González Ordiano *et al. J Big Data (2018) 5:11*
https://doi.org/10.1186/s40537-018-0119-6

Journal of Big Data

RESEARCH Open Access

Concept and benchmark results for Big Data energy forecasting based on Apache Spark

Jorge Ángel González Ordiano*†, Andreas Bartschat†, Nicole Ludwig†, Eric Braun†, Simon Waczowicz†, Nicolas Renkamp†, Nico Peter†, Clemens Düpmeier†, Ralf Mikut† and Veit Hagenmeyer†

*Correspondence: jorge.ordiano@kit.edu
†Jorge Ángel González Ordiano, Andreas Bartschat, Nicole Ludwig, Eric Braun, Simon Waczowicz, Nicolas Renkamp, Nico Peter, Clemens Düpmeier, Ralf Mikut and Veit Hagenmeyer contributed equally to this work
Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

Abstract

The present article describes a concept for the creation and application of energy forecasting models in a distributed environment. Additionally, a benchmark comparing the time required for the training and application of data-driven forecasting models on a single computer and a computing cluster is presented. This comparison is based on a simulated dataset and both *R* and *Apache Spark* are used. Furthermore, the obtained results show certain points in which the utilization of distributed computing based on *Spark* may be advantageous.

Keywords: Big Data, Forecasting, Energy, Data-driven, EnergyLab 2.0

Introduction

The transformation of the current energy grid into a Smart Grid [1] is an ongoing challenge in the pursuit of an environmentally-friendly energy supply. This transformation is exemplified—from a data perspective—by the European Union's decision to replace 80 percent of electricity meters with smart meters by the year 2020 [2] and by the desire to automate and monitor each of the power grid's voltage levels [3]. The increasing installation of information and communication technologies (ICT) comes hand in hand with an increment in the volume and variety of the collected data, i.e. Big Data. The difficulties present in the analysis and utilization of this Big Data—in the context of the Smart Grid—have caught the interest of the energy research community. Possible solutions have been proposed in the literature, e.g., the use of cloud computing [4, 5]. Nonetheless, the utilization of Big Data is not the only complication in the development of the future energy grid. The continuous integration of volatile renewable power systems (e.g., photovoltaic (PV) and wind power systems) poses an additional challenge, since power generation volatility complicates the required balancing of energy supply and demand [6]. However, data-driven forecasting models trained using the available Big Data may be a possible solution.

Unlocking the hidden potential in Big Data requires distributed algorithms and data storage systems. To this end, the present contribution offers a description of a Big Data forecasting concept that utilizes the distributed computing framework *Apache Spark*[1]

[1] spark.apache.org/.

for the creation and application of forecasting models. *Spark* possesses a number of Big Data processing methodologies that may be helpful when analyzing and using Smart Grid Big Data [7]. In addition, the presented Big Data forecasting concept can serve in the development of Big Data forecasting tools, for example, in the Helmholtz Association's Energy System 2050 (ES 2050) project.[2]

There is widespread belief that the utilization of Big Data can improve forecasting results if its underlying patterns can be analyzed [8]. However, the creation of data-driven forecasting models with Big Data proves to be challenging, since most data-driven approaches have not been designed to work on a distributed environment [8]. Therefore, the present contribution presents a benchmark to determine the possibility of training and applying data-driven forecasting models on Big Data. The goals of the benchmark are the assessment of the necessary time to obtain data-driven forecasting models when using Big Data and to determine the point at which a distributed computing framework based on, e.g., *Spark* becomes necessary. These goals are achieved by comparing the required time for training and applying different data-driven forecasting models on a computing cluster (using *Spark*) and on a single computer (using *R* and *Spark*). A similar study with a focus in the analysis of smart meter data using distributed computing can be found in [9]. It is important to mention, that the discussion and conclusion of the present work does not come from a *Spark*/Big Data expert point of view, but rather from a user's (e.g., energy researcher) perspective.

The present work is structured as follows: first general information on energy related forecasting is given. Thereafter, the Big Data forecasting concept and the conducted benchmark are presented. Afterwards, the obtained results are shown and discussed. Lastly, the conclusion and outlook are offered.

