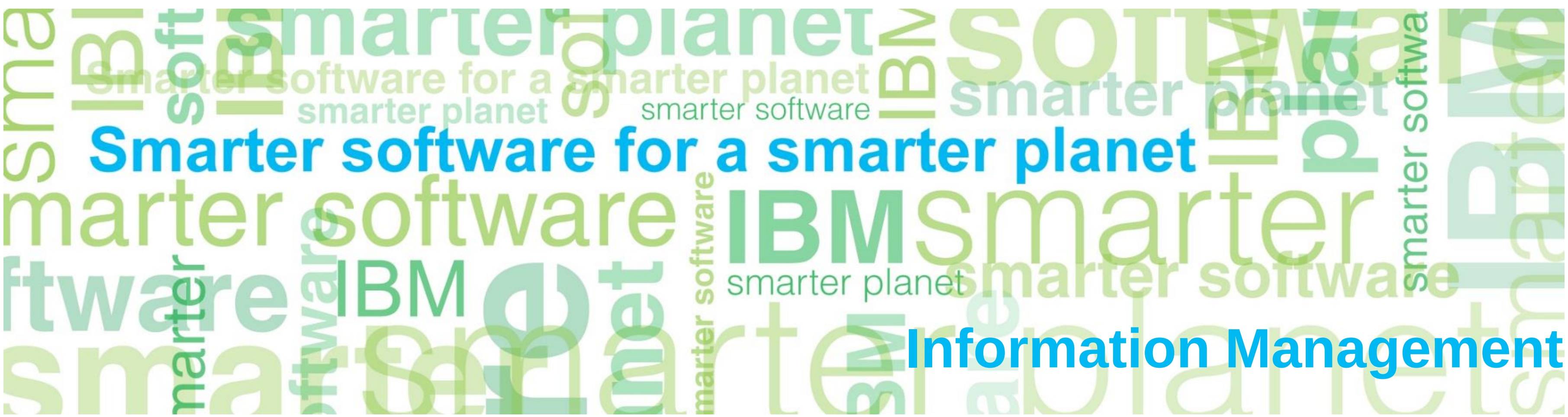
# Wieso eignet sich Informix besonders für das Internet der Dinge?

**Alexander Körner**
**akoerner@de.ibm.com**

Alexander Koerner (akoerner@de.ibm.com), June 2016

# Flexible and Scalable Sensor Data Management

## For Industry 4.0 and the IoT. In the Cloud and on the Edge by IBM



# A typical „IoT-/Industrie 4.0“ Scenario

**Industrial Shop Floor Sensors**

Internet of Things
IBM

IBM Bluemix™

Time Series Database
IBM

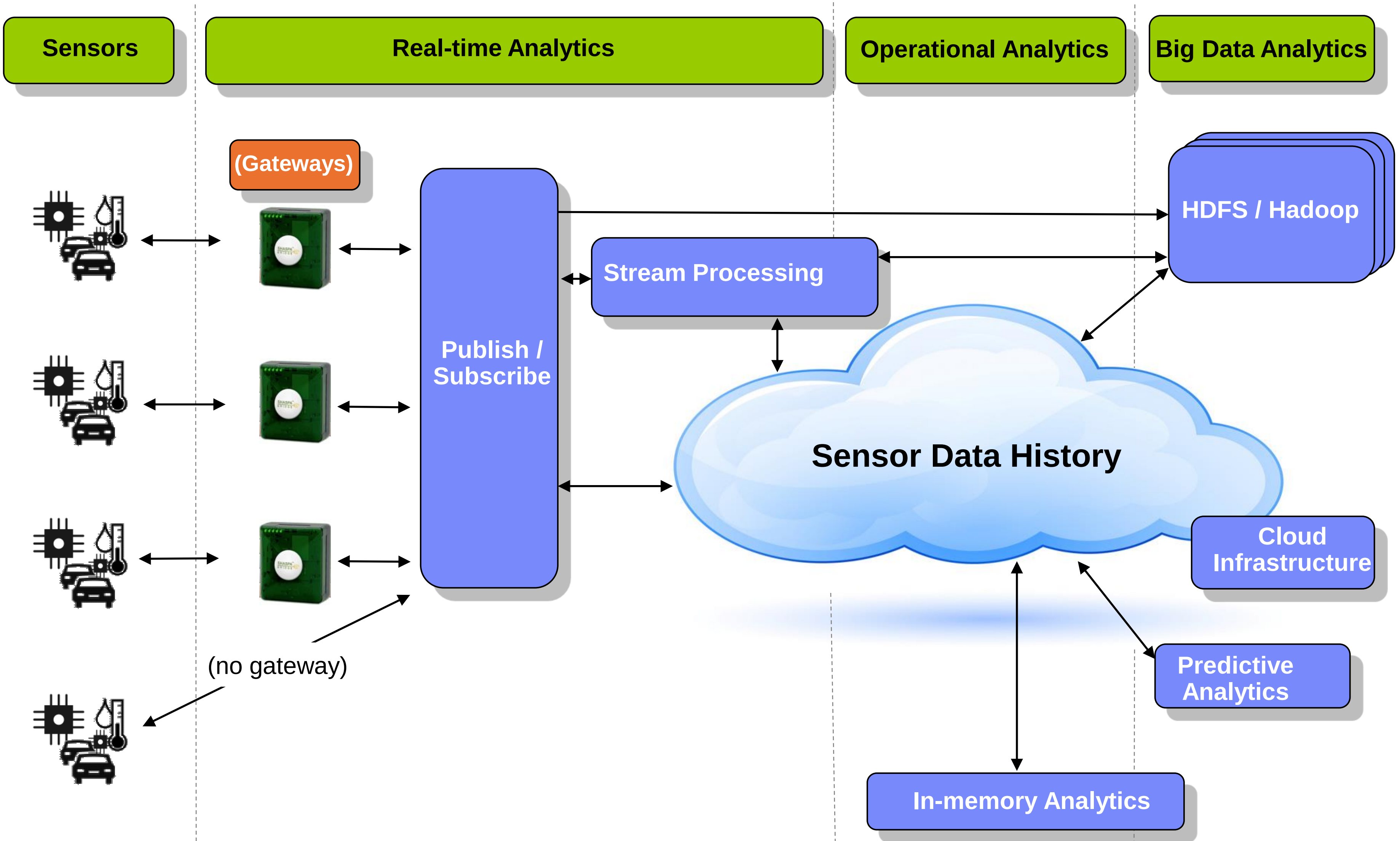
Direct sensor data transfers

Raw Sensor Data
via industry specific
automation protocols
(e.g. PROFINET, BACnet,
OPC UA, Modbus etc.)

Aggregated / Filtered Sensor Data
via standard communication protocols
and formats (e.g. MQTT/JSON) into
the Cloud (IBM Bluemix, IBM IoT Foundation),
IBM PMQ plus on premises solutions etc.

**Industrial IoT-/Edge-Gateway(s)**
with specific industry communication stacks
and local Analytic capabilties (in motion / at rest analytics)

# A Simplified IoT/Industry 4.0 Sensor Data Flow



# What are the IoT/Industry 4.0 Analytics Requirements for the Cloud?

- Requirements:
  - Potentially 1000’s of servers means **zero administration** is a must
  - Data volume adds up very quickly so **low storage overhead** is required
  - Data flows into the cloud continuously and must be **processed in real-time**
  - Must be able to handle **time series, spatial, and NoSQL data** natively
- Additional requirements
  - Must be able to **scale-out**
  - Must be available as a **service**

The database must be able to *ingest, process and analyze data in real-time*

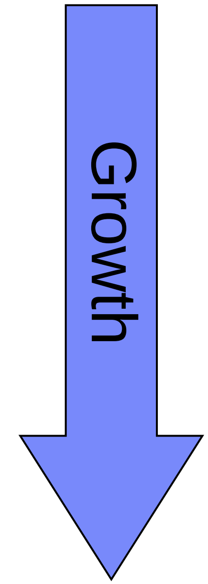
# What are the Analytics Requirements for an IoT (Edge-) Gateway?

