TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

ANTON MAKLAKOV

DESIGN AND IMPLEMENTATION OF INFORMATION WAREHOUSE FOR MANUFACTURING FACILITY SUPPORTING HOLISTIC ENERGY MANAGEMENT

Master of Science Thesis

Examiner: Prof Jose L. Martinez Lastra
Examiner and topic approved by the
Faculty Council of the Faculty of
Engineering Sciences
on 4 December 2013.

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Machine Automation
MAKLAKOV, ANTON: Design and Implementation of Information Warehouse for Manufacturing Facility Supporting Holistic Energy Management
Master of Science Thesis, 68 pages, 3 Appendix pages
March 2014
Major subject: Factory Automation
Examiner: Professor Jose L. Martinez Lastra
Supervisor: Anna Florea
Keywords: Energy Management, Energy Efficiency, Information Warehouse, Big Data, NoSQL, Cassandra, Service Oriented Architecture, Complex Event Processing, Manufacturing

Energy management is one of the most critical tasks, which needs to be performed in manufacturing facility, since manufacturing consumes 1/3 of world's energy. Same time, manufacturing facilities are equipped with large amounts of field devices, which generate vast amounts of information every second. With such a huge amount of real-time data, which has a potential to provide insight information for energy management needs, capturing, storing and processing of it becomes a challenge. In this thesis an information warehouse system supporting holistic energy management is designed and implemented. The main goal is to provide a system, which can capture, store and provide information relevant for energy management purposes in manufacturing facility.

The thesis consists of three main parts. In the first part current and most relevant for energy management concepts and technologies, including Big Data, NoSQL, Service Oriented Architecture and Complex Event Processing, are explored, analyzed and compared. In the second part an architectural design of information warehouse is presented. During this step a set of tools and technologies is selected for implementation. In a third part, an information warehouse system is implemented and tested in a manufacturing line test-bed.

Implemented information warehouse is based on multi-layered architectural pattern, where layers are communicating with each other via services. The most important advantage of this modular architecture is an ability to use implemented solution in any manufacturing facility, as modules can be easily reconfigured in order to adjust to different context. The designed information warehouse system was tested for a manufacturing line located in premises of Tampere University of Technology. The results of this thesis demonstrate that the developed information warehouse system is capable of collection, processing and providing access to crucial for energy management information.

PREFACE

It is very hard to believe, that my student life journey comes to its end. I have left my home country 2.5 years ago to study in Tampere University of Technology, and I can confidently say that it has been the most valuable experience of my life so far.

First of all, I would like to thank Professor Jose L.M. Lastra and all personnel of FAST Laboratory for providing interesting, broad and challenging courses. Also, I am very thankful to Prof. Lastra for selecting me to be admitted to the Machine Automation Degree program in the first place, and having faith in me throughout the whole studies.

Secondly, I would like to express my gratitude to my supervisor Anna for guiding me through the process of writing thesis, offering help and advices. Without her valuable feedback and deep understanding of thesis writing process, this thesis work wouldn't be the way it is now.

Thirdly, I am very thankful for my parents and my family for the huge support and love that they always share with me.

Fourthly, I am very grateful to my friends, who have always been ready to offer their full support, and with whom I could always share my thoughts and feelings: Ahmed, Mahmud, Ville, Juha, Zoran, Sohail. Also, I am thankful to my friends, who have not been physically with me here, but still could provide a mental support: Sebastien and Ilya.

Finally, and most importantly, I would like to thank my fiancée Rebekka for the incredible support, patience, carefulness and love. Thank you for sharing the best and the hardest moments with me, and being always close and ready to help in any situation.

Tampere, March 19th, 2013

Anton Maklakov

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ACRONYMS

IT	Information Technology
IoT	Internet of Things
NoSQL	Not Only SQL
BI	Business Intelligence
EDW	Enterprise Data Warehouse
RDBMS	Relational Database Management System
SQL	Structured Query Language
KPI	Key Performance Indicator
API	Application Programming Interface
CEP	Complex Event Processing
WS	Web Service
OSGi	Open Services Gateway Initiative
ESB	Enterprise Service Bus
EnMS	Energy Management System
QoS	Quality of Service
ACID	Atomicity, Consistency, Isolation, and Durability
CAP	Consistency Availiablity Partition
DBMS	Database Management System
CQL	Cassandra Query Language
SOA	Service Oriented Architecture
WSDL	Web Service Description Language
UDDI	Universal Description, Discovery and Integration
XML	eXtended Markup Language
XSD	XML Schema Definition
SOAP	Simple Object Access Protocol
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
URI	Unique Resource Identificator
JSON	JavaScript Object Notation
EPL	Event Processing Language
POJO	Plain Old Java Object
JPA	Java Persistence API
JPQL	Java Persistence Query Language
DAO	Data Access Object
IoC	Inversion of Control

CRUD	Create, Update and Delete
JAR	Java Archive
HMI	Human-Machine Interface
RMS	Root Mean Square
CAMX	Computer Aided Manufacturing using XML
LAN	Local Area Network

1. INTRODUCTION

Manufacturing has been a driving force for the growth of world economy for the last three centuries. Technical developments in manufacturing industry do not only affect growth of productivity and innovations, but also face a need to reduce energy consumption and carbon footprint [1]. Nowadays, manufacturing organizations consume one third of the world's energy. Due to this fact, authorities have realized the need to invest into improvement of energy management solutions.

According to report "Intelligent Manufacturing: targeting better energy efficiency" [2], the most common way to reduce the energy consumption in factories is installation of energy efficient lighting, conditioning and heating systems. However, the future of efficient energy consumption lies in an area of software development. It can be explained by the fact, that monitoring and control of energy consumption is crucial for effective energy management. It can only be achieved by development of software applications, which could collect all energy related information from the facility and analyze it. Therefore, advanced methods of processing and managing of energy data carry huge opportunities for improving the energy management in manufacturing facilities.

1.1. Background

Starting from the beginning of 21st century the amount of digital data started to increase rapidly as Web Technologies have been developing. The term Big Data emerged by the middle of 2000s when the major IT companies, including Google, Yahoo! and Amazon had to find solutions to deal with enormous amount of Web data arriving at very high speed [3]. According to reports world-wide, almost 90% of world's data has been generated in the past 2 years, and it is predicted that by the year 2025 the amount of data in the Internet will surpass the brain capacity of whole planet's population [4].

Such a dramatic growth of digital data is directly connected with the effect of phenomena called "Internet of Things" (IoT), which is nowadays one of the most important enablers of modern industrial development [5]. Appearance on global market of cheap and affordable storage systems, sensors and various communication devices has also led to the dramatic increase of digital data [6].

The concept of Big Data includes size of data and technologies needed to effectively process it and present its structure. Emergence of Big Data has resulted in a shift from traditional relational data management approaches to constantly developing

new data management strategies with NoSQL (Not Only SQL) platforms at their core. With these data management strategies it becomes possible to transform information from different independent sources (human-sourced, process-mediated and machine-generated information) into extensive resource of business information. In other words, Big Data management systems allow to get rid of different informational contexts and provide a universal information sphere, where every aspect of physical world and every event occurred in it can be captured and recorded [3]. The most important values brought by Big Data to a business include transparency of data for interested parties, deep level of detail for all available information, possibilities to perform advanced analytics, and opportunity to develop innovative and efficient services [7].

In manufacturing facilities sensors, actuators and controllers are generating vast amounts of real-time data, covering all processes and operations on-going on the factory floor, and which needs to be collected, processed, stored and analyzed in order to have improved process control and decision making.

This problem can be solved with the help of Big Data solutions based on technologies, which emerged in the past couple years, mainly built on open-source project of Apache called Hadoop, which is intended for reliable and scalable distributed computing [8]. Google has also invested a lot of resources in development of MapReduce technology for parallel processing of large data sets. Moreover, new technologies arise in the IT community, which are intended for the Big Data handling, including advanced Business Intelligence (BI), cloud computing, complex event processing and NoSQL databases [9]. NoSQL systems allow eliminating a need to have a fixed structure of the data and making it possible to scale out databases when the amount of data grows. In other words, NoSQL databases are capable of storing all incoming unstructured data independently and can be flexibly extended with the increase of data flow. Also, complex event processing enables to extract valuable information from the data on a real-time as data enters the system, based on complex patterns and conditions.

However, it is very important to understand that adopting Big Data to the manufacturing facility information system requires huge changes in infrastructure and data handling principles. Traditional Enterprise Data Warehouse (EDW) systems, which are currently widely used, are based on Relation Database Management Systems (RDBMS) with Structured Query Language (SQL) databases at its heart. Traditional EDW requires having a strongly defined structure for the data, where all categories of data are connected and related with each other. But traditional EDW is not ready to face a vast flow of unstructured incoming data, and it is not possible to scale relational databases. Another factor, making traditional approach ineffective for processing large amounts of data, is transactional nature of RDBMS, where database operations are optimized for processing information which needs to be always consistent. Same time, data arriving from smart meters, sensors, actuators and controllers is a real-time log data, which is sent from devices often every second, needs to be stored, processed and always be available to be requested. Given to these factors, the latest developments in a

warehousing allow neglecting these problems by shifting to NoSQL infrastructures, where data is collected, organized and analyzed [10]

1.2. Problem Definition

1.2.1. Justification for the work

According to report "Big Data Comes of Age" [3] the access to Big Data in manufacturing is mainly intended for business analysts, business executives and IT analysts. This means that usage of Big Data in manufacturing organizations is focused on the enterprise level functions, where responsible parties are directly interested in improving production rates, quality control and incomes, while area of energy management is not considered as a use case of Big Data handling. Moreover, manufacturing industry can be classified as a late adopter of Big Data technologies, meaning that, comparing to other industries, like media, finance and leisure, is Big Data solutions are now being only investigated and planned to be implemented.

Given to the fact, that software development and data processing is a key for improved energy efficiency in factories, facing Big Data is an important step needed to be made. Big Data handling carries huge opportunities for getting valuable insights into energy consumption information, which, with Big Data technologies implemented, can make a big and real change.

1.2.2. Problem statement

It is important to realize the need of shifting to modern warehouse system, intended to make use of the Big Data, in manufacturing facilities. Same time it is crucial to understand that Big Data helps to gain useful information and gives a ground not only for a better production planning, decision making, but also enables improved monitoring of energy consumption trends. The main question that needs to be answered is:

- *How information relevant for energy management purposes can be captured, stored and made transparent in manufacturing facility?*

1.3. Work description

The main purpose of this thesis work is to create an infrastructure of information warehouse, capable of turning large amounts of data, coming from factory shop floor devices, into valuable information, which allows detecting energy trends and improving energy consumption patterns. The warehouse infrastructure will consist of a data acquisition, complex event processing and database management system and services providing access to all the available data to anyone who might need it. The variety of

different parties, interested in the data, should be taken into consideration, in order to develop universal and flexible methods for providing the data.

1.3.1. Objectives

1. Develop modular system, which allows to decrease energy consumption and carbon emissions in manufacturing facilities and provides tools for efficient energy management
2. Design and implement an information warehouse system capable of capturing and storing energy related information in manufacturing facility
3. Demonstrate applicability and effectiveness of NoSQL database management system in manufacturing domain
4. Make energy efficiency information available for interested parties

1.3.2. Methodology

The declared objectives will be achieved in two main steps. First, a review of academic literature will be conducted, in order to analyze existing solutions for energy management and relevant technologies. Based on technologies discussed in review, an infrastructure for information warehouse will be designed and implemented. These two steps can be divided into several sub-steps, which are described below.

Theoretical review

In order to achieve objectives of the thesis, the most relevant concepts and technologies need to be selected. The research about existing technologies and already existing solutions will be held in order to analyze their applicability for manufacturing domain and most relevant technologies will be selected for implementation.

Methodology for architectural design

In order to complete an implementation, first overall architecture will be defined. The architecture will follow principles of layered pattern. The main elements of the information warehouse system are as following: Devices, Application for Data Processing and Presenting, Database Management System. Devices in the manufacturing system represent all sensors, energy meters and controllers, which generate a real-time energy data that needs to be processed by the application. Interactions between the elements will be described, in order to present the flows of information.

Database management system deployment

In this step a database management system, which will provide a possibility to physically store the manufacturing facility data needs to be selected. Based on a technologies review, the most appropriate database will be selected for implementation. In order to provide ability for proper functioning of database, a data model will be also designed.

Application development

Application will be responsible for providing of interface between the manufacturing system and the database, allowing all incoming data to be stored, analyzing the data in real-time in order to generate useful and meaningful information in form of Key Performance Indicators (KPI) for energy management. Also, application will provide a set of Web Services, which will be used as an Application Programming Interface (API) to retrieve all required data by responsible personnel. The following steps are required in order to fulfill requirements for the application.

- **Implementation of data endpoint**
  In order to deliver data from the facility inside of the application, a communication link between devices and application server will be set by creating an endpoint inside of the application, which can subscribe to the all available data in the manufacturing facility
- **Implementation of KPI computation**
  When data is continuously delivered to the application, it becomes available for processing and analysis. Complex Event Processing (CEP) engine is the central element of the application. By using even processing rules, the engine is capable of processing the incoming data and calculating KPIs on-the-fly. CEP engine will be configured in order to correctly handle incoming messages, and rules will be defined.
- **Development of Data Access layer**
  Data Access layer is a communication layer between the application and the database. It maps data inside of the application to the data model of database, thus providing easy and flexible way to save and retrieve data from the database.
- **Development of API**
  API is intended to provide an access to the data for all interested parties. API will be implemented in form of Web Services (WS), which allow accessing the data by different applications, built in any possible programming language.
- **Development of KPI definition interface**
  User interface is needed when personnel, responsible for energy management, wants to add new or edit current KPIs and rules for their calculation in the application.

- **Integration to OSGi infrastructure**
  Proposed solution will be also integrated to Enterprise Service Bus (ESB) messaging system, where it will be accessible inside for other components of enterprise infrastructure.

1.3.3. Assumptions and limitations

This thesis work has a set of assumptions and limitations that had an effect on a design and development of information warehouse system.

1) The main use case of the work is energy management, therefore energy related information is prioritized comparing to the rest of information available inside of manufacturing facility
2) Structure of raw data is predefined in controllers of manufacturing facility; therefore it might vary in different vendors. In this work only messages, which are generated by controllers of implementation test-bed, are considered
3) In order to achieve the best result from implementation of proposed solution, at least 3-4 database servers are required. However, only one server was available at the time when this thesis work was done.

1.4. Thesis outline

This thesis is structured as follows: Chapter 2 presents a review of literature containing technologies and concepts relevant for the thesis work. In Chapter 3 a methodology approach for implementation of the thesis and used tools and frameworks will be presented. In Chapter 4 implementation of the thesis is presented in details. Chapter 5 describes the implementation of proposed solution and presents the results. Chapter 6 provides a final conclusion of the thesis work.

2. THEORETICAL BACKGROUND

This chapter provides literature and technology review. Current solutions of information warehouses architecture and energy management in manufacturing domain are analyzed. Chapter 2.1 focuses on Energy Management concepts, principles and already existing solutions. Chapter 2.2 provides description of Big Data and overview of relevant technologies. Database management system technologies are reviewed in details with focus on NoSQL in Chapter 2.3. Also, two most advanced NoSQL databases are analyzed and compared in this chapter. Chapter 2.4 is focused on Service Oriented Architecture and Web Services. Chapter 2.5 provides a review of main concepts of Complex Event Processing.

2.1. Energy Management

Energy Management can be defined as a set of measures, which are designed and implemented with a purpose of minimizing of energy consumption. Energy Management System (EnMS) is concerned with capturing of energy data in order to provide a basis for decisions regarding energy efficiency [11].

Manufacturing industry consumes 1/3 of world's energy and emits 1/3 of carbon dioxide. European Commission has set a target of reducing yearly consumption by 20% by 2020 [12]. This means that energy management becomes one of the most crucial tasks in manufacturing and requires a lot of attention. There are examples of companies in a process industry, which had to reduce or stop their production, when electricity prices were high [13]. K.Bunse et al. in their research [12] list 3 most important drivers for introduction of energy efficiency improvements in manufacturing companies (Table 1).

Table 1: Energy Efficiency introduction drivers (adapted from [12])

Driver	Comment
Rising energy prices	Prices for oil, gas and other fossil fuels are continuously rising, therefore resulting in need to reduce consumption, especially in energy-intensive manufacturing industries
New environment regulations concerning CO2 emissions	Companies have to reduce energy consumption, which consequently results in reduction of carbon dioxide emissions, in order to be able to face challenges and costs resulting from CO2 regulations
Shift of customer's preferences to green and efficient products	Energy efficiency in manufacturing has a considerable effect on company's competitiveness, as end users treat energy efficiency in usage of product as one of the most important criteria in a purchasing decisions

In other words, energy efficiency has a huge impact not only on the environmental situation in the world, but also on the economic development and social role of individual manufacturing companies and industry as a whole. Therefore, main aspects of energy efficiency in manufacturing can be defined, which are presented on a Table 2.

Table 2: Main aspects of energy efficiency in manufacturing (adapted from [12])

Economic aspects	Environmental aspects	Social aspects
Energy costs	CO2 emissions	Energy-awareness of customers and workforce
Resource costs	Resource scarcity	Ensure resource and energy security for future generations
Cost internationalization	Other emissions	Interaction with political and company stakeholders
Risk of future liability cost		Image in society
Resource productivity		

According to [14] in order to improve energy efficiency, personnel responsible for energy management needs to follow a cycle "Plan, Do, Check and Action", where the first step includes identifying key energy performance indicators, adhering to rules that affect management systems, highlighting energy benchmarks, setting energy targets and designing an efficient energy management platform. Identification of energy factors

gives a comprehensive picture of the energy consumption in facility and possible way to improve resources management (Figure 1).

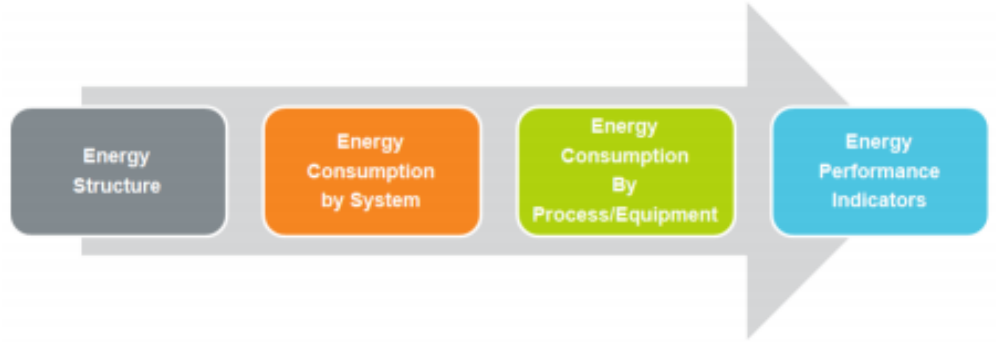

Figure 1: Energy factors identification flow (adopted from [14])

Moreover, "planning" step includes data acquisition [11], or in other words collection and provision of energy information, forming a basis for energy efficient decisions and actions. Measuring and controlling of energy consumption in manufacturing facilities is the first step towards energy efficient manufacturing [12].

