A Big Data Platform for Smart Meter Data Analytics

Abstract

Smart grids have started generating an ever increasingly large volume of data. Extensive research has been done in meter data analytics for small data sets of electrical grid and electricity consumption. However limited research has investigated the methods, systems and tools to support data storage and data analytics for big data generated by smart grids.

This work has proposed a new core-broker-client system architecture for big data analytics. Its implemented platform is named Smart Meter Analytics Scaled by Hadoop (SMASH). Our work has demonstrated that SMASH is able to perform data storage, query, analysis and visualization tasks on large data sets at 20 TB scale. The performance of SMASH in storing and querying large quantities of data are compared with the published results provided by Google Cloud, IBM, MongoDB, and AMPLab. The experimental results suggest that SMASH provides industry a competitive and easily operable platform to manage big energy data and visualize knowledge, with potential to support data-intensive decision making.

Keywords: Big data, Smart grid, Meter data analytics

1. Introduction

Smart grids operation and future energy management will be increasingly data-intensive. For example, smart metering generates greater time-resolution data. It is certain that the increased granularity and consequently increased quantity will enable deeper analysis, and thus better understanding to energy consumption and better demand side management [1]. The capability to handling big data will be a prerequisite of mining smart meter data for insightful knowledge [2].

Research literature in big data for smart grids has mainly focused on requirements, concepts, design, issues, challenges and future directions [3] [4], whereas lack of published work on how solutions are purposely developed. Our work will provide a demonstrable working solution to fill this gap. And we will address the challenges of hardware, and software constraints for big data analytics [5]. The experimental studies will demonstrate how such problems have been addressed with comparison with the related work.

This paper present a solution to the "big data" problems in the domain of meter data management, including the design of a new system architecture, the rationale of choices of technologies, and its implementation. This work will also examine the performance of the proposed big data architecture, tested on data for a real-world challenge: at the similar scale when the UK smart metering is fully roll-out.

1.1. Smart grids and meter data

Smart grids consist of two interlinking and integrated sub-systems [6]: (a) the information infrastructure, where data flows in the cyber part of smart grids [7] [8] [9]. For example, measurement data and control signals are communicated by computer networks; and (b) the power infrastructure which delivers electricity in the physical part of smart grids consisting of smart meters and such power devices as generators, towers and transformers. The IT component of the information infrastructure includes modeling, analysis, commercial transactions, information sharing [10] and management [11]. This work focuses on meter data management and analytics at large scales, which are core parts of the IT component of smart grids.

One of the major challenges in smart grid is big data management and analytics [3] [12]. Smart metering is creating an explosion in the scale of data available. For example, in the UK, the 27 million domestic electricity consumers currently just have over 100 million data points per year collected quarterly or half-yearly for energy suppliers to record, store, and use in billing and other business operation. When smart metering is fully deployed and operated at 30-minute sampling rate, energy suppliers will need to ingest, store and process at least around 4500 to 9000 times more of the current data size, reaching 50 terabytes (the unit symbol for terabyte is TB. 1 TB $= 10^{12}$ bytes $=$ 1000 gigabytes) or 500 billion data-points annually.

In future, the ability to deal with this big data challenge will significantly support several important applications in smart grid, such as situation awareness, state estimation, event detection, load forecasting, demand response management [12] [2].

1.2. Meter Data Analytics

Data mining methods have been applied to analyse meter data for various application purposes.This will discover knowledge and gain insights from big data to support energy management [13]. However, most work has been demonstrated by using relatively small data sets, for example: demand or load forecasting [14] [15] [16] [17], customer segmentation [18] [19], pattern classification [20] [21], energy tariff recommendations [22], energy use of appliances in individual home[23] [24], and demand side management [10][25][26]. One of the largest data sets reported recently is over 1 million data points [27], which is still not yet at a comparable scale to what is expected in the near future.

2. Related work in big data

In order to take the best advantage of smart metering and reduce the risk of information overload, more powerful and automated data filtering, processing and analysis are needed. For the information infrastructure in smart grids, a variety of new scalable and distributed architectures or frameworks have been proposed to tackle communication problems [28] [6] [29] [30] and tackle data processing problems [31].

2.1. Big data architectures

Recent years have seen an increasingly large number of business cases in big data systems, utilizing various technologies, products, services. The typical Big data architectures can be classified briefly as empirically-grounded designs, service-orientated designs

[32]. The high level design of big data architectures always share the following components in term of functionality: data sources and storage, data extraction, data loading, data processing, data analysis, interface and visualization. However for different applications, a rich variety of architectures have been purposely built [32]. For example, Facebook specifies ad hoc analysis; Linkedin accommodates end user services on the basis on data analysis [33]; Twitter processes in real time; Netflix processes video recommendations on-line, off-line, and near-line; and Computer network traffic measurement [34] are Hadoop/MapReduce based and from the user interface, end user makes requests

For smart grids, random matrix theory has been integrated to a big data architecture design [35]. The random matrix theory was used for analysing multivariate data with moving split window. Their big data architecture focuses on decentralized control systems. The case studies were carried on relatively small amount of data: for example 500 sample points and 59 000 data. Limited by practical constraints, they only carried out the analytics at individual sites but not on a larger scale. The data we work on is at 20 TB scale.

An energy saving system for individual building [24] was designed to have a layered architecture. It includes a metering infrastructure in the building, a data collection service, a data analysis bench and a web-based portal. Energy uses from appliances in the building were recorded. Whereas our proposed work aims for the national scale of data, not individual building. And each data point in our data sets is the sum of appliances in a household, which how energy bills are calculated.

2.2. Cloud computing

For real-time or short term management and decision making, other approaches as below would be more appropriate. Cloud computing has been presented as one of ideal solutions for both communication and data processing problems [29] [3] . Although the cloud offers a cost-effective way to support big data analytics [36] [37] [38] [39]. In future, a cloud solution might be desirable in scenarios where real-time responses are required from a given smart meter stream data, such as for emergency situations [28] and detecting power grid failure [40]. Apache Spark is one of computing platforms which have been studied for smart grids [41]. It is suggested that it can be used for real time sensor data processing and on-line data stream analysis, although it was tested on a small set-up: 16GB RAM and 2 TB hard disk [41]. Another solution, in-memory multi-core processing has been suggested to outperform batch processing systems, such as Hadoop, for real-time calculation of prices and tariffs [31].