- Very similar core requirements to the Cloud, but...
  - Have a small install footprint, **less than 100 MB**
  - Run with low memory requirements – **less than 256 MB**
  - Use **compression or other techniques** to minimize storage space
  - Have built-in support for common types of IoT data like **time series and spatial/GIS** data
  - Simple application development with support for both **NoSQL and SQL**
  - It must require **absolutely no administration**
  - Ideally it should be able to **network multiple gateways** together to create a single distributed database

The database must be powerful enough to *ingest, process and analyze data in real-time*

# Sensor Data Management – A typical Approach

| Sensor_ID | TimeStamp | phase1 | phase2 | ... | temp |
|---|---|---|---|---|---|
| 1 | 2010-06-01 00:00 | 1.3 | 0 | | 15.6 |
| 1 | 2010-06-01 00:30 | 1.6 | 0 | | 15.6 |
| 1 | 2010-06-01 01:00 | 1.4 | 0 | | 15.5 |
| 1 | 2010-06-01 01:30 | 1.4 | 0 | | 15.4 |
| 1 | 2010-06-01 02:00 | 1.4 | 0 | | 15.5 |
| ... | | | | | |
| 2 | 2010-06-01 00:00 | 0.4 | 0 | | 12.3 |
| 2 | 2010-06-01 00:30 | 0.3 | 0 | | 12.3 |
| 2 | 2010-06-01 01:00 | 0.2 | 0 | | 12.2 |
| 2 | 2010-06-01 01:30 | 0.5 | 0 | | 12.3 |
| ... | | | | | |
| 3 | 2010-06-01 00:00 | 0.0 | 3.5 | | 13.6 |
| 3 | 2010-06-01 00:03 | 0.0 | 4.3 | | 12.2 |




- Each row stores only the values for one meter per measurement = **Billions of data rows ☹**
- Meter IDs and time stamps need to be stored in each record = **Storage overhead ☹**
- Additional Indexes are required for fast data access = **Even more additional storage ☹**

# Sensor Data Managemet – The IBM Informix Advantage

| Sensor_ID | Origin | 00:00 | 00:30 | 01:00 | 01:30 | ... |
|---|---|---|---|---|---|---|
| 1 | 2010-06-01 | (1.3,0...15.6) | (1.6,0...15.5) | (1.4,0...15.5) | (1.4,0...15.4) | |
| 2 | 2010-06-01 | (0.4,0...12.3) | (0.3,0...12.3) | (0.2,0...12.2) | (0.5,0...12.3) | |
| 3 | 2010-06-01 | {v1: 1.5} | {v1: 9.1} | {v1: 0.3} | {v1:0.3,v2:4} | |

Growth →

- Only one data row per sensor (e.g. Smart Meter)
- Optimized sensor data storage (clustered, compressed, high frequency data)
- Time series elements can be JSON documents
- Built-In time series data high performance loader API (> 60000 values/sec/CPU core)
- Simplified application development (>100 built-in SQL time series functions)
- Available also for embedded platforms (e.g. ARM or Intel Quark CPUs)

That leads to…

(*) For regular Time Series

**…reduced disk space usage (about 60% less) ☺**

**…significantly increased performance and scalability ☺**

**…faster data load and access ☺**

# Proof Point 1: TimeSeries Meter Data Management Benchmark

- *30 million smart meters sending data every 15 minutes*
- *2.88 billion records inserted each day*
- *Workload: data Ingestion, data cleanup, and a daily billing cycle*

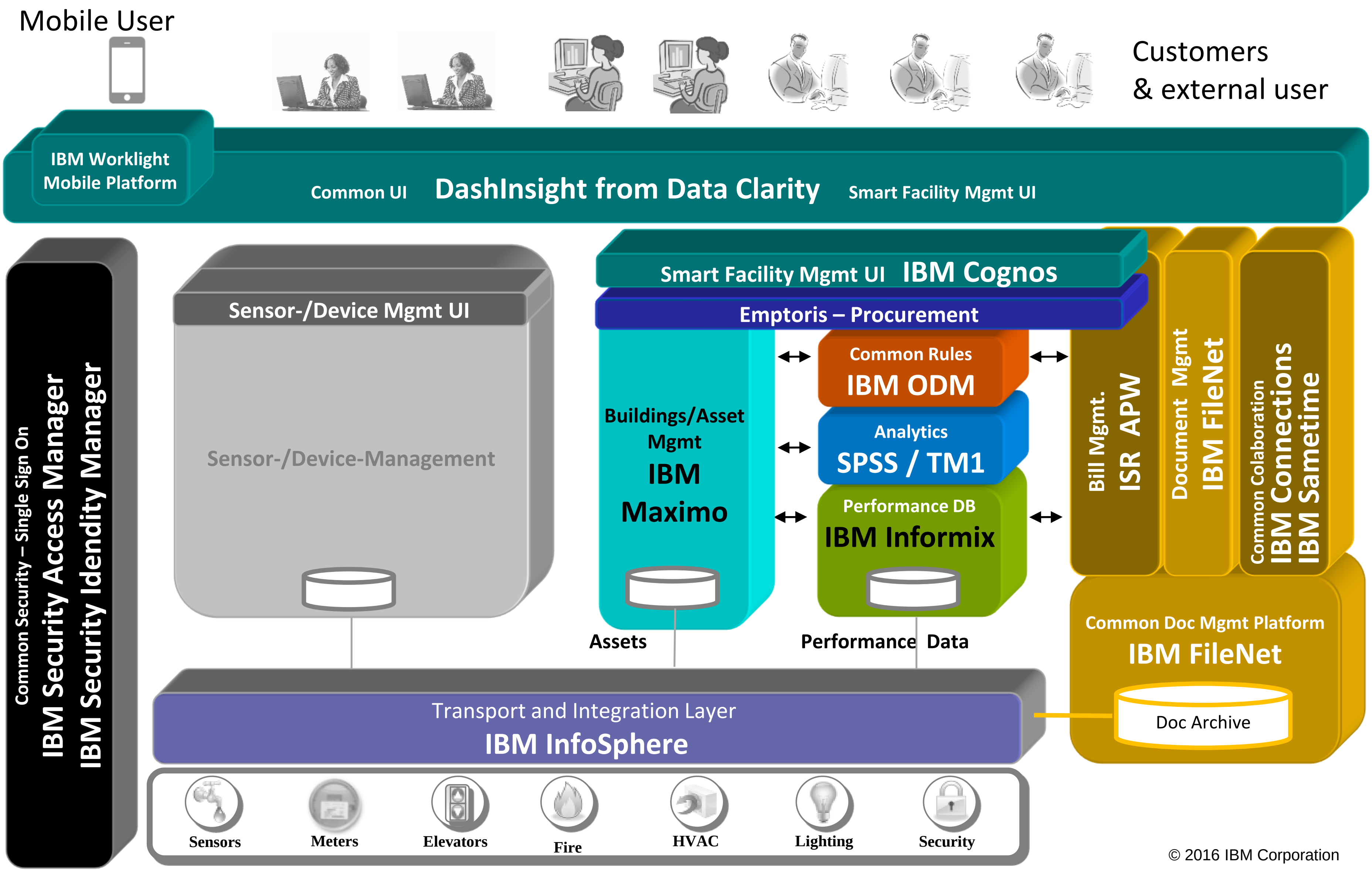
| Metric | Competitor | Informix |
|---|---|---|
| Daily processing time | **11 hours** | **5 hour 50 min** |
| Maximum number of cores used | **62** | **22** |
| Size of database per month of data | **15TB** | **5TB** |
| # Records processed each day | **2.88 Billion** | **2.88 Billion** |
| Billing determinants creation (1/21 of the total meter population) | **51,322** | **~2 million reads per second** |

# Proof Point 2: Oncor's Advanced Meter System (Texas, USA)

Now in Production with > 3.5 million meters!