Energy data in manufacturing industry should be collected on all discrete levels in a facility with a time intervals ranging from milliseconds to years. However, energy monitoring and controlling is not enough to provide efficient energy management, as holistic approach to energy efficiency is needed, meaning that data from all levels of factory floor and factory building must be correlated and evaluated [15].

2.1.1. Key Performance Indicators

KPI allows getting a relevant for improvement of energy management information out of raw data, coming from devices in a manufacturing facility. These indicators usually show the amount of benefit derived from particular energy use and can be compared with both internal and external targets. Table 3 demonstrates a classification of energy performance indicators, described in research of [15].

Table 3: Energy Performance Indicators classification

Indicators of inefficiencies in facility's energy usage (consumption profiles)
Indicators used to catalyze energy efficiency improvements and tracking of changes
Indicators projecting energy usage onto monetary values
Indicators intended for improvement of input, output and measurement points understanding

A lot of research work has been conducted in the area of energy KPI definition and analysis. Table 4 summarizes energy KPIs that have been defined and analyzed in literature [12, 15, 16].

Table 4: Energy KPIs

Key Performance Indicator	Description
Unit Energy Consumption	Energy consumption to produce one unit of economic output (product)
Specific Energy Consumption	Total energy consumption in facility with value-added and non-value-added operations
Process Energy Consumption	Energy consumption for process
Significant Energy User Consumption	Energy Consumption of most energy-demanding production units in facility
Energy Consumption Per Product	Total energy consumption of producing one product by facility
Power Consumption	Single or average power used by process
Energy Cost	Projection of consumed energy onto monetary value
Specific Energy Cost	Energy cost per one product
Energy Losses	Energy consumption of non-value-added operations
Energy Efficiency	Percentage of energy input to process against output of energy
Final Energy Efficiency Improvement	Energy savings per year

Furthermore, temporal resolution of KPIs is an important characteristic, enabling holistic energy management, which can provide long-term, medium-term and short-term trends of energy consumption (Table 5) [17].

Table 5: Temporal classification of energy KPIs

Time resolution	Comments
Year	Long-term energy consumption trends
Month	Allows to detect seasonal variations
Week	Medium-term trends. Helps to identify baseline energy demand
Day	Identifies energy consumption variations during days if week
Hour	Identifies daily trends and behavior of energy consumption, allows to optimize production schedule

2.1.2. State of the art in energy management

Problem of energy management in industrial and manufacturing domain has been addressed in many research works in past years. Common structure and concepts can be noticed in energy management systems proposed in various sources. Particularly, the most fundamental functionality of energy management system is collection and storing of data from field level devices. Workflow of energy management system for chemical manufacturing, designed by Drumm et al. in [18], consists of two steps – *Energy Efficiency Check* and *Energy Efficiency Management* (Figure 2), where at the first step all energy-related data, containing information about total energy consumption and energy costs, is collected, analyzed and stored in database. Vikhorev et al. in [15] also implements a real-time data acquisition as core functionality of energy management framework. Same time in [19] data collection step is only performed after energy profiles, energy indicators and relevant devices are identified and selected.

Figure 2: Workflow for Energy Efficiency management (adopted from [18])

The second important functionality of reviewed energy management systems includes static [19] and dynamic [15, 18] analysis of collected data with a purpose of obtaining KPIs, charts and metrics, which need to be monitored and compared with energy efficiency targets. The most wide-spread approach for real-time calculation of KPIs uses complex event processing on a stream of events incoming from devices, as it is described in [15].

The final step provides users of energy management system with feedback in a form of continuous monitoring of energy KPIs, decision support dashboards and visualization, in order to optimize energy consumption and improve energy efficiency. In [18] a reporting tool is implemented, which presents relevant energy consumption and energy processes together with monitoring, which can be scaled to different time resolution starting from daily to annual. Architecture of the proposed software tool is illustrated on a Figure 3.

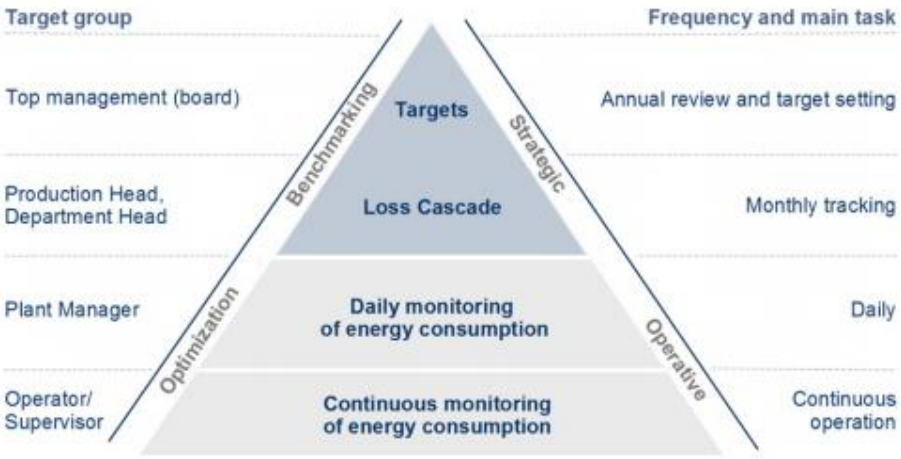

Figure 3: SCTRUCTese Integrated efficiency management tool (adopted from [18])

According to these fundamental steps, universal model of energy management system can be created, as illustrated on a Figure 4.

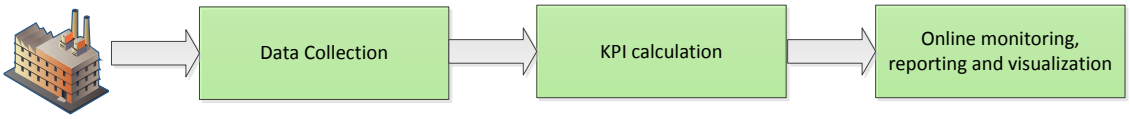

Figure 4: Unified energy management system workflow

However, with a broad amount of sources of data in manufacturing facilities, processing and monitoring of energy-related information faces a problem of common link and integration between all available systems. May et al. in [17], as part of European project PLANTCockpit, addresses a problem of integration of information from various vendors inside of manufacturing systems for building an enhanced energy management system. Cannata et al. in [20] discusses benefits of cross-layer architecture for energy efficiency, as it allows performing context-aware control and decoupling business processes.

Energy management in manufacturing has been a target of many research works in the past few years, which all share similar architectural approach, concerned with utilizing of energy data by collecting and transforming it to valuable KPIs.

2.2. Big Data

As it was stated in section 2.1, energy-related information from manufacturing facility is required for improvement of energy efficiency. Also, it was discussed in Introduction that amount of information coming from various devices on factory shop floor is very large, and often is referred as Big Data.

There are many different interpretations and definitions of Big Data available in literature. In [9] Big Data is defined as

"Data sets that grow very large or fast, so that they are difficult to handle using traditional technology",

In TechAmerica Foundation's report [21] more specific definition is offered:

"*Big Data is a term describing large volumes of high velocity, complex and variable data, which requires advanced techniques and technologies to enable the capture, storage, distribution, management and analysis of information*".

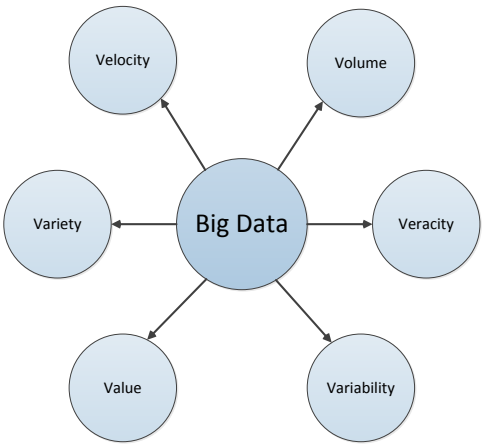

Figure 5: 6 Vs of Big Data

Big Data is traditionally characterized with 3 "V-words" : *variety, velocity and volume* [10, 21, 22], and even more characteristics are being introduced: *veracity* [23], [24, 25], *value* [10, 25], *variability* [26, 27] (Figure 5). Summary of these attributes is presented on a Table 6.

Table 6: Description of Big Data attributes

Attribute	Description
Volume	Represents amount of data. Approximately 2.5 Zettabytes (*1ZB=10^{21}bytes=1000 Exabyte=1 billion terabytes* [28]) of data were generated world-wide in 2012. Figure 6 shows estimation of data growth until 2020.
Velocity	Speed and frequency, with which data is being produced, changed and received. Velocity results in latency - a delay between a moment when data is created or saved and a moment when it is available.
Variety	Represents incredible amount of information of different types (*structured, semi-structured, unstructured*), coming from various sources. Results in a crucial requirement for database management systems and data warehouses to be able to dynamically adapt to various changing data formats by being capable of scaling
Veracity	Describes a level of reliability and trustworthy of an incoming data.
Value	Identifies which value data carries and how this value can be extracted for further analysis.
Variability	Assumes that data flow is inconsistent and may have varying semantics.

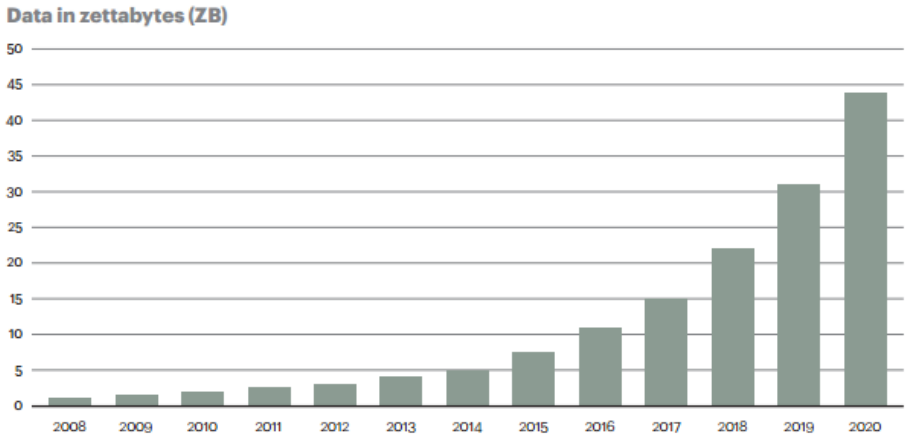

Figure 6: Data growth estimation (adopted from [29])

In order to better understand the nature of Big Data, it is important to thoroughly classify its sources and their relations. As defined in [3] sources of Big Data can be divided into 3 groups: *human-sources, process-mediated and machine-generated data,* which are described in Table 7.

Table 7: Three types of Big Data

Type of data	Description
Human-sourced	Subjective expression of human's experience. It is a biggest source of unstructured data that is often unreliable, thus cannot be a basis for critical decisions in business
Process-mediated	Traditional highly structured data from business and enterprise organizations. Characterized by transactions, relationships and well-defined contexts
Machine-generated	Structured data from smart devices, generated with a very high frequency, so that traditional relational database management systems become inappropriate for handling of it

The use of Big Data was first implemented by media corporations, which were innovators in sphere of Big Data. However, the importance of Big Data handling was realized by organizations from finance and industrial spheres, which became early adopters of the new technologies, followed by major implementations in utilities infrastructures and public services. Manufacturing organizations belong to the wave of latest organizations, which started to use Big Data [3].

2.2.1. Big Data technologies overview

Various technologies intended for benefiting from Big Data are collected in [7], including could computing, NoSQL databases, distributed systems, MapReduce and complex event processing. Cloud computing, which is becoming wide-spread, is defined in [30] as *"Set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing infrastructure on demand, which could be accessed in a simple and pervasive way."* Cloud computing allows to perform effective and scalable data analytics as cloud services provide organizations with already configured data management infrastructure [31]. The major providers of cloud computing are Amazon [32] and Google [33]. Apache Hadoop Distributed File System with its implementation of MapReduce framework, which enables parallel processing of huge amounts data, has already became a standard for Big Data processing [31].

2.3. Database management systems

The main goal of this thesis work is an implementation of information warehouse for the manufacturing facility; therefore an issue of data storage is a cornerstone of the work. For the long time RDBMS were dominating in an area of data storage solutions. RDBMS are based on ACID (Atomicity, Consistency, Isolation, and Durability)

concept, which guarantees that database transactions are processed reliably. Traditional databases have fixed table structures, where complex queries with joins can be used to fetch data from multiple tables in database [34]. One of the limitations of relational databases is lack of capability to scale in response to changing requirements [31]. This limitation includes inability to have a flexible data model, capable of processing heterogeneous and unstructured data [35], and complexity of horizontal scaling as the amount of data grows.

2.3.1. NoSQL

NoSQL databases have emerged in the beginning of 2000s as an opposition to traditional relational SQL databases, and currently are widely used in use cases when relational databases cannot handle huge amounts of data. NoSQL databases eliminate one of the biggest drawbacks of relational databases – inability to scale horizontally within the growth of demands and data quantities [31]. The first developments of distributed NoSQL databases have been conducted within Google (BigTable [36]) and Amazon (Dynamo [37]), where developers managed to provide distributed databases with high availability, applicability, scalability and performance. After that various open-source databases were developed by big web companies, including LinkedIn, Facebook and Twitter.

In book "Principles of Distributed Database Systems" [38] distributed database is defined as "*collection of multiple, logically interrelated databases distributed over a computer network*". Distributed database is often referred as a cluster of nodes, where individual node is a server running separate independent database instance. A CAP theorem, defined by Eric Brewer, says that distributed computer system cannot provide simultaneously the following 3 guarantees and have acceptable latency [39, 40]:

- Consistency (all nodes see same data at same time)
- Availability (every request receives a response)
- Partition tolerance (system can operate despite of failure of its part),

but it is possible to provide only 2 of them at once.

Therefore, it means that it is possible to create distributed database, which will be consistent and available, consistent and partition tolerant or available and partition tolerant. Thus, it is very important to understand the consequences of each of the guarantee. Having partition tolerance as an essential requirement for distributed system to operate, database can provide also either availability or consistency. According to the requirements for applications working with data, data should be transparent and always available, and thus availability should be preferred. However, while it is impossible to provide a consistency for distributed system, it is possible to provide *eventual consistency* [41]. Eventual consistency guarantees that in a sufficiently long time period when no changes were made to data in database, it is expected that all nodes will have an updated data, thus they will be consistent [42].

NoSQL DBMS are generally classified in literature to 4 types, according to their data modelling principles, which are described in details in [43] and summarized in Table 8.

Table 8: NoSQL databases classification

Type of database	Description
Key-value	This database is hash table containing columns of key-value pairs, where value can have a nested structure. Suitable for web applications, where unique user IDs and sessions need to be saved and requested.
Document	Data is saved as document objects, which can have very complicated and nested structure, and no data model definition needed. Perfect fit for applications, where event logs or data for analytics needs to be stored.
Column-family	Data is stored as rows, which are identified by row key, containing columns with data relevant to the key. Very effective for event logging and content management application.
Graph	Stores entities and relationships between them in form of graph. It can be used to follow and trace bidirectional connections between entities. Main area of application is social networking and location-based services.

The online rating of DBMS [44] shows that the most wide-spread NoSQL databases at the moment when this thesis was written are MongoDB and Cassandra (the list of all available databases is presented in Appendix 1). Their architecture and main characteristics will be discussed next.

2.3.2. MongoDB

Documentation of MongoDB is available online [45], but it is important to discuss the basic architectural principles. MongoDB represents a document store class of databases, meaning that it stores data in types, corresponding to native data types of various programming languages, where no structure of data needs to be defined. MongoDB provides high availability by using replication using master-slave architecture, and can be horizontally scaled using sharding.

Replication

In MongoDB replication is achieved according to the principle of master-slave architecture, where master database receives all write operations and sends data set to secondary (slave) databases in order to provide data redundancy. The concept of replication is depicted on a Figure 7.

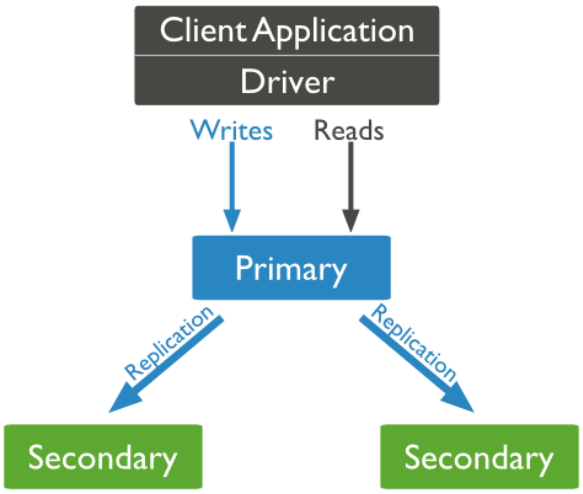

Figure 7: Replication in MongoDB (adopted from [45])

However, while failure of slave node in master-slave architecture doesn't have any considerable impact on the overall system's performance, a failure of master node always has a serious effect on a performance of whole system [46].

Data model

Data in MongoDB doesn't need to have a defined schema, unlike in SQL databases, due to the fact that it is saved as documents composed of fields and value pairs like in example on Listing 1.

```
{
      cell_id:1,
      phase: "A",
      timestamp: 1389349483,
      energy: 23.1,
      power: 10
}
```

Listing 1: Document structure

Documents are stored in collections and inside of single collection documents can have any structure. This approach provides high flexibility for data storage and retrieval.

Storing data

Data is stored in memory-mapped files, which are placed directly by operating system in memory, meaning that they are mapped to a region of virtual memory. This allows treating contents of data files as if they are stored in memory, thus data access is very fast and efficient.

Queries

Queries in MongoDB are performed on a collection of documents and can contain various search criteria and conditions. The syntax is different from traditional SQL syntax and can be expressed as following:

```
db.EnergyMeter.find({cell_id:1,phase:"A",power:{$lt20}}).sort({
timestamp:1})
```

In a query above all documents with specified cell, phase and power conditions will be returned sorted in order of ascending timestamp.