2.3. Commercial products

Commercial software products, for example IBM's Informix TimeSeries (TM), aim to manage big data for smart grids and smart meter data. However Informix TimeSeries's internal structure and the description of data sets are not reported [42]. Our design mainly chooses openly available software. This will advise industrial users how to build and modify their data management system flexibly to meet their various requirements, such as hardware resources, and technical constraints.

2.4. Requirements

Our aim is to analyse a large number of smart meter data at a national scale and over a relatively long time period. The cloud or Apache Spark targeting real-time analysis are not chosen as a data analysis platform here. This is because: (1) our research focus is to manage and plan at the national scale for long term, rather than real-time data processing. Therefore the ability to calling on every new data instance is unnecessary and not economic, whereas the ability to perform periodic batch processes is desirable; (2) to achieve real-time analysis, data has to be collected and transmitted in real time as a re-requisite. But at present, smart metering is rolling out, and the communication infrastructure of smart grid is under development. Therefore to collect data in a large scale and at real time is unavailable yet. In research literature, most of data sets used had been collected before analysis. Therefore direct communication with smart meters is not a concern; (3) data sets available for this work contain confidential and sensitive data such as household incomes and their property values, thus the data owners have granted no permission to store data in the public cloud. Moreover only the analytic outcomes and summarized data can be accessed by authorized users. This would be the practical situations for many real-world problems as well. So an off-line platform for big data analytics must be established. In summary off-line and centralized computation will meet the requirements.

3. System architecture of SMASH

To address the challenges of facilitating data management and data mining in a vast volume of smart meter data, this work proposes a new core-broker-client system architecture. Its implementation is a big data platform named SMASH, shortened for "Smart Meter Analytics Scaled by Hadoop". Fig. 1 illustrates its major components, namely data storage, big data platform including hardware, middle-ware and data mining applications, and web-based graphic user interface. The round circle indicates the scope of this study. Fig. 2 designs the technology stack for the SMASH system. The forthcoming sections will explain it in details.

3.1. SMASH System Modules Overview

The three principal SMASH components (modules) specific for the smart meter data: the Core, the Broker and the Web Client. The Core and the Broker sit on the same software "stack" as shown in Fig.2. And Fig.3 shows event flow over the major SMASH components.

3.1.1. SMASH Core

This module imports functionality from the Pig (to be explained in Section 4) and configuration modules to implement the Core API. API is short for application programming interface. It contains a client for Hadoop and HBase. It also includes many different implementations of the interfaces provided by the Core API. This module coalesces into a library and service bundle to provide features supporting batch-oriented analysis of smart meter data stored in HDFS (as large CSV files) and HBase (in several tables).

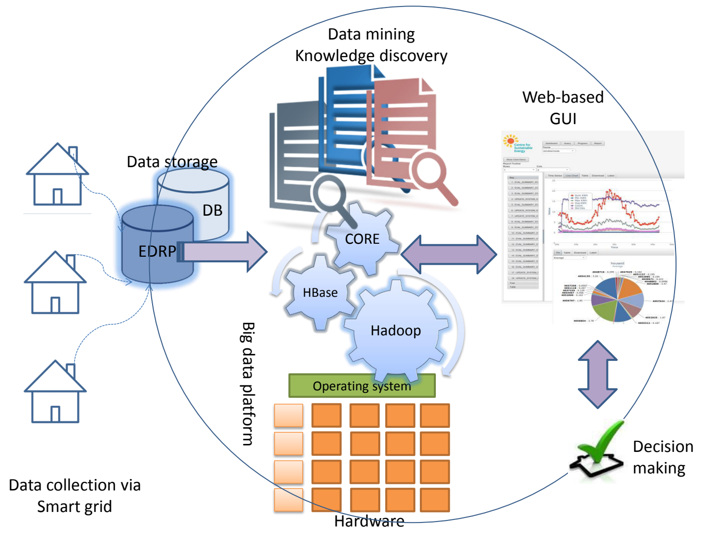


Figure 1: The system architecture of the SMASH. The round circle indicates its scope. The arrows indicate the direction of information flow. EDRP is the Energy Demand Research Project. The data set was from this project. CORE, Hadoop and HBase are the components in the SMASH, to be explained later.

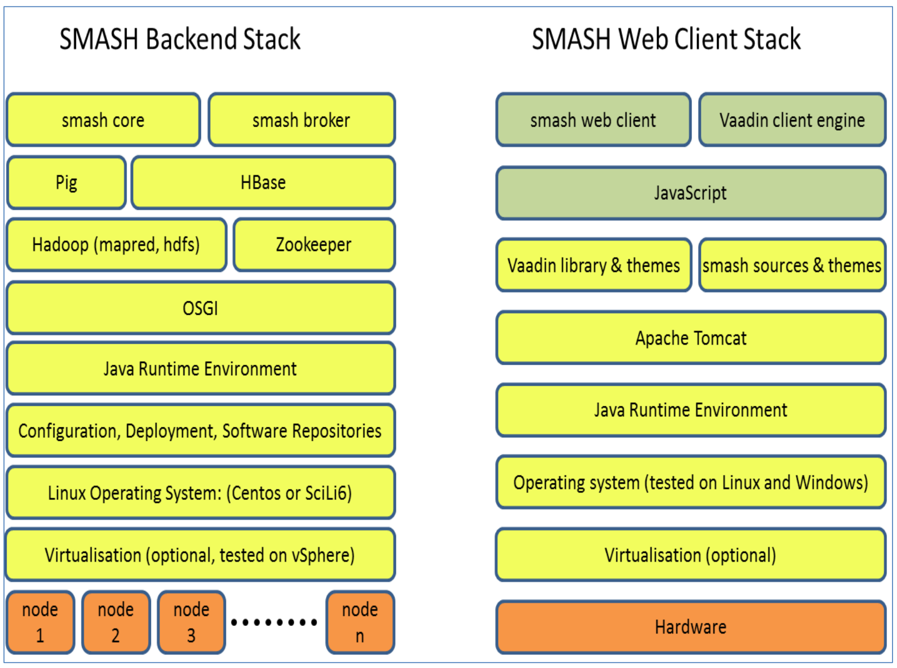


Figure 2: Technology stack for the SMASH system. Note that the backend stack is not the same across all nodes in the cluster. This diagram shows the superset of technologies required for the SMASH system. In particular, OSGi is only required on the nodes that run the SMASH Core and Broker bundles.