- Very Successful Meter Data Management Scalability PoC
- Informix took about **18 minutes** to validate and load a day's worth of data for 1 million meters (data collected at 15 minute intervals)
  – Their prior RDBMS took about **5 hours**
- Informix took about **6 minutes** for each mandatory ERCOT compliance report and **about 25 seconds** if the data was already cached in memory
  – Their prior RDBMS took from **2 to 3 hours** depending on the report
- Disk space used by Informix was about **350 GB**
  – Their prior RDBMS used about **1.3TB** for the POC
  - Their estimate for storing 3 years worth of 3.5 million meters was **55 TB**
  - Our estimate for 3 years worth of 3.5 million meters using the Informix TimeSeries was **15 TB**
- Results were very linear for Informix
  – Better results if you increase cpus and storage
  – If less performance is OK then cpus and storage can be reduced

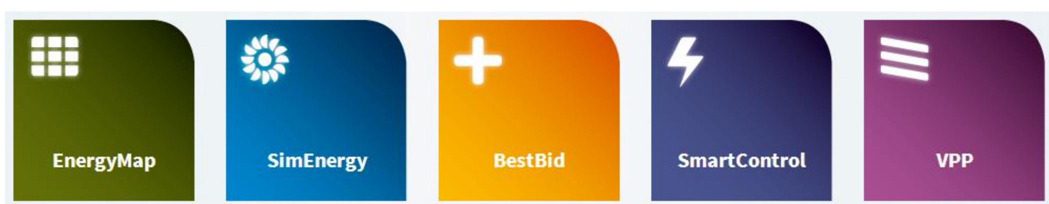
# Use Case 1: Smart Building / Facility Management



# Use Case 2: Energy Optimization - Misurio AG (Switzerland)

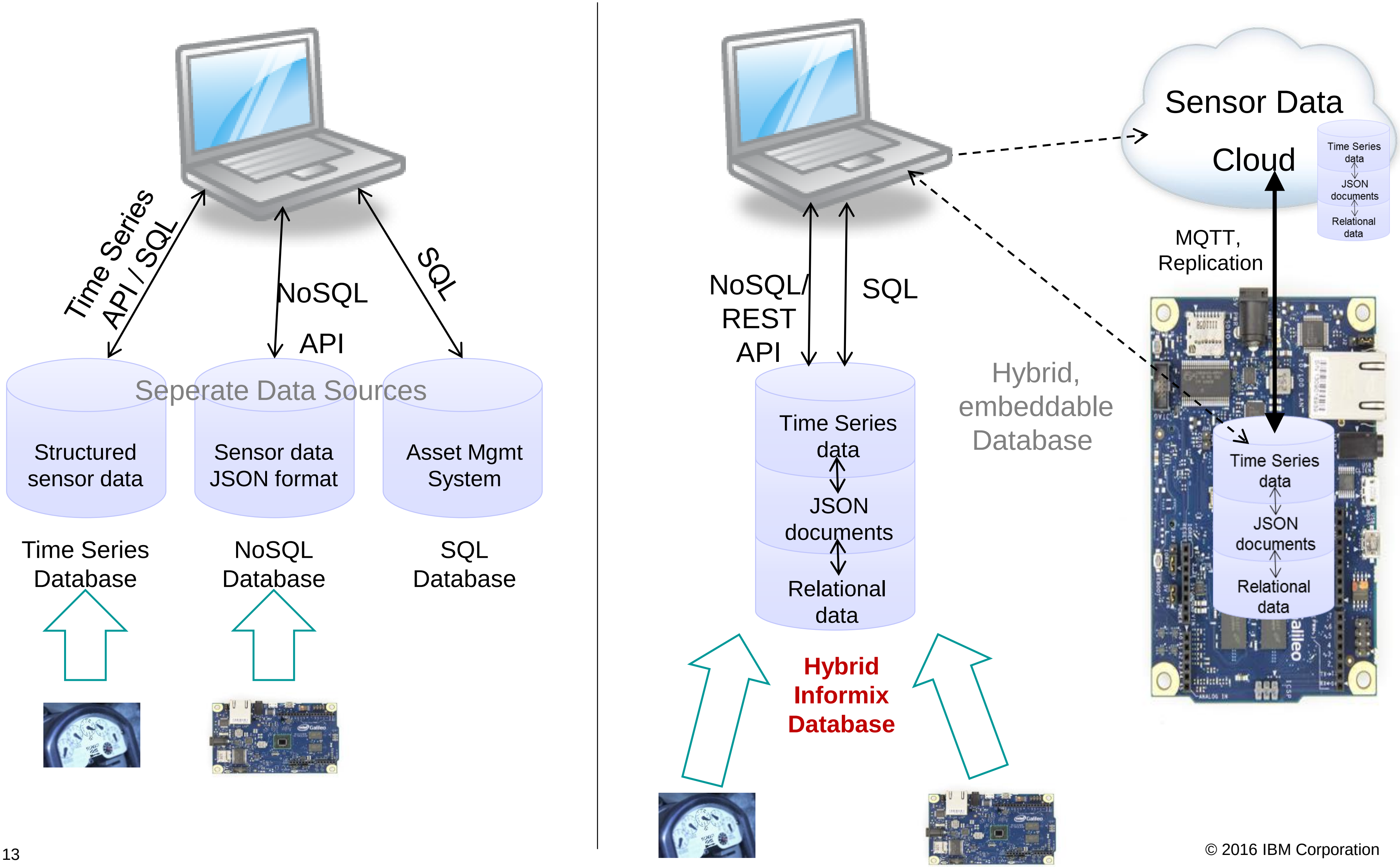
- The Misurio AG (Visp, Switzerland) is focusing on mathematical optimizations for the E&U industry
  - Lead product: The EnergyOn Platform
  - Input: Time series data from different data sources (energy data, meteorological data, energy markets etc.)
  - Output: Price- and demand-forecast with adjustable accuracies
- The Problem: The original sensor database (MySQL) for the Misurio Optimizer didn‘t scale with the growing requirements of Misurio‘s customers
- The Solution: Informix 12.10 with optimized TimeSeries, JSON and REST support as the new sensor database for the Misurio Optimizer

**Misurio Optimizer Applications**




| Research | Simulation | Proposals | Operation Optimization | Virtual Powerplants |
|---|---|---|---|---|

# Simplified application development: Just one hybrid and embeddable Informix DB instead of multiple, separate data sources