Energy forecasting

Time series forecasting models are useful at predicting values that are changing over time [10]. Hence, they are commonly used to forecast energy values, as e.g., electrical load and volatile renewable power generation. Energy forecasting models can generally be divided into white-box models, data-driven models (i.e. black-box models), and their combination (i.e. gray-box models) [11]. While white-box models conduct their forecasts through the utilization of known relations and expert knowledge (e.g., physical models for volatile renewable power generation [12, 13]), data-driven models try to infer—via data mining techniques—the relation between their input values and the future time series values.

Data-driven models (e.g., artificial neural networks, regression models, support vector regressions) have become quite popular in the energy forecasting community [11]. These models have the advantage of not requiring an explicit description of system specific properties (e.g., wind power curves, PV modules' tilt, power line losses, customer behaviour) to conduct their forecast, as this information is implicitly contained in the measured data. Examples of data-driven energy forecasting models found in literature are given in [14–16].

[2] helmholtz.de/en/research/energy/energy_system_2050/.

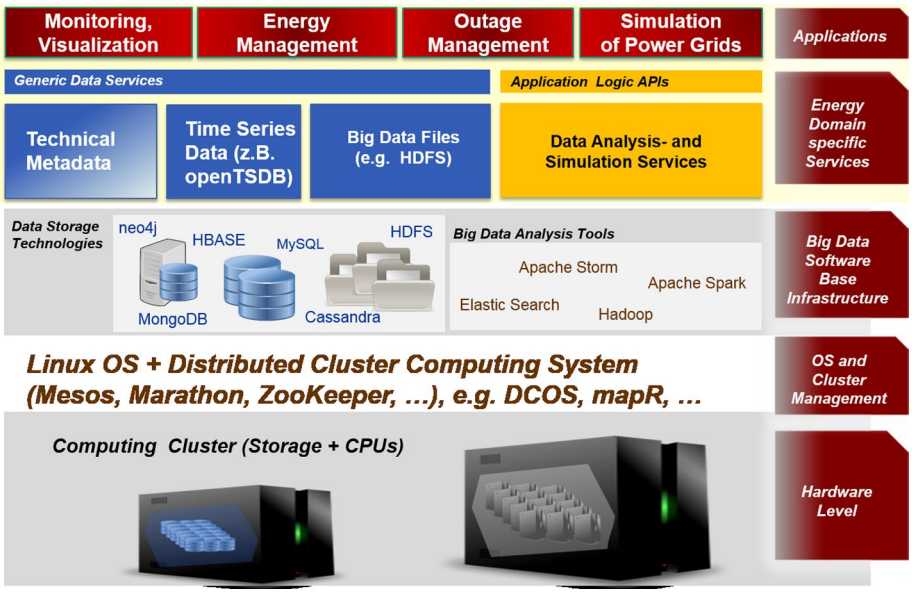

Fig. 1 Big Data energy forecasting infrastructure

Time series forecasting models aim to estimate the future values of a time series $\{y[k]; k = 1, \ldots, K\}$ at a specific forecast horizon H (e.g., 24 h) using all available information. For example, a forecasting model using current and past auto-regressive values as well as, current and past values of other exogenous time series (e.g., forecast weather data, calendar information) can be described by the functional relation:

$$\hat{y}[k+H] = f(y[k], \ldots, y[k-H_1], \mathbf{u}[k], \ldots, \mathbf{u}[k-H_1]; \boldsymbol{\theta}), \quad k > H_1; \tag{1}$$

where $\hat{y}[k+H]$ is the forecast value, H_1 describes the number of used time-lags, $\mathbf{u}[k]$ to $\mathbf{u}[k-H_1]$ are vectors containing the exogenous time series values, and $\boldsymbol{\theta}$ is a vector containing the parameters defining the model.

As already mentioned, forecasting models that are able to predict the future power generation and/or load are of major importance in assuring the power grid's stability. Therefore, both load and renewable power forecasting have been thoroughly discussed in literature. Several reviews outlining the state of the art of energy forecasting are presented in [17] (PV power forecasting), in [18] (load forecasting), and in [19] (wind power forecasting).

Big Data forecasting concept

Preliminary work regarding the Big Data concept described in the present section, including a first concept and benchmarks, can be found in [20]. The new widened concept for the Big Data forecasting infrastructure—i.e. an extended view of the infrastructure presented in [21] and [22]—is depicted in Fig. 1.