2.3.3. Cassandra

Cassandra's documentation is available online [47] and has a very comprehensive description of architecture. Cassandra has been designed with a possibility to handle large amount of data inside of a cluster of nodes without a single point of failure (partition tolerance). In order to provide partition tolerance, instead of traditional master-slave architecture, Cassandra implements peer-to-peer distributed architecture, where every peer, representing a single node in cluster, is exchanging information across the cluster every second and stores own share of data. Peer-to-peer communication is achieved by using protocol called *Gossip*, which allows nodes to exchange information about themselves and other nodes every second with up to 3 nodes in cluster. It also allows nodes to learn about the structure of cluster. Due to this Cassandra can be scaled horizontally by adding new nodes to cluster.

Replication

Replication of data in Cassandra can be flexibly configured, using configuration parameter called *Replication Factor*. For instance, if replication factor is 3, a piece of saved data will be saved on 3 nodes. It allows person who is responsible for database management to choose required and relevant behavior of database depending on the requirements to data availability and size of database cluster.

Data model

Cassandra is different from MongoDB, and rest of NoSQL databases, due to the fact that it needs to have a data model to be defined. What is more, in contrast to all other databases, data model in Cassandra needs to be designed based on query model, in other words, basic query use cases have to be determined first [42].

Cassandra uses CQL (Cassandra Query Language), which is an analogue to SQL, for defining data structures. Cassandra has keyspaces (usually one per application), and each keyspace has tables with defined columns and relevant data types. Data structure in tables resembles of SQL data structure, however a big difference is that Cassandra is a distributed database and data is denormalized.

Every table in Cassandra needs to have a compound primary key that includes partition key, which defines on which nodes data is stored, and one or more additional clustering keys, which are responsible for clustering and sorting of data. This approach allows performing very fast querying of data. Example of a Cassandra table having compound key, which consists of *cell_id* (partition key), *phase* and *timestamp* (clustering keys) is presented on a Figure 8.

Cell_id	***Phase***	***Timestamp***	***Energy***	***Power***	***Temperature***	**…**	***Humidity***
1	A	1389349483	23.1	10	30	…	0
1	B	1389339483	24	10	30	…	0
2	A	1389320483	30.5	10	31	…	0

Figure 8: Cassandra table structure example

Storing data

Data storage is organized in a following way: each primary key has an own hash value; every node is responsible for a data based on hash values, thus data associated with each primary key is saved on a relevant node. The data inside of node is distributed to virtual nodes, holding large amount of small partition (hash values) ranges. This principle is presented on a Figure 9.

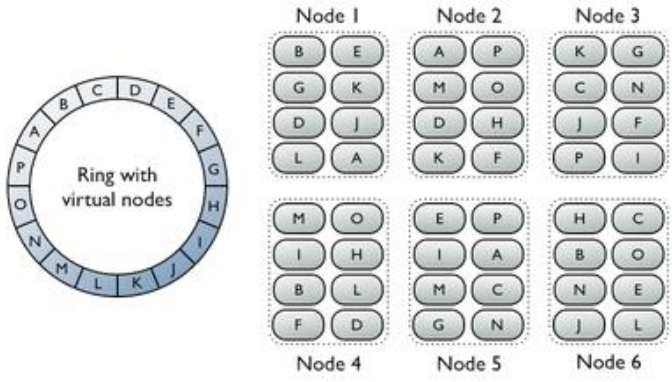

Figure 9: Cassandra's data storage organization (adopted from [47])

Client requests

Cassandra has a very efficient approach for client requests handling that provide one of the most important benefits – very fast data reads and writes. When a client connects to cluster to read or write data, the node that client connected to acts as a *coordinator* for the user operation. The task of coordinator is to find nodes that are needed to fulfil user's operation. For write operations coordinator sends request to all nodes, which belong to partition range of written key, and waits for response from number of replicas specified by *Consistency level* parameter, which defines the amount of nodes that have to respond with success acknowledgment to coordinator, in order to consider write to be successful. This sequence is demonstrated on a Figure 10.

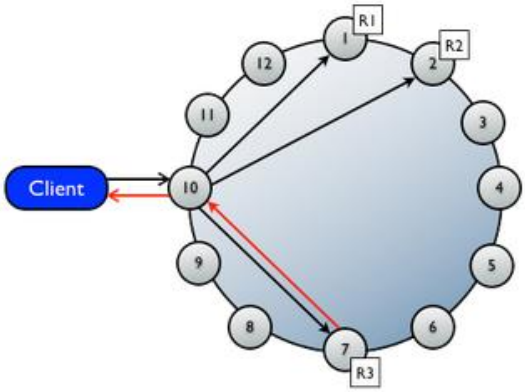

Figure 10: Cassandra's write operation in cluster (adopted from [47])

The Figure 10 demonstrates a situation, when replication factor is configured to be 3 and consistency level ONE. Node number 10 serves as a coordinator for client's write request and propagates the data to 3 nodes, responsible for storing it. Node number 7 sends success acknowledgment first, and, according to configured consistency level, coordinator replies to client with a success message.

In case of read request (Figure 11), coordinator contacts nodes according to consistency level, by sending request to those nodes, which currently have the fastest response in cluster. If multiple nodes are queried for data, their responses are compared in memory, in order to define the most recent version of data, which is then sent back to client, and at the same time coordinator updates in a background data in other nodes, in case if they had different data comparing to final response. This principle is demonstrated on a Figure 11.

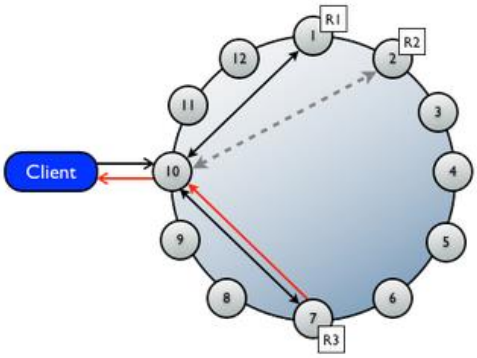

Figure 11: Cassandra's read operation in cluster (adopted from [47])

Client sends a read request to node number 10, which serves as a coordinator, which directs request to replica nodes 7 and 1 with highest performance rate. When result from them is received, it is compared to identify which node has the valid information, and this data (from node 7) is returned to client. Same time coordinator updates data in nodes 1 and 2 to be same as it was in node number 7.

The concept of client requests is very crucial to understand, because it clearly demonstrates one of the biggest advantages that Cassandra has over other NoSQL

databases: high availability and replication of data. It is guaranteed that client will get the latest and truthful information as a response to request. Moreover, even if some of the nodes are down, data will be still available through the other independent replica nodes.

Data queries

Queries in Cassandra have several limitations and cannot be as flexible as queries in MongoDB. Queries are done using CQL with the same syntax as SQL:

```
SELECT energy, power from EnergyMeter where cell_id=1 and phase=”A”
```

But queries are restricted in the way that conditions can be set only on a compound key in order from partition key to last clustering key. Therefore, it would be impossible to select all data for all cells and phase A. Instead it is possible to select all data for specific cell and phase for all the time or for specific time range.

2.3.4. Comparison of MongoDB and Cassandra

Table 9 presents the most crucial technical and architectural specifications of both databases for the purpose of comparing.

Table 9: Comparison of MongoDB and Cassandra

Feature	MongoDB	Cassandra
Replication	Master-slave strategy	Independent nodes, sharing same data
Scaling	Vertical scaling and horizontal scaling (achieved by sharding)	Horizontal scaling by adding new nodes to cluster, no extra configuration needed
Data model	Documents. No data model definition	Tables with compound keys. Data model needs to be defined
Writes	Fast, but can be locked in case of concurrent reads	Fast, constant-time with no locking
Reads	Very fast	Very fast
Queries	Flexible, allowed on any attribute inside of document	Restricted, allowed on compound key, weak support of secondary indexes

The Table 9 demonstrates the main differences between two databases and helps to make a conclusion that MongoDB is more suitable for cases, when scaling is not critical issue and incoming data can change within a time. Cassandra is the best fit for situations when system needs to be ultimately scalable and always available, which is achieved by putting a restriction of necessity to have a fixed data model. According to comparison research for sensor applications, conducted in [34], Cassandra is optimal for large critical applications, while MongoDB is better for small or medium-sized applications.

2.3.5. State of the art in NoSQL databases

There are very few research works and projects that have been done using NoSQL databases in a manufacturing domain, which also proves the fact, stated in chapter 2.2, that manufacturing organizations belong to group of late adopters of Big Data.

Thantriwatte and Kepetiyagama in [48] developed NoSQL query processing system for Wireless ad-hoc and sensor networks and compared them with already implemented SQL solution. Energy problems can often occur in sensor networks resulting with a failure of sensor nodes, thus data can be lost and ACID properties cannot be guaranteed.

Based on this, and also due to the fact that NoSQL provides good performance and scalability, authors decided to implement NoSQL system. They have shown that for small datasets performance of SQL and NoSQL queries in terms of execution time is same, but for huge data sets NoSQL is almost twice faster (Figure 12). The main reason for this difference is a slow processing time of SQL query.

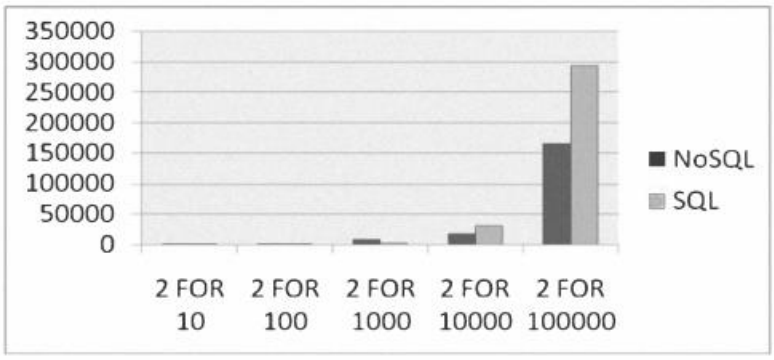

Figure 12: Comparison of NoSQL and SQL query performance (adopted from [48])

Yamamoto et al. in [49] implemented platform, called Scallops4SC (SCALable Logging Platform for Smart City), for storing and processing large-scale house data, where NoSQL database Hbase was used for log data, and MySQL for configuration data. This platform demonstrates a solution for handling large amount of machine-generated data (log data, which is collected periodically from different appliances and sensors in several houses) with NoSQL technology.

The log data is stored as key-value pairs, so that no schema is needed, where key contains information about date, time, house id, log type and device id (Figure 13).

Key	Value
2012-07-28T12:34:56.cs27.Energy.tv01	{value:600, unit:W}
2012-07-28T12:34:56.cs27.Device.tv01	{status:[power:off]}
2012-07-28T12:35:00.cs27.Device.tv01	{operation:on()}
2012-07-28T12:34:56.cs27.Env.temp2	{value:24.0, unit:celsius}
2012-07-28T12:34:56.cs27.Env.pcount3	{value:3, unit:people}

Figure 13: Key-value pairs for storing data (adopted from [49])

The rest of available research works about NoSQL are mainly concerned with performance analysis and comparison of different database vendors. Structured heterogeneous log analysis operations are evaluated for Cassandra, MongoDB, Membase, Neo4j and OrientDb in [50]; in-memory index structure developed in [51] is compared with other indexes implemented in NoSQL databases.

2.4. Service Oriented Architecture

As it was discussed in [20], cross-layer architecture can have huge effect on development of energy management system. Nowadays, Service Oriented Architecture (SOA) is a de-facto standard for enterprise organizations architectural models.

Thomas Erl gives a general definition of SOA in his book [52] as

"*An architectural model that aims to enhance the efficiency, agility and productivity of an enterprise by positioning services as the primary means through which solution logics is represented*".

A more specific definition is available in book [53], where SOA is defined as

"*Architecture that constitutes a distributed computing environment in which applications call functionality from other application either locally or remotely over an internal network of an IP-network in a loosely-coupled way*".

In other words, SOA approach allows dividing system functionality into various independent services, which are capable of communicating with each other at any time. SOA has 3 components, as described in [54]: *consumer*, *service* and *service broker*.

Services provide functions to consumers, which are defined in WSDL (Web Service Description Language) [55], and consumer is a software application which uses service through the defined service interface. Service broker is used for providing a registry of existing and known services, based on UDDI (Universal Description, Discovery and Integration) protocol, so that consumers can easily get required for them services. The interaction between these 3 components is presented on a Figure 14.

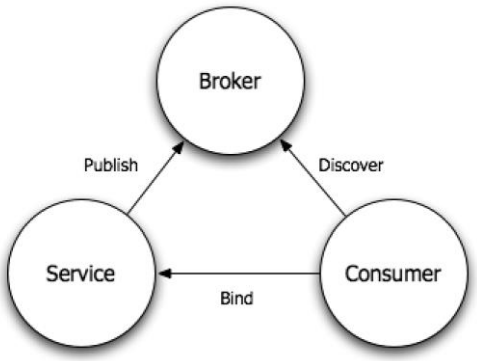

Figure 14: Interaction between components of SOA (adopted from [54])

The main characteristics of SOA, which are frequently mentioned in literature are collected in [56] and presented in a Table 10.

Table 10: Characteristics of SOA (adapted from [56])

Characteristic	**Comment**
Autonomy	Services are independent and structurally decoupled
Interoperability	Provision of interface describing available services and interaction patterns
Platform independence	Services are described with standard eXtended Markup Language (XML) and WSDL formats, which are universal and can be processed by any operating system, computer architecture, programming language or technology
Encapsulation	Services hide unnecessary details of their functionalities by exposing a user defined interface
Availability and Discovery	Services can be published for private or public use, and can be found in relevant registries

The most important benefit of SOA adoption in manufacturing is possibility to create a distributed system with loosely-coupled discrete components, which can be recomposed, reconstructed and reused to create new applications [57].

The core of SOA is Web Services, which give an opportunity for easier deployment of distributed system, as they can be accessed and invoked through Internet and can be used at any time [53]. W3C [58] defines Web Service as:

"A software system, designed to support interoperable machine-to-machine interaction over network. Other systems interact with Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP and XML serialization in conjunction with other Web-related standards."

Web Services are built upon several standards and protocols: WSDL, XML, XSD (XML Schema Definition), SOAP (Simple Object Access Protocol), UDDI and HTTP (Hypertext Transfer Protocol) [53]. There are two types of Web Services [59]:

1. SOAP Web Services
2. Representational State Transfer (REST) services

2.4.1. SOAP Web Services

SOAP Web Services are based on SOAP protocol, which defines message architecture and format using XML language. SOAP messages have top element *Envelope*, which contains of two elements: *header* and *body*, described in Table 11.

Table 11: SOAP envelope contents

Element	Contents
Header	Message-layer infrastructure information used for routing, security and configuration of transactions, security and reliability.
Body	Payload of message

With the help of SOAP engines, clients, who consume WS, can marshal and unmarshal (translating application's native language to and from SOAP protocol [60]) SOAP message to perform further required processing of data. Table 12 presents advantages and disadvantages of using SOAP Web Services [61].

Table 12: Advantages and disadvantages of SOAP Web Services

Advantages	Disadvantages
Protocol transparency and independence	Possibility of abstraction leakage
Service interface provides abstraction from communication and implementation protocols	Can be hard to define correct data model, that will support interoperability
Supports asynchronous services	

Process of marshalling and unmarshalling (or, serialization and deserialization) is crucial for understanding of implementation abstraction concept. They are performed on a server-side of each service, when SOAP request or response is received, in order to translate XML message to native language objects. This process is presented on a Figure 15.

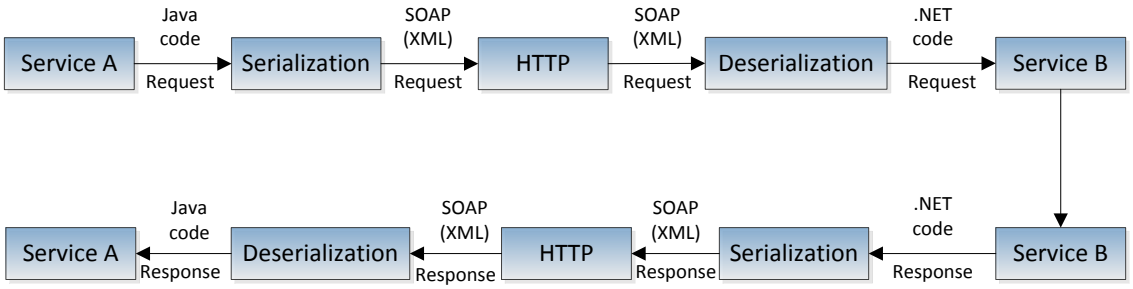

Figure 15: Serialization of XML messages (adapted from [60])

2.4.2. REST web services

REST Web Services are designed to have a tighter integration with HTTP protocol, and are more lightweight comparing to SOAP WS [59]. RESTful WS provide for clients a set of resources, which are identified with an URI (Uniform Resource Identifier) and can be accessed through Internet. Resources are manipulated using typical HTTP operations PUT, GET, POST and DELETE, responsible for creating new resources, retrieving current state of resources, transferring new states to resources (updating) and removing resource, respectively. Usually, JSON (JavaScript Object Notation) is a preferred format for messages in REST, as it is lightweight and has optimal performance. Table 13 presents advantages and disadvantages of REST services [61].

Table 13: Advantages and disadvantages of REST Web Services

Advantages	Disadvantages
Simple and lightweight	Connections might be blocked by firewalls
Built with minimal effort	Can be hard to define correct data model that will support interoperability
Discovered through Web	
Can be scaled using caching, clustering and load balancing	

2.4.3. SOA in manufacturing

A lot of research has been conducted for development of SOA in manufacturing in the last few years. The main focus of researches is directed towards design of layer architecture in enterprise and integration of applications from different levels of architecture. Adoption of SOA and Web Services allows achieving loose coupling, where services can be easily reconfigured and flexibly combined like it is shown in [62] and [63]. Cucinotta et al. in [64] use SOA to fulfill need for development and

deployment of new hardware and software solutions supporting real-time systems, in order to satisfy increasing demand for efficiency of manufacturing. They move further by proposing an infrastructure with Plug & Play services, which hide complexity of used devices, and allows having reconfigurable manufacturing system.

Enterprise Service Bus is another technology that comes together with SOA, which is responsible for orchestration, integration and management of services at the runtime. It is based on OSGi framework, which implements SOA concepts for dynamic discovery of components, developed in Java, in order to create real-time, reconfigurable and dynamic application. The main functional principles of OSGi framework are described in deeper details in [79]. Functionalities and design approach of ESB are described in [65], and [62] has it in a core of proposed architecture. Chen et al. in [66] developed an architecture that uses several service buses at the same time in order to have a smoother integration between services of collaborative manufacturing system.

It can be clearly seen that SOA proves to be an efficient architectural approach, which is actively adopted by enterprises in manufacturing and industry.