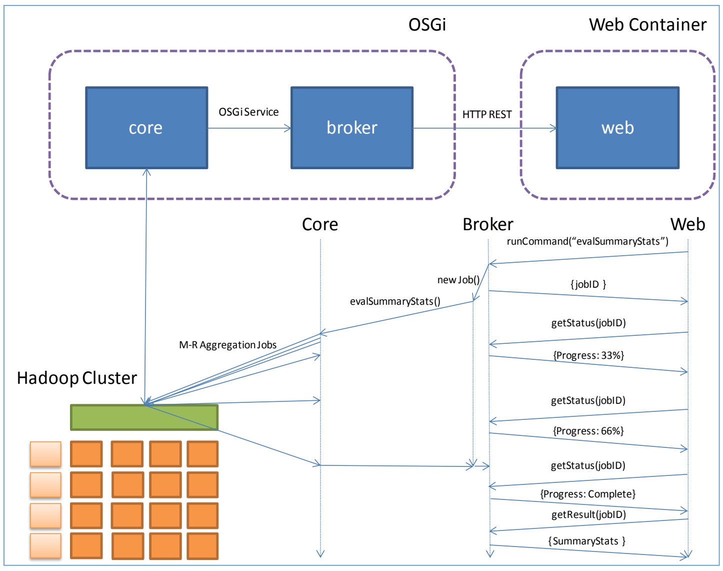


Figure 3: Event flow among SMASH's major components

3.1.2. SMASH Broker

This module provides the functionality for exposing Core services over a REST interface. Firstly it creates, queues, executes and manages jobs to implement Core services. It also queries the state of the system, including meta-data information about the Broker itself. Secondly, the Broker services asynchronously create and execute jobs which employ those services exported by the Core component, via an asynchronous API.

3.1.3. SMASH data mining

This module contains the source code used for data mining. A variety of data mining software tools is installed on SMASH on the top of stack 'SMASH Core', including feature selection, subgroup discovery [43] [44], classification [45], regression [45] and clustering [46]. For example, data mining experiments have found that when the number of bedrooms of the residual building is 4, a linear regression model: $PropertyValue = 0.45 \times HouseholdIncome + 10.70$ can be generated [44]. Moreover, on the basis of timestamp, new features such as maximum usage in the morning can be created [46], for further data analysis .

It is integrated with the other Core modules to exploit the distributed computing environment and provide data mining tools for use in data analysis. Data mining software tools, such as Weka (Version 3.6.4) [45] and Cortana (Version 2.0) [43] are integrated with the core modules for a range of data mining tasks.

3.1.4. SMASH Web client

SMASH presents users with a browser based interface powered by a dynamic web application, called the SMASH Web Client. The SMASH Web Client communicates via a Broker with the big data back-end platform, i.e. the SMASH Core, sitting on Hadoop and the cluster hardware. Its functions and the results will be reported in Section 6.4.

3.2. Key Features of SMASH's core-broker-client architecture

3.2.1. Initialisation

The SMASH system includes an initialisation feature which allows the software to be installed onto a fresh Hadoop (HBase-enabled) cluster with an initial set of trial data. The SMASHInitialisationService provides methods to install resources included in the SMASH Core module into HDFS. These methods attempt to install CSV files into HDFS before using them to create and populate HBase tables for storing smart meter data. The initial data set contains around 500,000 data points from a smart meter trial corresponding to 20 sample households. This enables the user to quickly execute a fresh installation of the SMASH system by supplying initial data to explore.

3.2.2. Ingress

The ingress feature of the SMASH system provides functionality for importing new data sets into HDFS and HBase. The methods for importing a new set of data into the backend are exploited by the initialisation process. "Helper utility" methods implementing the Hadoop and HBase clients support the fundamental process of storing smart meter data. An API (and endpoint) providing ingress features for importing batches of data into the SMASH system would require a trivial extension of the existing initialisation features to append additional data into existing files/tables and provide validation methods. As the instant access to a live data feed from smart grids is highly prohibited, we have been working with static data from smart electricity meter data sets which have been anonymized and with a consent from data owners.

3.2.3. Query Engine

The SMASH system provides a mechanism for the user to submit queries that compute aggregations on the data stored in the back-end. The user can specify query parameters by creating and populating an AggregationQuery object. This object is passed to the back-end as a parameter in a method call to an appropriate Core API service provided by the IAggregationService. There it is used to select and aggregate data, which is then returned in a result object. This approach for submitting queries is intended to support batch query processing in the back-end. By specifying the parameters in an object which is passed to the back-end, it delegates responsibility for interpreting and processing to the Core component. This flexible approach enables the Core component to exploit the query services available at run time to extract the relevant data. It also makes it easier to extend the services offered by the back-end for handling more sophisticated queries by simply adding parameter fields to the AggregationQuery object (a simple Java Bean acting as a container for a query parameters as fields) and enhancing the aggregation services in the Core to handle those additional parameters. The system also supports other types of queries such as requests for system state information and meta-data. Alternative query types, other than aggregation, could be supported using

Table 1: DICE

DICE Hadoop cluster capacity
11-node HDFS cluster with 227TB capacity
3-node HBase cluster (running on HDFS nodes)
3-node Zookeeper cluster (running on HDFS nodes)
2-ingress nodes for external access
2-nodes for hosting virtual machines (including the configuration management server)
3-nodes running a BigCouch cluster

Node's specification	
Worker nodes (x11)	Namenode or Jobtracker (x2)
CPU: 2 x Intel Xeon E5645, 6 cores, 2.4GHz, 12MB cache	CPU: same as the left
RAM: 24GB, 1333MHz	RAM: 64GB, 1333MHZ
Disk: 12 x 2TB SATA, 7.2k	Disk: 3x 1TB Near-line SAS, 7.2k

the same model of query submission with different parameter containers. For instance, a complete Extract Transform Load (ETL) service could be developed with this approach.

4. Implementation of SMASH

4.1. Hardware and operating system

As seen in Fig. 2, the SMASH is built on computing hardware. The Data Intensive Cluster Experiment (DICE) is a computing cluster provided by the University of Anonymization for analysis and storage of large volumes of data.

It includes a Hadoop cluster consisting of 11-node Hadoop Distributed File System (HDFS) cluster with 227 TB capacity, 3-node HBase cluster running on HDFS nodes, and 3-node Zookeeper cluster running on HDFS nodes. Hadoop, HBase, HDFS and Zookeeper are explained in the coming subsection. Along with the nodes outlined above, the DICE cluster also includes 2 ingress nodes for providing external access, 2 nodes for hosting virtual machines and 3 nodes running a BigCouch cluster. Each node in the DICE Hadoop cluster is running the Scientific Linux 6 operating system. Table 1 lists the capacity of DICE and the specification of its nodes.