The foundation of the infrastructure shown in Fig. 1 is a series of Linux nodes with many CPUs and large data storage arrays. To create a computing cluster using the available Linux nodes, a distributed cluster computing system needs to be installed. Two systems that are currently being considered for the Big Data forecasting infrastructure are

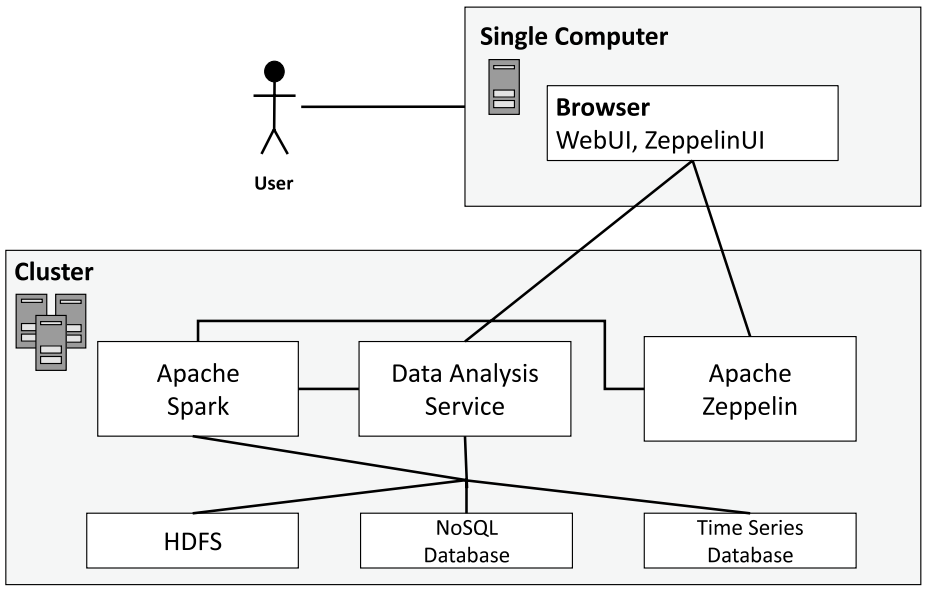

Fig. 2 Usage scenario of the Big Data forecasting concept

MapR[3] and *DC/OS*.[4] These systems allow the installation of a large range of Big Data components, as e.g., *Apache Hadoop*[5] [23–25] and *Spark* [26], and different databases for the storage of time series data.

As shown in Fig. 1, different services are situated—in a microservice-based architecture—on top of the Big Data software stack. The data analysis service is the one that allows frontend applications to start, monitor, and manage forecasting computations on the cluster. Such applications can access the service through a REST API. Additionally, with the help of a Web UI, i.e. a more convenient frontend application, the data analyst can operate the services using a highly customizable and dynamic user interface. These frontend applications spare the data analyst any interaction with the actual cluster and hide the complexity of the different Big Data tools. Furthermore, to develop different algorithms using *Spark* the data analyst can use the *Apache Zeppelin*[6] tool to implement and document new computation jobs dynamically.

A usage scenario for the Big Data forecasting concept is illustrated in Fig. 2. First, the data analyst implements a forecasting model using the *Zeppelin* software. After thorough tests, the source code can be uploaded to the cluster using the Web UI. Thereafter, the data analysis service triggers the compilation of the source code and the creation of a new *Spark* job that is then persisted in the *Hadoop Distributed File System* (*HDFS*). In the next step, the data analyst selects a time series training set and starts the training algorithm. The resulting model and parameters are stored in the cluster. Using these results, the forecasting model can be applied to new data. Afterwards, the forecast

[3] mapr.com/.
[4] dcos.io/.
[5] hadoop.apache.org/.
[6] zeppelin.apache.org/.

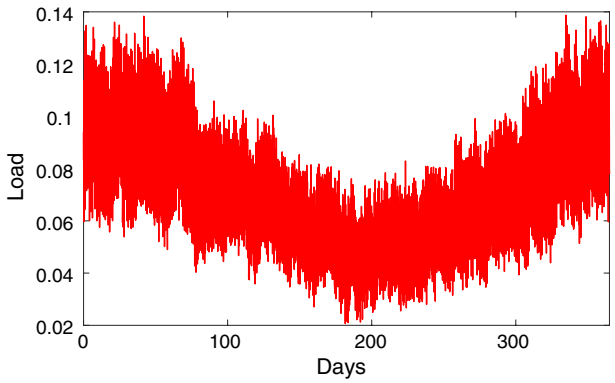

Fig. 3 One year of the simulated household load time series

results are stored in the time series database with a link to the forecasting model for later retrieval. The *Spark* job itself has to be able to access the databases in order to load the data in a distributed way. This is an important criterion in terms of performance.