2.5. Complex Event Processing

As it was already mentioned in Section 2.1.2, the second important step in energy management system is calculation of KPIs, which is usually completed with help of Complex Event Processing.

CEP has become one of the most common and useful tools for processing big amounts of data in a real-time. Generally, CEP could be described as a set of tools and techniques, intended for analyzing and handling real-time data in distributed information systems [16]. The main application areas for CEP include business process management and automation, finance, network and application monitoring and sensor network applications [67]. CEP has been heavily used in many projects, where real-time data management is a crucial requirement, for instance in sensor networks [68], for RFID data processing [69], as well as for energy efficient asset management in manufacturing facility [16].

A classification of event processing applications is proposed in [70] with 5 categories, described in a Table 14.

Table 14: Classification of CEP applications

Application	Description
Observation	Monitoring of systems and producing alerts in case of error or exceptional behavior
Information dissemination	Granulazing of information for a delivery to specific interested party
Dynamic operational behavior	Reacting to incoming events by providing fast, reliable and efficient decisions
Active diagnostics	Finding common attributes of problem and diagnose its reasons
Predictive processing	Prediction and prevention of undesired events

The operational principle of CEP is presented on Figure 16. Event producers are devices, which generate real-time events, streaming this data to an engine of CEP, where processing of the data is performed. Processed data then moves upstream to event consumers – applications and database management systems that use the data.

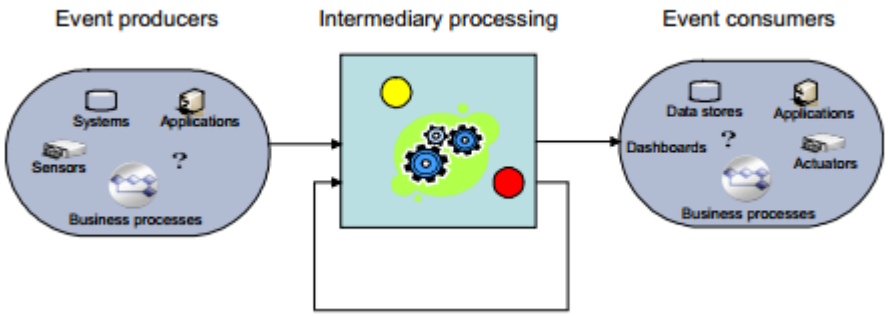

Figure 16: Main concepts of event processing (adopted from [70])

Complex event detection is performed in 4 steps, as described in [71]:

1. Primitive events are extracted from large data
2. Events are correlated or aggregated according to specific rules to create a new business event
3. Primitive and composite events are processed to extract their relationships
4. Response is sent to subscribers

Finally, CEP is used as an effective tool for processing large amounts of incoming data with a various complex structures according to complex rules.

3. METHODOLOGY

Required technologies for implementation of information warehouse and architectural approach are described in this section.

3.1. System Architecture

Layered architecture is used for development of an information warehouse system. In this architecture application consists of various layers, where each layer provides certain services for higher level without knowing specifics of a lower layer.

According to layered architecture principles, designed system's architecture consists of 6 layers:

- *Physical layer,* which is represented by all factory shop floor devices
- *Network layer,* which is responsible for routing data from devices to the application server. Event hub first receives incoming devices data and forwards it to subscribers as SOAP message. Application has an endpoint, which is responsible for receiving messages from event hub and sending them to be processed by application.
- *Application layer* performs data aggregation and processing by means of CEP engine
- *Data layer,* where data from application layer is stored in a database cluster
- *Service layer* provides various services for accessing data by users
- *User Access layer* represents all users, application and interfaces which use application services for energy management.

The described architecture is presented on Figure 17. Technologies and tools used to implement functionalities of components in each layer are described next.

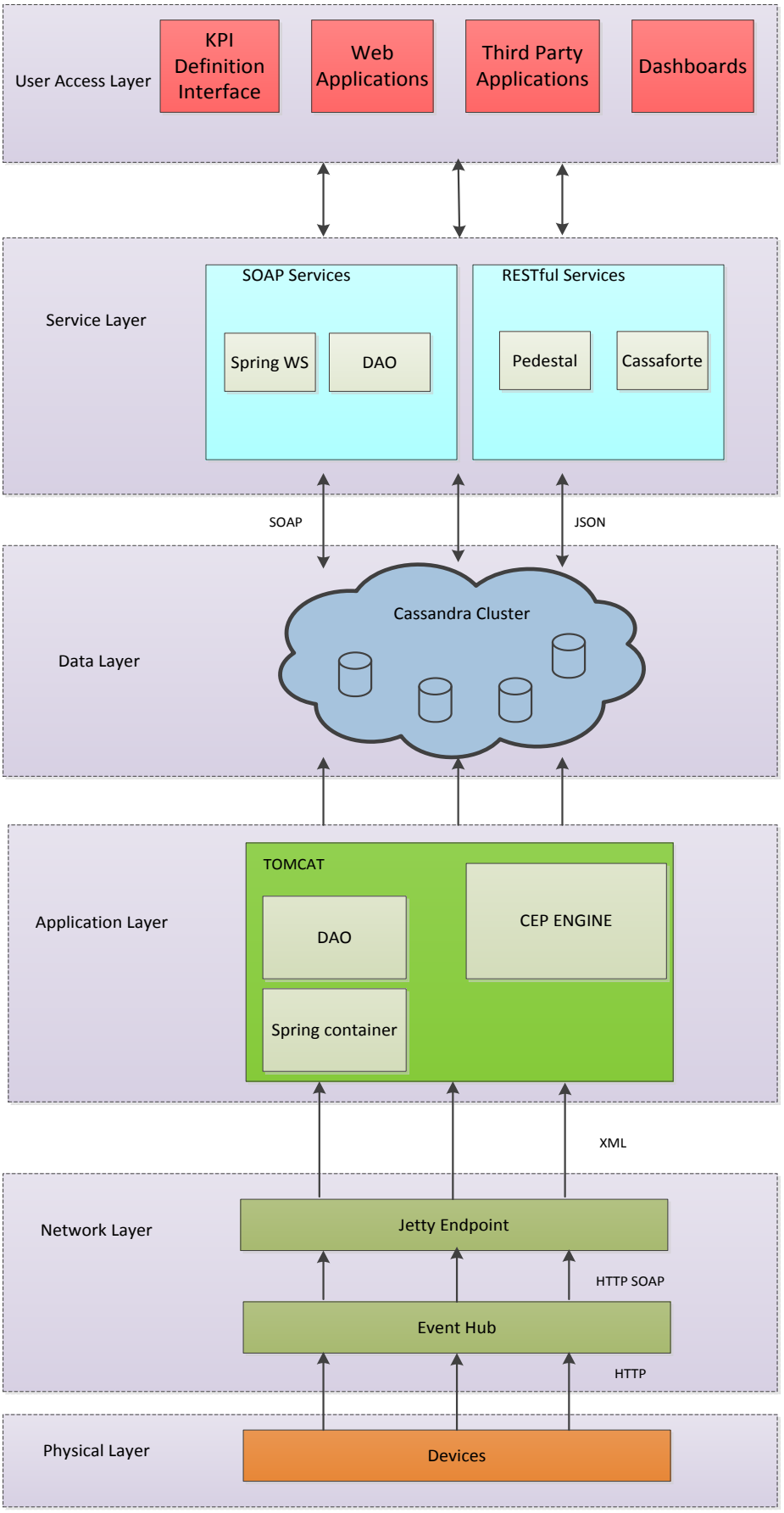

Figure 17: Layered architecture of designed system

3.2. Tools and Frameworks

For the implementation of functionalities of each layer an open-source tools and frameworks were selected.

3.2.1. Event Hub

Event Hub is a Java application, which subscribes for messages from all available devices and routes them to other applications, acting as a gate through which all data is passed. Event Hub is running as Apache Camel Jetty HTTP endpoint, which consumes and produces HTTP requests, forwarding messages to other registered HTTP endpoints.

3.2.2. Cassandra

As it was discussed in the technology review, a NoSQL DBMS will be used in this thesis work for storing the data. However, it is important to make a choice from the two considered major database solutions, Cassandra and MongoDB, which will be the most beneficial. The selection is done based on a requirement to have a data available all the time, and to be able to scale with the increasing demands. According to it, as it was discussed in chapter 2, Cassandra is the best fit. Having only well-structured data coming from the devices in manufacturing facility, it is possible to focus on data availability and provide interfaces for fast and efficient access to data. For the implementation a version 1.2.10 is used.

3.2.3. Esper

Esper engine is using SQL-like syntax for making queries on data; however, instead of querying static data from database, it stores queries and allows incoming real-time data to run through queries, thus triggering data analysis when it matches configured conditions [72]. With the help of query language of Esper, Event Processing Language (EPL), it is possible to derive and aggregate information from event streams, perform filtering, pattern matching and to group the data into various views, based on time window or event stream size. Events in Esper can be represented by Plain Old Java Objects (POJO), Java Maps, Object arrays and XML, which gives a flexibility of configuration for any specific case. The latest available version (4.9) at the time, when thesis was written, is used in implementation.

3.2.4. Kundera

Kundera is an open-source object-datastore mapping library, developed by Impetus [73], and intended for cross-datastore usage (in the moment of writing of thesis supports Cassandra, Hbase, MongoDB, Neo4J and relational databases). Kundera is based on JPA 2.0 (Java Persistence API) and allows usage of JPQL (Java Persistence Query Language) for fetching data from database. Kundera provides a very high abstraction

from database technologies, where switching between different databases can be done by changing one configuration file. Data Access layer uses *Entities*, also called Data Access Objects (DAO), which are POJOs containing specific annotations, in order to map data inside of them to database tables. The mapping rules for annotations and database data structures are presented in Table 15. The version 2.9 of Kundera is used in implementation.

Table 15: Annotations in Kundera (adapted from [73])

Annotation	***Cassandra***
@Table	CQL Table
@Column	Column
@Embeddable,@EmbeddedId	Compound Key

3.2.5. Spring Framework

Spring framework is a platform intended for development of enterprise software. The main concept of Spring is based on container and component model (Figure 18) and Inversion of Control (IoC) – all objects (transactions, database access, web applications), which in terms of Spring are called *beans*, are built on top of container, which can inject dependencies needed for applications in order to configure and integrate infrastructure [74]. Main advantages of using Spring include simplicity, which is achieved by usage of lightweight POJOs, testability and loose coupling, achieved through dependency injections, of applications [75].

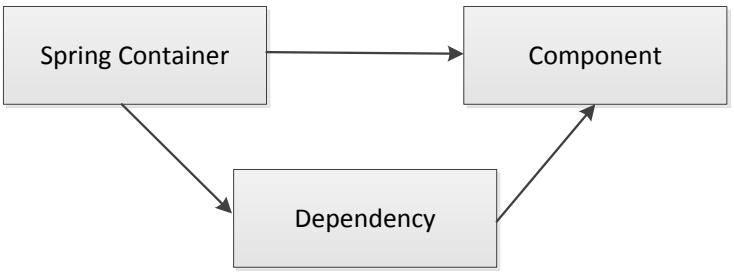

Figure 18: Spring Framework's concept (adapted from [74])

Spring is configured to manage specific beans in an application context with help of XML configuration files. In this thesis Spring will be used for configuration of application server, injection of Entity Managers, which are needed for DAO, and provision of SOAP Web Services. Version 3.2 of Spring Framework is used in this thesis.

3.2.6. Pedestal

Pedestal is a web application framework and used for development of REST web services, which are written in a functional language Clojure [76]. Clojure, being a LISP

programming language, allows performing fast operations over large sets of data, as it is presented in form of collections. For database access a Cassaforte library is used [77]. A version 0.0.1 of Pedestal is used in implementation.

3.2.7. Node.js

KPI definition user interface is developed with an open-source framework for JavaScript server-side programming called Node.js [78]. Node.js uses event-driven model for handling user requests, therefore providing possibility to create efficient real-time and concurrent web applications. For the implementation a version 0.9.12 is used.

3.2.8. Apache Service Mix

Apache Service Mix [79] is an open source ESB, which encompasses SOA and OSGi functionalities. Service Mix uses messages in order to communicate and exchange information between bundles, thus providing modularity for SOA. Bundles can be flexibly combined for composing various complex services, updated, reconfigured and reloaded during the runtime of application. The version 4.5.3 of Service Mix is used in this thesis.

4. IMPLEMENTATION

4.1. Overall system architecture

Proposed system contains of 3 components: controllers installed in manufacturing line, application server and database server. Application server contains an application, which received data, processes it, stores to database and provides services to access it. Database server is used for storing all generated raw data and calculated KPIs.

The real-time data from the manufacturing facility arrives to the application server on the available HTTP endpoint, implemented with Jetty. Upon the arrival to the endpoint, data is directed to CEP engine, implemented with Esper, where data is processed according to defined EPL rules. All processed data from CEP engine (raw data and KPIs) is then saved to the Cassandra database, deployed on Linux server with Ubuntu operating system, through Data Access Layer. Application server has SOAP Web Services published on the Internet, which provides interfaces for accessing data by users. The overall system architecture is presented on a Figure 19.

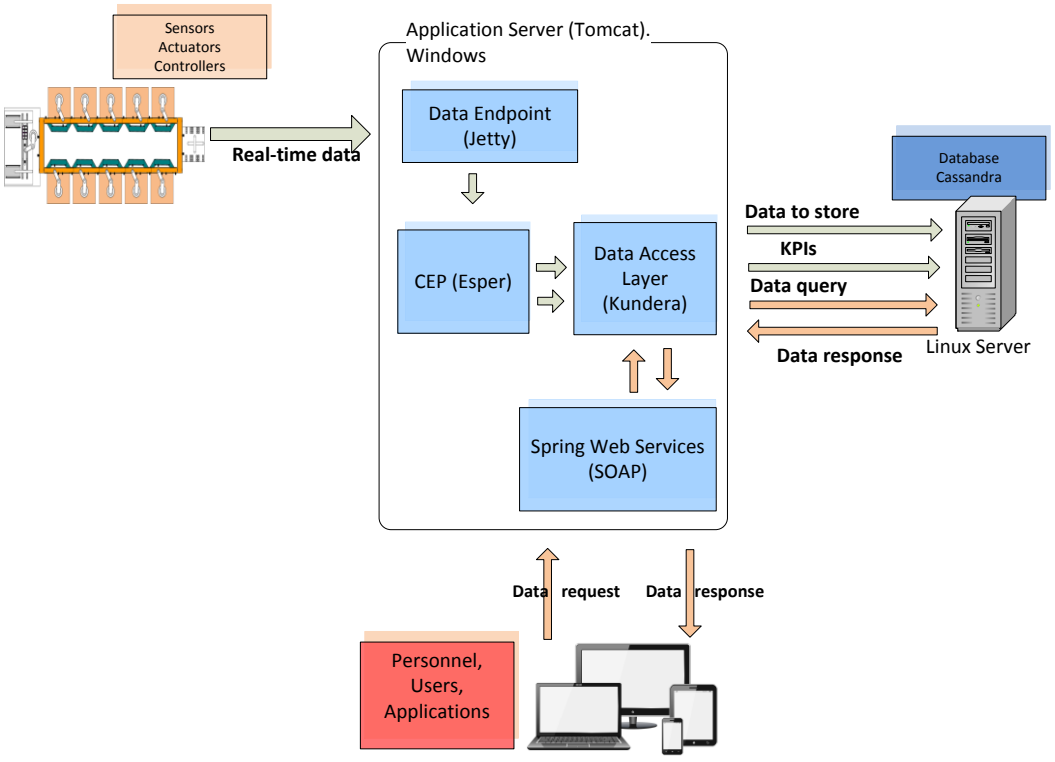

Figure 19: Application overall architecture

In next sections each components in the architecture is describe in more details.

4.2. Database data model definition

Based on existing types of messages in manufacturing facility, data model for Cassandra database can be designed. As it was mentioned in literature review, query model needs to be taken into consideration in order to design a sufficient and efficient data model. The basic concept for definition of query model requires distinguishing of all energy consumers, in order to provide an opportunity to analyze information in an atomic way by separate consumer or whole manufacturing line. Distinct element of the manufacturing line is cell, therefore users might need information about specific cell or set of cells. According to this, general energy message in manufacturing facility can contain information about energy values for a single cell, as it is illustrated on Listing 2.

```
<SampleEnergyMessage cellID="cell_1"
dateTime="2014-01-03T13:26:41"
watt="0.26" ="65.81" watthr="1255" var="-62.84"
vrms="-1217”/>
```

Listing 2: General energy message structure

Allegedly, the possible queries for energy message data from database are:

- *Select watt values for last day in cell_1*
- *Select all energy values in cell_1 for a specified time range*
- *Select watthr values for whole manufacturing facility for last day*

These use cases demonstrate that in all queries data is identified by cell id and is sorted according to timestamp. Therefore, cell name can be defined as a partition key, and timestamp as a clustering key. This data model will allow performing fast queries for a single cell or set of cells.

4.3. Energy Key Performance Indicators

Energy KPIs need to be calculated in order to give a valuable feedback for personal responsible for energy management. These KPIs can be divided into 2 groups, based on user needs:

1. *Real-time KPIs*
2. *On-request KPIs*

The first type of KPI is always calculated for a specific period of time in a real-time. These KPIs have to be stored in a database and be always available for users. They can be calculated by means of CEP engine. For instance, hourly energy consumption of robot cell can be calculated in a real-time as the time period is always fixed.

The second type of KPIs is calculated only when user needs it and doesn't need to be stored. For example, user needs to know an energy consumption of robot cell for flexible periods of time – for one day, one week or one month, which he or she can freely choose. In this case it makes sense to calculate the KPI by accessing directly historical data from the database. In order to calculate these KPIs, a calculation logics is added to Web Services.

4.4. Application development

4.4.1. Implementation of data endpoint

In order to receive a data from the manufacturing facility, a starting point inside of application server is created, which acts as a gateway for stream of incoming data. Every message from Event Hub is sent over HTTP as POST message and it is consumed by endpoint, where input message is transformed to XML Node format and then redirected to CEP engine. In case of success, server responds with HTTP 200 – a standard response for successful HTTP request. If error occurs when sending event to Esper engine, server will respond with HTTP 500 Internal Server Error. The case of successful message delivery is illustrated on Figure 20.