4.2. Software and Rationale of choices

Table 2 summarizes the main software components of the SMASH. The choices of appropriate big data techniques are evaluated and justified as following.

Firstly, the smart electricity meter data sets available to this study are data-at-rest, i.e. not in stream that is called data-in-motion. Data-at-rest needs to be stored before processing and analysis. Three popular storage options are relational database management system (RDBMS), Hadoop, and hybrid RDBMS with Hadoop and MapReduce integration [48]. Hadoop is an open software stack, capable to support data intensive distributed applications. It enables batch analytic applications over a vast volume of data. It is well suited to exploratory analysis. It is scalable to petabyte (the unit symbol for petabyte is PB. 1 PB = 10^{15} bytes) on purpose-built appliances or on cloud. To deal with practical variations, this work has chosen to store the data in a variety of forms as part of the experiments. The largest data sets (approx. 20 TB) were stored in simple CSV files, which in turn are stored in Hadoop's distributed file system. HBase (a columnar database similar to BigTable) is also used to store large data sets.

Table 2: Software for SMASH

	Main functions	references
Hadoop	a well established scalable solution enabling the distributed processing of large data sets across clusters of computers	http://hadoop.apache.org
HDFS	a robust file system for storing large files across multiple hosts hardware in a cluster composed of commodity	http://hadoop.apache.org
MapReduce	Map method filters and sorts data and Reduce method summarizes data	[47]
Java	computer programming language	https://java.com/en/
HBase	distributed, non-relational database, with a column-oriented datastore for large data sets	https://hbase.apache.org
Zookeeper	a centralized service for storing configuration information, naming and synchronization services	http://zookeeper.apache.org
Pig	a high-level data flow language that compiles to query plans executed as MapReduce programs	https://pig.apache.org
OSGi	manage dependencies and share the functionality provided across SMASH components in run-time	www.osgi.org
BCFG2	manage the configuration of each machine in the cluster	http://bcfg2.org
Vaadin Invient Charts	open source framework for building dynamic web applications add charts and diagrams to visualize the data	https://vaadin.com/home

In addition, there are six popular options for analyzing big data [48]. The two options, namely "Custom Hadoop MapReduce batch analytic applications using 'in-Hadoop' custom or Mahout analytics" and "MapReduce based Business Intelligence tools and applications that generate MapReduce applications" have been adopted in this research. Due to MapReduce's merits in scalability, fault-tolerance, ease of programming, and flexibility [49], it is ideal for batch processing of very large amounts of data, although it may not be the best choice for interactive or event-based online big data processing [50] [49]. The smart meter data sets collected for this research mainly contain structured data, and has no interactive data nor event-based data. Hence MapReduce is the ideal choice here.

This research tested the options above, and chose to write custom Hadoop MapReduce applications to pull out statistics by processing the complete data set. Hadoop has enabled us to store and query large data sets on a mature scalable system designed to run on clusters of commodity hardware. Then our own bespoke core-broker-client system architecture has developed for creating MapReduce applications and launching them on demand to obtain required statistics.

4.3. Systems Integration

Fig. 3 shows the flow of events occurring in the SMASH system. Here the following processes occur:

- The user clicks on a button to submit a query (or some other operation) from the client-side of the web application.

- A web request is sent to the web application server which executes a callback method to handle that request. This callback method will in turn make a request to the smash-broker web service.

- The Broker receives the request and executes the appropriate method in response. In this case, it creates a new job for the given parameters the Web Client has provided in an ICommand object.
- The Broker queues the job for execution and returns a job identifier value in a response to the web application.
- The Broker uses a managed thread pool to execute queued jobs. When a job is run, it will typically call a method from the Core with some parameters and wait for it to complete before returning a result. One of the parameters the Broker provides to the Core method is an IProgressMonitor object which the Core can use to update the status of the job as it is processed.
- After a job has been submitted to the Broker the web Client can use the job ID returned in the response from the Broker to make subsequent requests to get the status of the job and the result (when the job has completed).

This asynchronous execution model allows multiple jobs to be submitted and handled by the system simultaneously. The Broker provides the functionality to support this execution model whilst the Core can simply provide a thread-safe library of services to provide the desired features of the system.

5. Data sets

The SSE Energy Supply Ltd, UK has installed smart electricity meters in sampled residential(households) and collected meter readings at a 30-minute interval. Data had been collected by the Energy Demand Research Project (EDRP) [1], by the Office of Gas and Electricity Markets (Ofgem). Ofgem is a non-ministerial government department and an independent National Regulatory Authority. The meter reading and installation for collecting data are detailed at https://www.ofgem.gov.uk/gas/retail-market/metering/meter-reading-and-installation.

A record of meter reading mainly consists three parts: (1) the register number which links to the household; (2) the time stamp when the reading was taken; and (3) the electricity consumed in unit kilowatt hour (kWh) during the half hour starting from the last reading time and ending at its respective time stamp. Table 3 provides an example of four meter readings in a sequence for the same meter. Through "registerID", a meter associates to a "houseID". One household may have more than one meters, as shown in Table 4, for example. Note here the values of houseID, registerID, and meterID are anonymized.

In addition to the readings from smart meters, the SSE data sets also contain the data from other sources than smart meters. SSE data sets also have data on Grid Supply Point Group, Gas connection, and socio-demographic variables, such as Number of Bedrooms, of nearly every household in trials. These variables are explained in Table 5.

In total, the SSE data set contains approximately 100 million data points. This data set was used for small-scale machine learning experiments. In order to evaluate

[1]https://www.ofgem.gov.uk/gas/retail-market/metering/transition-smart-meters/energy-demand-research-project

Table 3: Records of meter readings

advanceID	registerID	advanceDatetime	advanceKWh	advanceImportDatetime
64667116	1252550	04/02/2017 17:00	0.0540	01:58.8
64667117	1252550	04/02/2017 17:30	0.3870	01:58.8
64667118	1252550	04/02/2017 18:00	0.2700	01:58.8
64667119	1252550	04/02/2017 18:30	1.0150	01:58.8

Table 4: Meter information

meterID	houseID	meter Fuel	meter Type	meterInstall Datetime	meterImport Datetime	meterSupplier Timestamp
123456	4037237	E	S	13/09/2007 00:00	20:33.5	13/10/2010 09:54
409876	4005834	E	D	20/10/1994 00:00	20:33.5	13/10/2010 09:54
400583	4005834	G	D	28/03/2001 00:00	20:33.5	13/10/2010 09:54

Table 5: Description of socio-demographic variables [44]. Note *: total 14 but only three GSP groups found in the SSE smart meter dataset.