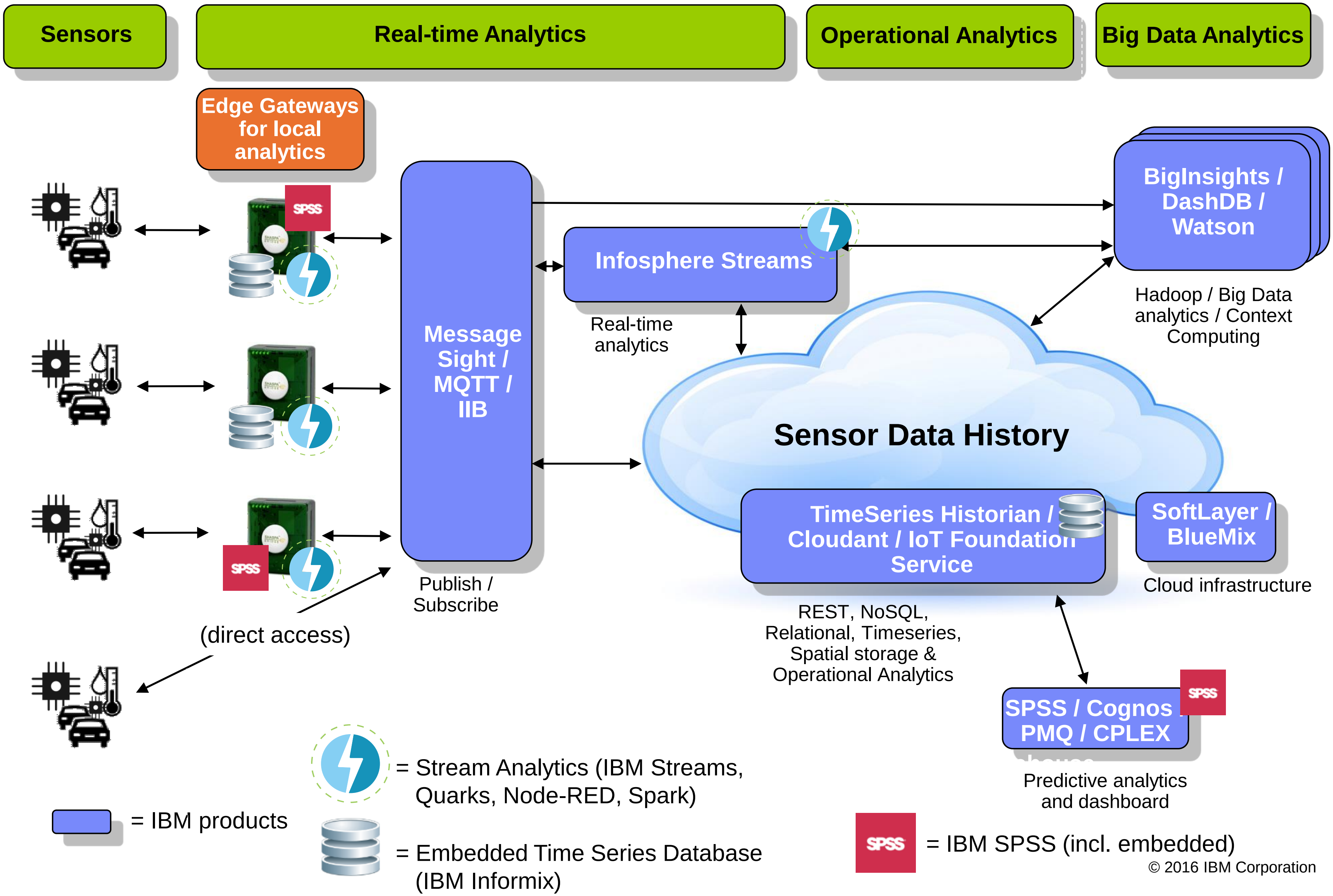


# Proof Point 3: Informix vs SQLite Embedded Platform Benchmark

| Tests on Intel Quark | Informix | SQLite |
|---|---|---|
| Data loading – high-speed performance (records per second) | **950 / 1050 (DK100 / DK50)[1]** | 700 (Average)[2] |
| Storage space that is required for 1 day of data | **275 MB** | 1200 MB |
| Aggregation query (seconds) | **2 secs** | 4-25 secs |
| Moving average (seconds) | **25 secs** | 259200[3] secs |
| Missing interval search (seconds) | **2 secs** | 14-30 secs |

1. The two figures for data loading with Informix reflect a slight difference in performance between the DK100 and DK50. DK100 had more running components causing a drop in performance vs DK50
2. Data loading with SQLite had significant variations in load performance as the database size increased.
3. The moving average result for SQLite is a projected figure that is based on a partial result after 10 minutes.

# IoT/Industry 4.0 Architecture Annotated with IBM Products



# Some IoT-/Industry 4.0-Gateway Use Cases...

- **Industrial Production**
  - Local Shop Floor Sensor Data (Pre-)Processing / Cleansing / Monitoring (Predictive)
  - Shop Floor Communication Protocol consolidation (Industrial Cloud Gateway with Analytic Capabilities)
- **Automotive**
  - On-Vehicle Data Buffering / Processing / Monitoring (Predictive)
- **Smart Facility-/Building-Management**
  - In-Building Sensor and Actor Data Collection and Control
  - Local Energy Optimization Capabilities with Cloud Access for more Complex Optimizations
- **Sports Statistics / Fan Experience**
  - In-Stadium Sensor based Statistics (Ball, Player and Referee Positions, Movements, Goals/Touchdowns)
- **Wind Turbine Monitoring**
  - Off-Site or Off-Shore locations with limited communication bandwidth require local operation data monitoring to optimize preventive maintenance

# How does Gateway Sensor Data processing helps IoT solutions?

- **IoT Gateways can reduce the cost of the backend cloud**
  - Reduces cloud storage by filtering/aggregating/analyzing data locally
  - Reduces cloud CPU requirements by precomputing values
- **Support of Real-Time requirements**
  - Intelligent gateways can detect and respond to local events as they happen rather than waiting for transfer to the cloud
- **Sensor Data Analytics move close to the data**
  - Gateways allow customers to capture and get value from their sensors without necessarily sending all data into the cloud
- **Protocol Consolidation**
  - Cloud does not need to implement the multitude of IoT protocols
- **Data Quality Optimization**
  - Duplicate elimination, Data Cleansing, Missing Values Handling, Sorting etc.

# IBM Powered / Cloud-Integrated IoT-/Edge-Gateways

EUROTECH
Imagine. Build. Succeed.

Custom M2M/IoT Gateways
MQTT, Informix
OSGi based Custom Protocol Stack

hilscher
COMPETENCE IN COMMUNICATION

netIOT Edge Gateway for Industrie 4.0/IoT
MQTT, Informix, SPSS
Industry Communication Protocols

ADLINK TECHNOLOGY INC.

PMQi Cognitive Gateway
MQTT, SPSS, Spark

PHŒNIX CONTACT

mGuard Security Gateway
QRadar SIEM

intel

intel IoT Gateways
MQTT, Informix

# Summary: Informix - The Sensor Data optimized Hybrid Database

- **Superior IoT Sensor Data Performance and Scalability in the Cloud, on Premises and on Edge Gateways**
  - Innovative Sensor Data Storage and Processing w/o the need for costly all-or-nothing In-Memory database technologies
  - Large Scale and Embedded Sensor Data Benchmarks

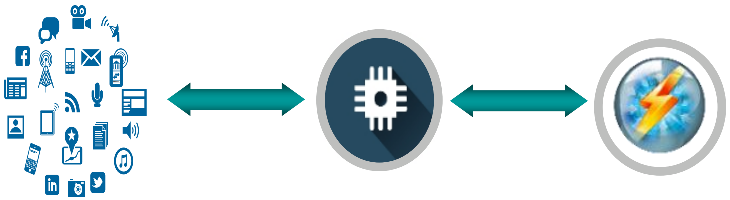
- **Simplified Application Development due to Integration of all Sensor Data relevant data types in one hybrid Database**
  - Relational, Time Series, JSON and Geospatial (all joinable!)
  - JSON based Time Series for full Schema Flexibilty
  - Free Developer Editions and Docker Images available

- **Open, Standardized APIs allow easy Integration into existing Architectures**
  - SQL, REST, MongoDB, Node-RED, MQTT, JDBC, ODBC, .NET etc.

- **Very Cost Effective Operations in all Deployment Models**
  - Very efficient HW utilization (less disk space, less CPU & Memory) compared to other approaches
  - Attractive license models