Some examples in which the high amounts of data may require the utilization of Big Data forecasting infrastructure are:

- Building-specific or consumer-specific forecasts for a whole region
- Coherent hierarchical forecasts [27] at various spatial/temporal aggregation levels
- Ensemble [28] or probabilistic [29] high resolution forecasts, e.g., 1-day or 1-week ahead ensemble forecasts with a high temporal resolution

Methods (benchmark)

The benchmark conducted in the present paper has two main goals: (i) to assess the necessary time to obtain data-driven forecasting models on a distributed environment and (ii) to determine the point at which a Big Data computing framework based on *Spark* becomes necessary. To achieve these goals, a test scenario is conducted in which the times needed for training and evaluating data-driven forecasting models on a single computer and in a distributed environment are calculated and compared. The specifics of the tested scenario, as well as of the data and data mining techniques used, are described below. Moreover, only the computation times for training and evaluating the forecasting models are of interest in the present contribution.

Data

The present contribution uses a dataset containing a simulated dimensionless single household electrical load time series. The time series values represent measurements taken every second over the course of 10 years, i.e. $K = 3.16 \cdot 10^8$. This time series is created using the signal generator developed by Stefan Klaiber that was also used to generate the data in [30]. The time series used in the present contribution was not selected for its timespan nor its temporal resolution, but rather for the amount of available measurements considered to be acceptable to achieve the present contribution's goals. Figure 3 depicts a year of the simulated household load time series. The amount of measurements

is quite conservative considering that electrical data recorders are currently able to obtain up to 256 measurements per period of a 50 [Hz] input signal [31].

Data mining techniques

In the present paper three different data mining techniques are used to obtain the various data-driven forecasting models: a multiple linear regression (MLR) [32], a least absolute shrinkage and selection operator (LASSO) [32], and a random forest [33]. All created models use only five previously selected past load values as input.

While the models created from the multiple linear regression approach and LASSO are linear combinations of their used features, the models obtained from the random forest are not. A random forest is what is called an ensemble learning method, meaning that its computed forecasting model is a combination of several different underlying models created using the same data mining technique. It is important to mention that all techniques used in *Spark* were taken from its machine learning libraries.

Test scenario

In the test scenario, data-driven forecasting models are trained on a single computer or on a computing cluster using the previously described techniques and an amount of training data corresponding either to 1 day (1D), 1 week (1W), 1 month (1M), 6 months (6M), 1 year (1Y), 5 years (5Y), or 10 years (10Y) of the load time series. *R* and *Spark* – with *Spark* using eight (*SC*8) computing cores—are used in the case of the single computer, while on the computing cluster only *Spark* is utilized (*SCl*). The combination of the different amounts of training data and the different approaches for training the models results in 21 different tests. The abbreviations used to refer to the conducted tests are contained in Table 1. The forecasting models are, thereafter, evaluated on their training data. This evaluation consists of applying the forecasting model and calculating a corresponding evaluation value (e.g., the mean absolute error). The application and evaluation procedures are coupled since *Spark* does not apply a forecasting model unless it is necessary (i.e. lazy evaluation). The necessary computation time for both the models' training and evaluation in each test is measured and compared. The time needed for loading the data in-memory is not measured in the present contribution. Additionally, since only the computation times and not the forecasting accuracy are relevant in the present article, the data set is not separated in a training and a test set.

Table 1 Conducted tests

Data amount	*R*	*SC*8	*SCl*
1D ($8.64 \cdot 10^4$ values)	R_{1D}	$SC8_{1D}$	SCl_{1D}
1W ($6.05 \cdot 10^5$ values)	R_{1W}	$SC8_{1W}$	SCl_{1W}
1M ($2.68 \cdot 10^6$ values)	R_{1M}	$SC8_{1M}$	SCl_{1M}
6M ($1.57 \cdot 10^7$ values)	R_{6M}	$SC8_{6M}$	SCl_{6M}
1Y ($3.14 \cdot 10^7$ values)	R_{1Y}	$SC8_{1Y}$	SCl_{1Y}
5Y ($1.58 \cdot 10^8$ values)	R_{5Y}	$SC8_{5Y}$	SCl_{5Y}
10Y ($3.16 \cdot 10^8$ values)	R_{10Y}	$SC8_{10Y}$	SCl_{10Y}

*SC*8 *Spark* with eight processing cores on a single computer, *SCl Spark* on the computing cluster

The single computer possesses an Intel Core i7-6700 processor with 3.40 GHz and 16 GBs of RAM, while the computing cluster is comprised of three nodes, each with two 8 Core Intel(R) Xeon(R) (with active hyper-threading) processors with 2.40 GHz and 128 GB of RAM. Moreover, the utilized versions of *R* and *Spark* are 3.3.3 and 2.1 respectively.