Socio-demographic variables	**Description**	**Nr. of categories**	**Example(s)**
GSP group	Grid Supply Point Group in UK, which are regional electricity distribution networks	14*	Southern; South Wales; North Scotland
Age	Age of head of household	6	Age 26-35
Decision Maker Type	Type of person deciding household matters	13	Young Couple
Family Lifestage	The combined stage of life and family status including children	14	Young family with children
Household Composition	People living together and their relationships to one another	13	Male homesharers
Household Income Band	Total household income per year	10	£30,000 to £39,999
Mains gas flag	Whether a household is connected to the Main gas network; if Yes, it's assumed that the household uses gas	2	connected to gas; not connected to gas
Mosaic Public Sector Group	Classification on citizen's location, demographics, lifestyles and behaviors	15	Young, well-educated city dwellers; Wealthy people living in the most sought after neighborhoods
Mosaic Public Sector Type	Subcategories of Mosaic Public Sector Group	69	Young professional families settling in better quality older terraces
Number of Bedrooms	Number of Bedrooms of the property	5	5 + bedrooms
Property Age	When the property was built	6	1871-1919
Property Type 2011	Type of property in 2011	5	Purpose built flats; Farm
Property Value Fine	Estimated property value	25	£500,001 to £600,000
Tenure 2011	Property ownership in 2011	3	Privately rented

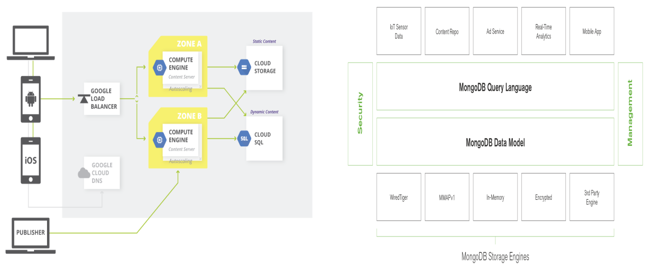


Figure 4: (left) GoogleCloud ContentMgmt ArchDiagram (right) MongoDB Storage Architecture [51]

the performance of SMASH over a suitably large dataset, the SSE smart meter data has been replicated with added variations to simulate a data set size similar to what would be produced by all UK domestic smart meters to be fully deployed. This larger data set is used for large-scale experiments to be reported in Section 6. This is estimated that all 27 million households in the UK will generate at a rate of approximately 13,000 records per second.

6. Performance of SMASH

6.1. Comparison of architectural patterns

The textbook software architectural patterns include layered pattern, broker pattern, client-server pattern. SMASH can be classified as in Broker pattern. SMASH is designed with components which interact with each other by remote service invocations. A broker component coordinates of communication among components. Servers publish their capabilities to the broker. Clients, in our case, through SMASH's web container, request a service from the broker, and the broker then re-directs these requests to a suitable service from its registry, as seen in Fig 3.

Fig 4(left) is GoogleCloud's content management architecture. It is a client-server pattern. The servers provide services to many clients. Clients send requests of services to the servers. In this case, a load balancer continuously listen to client requests and coordinate the distribution of requests to servers. Fig. 4(right) is MongoDB's flexible storage architecture [51]. It is a typical layered pattern. A layer provides services to the next higher layer. Every architectural patterns has the pros and cons. The choices of architectural patterns are often driven by needs and practical considerations.

The following subsections, the performance of SMASH will be compared with several products in the market or in literature.

6.2. Storing large quantities of data

The ability of data storage is typical measured by the speed of inserting records. Table 6 lists the performance results and comparison on this. SMASH has sustained a speed of inserting records (also called Average throughput rate) of 70,000 records per second without coprocessors and 200,000 records per second with coprocessors. These were tested whilst inserting records (rows) into HBase. In order to test the ingress capabilities of SMASH, we ran the system on the DICE cluster computing platform and

Table 6: Speed of inserting records. Unit : records/second

SMASH	Hbase	70,000	without coprocessors
		200,000	with coprocessors
SMASH - ingress	HDFS	10,000,000	
IBM's Informix TimeSeries		420,962	
Google Cloud		100,000	
MongoDB		80,000	
Expected demand for UK meter full roll out		13,000	

Table 7: Speed of querying records. Unit: records/second

SMASH			3,000,000
AMPLab	Redshift (HDD)	Query 1A	32,888
		Query 1B	3,331,851

measured it storing records to the distributed HDFS file system. In these experiments, SMASH has sustained more than 10 million rows per second, whilst being tested with up to 10 billion rows data, approximately 18.2 TB.

In comparison, Google Cloud Platform enables data inserting at a speed of 100,000 rows per second, per table [52]. While a user at Red Hat reported that MongoDB has reached a speed of 80,000 inserts per second on commodity hardware [53]. As an meter data management benchmark, the IBM's commercial product, Informix TimeSeries has reported their speed of data load is 420,962 records/sec, over a testing data load period: 3 hrs. 14 min [42].

6.3. Querying large quantities of data

The ability of processing data is typical measured by the speed of querying records. Table 7 lists the performance results and comparison on this. SMASH has been demonstrated processing approximately 3,000,000 records per second to analyse a large data set to produce a query result in less than 10 minutes whilst running on the DICE platform. In comparison on the speed of querying data, the Big Data Benchmarks [54] which are published by AMPLab, University of California, Berkeley, vary greatly. For the Scan Query which reads and writes table data, the top performed, Redshift (HDD) achieves 32,888 results within 2.49 seconds for Query 1A, and 3,331,851 results within 2.61 seconds for Query 1B.

These performance evaluations indicate that SMASH could feasibly store, query and analyze big smart meter data sets. This work also provides evidences on the requirements of hardware and big data software to facilitate big data analytics after the UK smart meter roll out is fully completed and operational.

6.4. GUI and data visualization

Web-based graphic user interface (GUI) is built to provide an interface to interact with users, typically, industrial users to support decision making. From industrial users' point of view, an ideal analytic system should enable data analysis, data manipulation and result visualization.