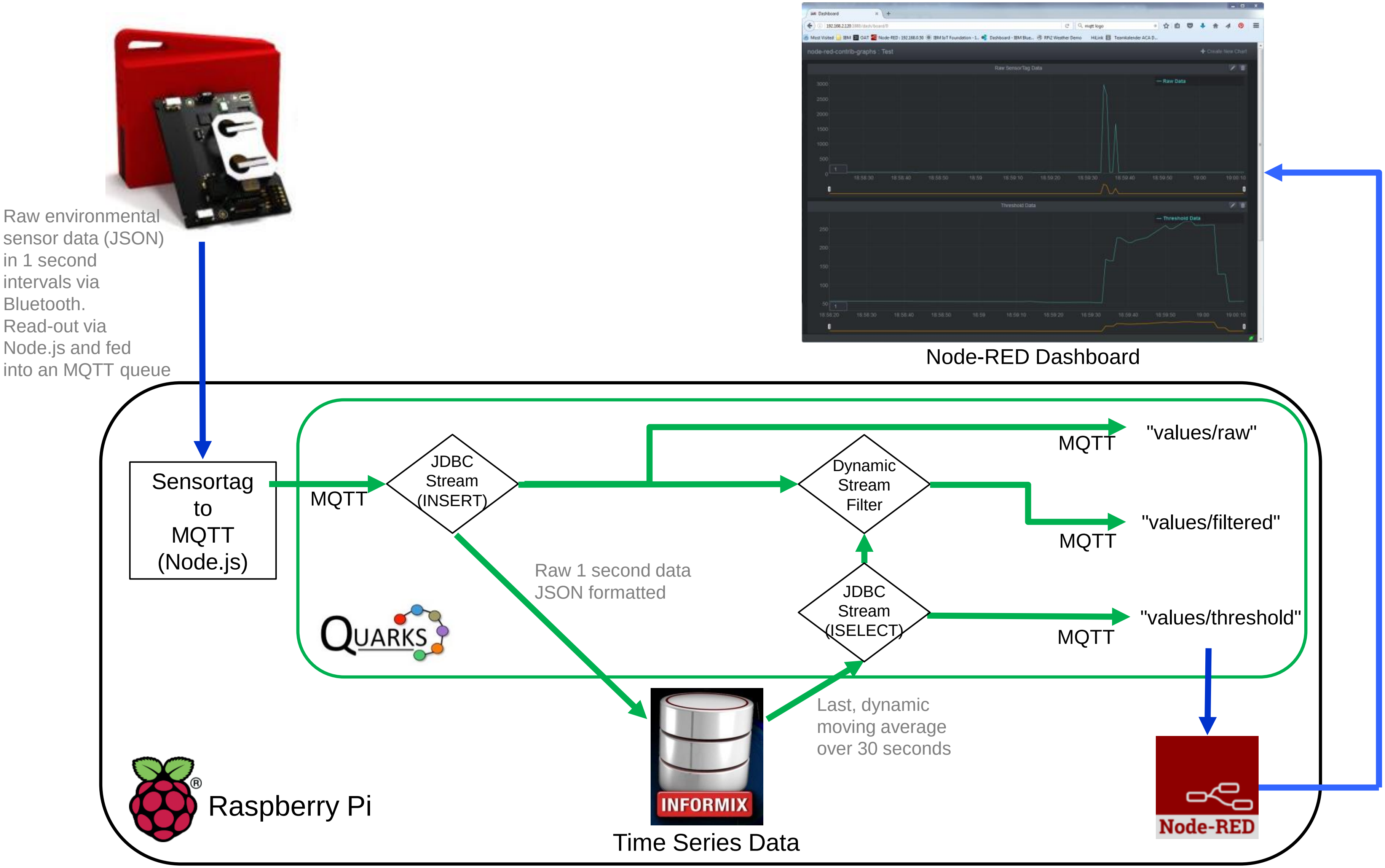
# Watson IoT Platform & Streams

- Quarks for edge analytics – on device or gateway
  - Lightweight embedded streaming analytics runtime
  - Analyze events locally on the edge
  - Reduce communication costs by only sending relevant events
- IBM Watson IoT Platform for optional device hub
  - Device management
  - MQTT message broker



- IBM Streams for streaming analytics
  - High performance, full featured streaming analytics
  - Build windows of state and correlate across devices
  - Have access to data-of-record systems, e.g. medical history
  - Control edge device based upon analytics
  - Central job management/health summary
  - Automatic application connectivity

# Live Demo: Sensor Data Flow - Quarks & Informix on a Raspberry Pi



# In-Motion & At-Rest Analytics on the Edge: Quarks & Informix

```
DirectProvider tp = new DirectProvider();
Topology t = tp.newTopology("InformixQuarksSensorTagMQTT");
JdbcStreams mydb = new JdbcStreams(t, () ->  InformixConnector.getIfxDataSource() ,
        (dataSource) -> dataSource.getConnection() );

MqttConfig mqttConfig = new MqttConfig("tcp://localhost:1883", "");
MqttStreams mqtt = new MqttStreams(t, () -> mqttConfig);
TStream<String> msgs = mqtt.subscribe("sensortag/cc2650", 0);
```

Create a new Quarks stream from an MQTT queue

```
msgs = msgs.map(tuple -> new JSONObject(new JSONObject(tuple), new String[] {"tstamp","lux"}).toString());
mydb.executeStatement(msgs,
 () -> "INSERT into quarks_json_ts_v values (?,?,?)",
 (g, stmt) -> { String ts = EpochtoTS(new JSONObject(g).getLong("tstamp"));
                stmt.setString(1, "RAW");
                stmt.setString(2, ts);
                BSONObject bson = new BasicBSONObject("value_r", new JSONObject(g).getDouble("lux"));
                stmt.setObject(3, new IfxBSONObject(bson));
} );
```

Insert raw sensor data from stream into an Informix time series

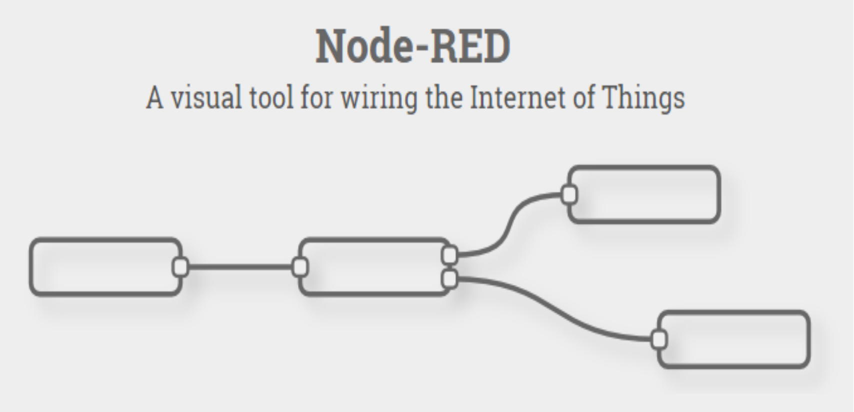
```
mqtt.publish(msgs, "values/raw", 0, false);

TStream<String> threshold = mydb.executeStatement(msgs,
        () -> "select elem.timestamp, elem.value FROM (select GetLastElem(Apply('TSRunningAvg($value_r, 30)',
                                                 CURRENT - 30 UNITS SECOND, CURRENT, sensor_values)::TimeSeries(one_value_t))
                   from quarks_json_ts where sensor_id = ?) AS t(elem);",
        (g, stmt) -> stmt.setString(1,"RAW"),
        (g, rs, exc, consumer) -> {
                if (exc != null) { System.err.println("Unable to execute SELECT: " + exc); }
                else {
                        rs.next();
                        Double last = rs.getDouble("value");
                        String ts = rs.getString("timestamp");
                        JSONObject jo = null;
                        if (ts != null)
                        {
                          jo = new JSONObject().put("threshold", setThresholdValue(last)).put("tstamp", TStoEpoch(ts));
                        }
                    consumer.accept(jo.toString());
                }
            }
        );
mqtt.publish(threshold, "values/threshold", 0, false);

msgs = msgs.filter(tuple -> new JSONObject(tuple).getDouble("lux") > thresholdValue);
```

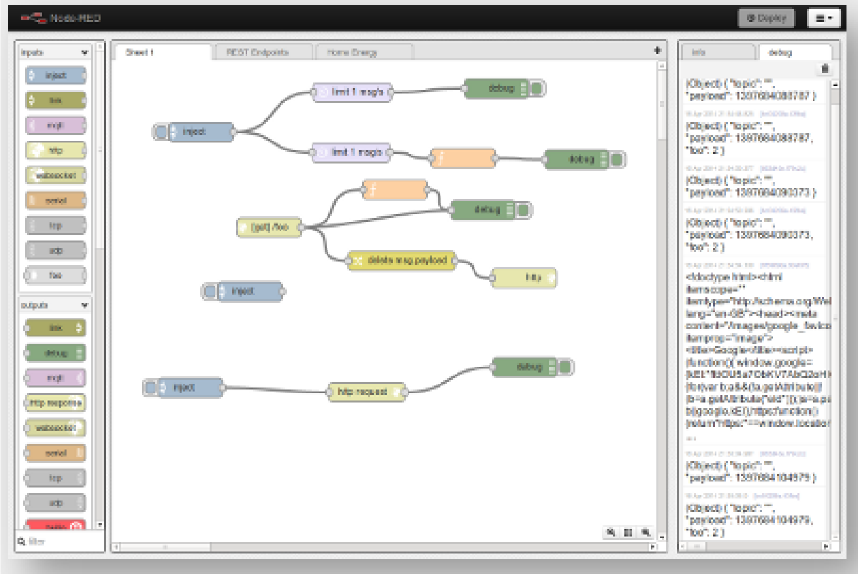
Dynamically calculate moving average from raw time series data

Apply dynamic moving average as a dynamic filter to the stream

# What is Node-Red?

- nodered.org
- A visual tool for wiring the Internet of Things
- Node.js application.  Uses building blocks of reusable code to control flow of data



- "Flows" are created to process data.
- Example:  Take data in, transpose, process and store in a database and/or publish to a message broker
- Collaboration among node developers

# The Watson IoT Platform - our next generation IoT platform

## IBM IoT Foundation Offerings

### IBM IoT Foundation Connect
Attach, Collect & Organize, Device Management, Secure Connectivity, Visualization

### IBM IoT Foundation Information Management
Storage & Archive, Metadata Management, Reporting, Streaming data, Parsing and Transformation, Manage unstructured data

### IBM IoT Foundation Analytics
Predictive, Cognitive, Real-time, and Contextual

### IBM IoT Foundation Risk Management
Security Analytics, Data Protection, Auditing/Logging, Firmware Updates, Key/Cert Mgmt, Org Specific Security

# What is Bluemix?

Bluemix is an **open-standards**, cloud-based **platform** for **building**, **running, and managing applications.**

### Build your apps, your way

Use the most prominent compute technologies to power your app: Cloud Foundry, Docker, OpenStack.