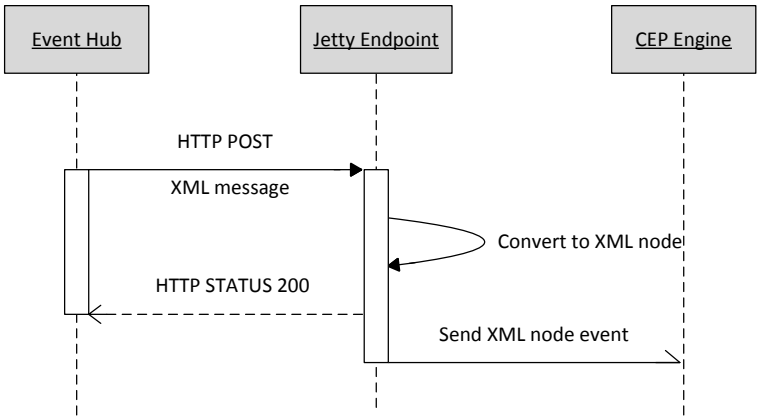

Figure 20: Message delivery to application

4.4.2. Implementation of data processing

From the endpoint data is delivered to Esper CEP engine, where it is processed and prepared for storing.

In order to make CEP engine aware of incoming messages, a configuration file *esper.cfg.xml* is used. This file configures Esper by defining types of existing messages. If incoming XML message is not predefined, an exception will be thrown by Esper engine, stating that the message is not configured.

Figure 21 illustrates the workflow of CEP engine. When XML event enters the engine, it matches the message to relevant rules. If the rule is satisfied, subscriber is triggered, which is responsible for processing of the data by getting relevant information from the message and passing it to data access layer. If XML event doesn't satisfy the rule, it is sent to memory of engine, where all messages for specific rule are aggregated until the rule is satisfied.

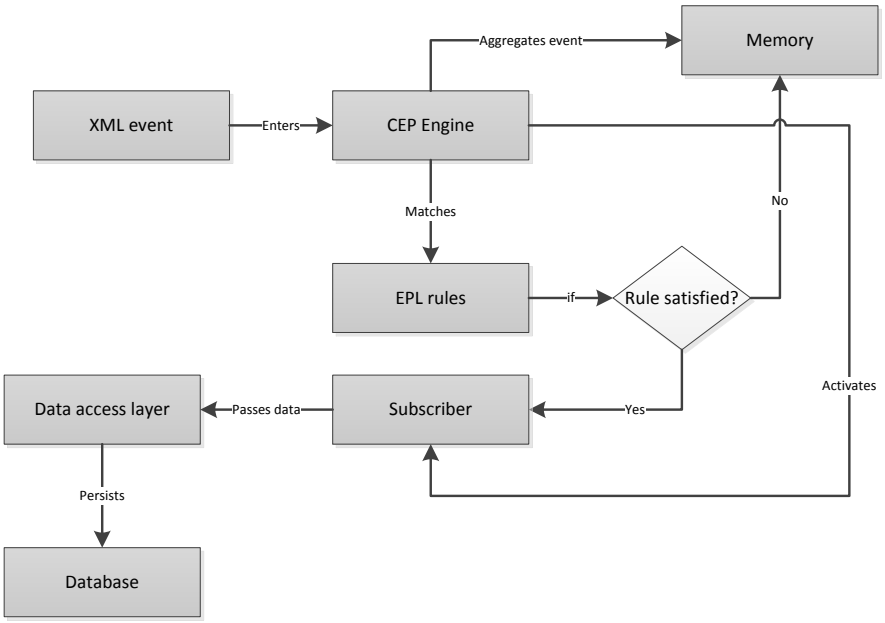

Figure 21: CEP engine workflow

CEP engine has two main functionalities:

- *Parsing of the incoming messages*
  It is achieved by having an EPL rule, which selects all attributes of message every time new message arrives, with subscriber receiving Java objects containing values of attributes
- *Calculation of KPIs*
  KPIs are calculated in a real-time according to existing EPL rules

4.4.3. Development of Data Access Layer

Data access layer is an intermediate layer between application logics and database. It provides a loose coupling for fetching data from database, by offering an interface with methods that are responsible for database queries, and separating it from application logics.

An XML configuration file persistence.xml is used for specifying database connection details, which is placed under /META-INF folder. The file is demonstrated on a Listing 3. It contains desired name of persistence unit (database cluster), list of mapped to database tables Java classes and connection-specific parameters. Any amount of persistence units can be added to the file. Java classes (called Entities), intended for

mapping to database tables, have @Entity annotation, which has a property referencing to relevant persistence unit.

```
<persistence-unit name="persistenceUnit">

<class>com.model.MessageModel</class>
<class>com.model.MessageModelKey</class>
<properties>
        <property name="kundera.nodes" value="localhost"/>
        <property name="kundera.port" value="9160" />
        <property name="kundera.keyspace" value="name" />
        <property name="kundera.dialect" value="cassandra" />
        <property name="kundera.client.lookup.class"
        value="com.impetus.client.cassandra.thrift.ThriftClientFact
        ory" />
        <property name="kundera.cache.provider.class"
        value="com.impetus.kundera.cache.ehcache.EhCacheProvider"/>
</properties>
</persistence-unit>
```

Listing 3: Database connection configuration file

Physical connection to database is made with an instance of *EntityManager*, which provides functionality to perform CRUD (Create, Update and Delete) operations on a database. An independent EntityManager is obtained for every single request from an *EntityManagerFactory*. EntityManagerFactory and EntityManager interfaces are available as a part of JPA 2.0. EntityManagerFactory is a heavyweight object, therefore it is instantiated only once during application start and is stored inside of Spring Container, so that EntityManager can be easily created at any time. EntityManager is injected into objects with *@PersistenceContext* annotation.

Queries for data access are performed with JPQL, which is wrapped by Kundera. The example of query formulation is illustrated on a Listing 4. This query selects some specific data, which is mapped to database with *EnergyData* entity, and limits search to particular cell and time range.

```
String query = "SELECT e.data from EnergyData e
where e.key.cellId = :cellId
and e.key.timestamp > :timeStart
and e.key.timestamp < :timeEnd"
```

Listing 4: JPQL query example

Query is using named parameters (“*:cellId”,”:timeStart”,”:timeEnd”*) in order to provide a flexibility in accessing the data. These parameters are further evaluated according to specific user request. After all, *Query* object is created, to which textual representation of query and named parameters are passed. Data can be fetched from database by calling a *getResultList* method of Query object.

4.4.4. Development of API

SOAP Web Services

API based on SOAP is implemented with Spring Web Services, provides various methods for accessing data in the database. Spring Web Service is a contract-first service, which means that XML schema needs to be created before development of service logics in Java. The schema contains description of methods used to access data, consisting of request and response messages, as it is illustrated in example on Listing 5.

```
<xs:element name="DataRequest"
   <xs:complexType>
      <xs:sequence>
         <xs:element name="cellId" type="xs:int" />
         <xs:element name="timeStart" type="xs:long" />
          <xs:element name="timeEnd" type="xs:long" />
      </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="DataResponse"
   <xs:complexType>
      <xs:sequence>
         <xs:element name="cellId" type="xs:int" />
         <xs:element name="value" type="xs:float" />
          <xs:element name="timestamp" type="xs:long" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Listing 5: Definition of data access request and response

Based on this schema and configuration file, which defines host of Web Service, JAXB mapping classes for serialization and deserialization of incoming XML requests, Spring automatically generates WSDL file, which becomes available for clients. The next step after contract definition is development of Java Web Service endpoints, which are mapped to relevant requests in WSDL with help of annotations, where request processing is performed. Further, endpoint makes a query to database through data access layer, prepares response message and sends data back to client. The whole cycle of operations and interaction between layers of application from SOAP request to SOAP response is depicted on a Figure 22.

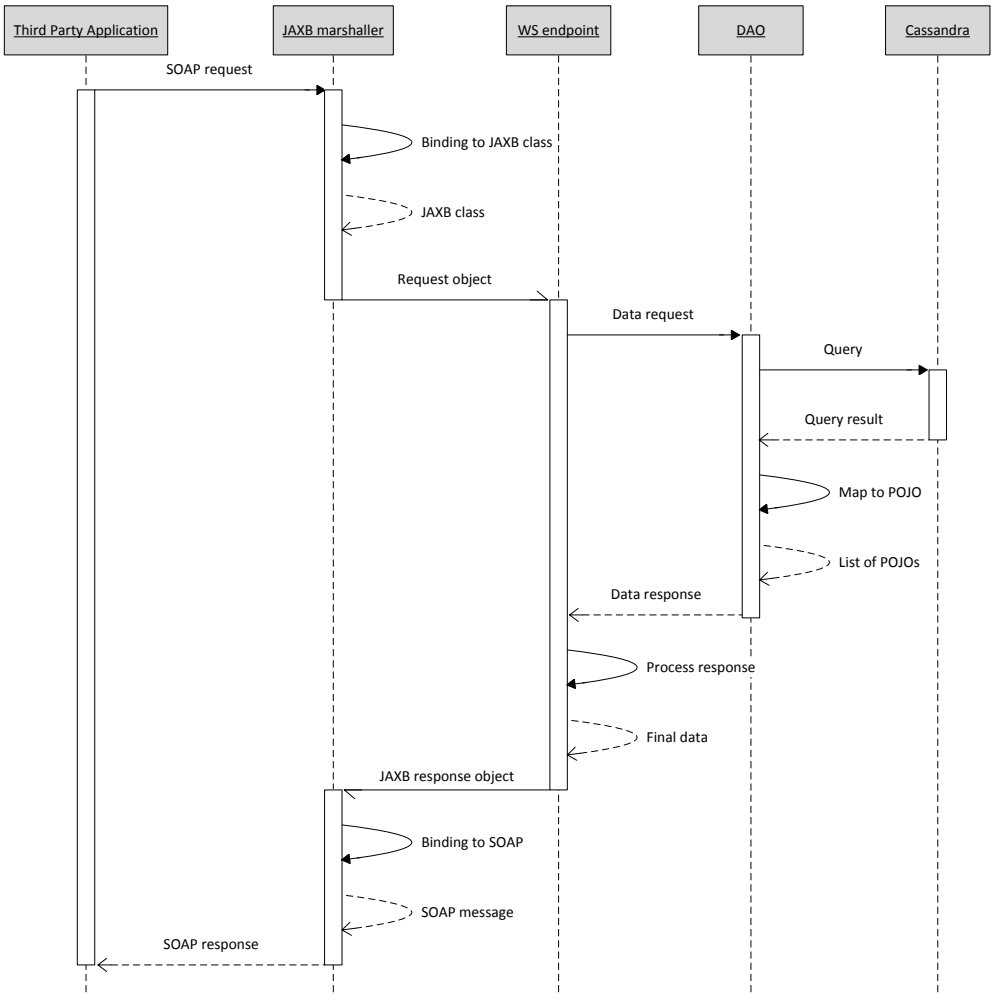

Figure 22: Web Service workflow

REST Web Services

RESTful API is developed with web application framework Pedestal for Clojure. Service is deployed with the help of build automation tool Leiningen. API provides flexible methods for reading historical data from the database. These services are intended for web applications, which need to present historical data by creating graphs and charts or analyzing the data.

Clojure application is running as a separate server application, receiving HTTP requests. For every request a mapping logic is defined, stating which function needs to be executed when request arrives. The function is responsible for handling of request by creation of relevant query to the database, processing the response from database to correspond to required response format, and returns JSON object back to the client. The whole workflow sequence is presented on a Figure 23.

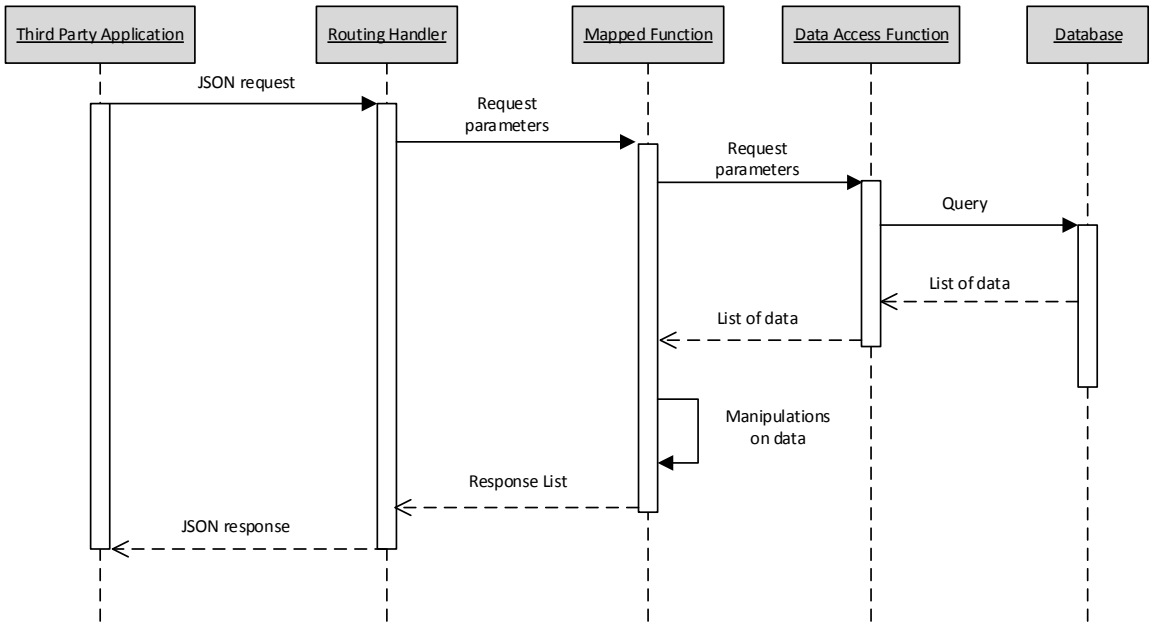

Figure 23: REST Web Service workflow

The processing stage can be performed very fast even on the very large data set, because the data is presented in form of a collection, which can be easily manipulated with Clojure.

4.4.5. Development of KPI definition interface

Due to the fact that CEP is one of the core functionalities of the information warehouse system, access to event processing rules for responsible personal needs to be transparent. KPI definition interface is created to satisfy this requirement. The interface is developed with JavaScript and HTML5.

The interface, presented on Figure 24, consists of 3 areas:

1. *Rule creation*
   User can add new rule and give it a name for identification purposes
2. *Rule selection, display and editing*
   User can navigate through already existing rules and select them for editing or deleting
3. *Message references*
   This area provides a reference to existing messages in the facility

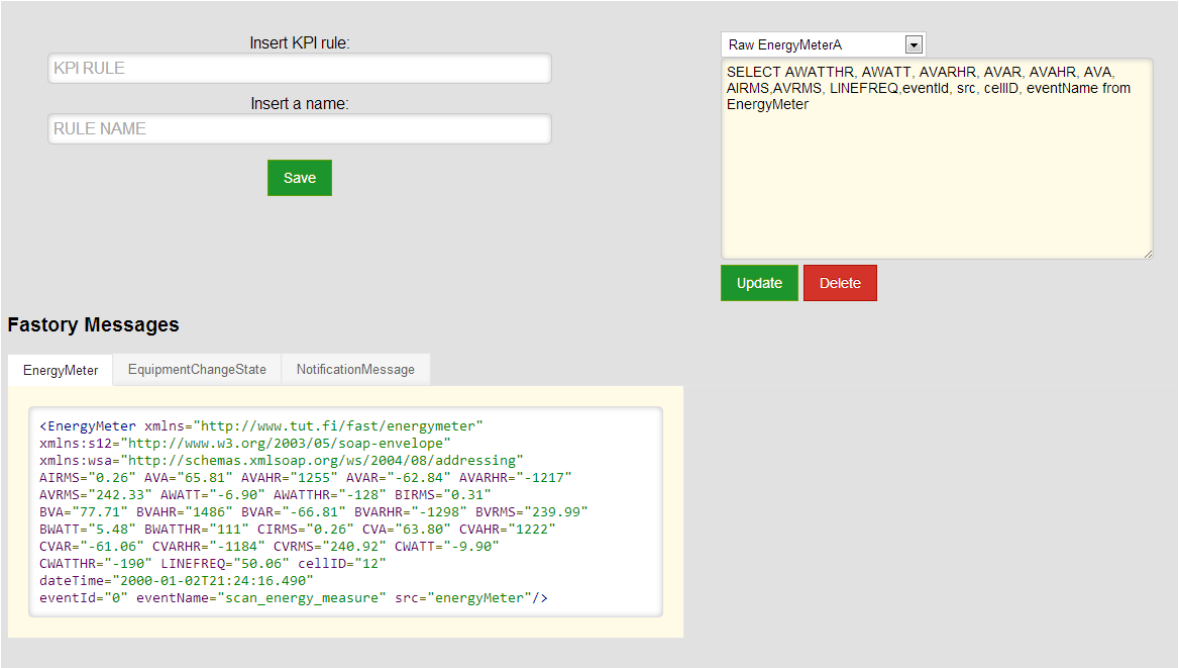

Figure 24: KPI definition interface

The interface is intended for a personal, which is familiar with EPL rules syntax.

4.4.6. Web application configuration

All modules described previously are initialized with spring configuration file *web.xml*, which is responsible for initialization of context configuration files and Web Service mappings. There are two context configuration files:

- *Application context*

  responsible for initialization of CEP engine, HTTP endpoint and Entity Manager Factory

- *Spring-ws-servlet*

  responsible for initialization of Spring Web Service

Figure 25 demonstrates the configuration and initialization process of the application.

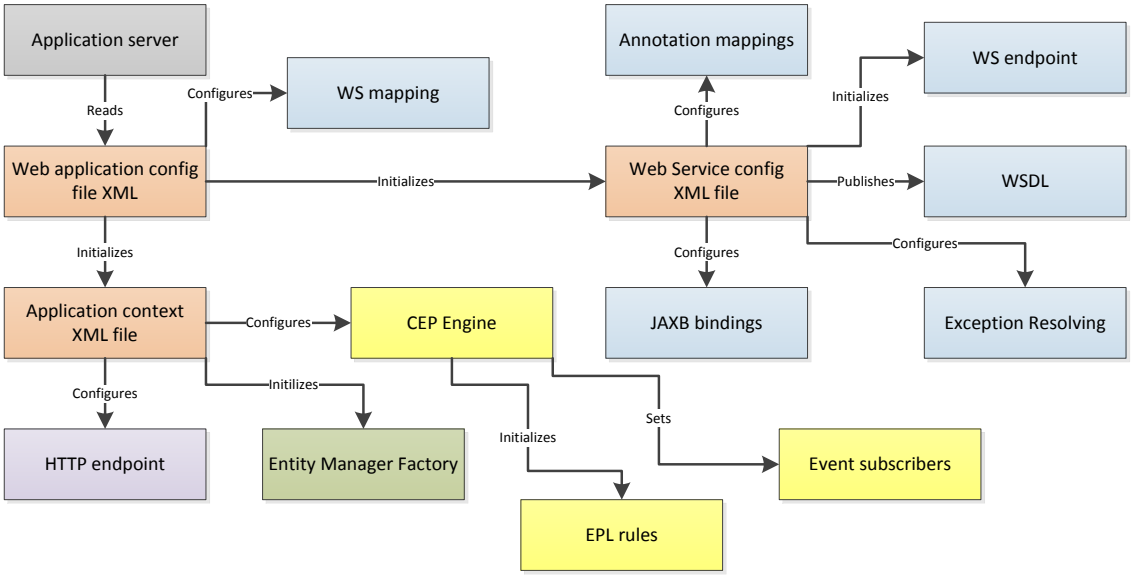

Figure 25: Application configuration with Spring

After configuration is completed, the application is ready to receive and process incoming data and Web Service requests.

4.5. Development of OSGi module

Application also needs to be made accessible inside of OSGi infrastructure. In order to complete the integration, a module responsible for access to Cassandra database needs to be created. This module can be used for generation of services, which can be used by other OSGi modules to access data from the database. However, the major challenge faced during this step was inability of Kundera library to run in OSGi environment. This is explained by the fact that Kundera is based on Thread context class loading, meaning that its context can't be loaded to OSGi thread, from where it could be accessible by other modules.

This problem was solved by creation of a new module, called *kunderaLibrary*, which exports all dependencies required by Kundera, and creates new methods to access the interface of Kundera library. The module contains all Java Archive (JAR) files, which are referenced by Kundera, resulting in a very heavy weight of the final module (34MB). In a bigger application heavy-weight module can reduce performance of the system, but this was the only possible solution to integrate Kundera library to OSGi environment.

5. RESULTS

5.1. Implementation test-bed

Designed approach system was implemented for a manufacturing line located in premises of FAST Laboratory in Tampere University of Technology. The line is composed of 12 cells, equipped with robots and conveyors (Figure 26), which is capable of performing simulation of mobile phone manufacturing by drawing parts of phone on a paper.

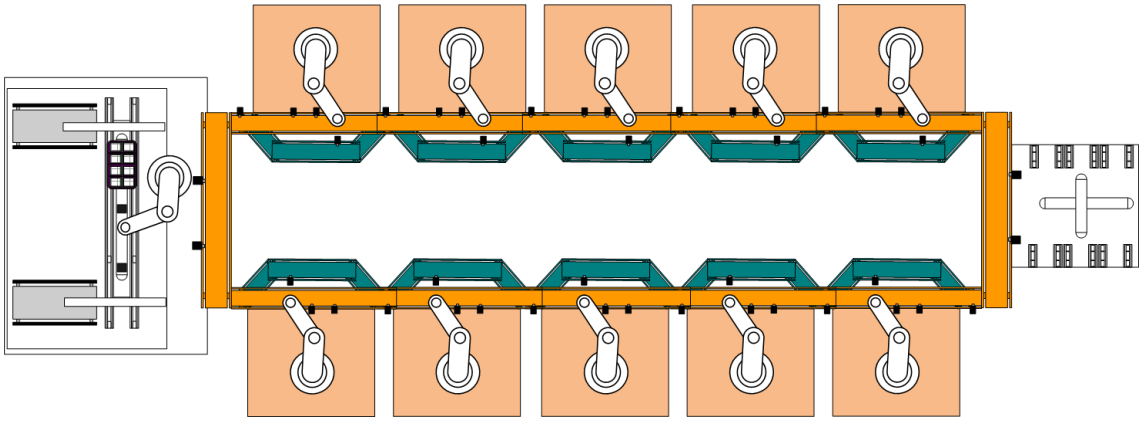

Figure 26: FAST manufacturing line

The line is equipped with S1000 controllers and E10 energy analyzers. S1000 controller is a programmable Smart Remote Terminal Unit device, designed to operate in industrial environment and compliant with all industrial signal types and levels, which provides web-based Human-Machine Interface (HMI) and Web Services integration. E10 is an additional module for S1000, which analyzes 3-phase electrical power consumption [80]. The following parameters are analyzed in E10:

- *Root mean square (RMS) voltage*
- *RMS current*
- *Active, reactive and apparent power*
- *Active, reactive and apparent energy*

5.1.1. Manufacturing line events

There are 3 types of events generated in a factory floor: energy message, equipment change state message and notification message.

Energy Meter message contains energy consumption related information of each cell, divided by 3 energy phases – A (robot), B (controller) and C (conveyor), as it is presented in example of message on Listing 6.