Results and discussion

The computation times required for the training and evaluation of forecasting models for the different tests and the three different data mining techniques are shown in Table 2. The presented results are mean and standard deviation values obtained by training and evaluating the different models ten separate times. Note that the tests for a random forest with *Spark* on the single computer using 5 and 10 years of training data were not conducted.

The obtained results show that *R* is only faster than both variants of *Spark* at training and evaluating MLR models when the lowest amounts of data (1 day to 1 month) are used. However, once amounts of data equal to or larger than 6 months are utilized, *Spark* on the computing cluster outpaces *R* at both training and evaluating the MLR models. In addition, the single computer running *R* runs out of memory when data amounts larger than or equal to 5 years are used. For the sake of illustration, Fig. 4 depicts—using a logarithmic axis—the computation times for training a MLR in *R* and *Spark* on the computing cluster.

Table 2 Mean computation times (seconds) for forecasting models training and evaluation

Tests	Training			Evaluation		
	MLR	LASSO	Random forest	MLR	LASSO	Random forest
R_{1D}	*0.014 (0.009)*	0.407 (0.074)	1773.078 (18.274)	*0.01 (0.008)*	*0.073 (0.026)*	583.438 (16.824)
$SC8_{1D}$	0.208 (0.015)	*0.223 (0.026)*	22.122 (1.639)	0.192 (0.027)	0.188 (0.015)	20.905 (1.755)
SCl_{1D}	0.399 (0.022)	0.404 (0.040)	*10.342 (0.202)*	0.462 (0.029)	0.440 (0.020)	*7.247 (0.199)*
R_{1W}	*0.07 (0.006)*	2.7 (0.075)	OoM	*0.072 (0.007)*	*0.27 (0.021)*	OoM
$SC8_{1W}$	0.271 (0.013)	*0.281 (0.013)*	108.993 (6.285)	0.348 (0.022)	0.353 (0.012)	71.187 (7.316)
SCl_{1W}	0.390 (0.022)	0.409 (0.025)	*28.418 (0.491)*	0.475 (0.025)	0.476 (0.018)	*14.416 (0.169)*
R_{1M}	*0.324 (0.056)*	13.438 (0.247)	OoM	*0.267 (0.074)*	1.346 (0.223)	OoM
$SC8_{1M}$	0.627 (0.015)	0.638 (0.018)	454.759 (3.311)	1.058 (0.15)	1.074 (0.031)	243.008 (6.786)
SCl_{1M}	0.463 (0.021)	*0.482 (0.032)*	*67.469 (2.125)*	0.644 (0.042)	*0.652 (0.036)*	*33.546 (1.908)*
R_{6M}	1.862 (0.146)	115.960 (17.589)	OoM	3.206 (0.311)	13.610 (11.116)	OoM
$SC8_{6M}$	2.822 (0.060)	2.857 (0.077)	3101.202 (14.726)	5.420 (0.101)	5.457 (0.150)	1722.941 (36.603)
SCl_{6M}	*1.133 (0.055)*	*1.115 (0.028)*	*357.388 (5.037)*	*1.747 (0.050)*	*1.757 (0.061)*	*161.561 (2.156)*
R_{1Y}	4.061 (0.165)	OoM	OoM	6.903 (0.718)	OoM	OoM
$SC8_{1Y}$	5.588 (0.050)	5.604 (0.041)	6291.860 (43.074)	10.809 (0.049)	10.800 (0.053)	3246.918 (35.248)
SCl_{1Y}	*1.934 (0.061)*	*1.903 (0.055)*	*784.018 (11.487)*	*1.934 (0.061)*	*1.903 (0.055)*	*304.891 (3.768)*
R_{5Y}	OoM	OoM	OoM	OoM	OoM	OoM
$SC8_{5Y}$	41.464 (0.575)	42.520 (1.937)	NT	72.940 (1.726)	73.728 (2.644)	NT
SCl_{5Y}	*16.104 (0.711)*	*15.474 (0.683)*	IOF	*26.528 (1.346)*	*26.380 (1.000)*	IOF
R_{10Y}	OoM	OoM	OoM	OoM	OoM	OoM
$SC8_{10Y}$	NT	NT	NT	NT	NT	NT
SCl_{10Y}	38.997 (1.869)	39.300 (1.755)	IOF	63.044 (2.157)	63.127 (2.301)	IOF