Fig. 5 is the web page sitemap. Fig. 6 includes four screen-shots representing four main stages where users make a query and receive results using the Web-based GUI of SMASH.

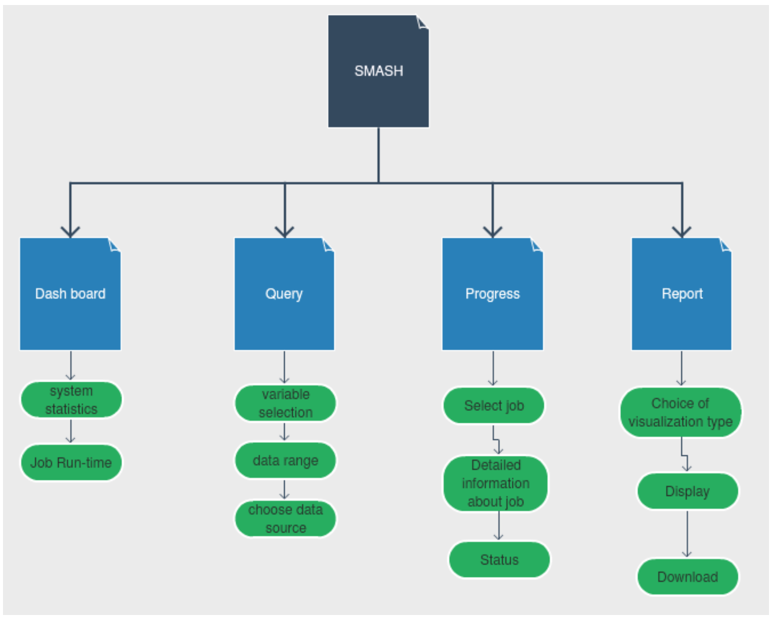


Figure 5: Web Page Sitemap

Fig. 6 (top left) The Dashboard View lists the system configuration and high-level system statistics.

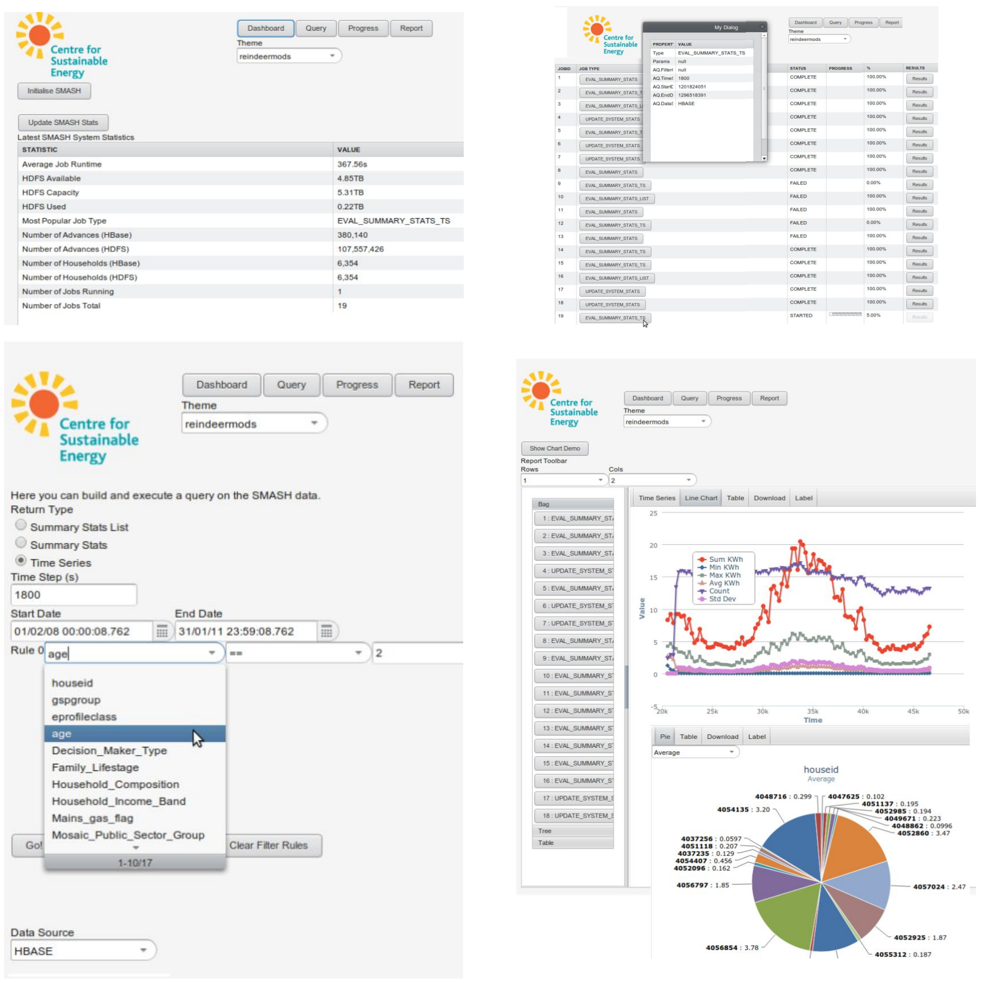
Fig. 6 (bottom left) The Query View allows users to choose input data and configure parameters such as the type of query and household sampling criteria. A query specifies the parameters for a job that will evaluate statistics of the data sets. For example, using the rules to filter the households by some sociodemographic criteria, a query could extract statistics for particular time interval which could be considered as candidate models of household consumption. Each query created in Query View is submitted as a job, and its progress would appear in Progress View.

Fig. 6 (top right) The Progress View shows a selected list of models for which we have a practical interest, that were built following initial results from data mining on the training data set. SMASH is also able to accommodate data mining to discover rule-based models, distance-based models and tree-based models. In practice, users decide the job types, output models and configurations to run on the full data sets. The execution status, progress and tabs linking to results are displayed.

Fig. 6 (bottom right) The Report View offers a range of options for visualizing results to provide insight and highlight various summary statistics.

7. Conclusions

Smart metering has started generating an ever increasingly large volume of data. Going from quarterly manual data collection to 30 minute automatically readings represents a radical re-instrumentation of energy infrastructure.