### Scale more than just instances

Development, monitoring, deployment, and logging tools allow the developer to run and manage the entire application.

### Extend apps with services

A catalog of IBM, third party, and open source services allow the developer to stitch an application together quickly.

### Deploy and manage hybrid apps seamlessly

Get a seamless dev and management experience across a number of hybrid implementations options.

### Layered Security

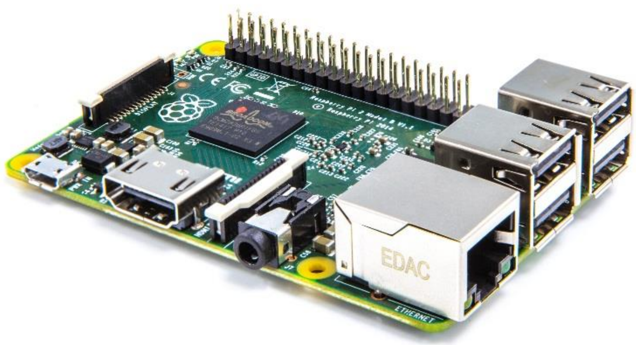
IBM secures the platform and infrastructure and provides you with the tools to secure your apps.

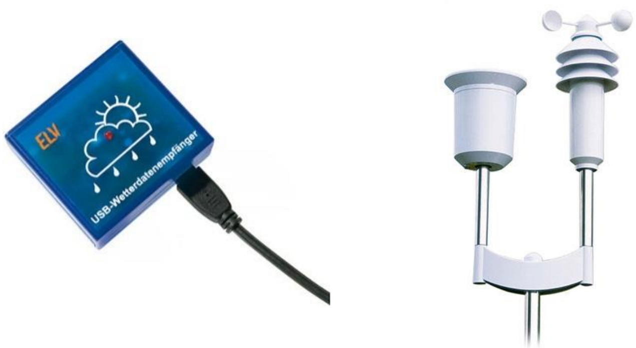
### Flexible Pricing

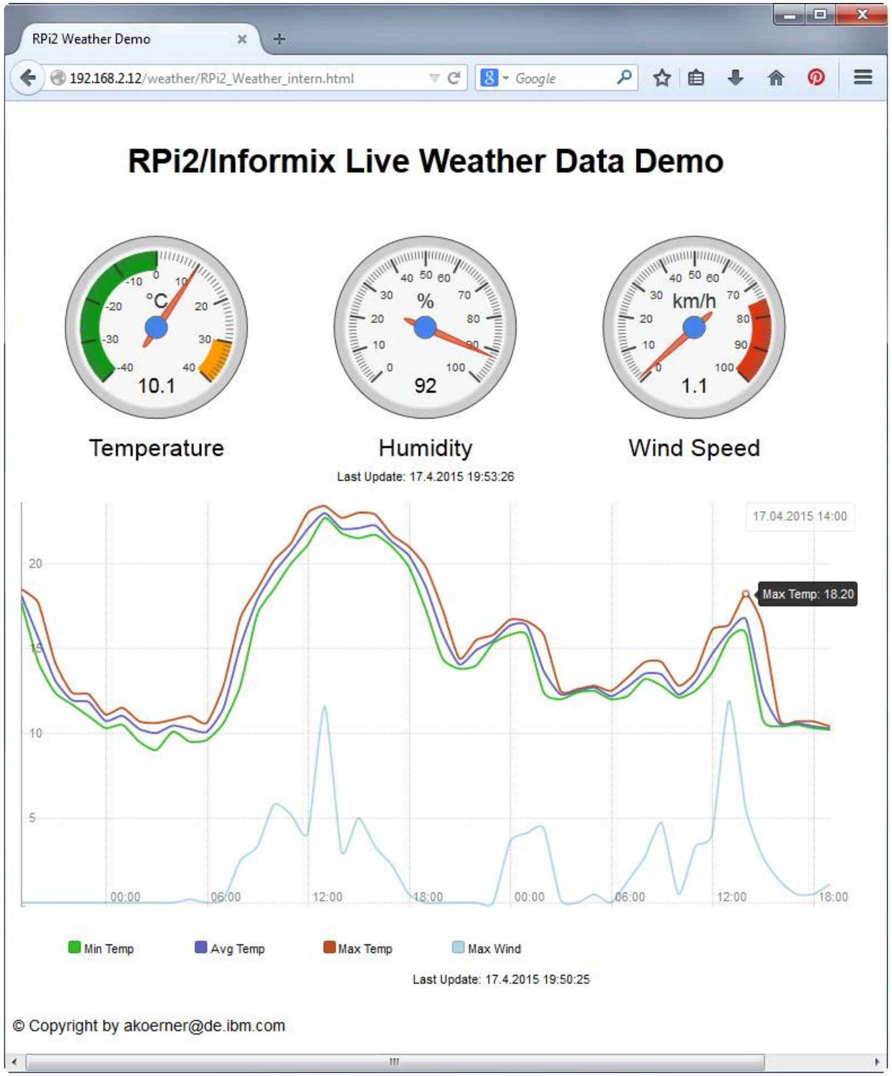
Try compute options and services for free and, when you're ready, pay only for what you use. Pay as you go and subscription models offer choice and flexibility.

# My IoT@Home Project ☺

- An Informix TimeSeries, REST API, JSON & Raspberry Pi 2 based Weather Station...







# Thank You


IBM

**Alexander Koerner**
*Analytics Architect*
*IBM Analytics*
*Industrie 4.0 & IoT*
*Informix SME*

*Hollerithstrasse 1*
*D-81829 Muenchen*
*Mail: Postfach 820364*
*D-81803 Muenchen*

*Phone +49-89 4504 1423*
*Fax +49-160 90737756*
*Mobile +49-160 90737756*
*akoerner@de.ibm.com*

Twitter: @AlexKoeMUC

# Backup Slides

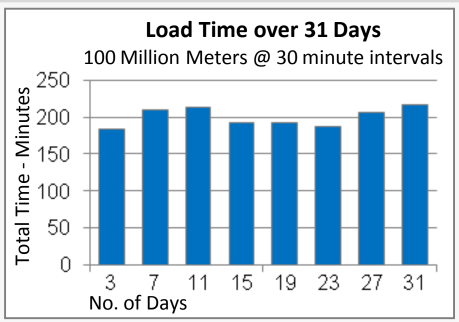
# Proof Point 4: Informix / AMTSybex 100 Mio Meter Benchmark

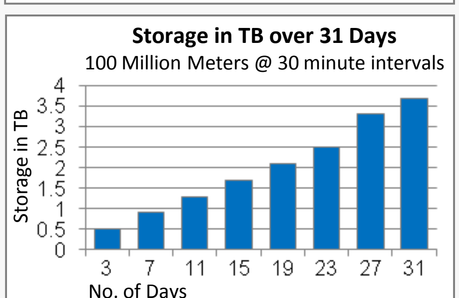
### 10 Million Meter Benchmark Results (36 minutes – total time)