The values in parenthesis are the standard deviation values

Italics: lowest computation time for a given data mining technique and a certain amount of training/evaluation data

OoM out of memory, *NT* not tested, *IOF* integer overflow

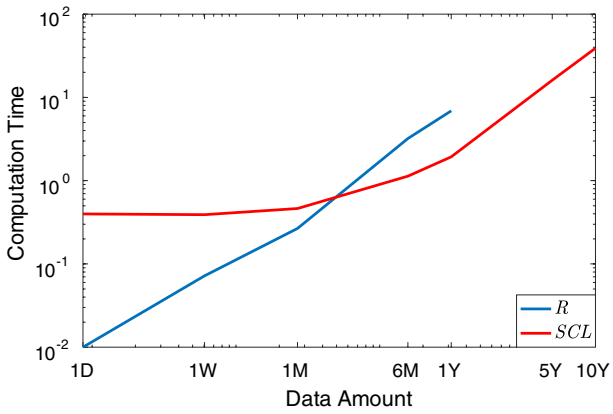

Fig. 4 Computation time required for training a MLR using different amounts of data

For the most complex approach in this contribution, the random forest, the *Spark* cluster shows its full potential and is clearly the fastest. Neither the single computer *Spark* variant nor the *R* variant comes close to the cluster's computation times. Additionally, the single computer using *R* runs out of memory quickly. The computer can only train a random forest with the data corresponding to one day. However, the computing cluster also fails to train a random forest with data larger than 1 year. This failure most likely stems from an integer overflow. As it can be seen in Table 2, finding a single winner for all models and amounts of data is not possible. Looking at the results for the LASSO model, *R* exhibits the worst computation times for training, regardless of the data amount. Yet, it is not clear which of the remaining two variants is best. While the Spark computing cluster is faster for data larger than 1 month, its single machine variant is the best choice for smaller amounts of data. With respect to the evaluation computation times, the Spark cluster is again the fastest for data larger than a week. This time however, R is the fastest for the smallest two amounts of data.

Certain features of *Spark* have to be taken into account to achieve low computation times. During the writing of the present work, caching was found to be one of those relevant features. The reason being, *Spark* is limited by available RAM and must know which data has to be kept in-memory in order to overcome this limitation. To determine the influence caching has on performance, the following test is conducted: MLR forecasting models are trained and evaluated using 6 months of data and utilizing both *Spark* on the single computer and *Spark* on the computing cluster. Furthermore, three different caching strategies (CS) are used:

1. CS1: No caching
2. CS2: Caching of input data, but not of intermediate results
3. CS3: Caching of input data and of intermediate results

The input data and the intermediate results—both represented as data frames—are manually cached if their caching is required by the used strategy. The results obtained from the three different caching strategies are contained in Table 3.

Table 3 Mean computation times (seconds) for MLR forecasting models training and evaluation using different caching strategies

Tests	Training			Evaluation		
	CS1	CS2	CS3	CS1	CS2	CS3
$SC8_{6M}$	356.695 (4.019)	*2.822 (0.060)*	2.925 (0.047)	233.737 (2.003)	*5.420 (0.101)*	9.467 (0.364)
SCl_{6M}	37.562 (0.290)	1.133 (0.055)	*1.128 (0.041)*	47.532 (0.374)	*1.747 (0.050)*	1.777 (0.055)

The values in parenthesis are the standard deviation values

Italic: lowest computation time for a given caching strategy

As seen by the results obtained for CS1, no caching results in poor computation performance; this can be explained by the fact that *Spark* is unable to keep the necessary data in-memory. Hence, *Spark* is forced to read the data from the provided source using slow read operations every time the data is required. Interestingly, caching the necessary input data and all the intermediate results (CS3) is not the optimal solution either; since the used workers may run out of memory. Nonetheless, CS3 still results in the fastest training of an MLR using *Spark* on the cluster. CS2, i.e. caching only the necessary input data, resulted in the lowest evaluation computation times. It is important to note that CS2 is the caching strategy used to obtain the results shown in Table 2.