Latest SMASH System Statistics

STATISTIC	VALUE
Average Job Runtime	367.56s
HDFS Available	4.85TB
HDFS Capacity	5.31TB
HDFS Used	0.22TB
Most Popular Job Type	EVAL_SUMMARY_STATS_TS
Number of Advances (HBase)	380,140
Number of Advances (HDFS)	107,557,426
Number of Households (HBase)	6,354
Number of Households (HDFS)	6,354
Number of Jobs Running	1
Number of Jobs Total	19

My Dialog

PROPERT	VALUE
Type	EVAL_SUMMARY_STATS_TS
Params	null
AQ.Filter	null
AQ.Time	1800
AQ.Start	1201824051
AQ.EndD	1296518391
AQ.Data	HBASE

JOBID	JOB TYPE	STATUS	PROGRESS	%	RESULTS
1	EVAL_SUMMARY_STATS	COMPLETE		100.00%	Results
2	EVAL_SUMMARY_STATS_T	COMPLETE		100.00%	Results
3	EVAL_SUMMARY_STATS_L	COMPLETE		100.00%	Results
4	UPDATE_SYSTEM_STATS	COMPLETE		100.00%	Results
5	EVAL_SUMMARY_STATS_T	COMPLETE		100.00%	Results
6	UPDATE_SYSTEM_STATS	COMPLETE		100.00%	Results
7	UPDATE_SYSTEM_STATS	COMPLETE		100.00%	Results
8	EVAL_SUMMARY_STATS	COMPLETE		100.00%	Results
9	EVAL_SUMMARY_STATS_TS	FAILED		0.00%	Results
10	EVAL_SUMMARY_STATS_LIST	FAILED		100.00%	Results
11	EVAL_SUMMARY_STATS	FAILED		100.00%	Results
12	EVAL_SUMMARY_STATS_TS	FAILED		0.00%	Results
13	EVAL_SUMMARY_STATS	FAILED		100.00%	Results
14	EVAL_SUMMARY_STATS_TS	COMPLETE		100.00%	Results
15	EVAL_SUMMARY_STATS_TS	COMPLETE		100.00%	Results
16	EVAL_SUMMARY_STATS_LIST	COMPLETE		100.00%	Results
17	UPDATE_SYSTEM_STATS	COMPLETE		100.00%	Results
18	UPDATE_SYSTEM_STATS	COMPLETE		100.00%	Results
19	EVAL_SUMMARY_STATS_TS	STARTED		5.00%	Results

Here you can build and execute a query on the SMASH data.

Return Type
( ) Summary Stats List
( ) Summary Stats
(•) Time Series

Time Step (s): 1800

Start Date: 01/02/08 00:00:08.762   End Date: 31/01/11 23:59:08.762

Rule 0: age   ==   2

Data Source: HBASE

Figure 6: Screen-shots of GUI of SMASH: (top left) Dash board View; (top right) Progress View; (bottom left) Query View; (bottom right) Report View.

This work addresses this challenge by presents a new core-broker-client system architecture for Smart Meter Data Analytics at a big data scale. It has demonstrated performing data storing, querying, analysis and visualization tasks on large data sets for smart meter, with competitive performance. In future, we plan to utilize this platform to facilitate data-driven management for smart grid.

References

[1] L. Olmos, S. Ruester, S.-J. Liong, J.-M. Glachant, Energy efficiency actions related to the rollout of smart meters for small consumers, application to the austrian system, Energy 36 (7) (2011) 4396 – 4409.

[2] D. Alahakoon, X. Yu, Smart electricity meter data intelligence for future energy systems: A survey, IEEE Transactions on Industrial Informatics 12 (1) (2016) 425–436.

[3] P. D. Diamantoulakis, V. M. Kapinas, G. K. Karagiannidis, Big data analytics for dynamic energy management in smart grids, Big Data Research 2 (3) (2015) 94 – 101.

[4] H. Daki, A. El Hannani, A. Aqqal, A. Haidine, A. Dahbi, Big data management in smart grid: concepts, requirements and implementation, Journal of Big Data 4 (1) (2017) 13.

[5] K. Kambatla, G. Kollias, V. Kumar, A. Grama, Trends in big data analytics, Journal of Parallel and Distributed Computing 74 (7) (2014) 2561 – 2573.

[6] Y.-J. Kim, M. Thottan, V. Kolesnikov, W. Lee, A secure decentralized data-centric information infrastructure for smart grid, Communications Magazine, IEEE 48 (11) (2010) 58–65.

[7] M. Jarrah, M. Jaradat, Y. Jararweh, M. Al-Ayyoub, A. Bousselham, A hierarchical optimization model for energy data flow in smart grid power systems, Information Systems 53 (2015) 190 – 200.

[8] M. Nabeel, X. Ding, S.-H. Seo, E. Bertino, Scalable end-to-end security for advanced metering infrastructures, Information Systems 53 (2015) 213 – 223.

[9] A. E. C. Mondragon, E. S. Coronado, C. E. C. Mondragon, Defining a convergence network platform framework for smart grid and intelligent transport systems, Energy 89 (2015) 402 – 409.

[10] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, G. Hancke, A survey on smart grid potential applications and communication requirements, Industrial Informatics, IEEE Transactions on 9 (1) (2013) 28–42.

[11] B. Alagoz, A. Kaygusuz, A. Karabiber, A user-mode distributed energy management architecture for smart grid applications, Energy 44 (1) (2012) 167 – 177.

[12] C. Tu, X. He, Z. Shuai, F. Jiang, Big data issues in smart grid a review, Renewable and Sustainable Energy Reviews 79 (2017) 1099 – 1107.

[13] K. Zhou, C. Fu, S. Yang, Big data driven smart energy management: From big data to big insights, Renewable and Sustainable Energy Reviews 56 (2016) 215 – 225.

[14] C. Borges, Y. Penya, I. Fernandez, Evaluating combined load forecasting in large power systems and smart grids, Industrial Informatics, IEEE Transactions on 9 (3) (2013) 1570–1577.

[15] Y. Zhang, G. Luo, Short term power load prediction with knowledge transfer, Information Systems 53 (2015) 161 – 169.

[16] N. Ding, Y. Besanger, F. Wurtz, G. Antoine, Individual nonparametric load estimation model for power distribution network planning, Industrial Informatics, IEEE Transactions on 9 (3) (2013) 1578–1587.

[17] M. S. Al-Musaylh, R. C. Deo, J. F. Adamowski, Y. Li, Short-term electricity demand forecasting with mars, svr and arima models using aggregated demand data in queensland, australia, Advanced Engineering Informatics 35 (2018) 1 – 16.