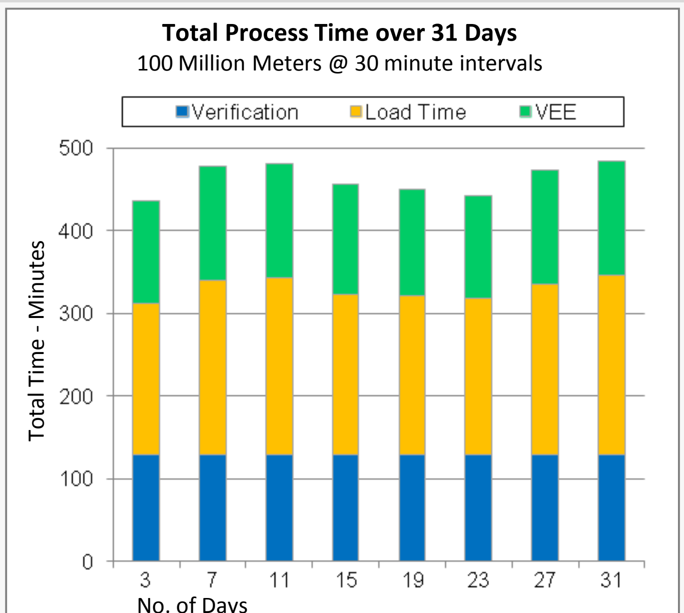
| Process | Elapsed Time | Avg. Throughput Rate |
|---|---|---|
| Preparation and Technical Verification | 10 min 02 sec | 797,342 records/sec |
| Data Load | 13 min 56 sec | 457,162 records/sec |
| Validation, Estimation, and Editing (VEE) | 11 min 18 sec | 707,964 records/sec |

### 100 Million Meter Benchmark Results (7 hrs 35 minutes – total time)

| Process | Avg. Elapsed Time | Avg. Throughput Rate |
|---|---|---|
| Preparation and Technical Verification | 2 hrs 10 min | 628,205 records/sec |
| Data Load | 3 hrs 14 min | 420,962 records/sec |
| Validation, Estimation, and Editing (VEE) | 2 hrs 11 min | 623,409 records/sec |



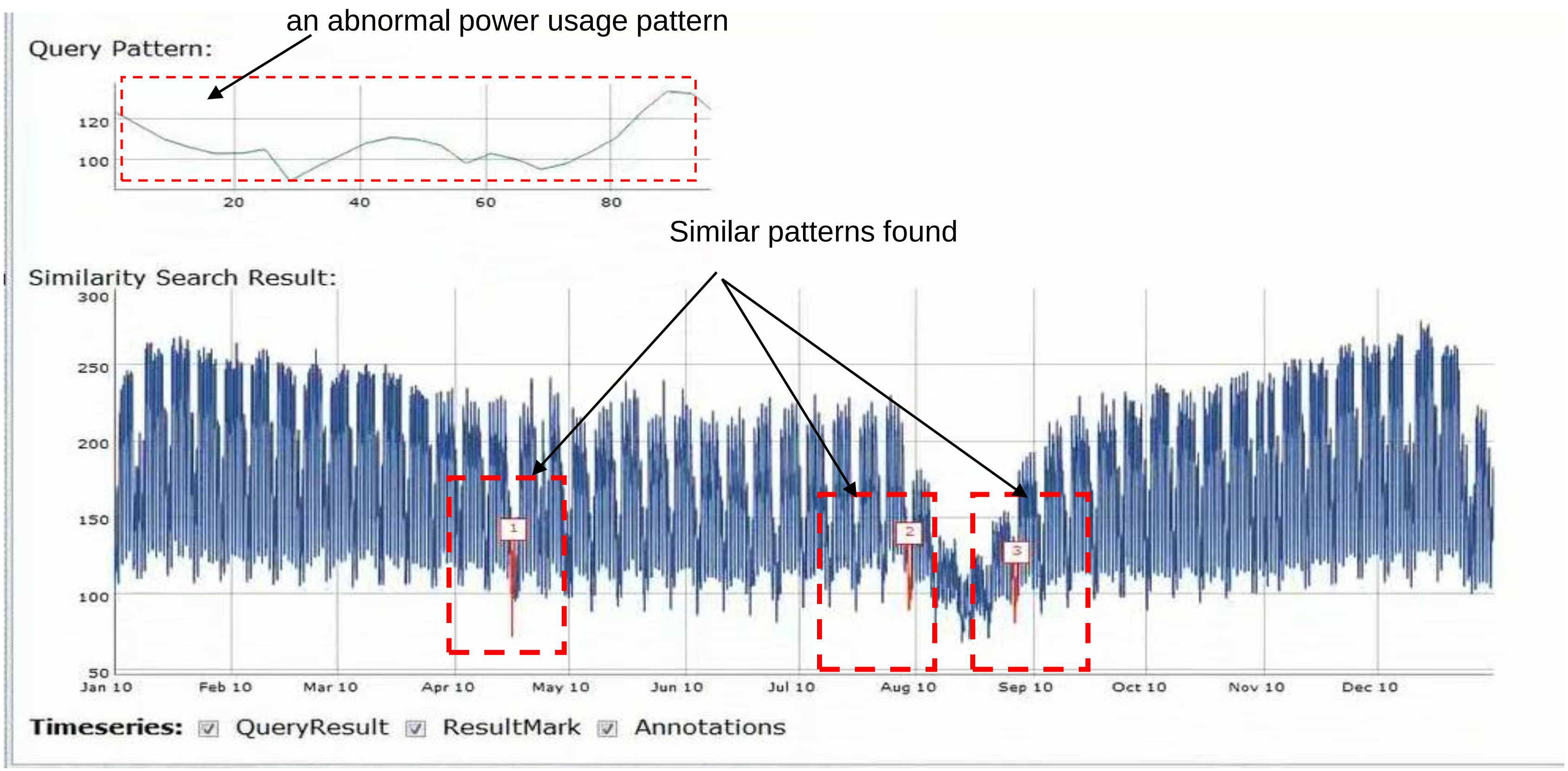




## Benchmark Results

- End to end processing time of 36 minutes for **10** Million meters at 30 minute intervals.
- End to end processing time of < 8 hours for **100** Million meters including billing queries.
- Processed 31 days of data using <4TB of XIV storage on a single 16 core Power 750 machine.
- Overall processing time remained constant with increased historical data.
- Storage used remained **linear** with increasing data.

# Informix Time Series Data Pattern Matching Search

- A Search tool for finding similar looking patterns not just an exact match




**A scenario of pattern matching for abnormal power usage detection**

# Spatiotemporal Search (STS) for Moving Objects

- **Manage location data (eg. GPS data) in an Informix TimeSeries**
  - Time based analysis
- **Use Cases**
  - Find the location of a moving object at a specific time
  - Find the last known time and location of a specific moving object
  - Find when a moving object was in a region around a point of interest
  - Find the trajectories of moving objects near a point of interest during a time range

  ...and more
- **Build index structures**
  - Time range and spatial analysis
  - Using R-Tree indexing
- **Operations**
  - Location questions base on time or timerange
  - Time questions base on location
  - Time and location questions

# Time Series data & JSON Integration

- Schema flexibility for time series applications
- Create a time series that contains JSON documents
- JSON documents are unstructured data
- No schema changes required, if the structure of the data changes
- Easy to load and query the data
- Ability to store time-based data as JSON / BSON docs
- Each JSON data element is accesible for time series functions (e.g. for aggregation)
- Supports time series manipulation through the MongoDB- & REST-API

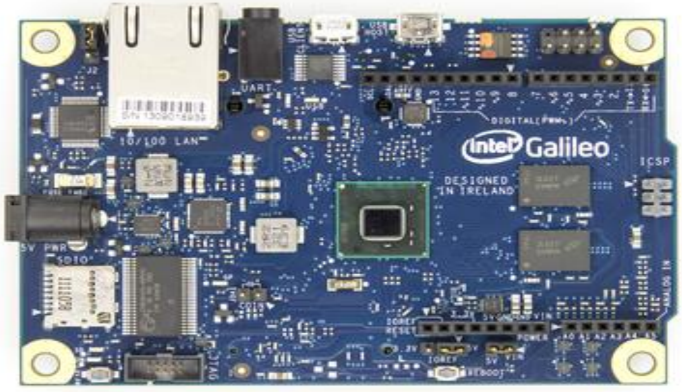
```
CREATE ROW TYPE ts_data_j (tstamp datetime year to fraction(5),
                           sensor_data BSON);

CREATE TABLE tstable_j (id int not null primary key,
                        ts  timeseries(ts_data_j));

EXECUTE PROCEDURE TSCreateVirtualTab(virt_tstable_j, tstable_j);

INSERT INTO virt_tstable_j values(1,
  "2014-01-01 01:00:00.00000",('{"v1":2.1,"v2":20.1}'::json)::bson);

INSERT INTO virt_tstable_j values(1,
  "2014-01-01 01:15:00.00000",('{"v1":2.1,"v2":20.1,"v3":7.9”}'::json)::bson);
```

# IBM Informix 12.10 on ARM

- **Full Informix 12.10 Edition, with TimeSeries and GeoSpatial**
- **Minimum Informix 12.10 – ARM System Requirements**
  - Supported ARM processors
    - ARM V6 (armv6l)
    - ARM V7 (armv7l) with floatingpoint support
  - Minimum 256 MB RAM (for Linux and Informix), recommended 512 MB
  - Storage space for the database files (e.g. SSD, SD, microSD etc.)
  - Minimum product deployment diskspace (w/o database files, customizable): < 100 MB
  - One of the following supported Linux distributions for ARM
    - Debian
    - Ubuntu
    - Raspian Wheezy / Jessie
    - Support for other Linux distributions upon request
  - Free Developer Editions available!