Another important factor to consider is the repartitioning of data. In order to distribute the computation, the original dataset must be split into partitions and distributed to the available workers. For small datasets, a one to one ratio between partitions and workers is recommended. As the amount of data increases, the number of partitions must increase accordingly; otherwise, the partitions will become too large to be processed by a single worker and will result in out of memory errors.

Conclusion and outlook

In the present contribution, a Big Data forecasting concept is described. Afterwards, a benchmark comparing the training and evaluation of data-driven forecasting models using different amounts of data, as well as *R* and *Spark* on a single computer and *Spark* on a computing cluster is presented. The obtained results show the points at which a Big Data computing framework based on *Spark* may be advantageous, for instance, when using a complex data mining technique or when surpassing a specific amount of data. The former is shown by the fact that *Spark* on the cluster has−—for the conducted benchmark—the lowest computation times for training and evaluating a complex data-driven model, i.e. a random forest; the latter is shown by *Spark* on the computing cluster outpacing both single computer approaches independently of the utilized technique once a data amount threshold is surpassed (i.e. in the presented benchmark a training set comprised of $1.57 \cdot 10^7$ values). The results also show that *Spark* is sensitive towards certain factors like caching and the repartitioning of data; factors that when disregarded may reduce the computational advantages provided by *Spark*.

Even though the present contribution showed certain benefits of utilizing *Spark* on a computer cluster for the creation and application of energy forecasting models, there is still a number of questions that need to be answered in future works. For example, how does the behaviour and computation times of *Spark* change if the necessary data

is loaded from and saved into a database or if a forecasting model is implemented as an online streaming service.

Authors' contributions
All authors contributed equally. All authors read and approved the final manuscript.

Acknowledgments
The authors acknowledge the support given by the "Deutsche Forschungsgemeinschaft" and by the Open Access Publishing Fund of the Karlsruhe Institute of Technology. The authors also acknowledge careful English language editing by Alexander Murray (KIT).

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
The data will not be shared.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
The present contribution is supported by the Helmholtz Association under the Joint Initiative "Energy System 2050 - A Contribution of the Research Field Energy". This work was also partially funded by the DFG Research Training Group 2153: "Energy Status Data - Informatics Methods for its Collection, Analysis and Exploitation".

Publisher's Note

Received: 5 December 2017 Accepted: 24 February 2018
Published online: 06 March 2018

References

1. Fang X, Misra S, Xue G, Yang D. Smart grid—the new and improved power grid: a survey. IEEE Commun Surv Tutor. 2012;14(4):944–80.
2. Zhou S, Brown MA. Smart meter deployment in Europe: a comparative case study on the impacts of national policy schemes. J Clean Prod. 2017;144:22–32.
3. Monti A, Ponci F, Ferdowsi M, McKeever P, Löwen A. Towards a new approach for electrical grid management: the role of the cloud. In: Proc., IEEE international workshop on measurements & networking (M&N). 2015. p. 1– 6.
4. Rusitschka S, Eger K, Gerdes C. Smart grid data cloud: a model for utilizing cloud computing in the smart grid domain. In: Proc., 1st IEEE international conference on smart grid communications (SmartGridComm). 2010. p. 483– 88.
5. Simmhan Y, Aman S, Kumbhare A, Liu R, Stevens S, Zhou Q, Prasanna V. Cloud-based software platform for Big Data analytics in smart grids. Comput Sci Eng. 2013;15(4):38–47.
6. Gottwalt S, Gärttner J, Schmeck H, Weinhardt C. Modeling and valuation of residential demand flexibility for renewable energy integration. IEEE Trans Smart Grid. 2017;8(6):2565–74.
7. Shyam R, Kumar S, Poornachandran P, Soman K. Apache Spark a Big Data analytics platform for smart grid. Proced Technol. 2015;21:171–8.
8. Hassani H, Silva ES. Forecasting with Big Data: a review. Ann Data Sci. 2015;2(1):5–19.
9. Liu X, Golab L, Golab WM, Ilyas IF. Benchmarking smart meter data analytics. In: EDBT. 2015. p. 385– 396.
10. Hyndman RJ, Athanasopoulos G. Forecasting: principles and practice. Lexington: OTexts; 2014.
11. González Ordiano JÁ, Waczowicz S, Hagenmeyer V, Mikut R. Energy forecasting tools and services. Wiley Interdiscip Rev Data Min Knowl Discov. 2018;8(2):1235.
12. Monteiro C, Bessa R, Miranda V, Botterud A, Wang J, Conzelmann G. Wind power forecasting: state of the art 2009. Argonne National Laboratory (ANL): Technical report; 2009.
13. Pelland S, Galanis G, Kallos G. Solar and photovoltaic forecasting through post-processing of the global environmental multiscale numerical weather prediction model. Prog Photovolt Res Appl. 2013;21(3):284–96.
14. Duran MJ, Cros D, Riquelme J. Short-term wind power forecast based on ARX models. J Energy Eng. 2007;133(3):172–80.
15. Fan S, Chen L. Short-term load forecasting based on an adaptive hybrid method. IEEE Trans Power Syst. 2006;21(1):392–401.
16. González Ordiano JÁ, Waczowicz S, Reischl M, Mikut R, Hagenmeyer V. Photovoltaic power forecasting using simple data-driven models without weather data. Comput Sci Res Dev. 2017;32:237–46.