```
<EnergyMeter xmlns="http://www.tut.fi/fast/energymeter"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
AIRMS="0.26" AVA="65.81" AVAHR="1255" AVAR="-62.84" AVARHR="-1217"
AVRMS="242.33" AWATT="-6.90" AWATTHR="-128" BIRMS="0.31"
BVA="77.71" BVAHR="1486" BVAR="-66.81" BVARHR="-1298"
BVRMS="239.99"
BWATT="5.48" BWATTHR="111" CIRMS="0.26" CVA="63.80" CVAHR="1222"
CVAR="-61.06" CVARHR="-1184" CVRMS="240.92" CWATT="-9.90"
CWATTHR="-190" LINEFREQ="50.06" cellID="12"
dateTime="2000-01-02T21:24:16.490"
eventId="0" eventName="scan_energy_measure" src="energyMeter"/>
```

Listing 6: Energy Meter XML message example

Equipment Change State message carries data about status of robot inside of each cell, specifying current and previous CAMX (Computer Aided Manufacturing using XML) states, as well as all relevant information, which is presented on a Listing 7.

```
<EquipmentChangeState xmlns="http://www.tut.fi/fast/robot"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
cellID="6" condition="NORMAL" currentState="READY-IDLE-BLOCKED"
dateTime="2000-01-04T00:18:41.970" eventID="xx"
eventName="ItemWorkComplete"
palletID="13" previousState="READY-PROCESSING-EXECUTING"
recipeNum="7"
src="robot" toolID="1" transID="2"/>
```

Listing 7: Equipment Change State XML message

Notification message contains status of conveyor and relevant to it information for each of the cells (Listing 8).

```
<NotificationMessage
xmlns="http://www.pe.tut.fi/fast/wsdl/ConveyorService"
xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
cellID="5" dateTime="2000-01-03T23:49:41.STOPPED" eventID="75"
eventName="palDeparture" fromZoneID="4" opResult="NA"
palletID="13" src="conveyor" toZoneID="5" transID="NA"/>
```

Listing 8: Notification Message XML message

5.1.2. Energy Meter data model

According to message structure, described in previous section, contents of Energy Meter message can be divided into phase-specific energy consumption data and common cell data. The potential query use cases are presented below:

- *Get active energy power data for robot in cell number 1 during previous day*
- *Get energy consumption for conveyors in each cell during last week*
- *Get specific energy parameters of all phases in several cells during specific time range*

From these use cases a conclusion can be made that data can be identified distinctly by cell number, energy phase and time. These three parameters define a compound key, where cell id is a partition key, energy phase and timestamp are clustering keys. According to this data model, energy data will be stored on different nodes of cluster based on a hash value of cell id, and will be clustered in accordance to energy phase and will be sorted by timestamp. Figure 27 illustrates the data model in an understandable way and shows the way data is clustered.

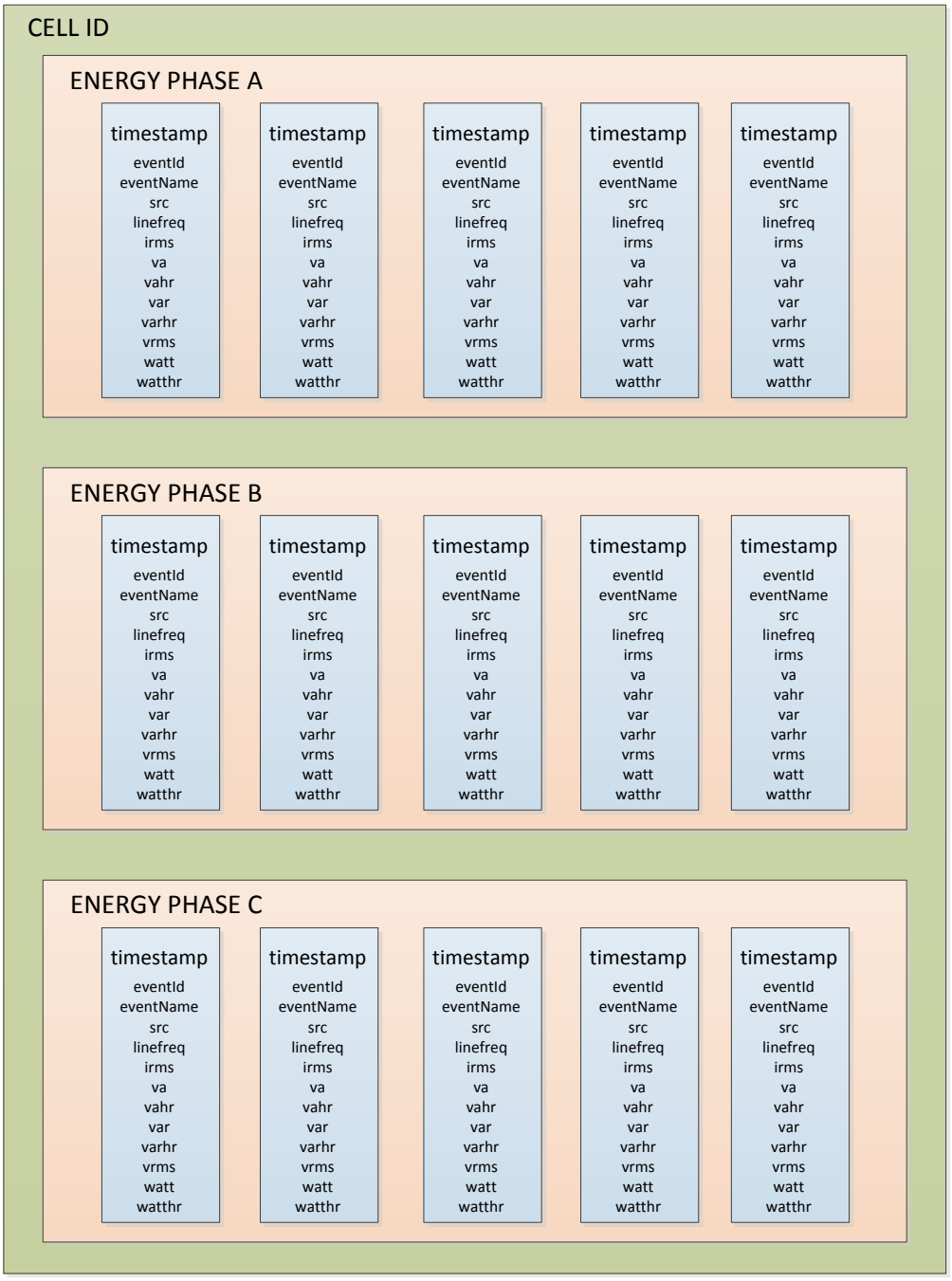

Figure 27: Data model of Energy Meter table

5.1.3. Equipment Change State and Conveyor Notification data models

As it was described earlier, equipment change state messages contain relevant information about status of robot in each cell. This information is tightly connected with energy efficiency, thus needs also to be stored and available for requests.

Potential query use cases look like as following:

- *Get amount of occurrences of "READY-PROCESSING-EXECUTING" state in cell number 1 during last hour*
- *Get timestamps of events when robot was blocked in cell number 3*

According to presented use cases and structure of XML message, it can be determined that equipment change state data can be uniquely identified by combination of cell id, event name and timestamp.

Conveyor Notification message also carries information that might be relevant for analysis of energy efficiency of manufacturing line. The query and data models are similar to Equipment Change State message. Illustrations of Equipment Change State and Conveyor Notification tables are available in Appendix 2.

5.1.4. KPI data model

KPI data is also needs to be stored in a database with predefined data model. However, in contrast to described earlier tables for data coming from devices, a single data model needs to be defined for any type of KPI. This approach will result in a possibility to focus in the future on development of KPIs without requirements for creating new data structure for each newly introduced KPI.

All information in a context of KPI can be divided into 3 types with following contents:

- KPI definition
    - KPI name
    - Cell ID (or consumer/scope)
    - Timestamp
- Actual value of KPI
    - Numerical value
- Metadata
    - Pallet ID
    - CAMX State
    - Product
    - Any other relevant information

According to these types of data, query use cases can be defined:

- *Get values of energy consumption KPI for cell number one for last month*
- *Get values of energy consumption KPI for whole manufacturing line for specified time period*
- *Get value of energy consumption KPI and its metadata of specific cell for specified time period*

The data model of KPI table thus should have a compound key, consisting of KPI name as partition key, cell ID and timestamp as clustering keys. Moreover, KPI metadata can include any relevant for particular KPI information, thus data model needs to support a flexible addition of KPI metadata parameters. This can be achieved with help of collections support realized in Cassandra, so that metadata column can be

defined as map containing key-value pairs of text variables. Key-value pairs can be added when needed for specific KPI and can always be fetched, when requesting the data. The data model illustration is available in Appendix 2.

5.2. Application integration

In order to complete the thesis work, the application has been integrated to the FASTory production line. Cassandra database has been deployed to Linux server, which has accessible public IP. Cassandra has also been configured to be bounded to the public IP of the server.

The application has been running on a Windows machine connected to Local Area Network (LAN) of the production line, so that messages from Event Hub could be redirected to the application.

Application has been configured to access the deployed database, and all used entities for mapping of data has been added, as it is illustrated on a Listing 9.

```
<persistence-unit name="fastoryDatastore_pu">
<provider>com.impetus.kundera.KunderaPersistence</provider>
<class>tut.fi.dpe.fast.datastore.model.EquipmentChangeState</class>
<class>tut.fi.dpe.fast.datastore.model.EquipmentChangeStatekey</class>
<class>tut.fi.dpe.fast.datastore.model.EnergyMeter</class>
<class>tut.fi.dpe.fast.datastore.model.EnergyMeterKey</class>
<class>tut.fi.dpe.fast.datastore.model.QualityInspectionShort</class>
<class>tut.fi.dpe.fast.datastore.model.QualityInspectionShortKey</class>
<class>tut.fi.dpe.fast.datastore.model.ThlValue</class>
<class>tut.fi.dpe.fast.datastore.model.ThlValueKey</class>
<class>tut.fi.dpe.fast.datastore.model.ConveyorNotification</class>
<class>tut.fi.dpe.fast.datastore.model.ConveyorNotificationKey</class>
<class>tut.fi.dpe.fast.datastore.model.SilentPowerUnit</class>
<class>tut.fi.dpe.fast.datastore.model.SilentPowerUnitKey</class>
<class>tut.fi.dpe.fast.datastore.model.FastoryKpi</class>
<class>tut.fi.dpe.fast.datastore.model.FastoryKpiKey</class>
<exclude-unlisted-classes>true</exclude-unlisted-classes>

<properties>
<property name="kundera.nodes" value="localhost"/>
<property name="kundera.port" value="9160" />
<property name="kundera.keyspace" value="FastoryDatastore" />
<property name="kundera.dialect" value="cassandra" />
<property name="kundera.client.lookup.class"
value="com.impetus.client.cassandra.thrift.ThriftClientFactory" />
<property name="kundera.cache.provider.class"
value="com.impetus.kundera.cache.ehcache.EhCacheProvider" />
property name="kundera.ddl.auto.prepare" value="update" />
</properties>
</persistence-unit>
```

Listing 9: Database configuration file persistence.xml

Http endpoint inside of the application has also been configured to listen to incoming HTTP POST requests on the IP address of local machine. This address was also added to configuration of routes inside of the Event Hub.

5.3. Web services

In order to make stored in database data transparent and available for relevant personal, Web services for data access have been developed. All created services are summarized in a Table 16.

Table 16: Developed Web Services

Name	Functionality	Type
Historical Data	Returns requested historical data	SOAP
KPI Data	Returns requested KPI data	SOAP
Full Energy Data	Returns historical data of energy meter message	REST
Full Robot Data	Returns historical data of robot equipment change state message	REST
Full Conveyor Data	Returns historical data of conveyor notification message	REST
Chart data	Returns historical energy data, which can be used for plotting charts and graphs	REST

5.3.1. Historical data

This service is used to access any available historical energy data in a various combinations. Inputs of the service include:

- List of cells
- List of phases
- List of energy parameters
- List of time ranges

In other words, this service allows user to form as flexible and detailed request for any amount of cells, energy phases, energy parameters and time ranges, as it is required by user. The example of the request message is illustrated on a Listing 10.

Response message, generated by this service, has very complex structure, which, however, can be easily parsed by a client's side application.

```
<HistoricalDataRequest>
   <cellsList>
      <cell>1</cell>
      <cell>2</cell>
      <cell>3</cell>
   </cellsList>
   <phasesList>
      <phase>A</phase>
      <phase>B</phase>
   </phasesList>
   <parametersList>
      <parameter>wattz</parameter>
      <parameter>watthr</parameter>
   </parametersList>
   <timeRangesList>
      <timeRange id=”1”>
         <timeStart>13768770040000</timeStart>
         <timeEnd>13768770080000</timeEnd>
      </timeRange>
   </timeRangesList>
</HistoricalDataRequest>
```

Listing 10: Historical Data request

5.3.2. KPI data

This service is intended for giving access to all available KPIs stored in the database. The input information to the service includes:

- KPI name
- Cell ID
- Time range

With KPI data service is possible to get values of particular KPI for specific cell during some period of time. The response of the service contains the list of *KpiData* objects, where each of them includes:

- KPI name
- Cell ID
- KPI value
- Timestamp
- Element containing list of metadata variables

5.3.3. Full energy data

This RESTful service is also used to access energy related information stored in the database. The input of the service include following information:

- List of cells
- List of phases
- Time range

The service is accessed via URI and inputs are added to request as query parameters:

http://localhost:8090/data/fastory/full/energy?cell=3,4&phase=A&timestart=1389621600000&timeend=1389632400000

For this input data service returns JSON object containing all energy data for specified cells and phases inside of time range. The example of a response is presented on a Figure 28.


