<td>mA</td>
<td>10-120</td>
<td>Main and sub</td>
</tr>
<tr>
<td>PowerFactor</td>
<td>\frac{1}{1000}</td>
<td>10-120</td>
<td>Main and sub</td>
</tr>
<tr>
<td>VAPower</td>
<td>mVA</td>
<td>10-120</td>
<td>Main</td>
</tr>
<tr>
<td>FwdActive</td>
<td>mWh</td>
<td>10-120</td>
<td>Main</td>
</tr>
<tr>
<td>RevActive</td>
<td>mWh</td>
<td>10-120</td>
<td>Main</td>
</tr>
<tr>
<td>ImportReactive</td>
<td>mVARh</td>
<td>10-120</td>
<td>Main</td>
</tr>
<tr>
<td>ExportReactive</td>
<td>mVARh</td>
<td>10-120</td>
<td>Main</td>
</tr>
</tbody>
</table>
DATA SET CHARACTERISTICS FROM TEST BED

Uncertainty in estimating $M_{Ts}$:
- Sampling period fluctuates
- Meter is offline
- Maintenance
- Manually querying

Percentage of data successfully retrieved:
$$e^{(i,k)} = \frac{M_{T_s}^{(i,k)} \times |M^{(i,k)}|}{\max M^{(i,k)} - \min M^{(i,k)}}$$

Sample period
$\text{Start time}$
$\text{End time}$

# of measurements

Percentage of data successfully retrieved:

![Bar chart showing percentage of data successfully retrieved by month, with bars for February, March, April, May, June, July, August, and September.](image-url)
EXPERIMENTAL SETUP

Node.js client module: bench-rest

Cloud system

Raspberry Pi

Node.js Client
- Port 9000
- Configuration parameters

SSH tunnel
- ssh -L 9000:localhost:8005 $target_address
- Measurement data

Target platform
- Port 8005
- Web Server
- DB&A Web App
- PostgresQL/SQLite
- gunicorn
## EXPERIMENTAL EVALUATION

<table>
<thead>
<tr>
<th>Function</th>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posting</td>
<td>One measurement</td>
<td>Replays measurement</td>
</tr>
<tr>
<td>Filtering</td>
<td>A week of consumption data</td>
<td>20 JSON formatted measurements and total number of measurements</td>
</tr>
<tr>
<td>Condensation</td>
<td>A week of consumption data condensed on daily basis</td>
<td>20 JSON formatted outputs and total number of measurements</td>
</tr>
<tr>
<td>Virtual</td>
<td>A week of virtual energy measurements</td>
<td>All entries without pagination.</td>
</tr>
</tbody>
</table>

![Graph showing time comparison for different functions](image)

- **Subset of SmartHG data set**
DISCUSSION

- Use of the Python language and Django framework
- Impact on experimental results:
  - Caching system
  - Reverse proxy
  - Placement of business logic: database queries vs. Django framework
  - Pagination ➔ Minimise size of response and better distribution of computational burden
    - Makes evaluation comparison difficult
- Evaluation of MDM data latency with composability setup
CONCLUSION & FUTURE WORK

- Complies with five faulty scenarios
- Provide common analytic functions for MDM
- Evaluated on a large and small scale
- Source code available at: https://github.com/dbservice/dbservice

Future Work

- Security and privacy
  - Better user model to give granular data access
- Quality assurance
  - Use metering hierarchy to detect inconsistencies in meter data
QUESTIONS?

smik@eng.au.dk