17. Antonanzas J, Osorio N, Escobar R, Urraca R, Martinez-de-Pison F, Antonanzas-Torres F. Review of photovoltaic power forecasting. Sol Energy. 2016;136:78–111.
18. Hong T, Pinson P, Fan S, Zareipour H, Troccoli A, Hyndman RJ. Probabilistic energy forecasting: global energy forecasting competition 2014 and beyond. Int J Forecast. 2016;32(3):896–913.
19. Jung J, Broadwater RP. Current status and future advances for wind speed and power forecasting. Renew Sustain Energy Rev. 2014;31:762–77.
20. Renkamp N. Short-term load forecasting in district heating networks. Master's thesis, Karlsruher Institut für Technologie. 2016.
21. Düpmeier C, Stucky K-U, Mikut R, Hagenmeyer V. A concept for the control, monitoring and visualization center in Energy Lab 2.0. In: Proc., of the 4th D-A-CH energy informatics conference. Berlin; Springer, Cham: 2015. p. 83– 94.
22. Hagenmeyer V, Cakmak HK, Düpmeier C, Faulwasser T, Isele J, Keller HB, Kohlhepp P, Kühnapfel U, Stucky U, Waczowicz S, Mikut R. Information and communication technology in Energy Lab 2.0: Smart energies system simulation and control center with an Open-Street-Map-based power flow simulation example. Energy Technol. 2016;4:145–62.
23. Manikandan SG, Ravi S. Big Data analysis using Apache Hadoop. In: Proc., of the 2014 international conference on IT convergence and security (ICITCS). 2014. p. 1– 4.
24. McAfee A, Brynjolfsson E, Davenport TH, Patil D, Barton D. Big Data. The management revolution. Harvard Bus Rev. 2012;90(10):61–7.
25. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop distributed file system. In: Proc., of the IEEE 26th symposium on mass storage systems and technologies (MSST). 2010. p. 1– 10.
26. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I. Spark: cluster computing with working sets. In: Proc., 2Nd USENIX conference on hot topics in cloud computing. HotCloud'10. Berkeley: USENIX Association: 2010. p. 10– 10.
27. Taieb SB, Taylor JW, Hyndman RJ. Coherent probabilistic forecasts for hierarchical time series. In: Proc., 34th international conference on machine learning. Sydney: PMLR, International Convention Centre; 2017. p. 3348– 3357.
28. Gneiting T, Raftery AE. Weather forecasting with ensemble methods. Science. 2005;310(5746):248–9.
29. Gneiting T, Katzfuss M. Probabilistic forecasting. Annu Rev Stat Appl. 2014;1:125–51.
30. Klaiber S, Waczowicz S, Konotop I, Westermann D, Mikut R, Bretschneider P. Prognose für preisbeeinflusstes Verbrauchsverhalten. at-Automatisierungstechnik. 2017;65(3):179–88.
31. Maaß H, Cakmak HK, Bach F, Mikut R, Harrabi A, Süß W, Jakob W, Stucky K-U, Kühnapfel UG, Hagenmeyer V. Data processing of high rate low voltage distribution grid recordings for smart grid monitoring and analysis. EURASIP J Adv Signal Process. 2015;1:1–21.
32. Fahrmeir L, Kneib T, Lang S, Marx B. Regression: models, methods and applications. Berlin: Springer; 2013.
33. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.