```
[{"cell_id":3,"va":265.489990234375,"timestamp":1389621603955,"eventName":"
scan_energy_measure","vrms":236.0,"irms":1.090000033378601,"src":"energyMeter"
                                        ,
"linereq":50.0,"watthr":167.0,"eventId":0,"energy_phase":"A","watt":101.040000
       91552734,"vahr":473.0,"varhr":-213.0,"var":-110.86000061035156}]
```


Figure 28: Full energy data response

5.3.4. Full robot data

This service provides a historical data of equipment state changes of robots. The input of the service needs to have:

- Cell ID
- Event name
- Time range

Full robot data, similarly to full energy data service, has input parameters in a query part of HTTP request:

http://localhost:8090/data/fastory/full/robot?cell=3&eventname=ItemWorkComplete&timestart=1389617346055&timeend=1390993127191

An example of response produced by this service is presented on a Figure 29.

[{"cell_id":3,"timestamp":1389618874566,"event_name":"ItemWorkComplete","trans_id":"2","src":"robot","eventId":"01","current_state":"READY-IDLE-BLOCKED","previous_state":"READY-PROCESSING-EXECUTING","tool_id":1,"condition":"normal","recipe_number":2,"pallet_id":6}]

Figure 29: Full robot data response

5.3.5. Full conveyor data

Similarly to two previously described services, this service provides user with a historical data, containing information about conveyor statuses. The input to the service needs to include:

- Cell ID
- Event name
- Time range

The HTTP request for the full conveyor data service contains the input information as a query part:

http://localhost:8090/data/fastory/full/conveyor?cell=3&eventname=palArrival&timestart=1389617346055&timeend=1390993127191

An example of the response is illustrated on Figure 30.

[{"cell_id":3,"timestamp":1389618008158,"event_name":"palArrival","from_zone_id":1,"trans_id":"5","event_id":2,"src":"conveyor","op_result":"inPosition","to_zone_id":4,"pallet_id":2}

Figure 30: Full conveyor data response

5.3.6. Chart data

This service can be used specifically for displaying the historical energy data, for instance, in form of charts. For the input containing cell id, energy phase, energy parameter and time range, the service returns an object with timestamp-value pairs, as it is presented on a Figure 31.

[[1389617205786,102.30999755859375],[1389617206053,102.62999725341797],[1389617206353,102.45999908447266],[1389617207086,102.45999908447266],[1389617207421,101.44000244140625]

Figure 31: Chart data response

5.4. Integration to OSGi

The developed information warehouse system was integrated into FASTory OSGi-based messaging infrastructure. At the moment, when this thesis was written, the infrastructure was in the development stage. FASTory OSGi infrastructure is running on Apache Service Mix ESB, and provides functionality of creation of custom *function blocks*. Function block is constructed from various available modules. For instance, if developer wants to create a service, which will be fetching some data from database, making analysis of data and returning the result to user, he or she will create function block that encapsulates database module, analysis module and parser module for response. Such function block can be created by sending XML message to *function block manager* module. Inside of function block modules are communicating also by exchanging XML messages.

Module, responsible for accessing data from Cassandra was created. This module uses kunderaLibrary, which is presented in Chapter 4.5, as a dependency, so that it can call all data access methods of Kundera. In the newly created module methods to access historical energy data were added. Modules were tested in Apache Service Mix, and in the result response from database was accessible inside of the OSGi environment.

5.5. Tests

In order to validate the applicability of developed information warehouse system, its functionalities have been test in real-life situations.

5.5.1. Data capturing and storage

Application was tested while the FASTory line was in operation mode, during which all data generated by controllers have been stored. The line has been running for 2 hours, and the results are presented on a Table 17.

Table 17: Data storage result

Theoretical amount of generated messages per cell	Amount of rows in database per phase of cell
1440	1376

Energy message for a single cell is generated every 5 seconds. Therefore, 12 messages are generated in 1 minute and 1440 messages in 2 hours. However, this number differs from the amount of stored data. This can be explained by failures of the manufacturing line, when the messages were not generated at all. This period has lasted for 8 minutes totally during these 2 hours.

Moreover, several times application was running to store simulated data for testing purposes. During that times all data has been successfully stored to the database.

Figures 32-33 present examples of historical data for different phases of cell number 3. The time periods of 8 minutes, when raw data was not available, are marked with red frame on the pictures.

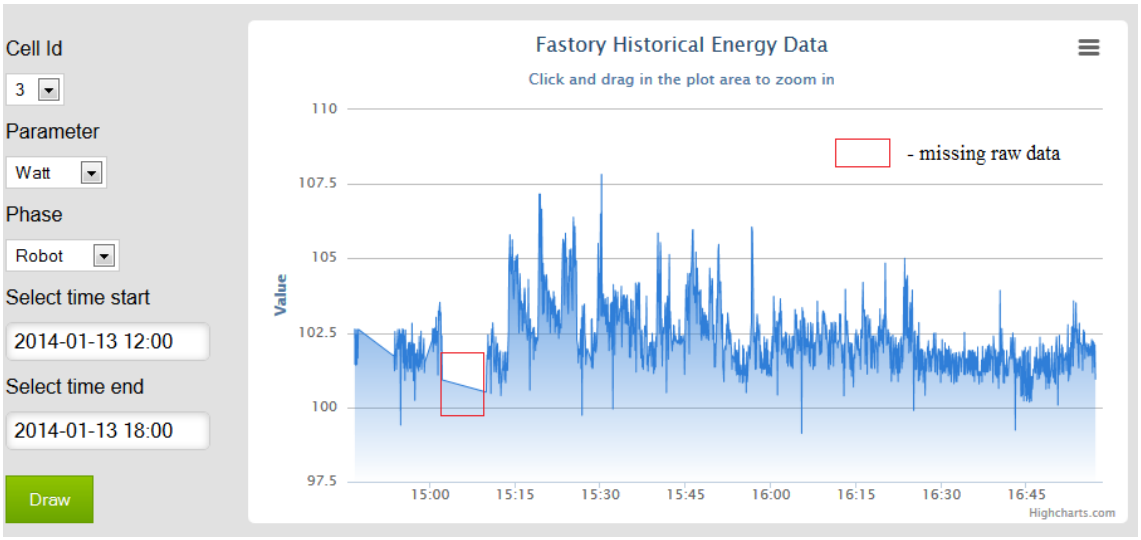

Figure 32: Historical power data for robot phase of cell 3

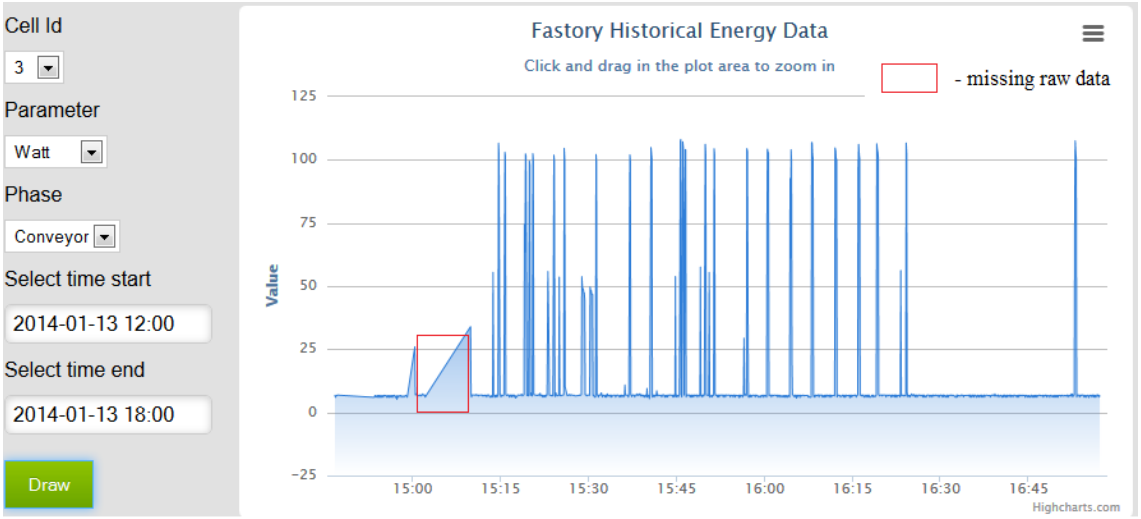

Figure 33: Historical power data for conveyor phase of cell 3

A clear pattern of energy consumption can be seen from the figures above.

5.5.2. Performance benchmarking

In order to compare performance of Cassandra database with a traditional relational database (MySQL), benchmarking test for writes and reads from databases were made.

Test environment

MySQL database was running on the same Linux server as Cassandra. Table, intended for storing energy data was created in a test keyspace in Cassandra database, as it is illustrated on Listing 11.

```
CREATE TABLE energy_meter (
cell_id int,
energy_phase text,
timestamp bigint,
event_id int,
event_name text,
irms float,
linefreq float,
src text,
va float,
vahr float,
var float,
varhr float,
vrms float,
watt float,
watthr float,
PRIMARY KEY (cell_id, energy_phase, timestamp)
```

Listing 11: Creation of energy_meter table for testing in CQL utility

Another table, corresponding to this table, was created in MySQL database, as it is shown on Listing 12. The same structure was chosen in order to be able to compare performance of databases on the atomic operations level.