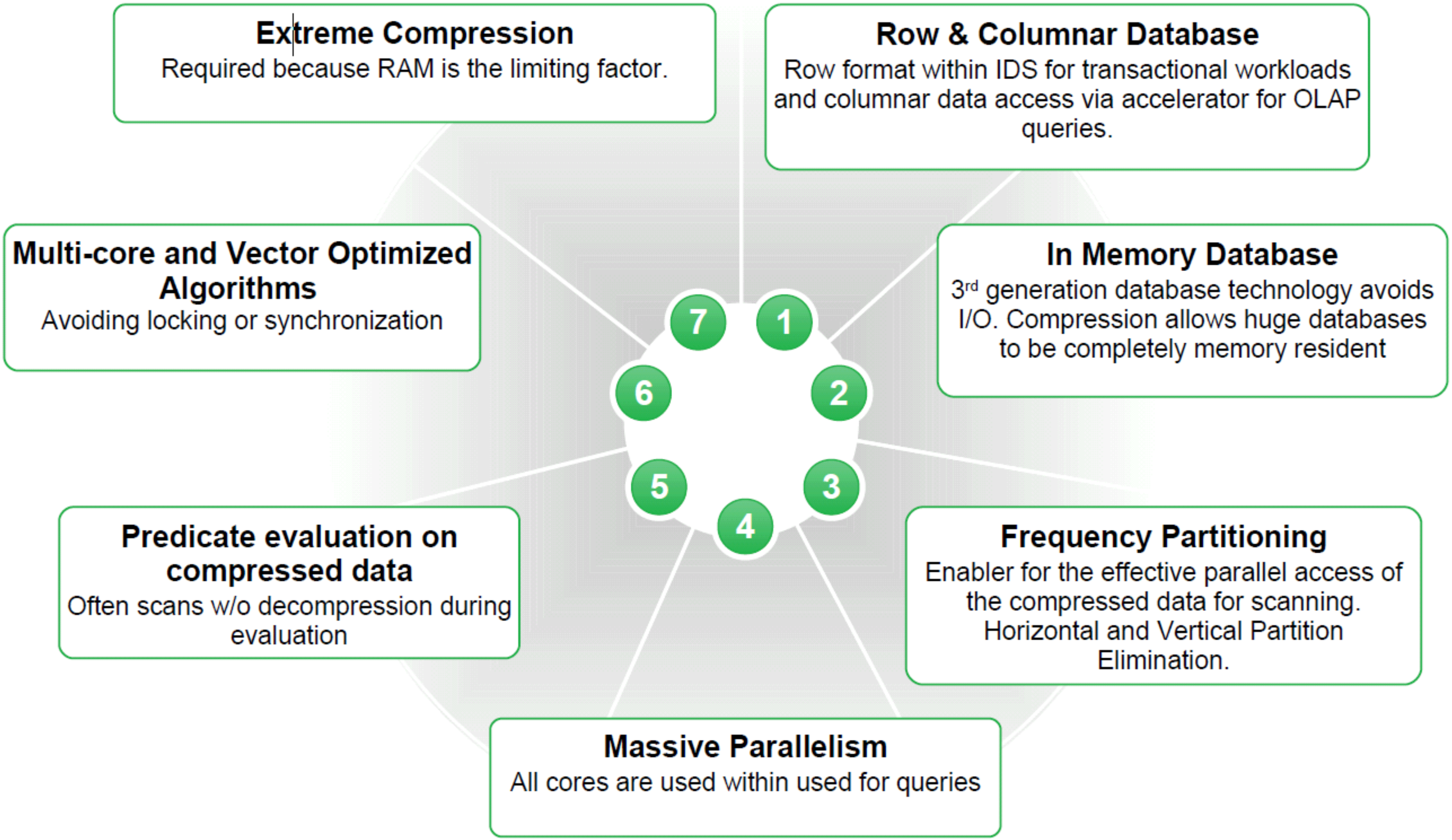
# IBM Informix 12.10 on Intel Quark / Atom

- **Full Informix 12.10 Edition, with TimeSeries and GeoSpatial**
- **Minimum requirments for  Informix 12.10 – on Quark / Atom**
  - 400MHz 32-bit Intel® Pentium
  - Minimum 256 MB RAM (for  Linux and Informix), recommended 512 MB
  - Writable media like  SSD, SD, microSD for Database data
  - Mimimum disk space for the database  < 100 MB
  - Linux Distributionen for Galileo board
    - Debian
    - Wind River Linux



# IBM Informix’s optional In-Memory Technology for Sensor Data Warehouses

# Informix (In-Memory) Warehouse Accelerator (IWA)



# Example for the IWA / Sensor Data Performance

- Use Case: 280000 simulated smart meters, 12 months of data, 15 minute intervals
  - About 9.8 billion measurements
  - About 232 GB in Memory consumption (compared to 524 GB on Disk in time series format or about 1.3 TB in relational format w/o Indexes)

- Execution of the following SQL-Query:

```
SELECT  METER_ID, YEAR(TIME_OF_MEASUREMENT) As JAHR,
        COUNT(*) AS ANZAHL,
        SUM (R.AVG) AS VERBRAUCH,
        MIN (TIME_OF_MEASUREMENT) MIN_TOM,
        MAX (TIME_OF_MEASUREMENT) MAX_TOM,
        MIN (HT) MIN_HT,
        MAX (HT) MAX_HT,
        MAX (HT) - MIN (HT) AS VEBRAUCH_AUS_HT
FROM READINGS_TAB R
      WHERE TIME_OF_MEASUREMENT >= "2013-04-01 00:00:00.00000"::DATETIME YEAR TO
FRACTION(5)
      AND TIME_OF_MEASUREMENT <= "2014-03-31 23:45:00.00000"::DATETIME YEAR TO FRACTION(5)
GROUP BY 1, 2
      ORDER BY 1, 2;
```

- The measured run time for that query (incl. the transfer of the 560000 result records to the client) was only about 58 seconds! ☺
  - If the query only fetched the first 2 data rows, the run time was about 43 seconds

# Links

- IBM Informix Database (Hybrid Sensor Database)
  - http://www-01.ibm.com/software/data/informix/
- IBM Informix product downloads (incl. free editions)
  - http://www-01.ibm.com/software/data/informix/downloads.html
- IBM Informix on the Raspberry Pi (HowTos)
  - https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=97199&p=674959 (Informix Intro and Installation)
  - https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=97772&p=678497 (An Informix Sensor DB - Part 1)
  - https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=100029&p=693935 (An Informix Sensor DB - Part 2)
  - https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=137392&p=912405 (The Informix REST API)
  - https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=140398 (Round robin sensor data storage with IBM Informix)
- IBM Redbook about the Informix TimeSeries Technology
  - http://www.redbooks.ibm.com/abstracts/sg248021.html?Open
- IBM Internet of Things Foundation
  - http://internetofthings.ibmcloud.com/
- IBM Bluemix
  - http://www.ibm.com/software/bluemix/
- Node-RED
  - http://nodered.org/