[18] W. Labeeuw, G. Deconinck, Residential electrical load model based on mixture model clustering and markov models, Industrial Informatics, IEEE Transactions on 9 (3) (2013) 1561–1569.

[19] A. M. Ferreira, C. A. Cavalcante, C. H. Fontes, J. E. Marambio, A new method for pattern recognition in load profiles to support decision-making in the management of the electric sector, International Journal of Electrical Power and Energy Systems 53 (0) (2013) 824 – 831.

[20] M. Biswal, P. Dash, Measurement and classification of simultaneous power signal patterns with an s-transform variant and fuzzy decision tree, Industrial Informatics, IEEE Transactions on 9 (4) (2013) 1819–1827.

[21] C. Beckel, L. Sadamori, S. Santini, Automatic socio-economic classification of households using electricity consumption data, in: Proceedings of the Fourth International Conference on Future Energy Systems, e-Energy 2013, 2013, pp. 75–86.

[22] K. Valogianni, W. Ketter, Effective demand response for smart grids: Evidence from a real-world pilot, Decision Support Systems 91 (2016) 48 – 66.

[23] L. M. Candanedo, V. Feldheim, D. Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy and Buildings 140 (2017) 81 – 97.

[24] J.-S. Chou, N.-T. Ngo, Smart grid data analytics framework for increasing energy savings in residential buildings, Automation in Construction 72 (2016) 247 – 257.

[25] D. Huang, H. Zareipour, W. Rosehart, N. Amjady, Data mining for electricity price classification and the application to demand-side management, Smart Grid, IEEE Transactions on 3 (2) (2012) 808–817.

[26] A. Taha, J. Panchal, Decision-making in energy systems with multiple technologies and uncertain preferences, Systems, Man, and Cybernetics: Systems, IEEE Transactions on 44 (7) (2014) 894–907.

[27] J. Moeyersoms, D. Martens, Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector, Decision Support Systems 72 (2015) 72 – 81.

[28] J. Zhou, R. Hu, Y. Qian, Scalable distributed communication architectures to support advanced metering infrastructure in smart grid, Parallel and Distributed Systems, IEEE Transactions on 23 (9) (2012) 1632–1642.

[29] X. Fang, S. Misra, G. Xue, D. Yang, Managing smart grid information in the cloud: opportunities, model, and applications, Network, IEEE 26 (4) (2012) 32–38.

[30] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, W. H. Chin, Smart grid communications: Overview of research challenges, solutions, and standardization activities, Communications Surveys Tutorials, IEEE 15 (1) (2013) 21–38.

[31] M. Arenas-Martinez, S. Herrero-Lopez, A. Sanchez, J. Williams, P. Roth, P. Hofmann, A. Zeier, A comparative study of data storage and processing architectures for the smart grid, in: Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on, 2010, pp. 285–290.

[32] P. Pkknen, D. Pakkala, Reference architecture and classification of technologies, products and services for big data systems, Big Data Research 2 (4) (2015) 166 – 186.

[33] R. Sumbaly, J. Kreps, S. Shah, The big data ecosystem at linkedin, in: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD '13, ACM, 2013, pp. 1125–1134.

[34] Y. Lee, Y. Lee, Toward scalable internet traffic measurement and analysis with hadoop, SIGCOMM Comput. Commun. Rev. 43 (1) (2012) 5–13.

[35] X. He, Q. Ai, R. C. Qiu, W. Huang, L. Piao, H. Liu, A big data architecture design for smart grids based on random matrix theory, IEEE Transactions on Smart Grid 8 (2) (2017) 674–686.

[36] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, S. U. Khan, The rise of big data on cloud computing: Review and open research issues, Information Systems 47 (0) (2015) 98 – 115.

[37] Intel, Big data in the cloud: Converging technologies, Intel IT Center.

[38] F. Xhafa, Advanced knowledge discovery techniques from big data and cloud computing, Enterprise Information Systems 10 (9) (2016) 945–946.

[39] A. A. Munshi, Y. A.-R. I. Mohamed, Big data framework for analytics in smart grids, Electric Power Systems Research 151 (2017) 369 – 380.

[40] H. Yang, P. Li, Z. He, X. Guo, S. Fong, H. Chen, A decision support system using combined-classifier for high-speed data stream in smart grid, Enterprise Information Systems 10 (9) (2016) 947–958.

[41] S. R., B. G. H.B., S. K. S., P. Poornachandran, S. K.P., Apache spark a big data analytics platform for smart grid, Procedia Technology 21 (2015) 171 – 178.

[42] IBM-Software, Managing big data for smart grids and smart meters, IBM Data magazine.

[43] M. Meeng, A. Knobbe, Flexible enrichment with cortana – software demo., in: Proceedings of BeneLearn, 2011, pp. 117–119.

[44] N. Jin, P. Flach, T. Wilcox, R. Sellman, J. Thumim, A. Knobbe, Subgroup discovery in smart electricity meter data, Industrial Informatics, IEEE Transactions on 10 (2) (2014) 1327–1336.

[45] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: An update, in: SIGKDD Explorations, Vol. 11, 2009.

[46] R. Al-Otaibi, N. Jin, T. Wilcox, P. Flach, Feature construction and calibration for clustering daily load curves from smart meter data, IEEE Transactions on Industrial Informatics 12 (2016) 1–10.

[47] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[48] M. Ferguson, Architecting a big data platform for analytics, IBM Data magazine.

[49] C. Doulkeridis, K. Nrvg, A survey of large-scale analytical query processing in mapreduce, The

VLDB Journal 23 (3) (2014) 355–380.

[50] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, A knowledge-based platform for big data analytics based on publish/subscribe services and stream processing, Knowledge-Based Systems.

[51] MongoDB, Mongodb architecture guide - mongodb 3.2, Tech. rep., MongoDB (2015).

[52] Google, Streaming data into bigquery, Tech. Rep. January 29, Google Cloud (2016).
URL `https://cloud.google.com/bigquery/streaming-data-into-bigquery`

[53] V. Mihalcea, Mongodb facts: 80000+ inserts/second on commodity hardware, Tech. rep., Red Hat (2013).
URL `http://vladmihalcea.com/2013/12/01/mongodb-facts-80000-insertssecond-on-commodity-hardware/`

[54] AMPLab, Big data benchmark, Tech. Rep. February, University of California, Berkeley (2014).
URL `https://amplab.cs.berkeley.edu/benchmark/`