```
CREATE TABLE 'energy_meter' (
ID bigint(20) AUTO_INCREMENT,
cell_id int(11),
energy_phase varchar(50),
timestamp bigint(20)
event_id int(11),
src varchar(50),
event_name varchar(50),
linefreq float,
irms float,
va float,
vahr float,
var float,
varhr float,
vrms float,
watt float,
watthr float,
PRIMARY KEY (ID))
```

Listing 12: Creation of energy_meter table for testing in MySQL

Tests included write and read operations. For write tests a special classes were created, which were generating random data. For read tests a real data was queried. Real data from Cassandra was copied to the MySQL database for this purpose.

Consequent writes

Firstly, time required to complete a specific amount of consequent writes to databases was tested. Tests were performed for 10, 100 and 1000 writes. This test represents a use case, when messages from the factory floor are consequently arriving to the application. Results of the test are presented on a Table 18.

Table 18: Consequent writes

Number of writes	Total time, ms (MySQL)	Total time, ms (Cassandra)
10	2106	771
100	12038	9831
1000	271806	140060

From the results we can see that on average performance on consequent writes of Cassandra database was twice faster, comparing to MySQL.

Concurrent writes

In a second test an average time per write for 10, 100 and 1000 concurrent writes was measured. This test measures the time needed to save messages under high load of concurrent connections to database. The results are presented on a Table 19.

Table 19: Concurrent writes

Number of writes	Average time, ms (MySQL)	Average time, ms (Cassandra)
10	182.9	16.8
100	89.3	12
1000	78	11.5

As we can see from the results, Cassandra's performance on concurrent writes is about 8 times faster than MySQL's performance.

Concurrent reads

In the last test 3000 rows were concurrently red from the databases for 10, 100 and 1000 threads, and average time per read was measured. This test represents a use case, when user is making a request to database to get a historical data for specific period of time. The results are presented on a Table 20.

Table 20: Concurrent reads

Number of threads	Average time, ms (MySQL)	Average time, ms (Cassandra)
10	980	1457.7
100	386.9	1262.7
1000	433.6	1441.7

According to the results of this test, MySQL performed 2.5 times faster on a reads, comparing to Cassandra.

Tests results

Performed tests demonstrate that Cassandra has very fast and efficient writes, what is very important for the process of storage of captured data. From the other side, MySQL outperformed Cassandra on reads, which means that Cassandra might have slower response time for client requests. However, this weak performance can be explained by the fact that tests were performed on a small amount of data and only one Cassandra node was used. In future, same tests, but on a cluster of Cassandra databases should be performed, in order to achieve optimal results.

6. CONCLUSIONS

6.1. Implementation conclusions

Energy management has already become one of the most crucial issues, faced by manufacturing industry. It has been realized by authorities that amount of energy consumption and carbon dioxide emissions need to be decreased, especially in manufacturing facilities. Currently available solutions for energy management are based on Service Oriented Architecture, which is now a de-facto standard architecture for development of energy management systems. This architectural approach allows decoupling application logics by providing various services, which can be combined in a flexible manner.

Another important aspect of energy management system is a requirement to capture and store huge amounts of data coming from all possible devices on a factory shop floor, in order to produce a valuable feedback and basis of energy efficient decisions. This data is often referred as a Big Data due to its complex structure and huge volume. Nowadays, energy management systems implemented in manufacturing facilities do not support Big Data handling. However, according to the estimations that the growth of amounts of data is going to be exponential in the next several years, Big Data support in manufacturing facilities needs to be addressed and considered.

In this thesis work an information warehouse system for manufacturing facility has been designed. The main advantage of designed system is a modular architecture, based on SOA, which allows the system to be implemented in any manufacturing facility. The developed information warehouse system is capable of providing the authorities and personal with meaningful information about the energy consumption of the facility, thus allowing making decisions leading to the decrease of energy consumption. This system provides a platform for efficient energy management by offering functionalities, responsible for capturing and storage of raw data, calculation of Key Performance Indicators and providing Web Services to access all stored information.

The proposed system has a multi-layered architecture, where each layers is responsible for a set of specific functions and provides services for the higher layer. The architecture encapsulates variety of technologies, frameworks and tools developed in Java, Clojure and JavaScript programming languages.

In order to prove the applicability of the developed system, it was implemented for a real production line. For the implementation a NoSQL Cassandra database has been used instead of a traditional relational database, in order to make the solution Big Data-ready.

Finally, the developed information warehouse system can serve as a foundation for energy management system in a real facility, where more facility-specific services can be added and behavior of system can be adjusted to correspond to relevant context. The developed system is flexible in the way that it is not linked to any specific facility and can be easily reconfigured.

6.2. Future work

In order to get the optimal performance from Cassandra database for the developed information warehouse system, more database servers need to be added to cluster. Besides, further research on applicability of more advanced Big Data technologies need to be conducted. These technologies include implementation of Hadoop Distributed File System and MapReduce parallel processing technique and have a great potential for more efficient and faster processing of big amounts of energy data.

REFERENCES

[1] McKinsey Global Institute, November 2012, “Manufacturing the future: The next era of global growth and innovation”, 24p

[2] 2013, “Intelligent Manufacturing: targeting better energy efficiency”, A report by the Economist Intelligence Unit sponsored by ABB

[3] Dr. Barry Devlin, Shawn Rogers & John Myers, 2012 “Big Data Comes of Age”, EMA and 9sight Consulting Report, 43p

[4] Markus Löffler & Andreas Tschiesner**,** 2013, “The Internet of Things and the Future of Manufacturing”, McKinsey&Company

[5] 2013, VTT, Espoo, ”VTT Visions: Productivity Leap with IoT”, 99p

[6] Srinivasa, S., Bhatnagar, V., “Scalable Analytics – Algorithms and Systems”, 2012, Springer, Big Data Analytics, First International Conference, New Delhi, India, pp. 1-7

[7] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Byers, A.H., “Big data: The next frontier for innovation, competition, and productivity”, 2011, McKinsey Global Institute, 156p

[8] Apache Hadoop, available online: http://hadoop.apache.org/ [accessed on 7.3.2014]

[9] March 2012, IDC #233485, Volume 1, Tab: Markets, “Big Data: Global Overview: Market Analysis”, 30p

[10] Oracle White Paper, June 2013, “Oracle: Big Data for the Enterprise”, 15p

[11] DIN EN 16001: Energy Management Systems In Practice A guide for Companies and Organizations, 2010, Federal Ministry for the Environment, Nature Conservation and Nuclear Safety (BMU)

[12] Bunse, K., Vodicka, M., Schönsleben, P., Brulhart, M., Ernst, F. O., "Integrating energy efficiency performance in production management – gap analysis between industrial needs and scientific literature”, Journal of Cleaner Production, Vol. 19, No. 6-7, 25 April 2011, pp. 667-679

[13] Rudberg, M., Waldemarsson, M., Lidestam, H., “Strategic perspectives on energy management: A case study in process industry”, 2013, Elsevier, Applied Energy, vol. 104, pp. 487-496

[14] BSR, “Energy Management Handbook”, April 2012, available online: www.bsr.org/reports/bsr-energy-management-handbook.pdf [accessed on 15.03.2014]

[15] Vikhorev, K., Greenough, R., Brown, N., “An advanced energy management framework to promote energy awareness”, 2013, Elsevier, Journal of Cleaner Production, vol. 43, pp. 103-112

[16] Zhang, B., Postelnicu, C., Lastra, J.L.M., "Key Performance Indicators for energy efficient asset management in a factory automation testbed", 2012, 10th IEEE International Conference on Industrial Informatics (INDIN), pp.391-396

[17] May, G., Taisch, M., Kelly, D., "Enhanced energy management in manufacturing through systems integration", 2013, 39th Annual Conference of the IEEE Industrial Electronics Society IECON, pp.7525-7530

[18] Drumm, C., Busch, J., Dietrich, W., Eickmans, J., Jupke, A., “STRUCTese® – Energy efficiency management for the process industry”, 2013, Elsevier, Chemical Engineering and Processing: Process Intensification, vol. 67, pp. 99-110

[19] Marinakis, V., Doukas, H., Karakosta, C., Psarras, J., “An integrated system for buildings’ energy-efficient automation: Application in the tertiary sector”, 2013, Elsevier, Applied Energy, vol. 101, pp. 6-14

[20] Cannata, A., Karnouskos, S., Taisch, M., "Energy efficiency driven process analysis and optimization in discrete manufacturing", 2009, 35th Annual Conference of IEEE Industrial Electronics IECON '09, pp.4449-4454

[21] TechAmerica Foundation, “Demystifying big data. A Practical Guide To Transforming The Business of Government”, 3 October 2012, Report, 39p

[22] Sagiroglu, S., Sinanc, D., "Big data: A review", 2013 International Conference on Collaboration Technologies and Systems (CTS), pp.42-47

[23] Philip Russom, “Big Data Analytics”, 2011, TDWI Best Practices Report

[24] IBM Global Services, “Analytics: The real-world use of big data”, October 2012, Executive Report

[25] Demchenko, Y., Grosso, P., De Laat, C., Membrey, P., "Addressing big data issues in Scientific Data Infrastructure", 2013, International Conference on Collaboration Technologies and Systems (CTS), pp.48-55

[26] Forsyth Communications, “For Big Data Analytics There’s No Such Thing as Too Big”, March 2012, 20p

[27] Katal, A., Wazid, M., Goudar, R.H., "Big data: Issues, challenges, tools and good practices," Contemporary Computing (IC3), pp.404-409

[28] Wikipedia, Zettabyte, available online: http://en.wikipedia.org/wiki/Zettabyte [accessed on 14.3.2014]

[29] A.T Kerney Inc, “Big Data and the Creative Destruction of Today’s Business Models”, 2013, available online: https://www.atkearney.com/strategic-it/ideas-insights/article/-/asset_publisher/LCcgOeS4t85g/content/big-data-and-the-creative-destruction-of-today-s-business-models/10192 [accessed on 14.3.2014]

[30] Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, C., “Cloud Computing: a Perspective Study”, 2010, Springer, New Generation Computing, vol. 28, issue 2, pp. 137-146

[31] Gupta, R., Gupta H., **“**Cloud Computing and Big Data Analytics: what is new from databases perspective?”, 2012, Springer, Big Data Analytics, First International Conference, BDA 2012, New Delhi, India, pp. 42-61

[32] Amazon EC2, available online: http://aws.amazon.com/ec2/ [accessed on 15.03.2014].

[33] Google Cloud Platform, Compute Engine, available online: https://cloud.google.com/products/compute-engine/ [accessed on 15.03.2014]

[34] van der Veen, J.S., van der Waaij, B., Meijer, R.J., “Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual”, 2012, IEEE 5th International Conference on Cloud Computing (CLOUD), pp.431-438

[35] Mohan, C., “History Repeats itself: Sensible and NonsenSQL Aspects of the NoSQL Hoopla”, 2013, Proceedings of the 16th International Conference on Extending Database Technology, pp. 11-16

[36] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach D. A., Burrows, M., Chandra, T., Fikes, A., Gruber R.E., “Bigtable: A Distributed Storage System for Structured Data”, 2008, ACM Transactions on Computer Systems, vol. 26 issue 2

[37] DeCandia, G., Hastorun D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W., “Dynamo: amazon’s highly-available key-value store”, Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, 2007, pp. 205-220

[38] Özsu, M., Valduriez, P., “Principles of Distributed Database Systems”, Springer, 3rd ed., 2011, 860p

[39] Wikipedia, CAP theorem, available online: http://en.wikipedia.org/wiki/CAP_theorem [accesses on 15.03.2014]

[40] Cassandra Wiki, Architecture Overview, available online: http://wiki.apache.org/cassandra/ArchitectureOverview [accessed on 15.03.2014]

[41] Redmond, E., Wilson, J. R., "Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement", the Pragmatic Bookshelf, 11.05.2012, 354p.

[42] Wang, G., Tang, J., "The NoSQL Principles and Basic Application of Cassandra Model", 2012, International Conference on Computer Science & Service System (CSSS), pp. 1332-1335

[43] Sadalage, P. J., Fowler, M., "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", Addison-Wesley Professional, 2012, 192p

[44] Knowledge Base of Relational and NoSQL Database Management Systems, DB-engines Ranking, available online: http://db-engines.com/en/ranking [accessed on 15.03.2014]

[45] MongoDB, The Mongo 2.4 Manual, available online: http://docs.mongodb.org/manual/ [accessed on 15.03.2014]

[46] Huang, S., Cai, L., Liu, Z., Hu, Y., "Non-structure Data Storage Technology: A Discussion", 2012, IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), pp.482-487

[47] DATASTAX Documentation, Apache Cassandra 1.2, Understanding the architecture, available online: http://www.datastax.com/documentation/cassandra/1.2/webhelp/index.html?pagename=docs&version=1.2&file=index#cassandra/architecture/architectureTOC.html [accessed on 15.03.2014]

[48] Thantriwatte, T. A. M. C., Keppetiyagama, C. I., "NoSQL query processing system for wireless ad-hoc and sensor networks", 2011, International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 78-82

[49] Yamamoto, S., Matsumoto, S., Nakamura, M., "Using cloud technologies for large-scale house data in smart city", 2012, IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 141-148

[50] Jayathilake, D., Sooriaarachchi, C., Gunawardena, T., Kulasuriya, B., Dayaratne, T., "A study into the capabilities of NoSQL databases in handling a highly heterogeneous tree", 2012, IEEE 6th International Conference on Information and Automation for Sustainability (ICIAfS), pp.106-111

[51] Ohene-Kwofie, D., Otoo, E.J., Nimako, G., "O2-Tree: A Fast Memory Resident Index for NoSQL Data-Store", 2012, IEEE 15th International Conference on Computational Science and Engineering (CSE), pp.50-57

[52] Erl, T., "SOA Principles of Service Design", 2007, Prentice Hall, 865p

[53] Wiehler, G., "Mobility, Security and Web Services", 2004, Publics Corporate Publishing, Erlangen, 220p

[54] Cohen, F., "Fast SOA: The way to use native XML technology to achieve Service Oriented Architecture governance, scalability, and performance", 2007, Morgan Kaufmann Publishers Inc, 279p

[55] Lastra, J.L.M., Delamer, I.M., "Semantic Web Services in Factory Automation", 2005, Tampere University of Technology, Institute of Production Engineering, Report 70, 67p

[56] Zhou, H.J., Cao, J.Z., Guo, C.X., Qin, J., "The architecture of intelligent distribution network based on MAS-SOA", 2010, IEEE Power and Energy Society General Meeting, pp. 1-6

[57] Li Da Xu, "Enterprise Systems: State-of-the-Art and Future Trends", 2011, IEEE Transactions on Industrial Informatics, vol.7, no.4, pp.630-640

[58] W3C, Web Services Architeecture, "What is a Web Service?", available online: http://www.w3.org/TR/ws-arch/#whatis [accessed on 15.03.2014]

[59] Laine, M., "RESTful Web Services for the Internet of Things", 2011, Department of Media Technology, Aalto University School of Science, 3p.

[60] Shi, X., "Sharing Service Semantics using SOAP-Based and REST Web Services" 2006, IT Professional, vol. 8, no. 2, pp. 18-24

[61] Pautasso, C., Zimmermann, O., Leymann, F., "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision", 2008, Proceedings of the 17th international conference on World Wide Web, pp. 805-814

[62] Zhu, C., Chai, M., Lu, Y., Guo, Y., "Service Oriented Architecture Design of Energy Consumption Information System about Petroleum Enterprise", 2013, Fifth International Conference on Computational and Information Sciences (ICCIS), pp. 1174-1177

[63] Yang, J., Li, J., Deng, X., Xu, K., Zhang, H., "A web services-based approach to develop a networked information integration service platform for gear enterprise", 2012, Springer, Journal of Intelligent Manufacturing, vol. 23, issue 5, pp. 1721-1732

[64] Cucinotta, T., Mancina, A., Anastasi, G.F., Lipari, G., Mangeruca, L., Checcozzo, R., Rusina, F., "A Real-Time Service-Oriented Architecture for Industrial Automation", 2009, IEEE Transactions on Industrial Informatics, vol.5, no.3, pp. 267-277

[65] Wu, J., Tao, X., "Research of enterprise application integration based-on ESB", 2010, 2nd International Conference on Advanced Computer Control (ICACC), vol.5, pp.90-93

[66] Chen, Q., Shen, J., Dong, Y., Dai, J., Xu, W., "Building a Collaborative Manufacturing System on an Extensible SOA-based Platform", 2006, 10th International Conference on Computer Supported Cooperative Work in Design CSCWD '06, pp.1-6

[67] EsperTech, available online http://esper.codehaus.org/ [accessed on 15.03.2014]

[68] Saleh, O., Sattler, K.-U., "Distributed Complex Event Processing in Sensor Networks", 2013 IEEE 14th International Conference on Mobile Data Management (MDM), vol.2, pp.23-26

[69] Wang, G., Jin, G., "Research and Design of RFID Data Processing Model Based on Complex Event Processing", 2008, International Conference on Computer Science and Software Engineering, vol.5, pp.1396-1399

[70] Etzion, O., Niblett, P., "Event Processing in Action", 2010, Manning Publications Co, 360p

[71] Wang, Y.H., Cao, K., Zhang, X.M., "Complex event processing over distributed probabilistic event streams", 2013, Elsevier, Computers & Mathematics with Applications, vol. 66, issue 10, pp. 1808-1821.

[72] EsperTech, Esper Reference, available online: http://esper.codehaus.org/esper-4.10.0/doc/reference/en-US/html/index.html [accessed on 15.03.2014]

[73] GitHub, impetus-opensource/Kundera, available online: https://github.com/impetus-opensource/Kundera/wiki [accessed on: 15.03.2014]

[74] Lui, M., Gray, M., Chan, A., Long, J., "Pro Spring Integration", 2011, Apress Berkley, CA, USA, 641p

[75] Walls, C., "Spring in Action, Third Edition", 2011, Manning Publications Co, 424p

[76] GitHub, Pedestal repository, available online: https://github.com/pedestal/pedestal [accessed on 15.03.2014]

[77] GitHub, clojurewerkz/cassaforte repository, available online: https://github.com/clojurewerkz/cassaforte [accessed on 15.03.2014]

[78] Node.JS, available online: http://nodejs.org [accessed on 15.03.2014]

[79] Higuera-Toledano, M.T., Wellings, A.J., "Distributed, Embedded and Real-Time Java Systems", 2012, Springer US, 378p

[80] Inico S1000 User Manual, available online: http://www.inicotech.com/doc/ [accessed on 15.03.2014]

[81] Bean, J., "SOA and Web Services Interface Design. Principles, Techniques, and Standards", 2009, Morgan Kaufmann, 384p

APPENDIX 1 – AVAILABLE NOSQL DATABASES

Name	Database Model Type
Redis	Key-value store
Memcached	Key-value store
Riak	Key-value store
DynamoDB	Key-value store
Ehcache	Key-value store
SinpleDB	Key-value store
Berkeley DB	Key-value store
Hazelcast	Key-value store
Coherence	Key-value store
Oracle NoSQL	Key-value store
Infinispan	Key-value store
ZODB	Key-value store
GT.M	Key-value store
Aerospike	Key-value store
LevelDB	Key-value store
FoundationDB	Key-value store
MongoDB	Document store
CouchDB	Document store
Couchbase	Document store
RavenDB	Document store
GemFire	Document store
Datameer	Document store
Mnesia	Document store
Cloudant	Document store
RethinkDB	Document store
Neo4j	Graph database
OrientDB	Graph database
Titan	Graph database
FlockDB	Graph database
Cassandra	Column oriented database
Hbase	Column oriented database
Accumulo	Column oriented database
Hypertable	Column oriented database

APPENDIX 2 – CASSANDRA DATA MODELS

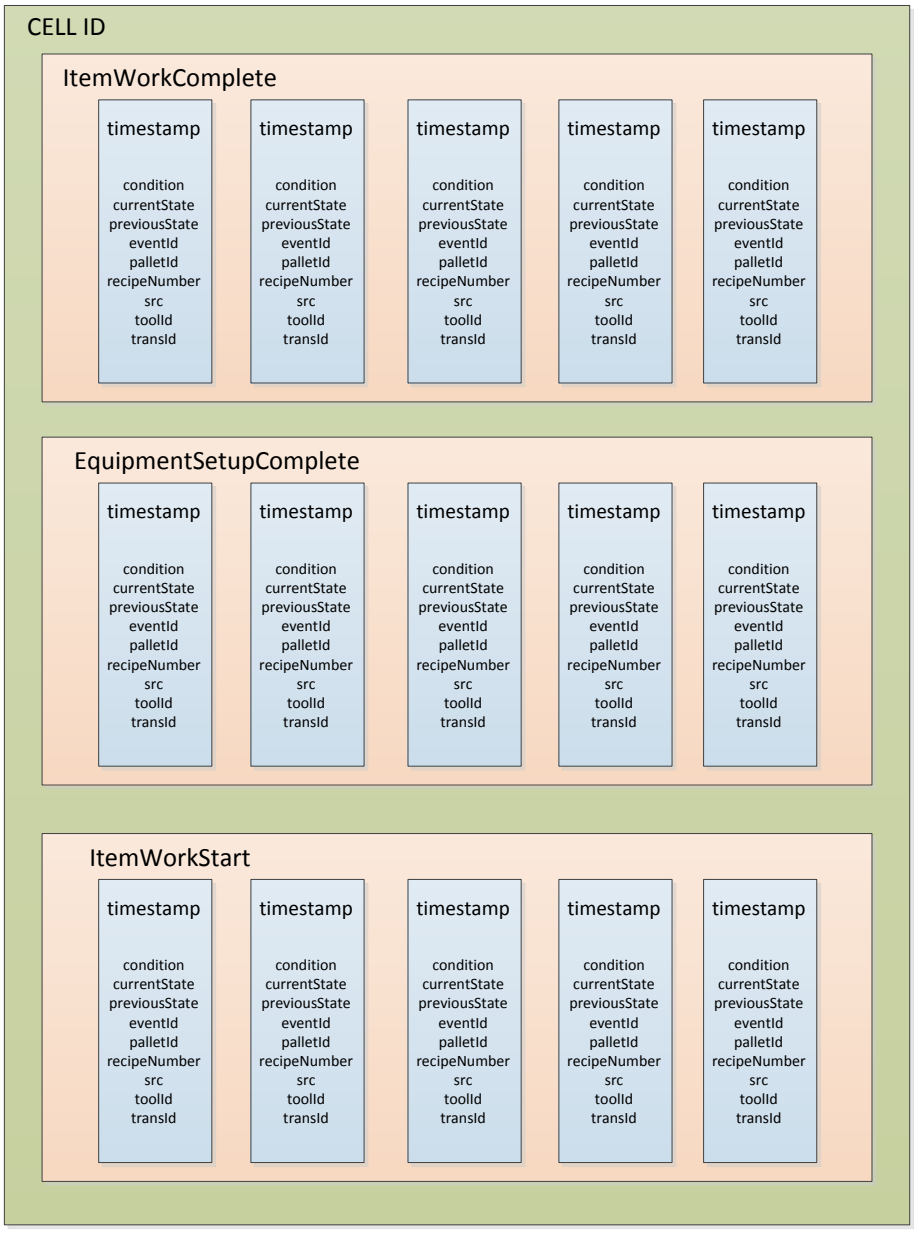

Figure 34: Data model of Robot Equipment Change State table

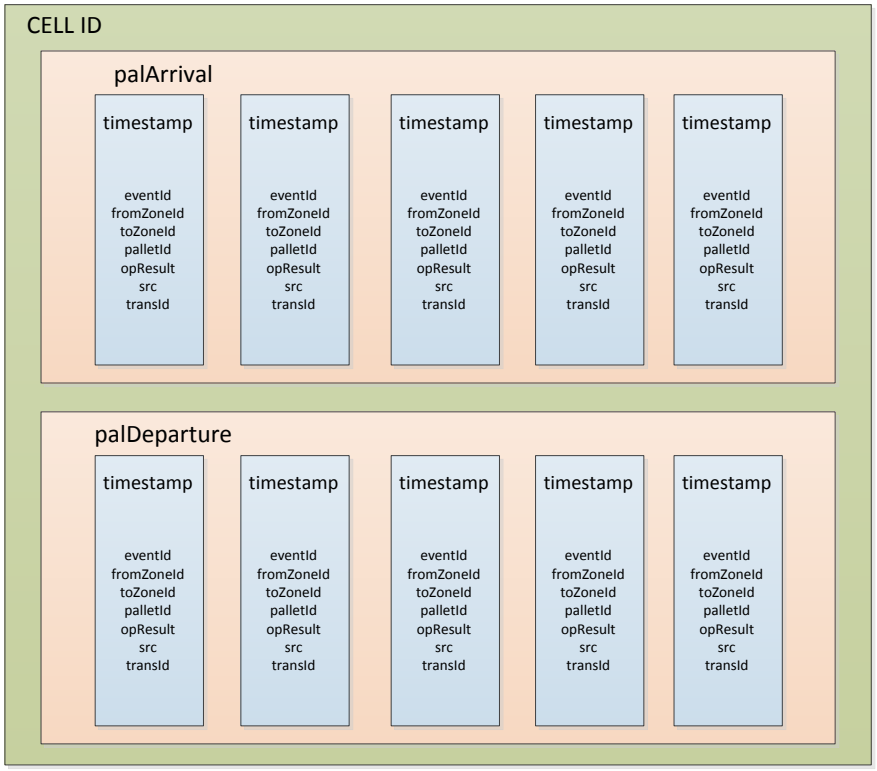

Figure 35: Data model of Conveyor Notification table

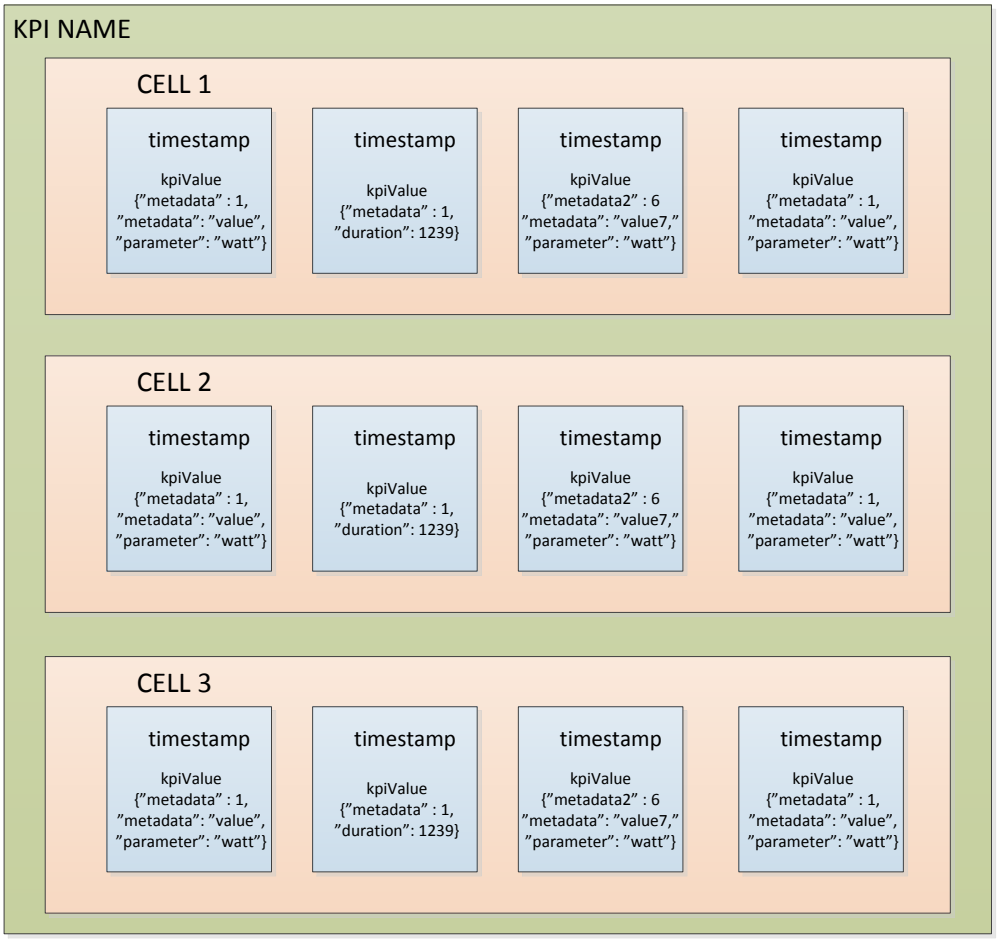

Figure 36: Data model of KPI table