Temporal and Geospatial Patterns Through Local Weather Simulation in BigPetStore

Ronald J. Nowling
Red Hat, Inc.
Raleigh, NC 27601
Email: rnowling@redhat.com

Jay Vyas
Red Hat, Inc.
Raleigh, NC 27601
Email: jvyas@redhat.com

***Abstract*—Generating large amounts of semantically-rich data for testing big data workflows is paramount for scalable performance benchmarking and quality assurance in modern machine-learning and analytics workloads. The most obvious use case for such a generative algorithm is in conjunction with a big data application blueprint, which can be used by developers (to test their emerging big data solutions) as well as end users (as a starting point for validating infrastructure installations, building novel applications, and learning analytics methods).**

We present a new domain-driven, generative data model for BigPetStore, a big data application blueprint for the Hadoop ecosystem included in the Apache BigTop distribution. We describe the model and demonstrate its ability to generate semantically-rich data at variable scale ranging from a single machine to a large cluster. We validate the model by using the generated data to answer questions about customer locations and purchasing habits for a fictional targeted advertising campaign, a common business use case.

***Keywords*—big data, synthetic data sets, data generation, benchmarking, testing, probabilistic models**

I. Introduction

Big data applications process large, dynamic, multidimensional data sets with the general goal of information and knowledge extraction. With the wide variety of big data tools available and lagging documentation, both developers and users of big data systems benefit from the availability of realistic example applications. For developers, such applications are useful for testing, benchmarking, and evaluating design choices; for users, example applications provide starting points for learning methods, developing their own applications, and implementing new analytics workflows.

BigPetStore is one such big data application blueprint built around processing transaction data for a fictional chain of pet stores. BigPetStore targets the Hadoop [1] ecosystem with examples implemented for loading, cleaning, aggregating, and performing analytics on data using Hive [2], Pig [3], [4], and Mahout [5]. BigPetStore has been incorporated into the open-source Apache BigTop distribution [6], where it is used for testing and as a reference application. One of BigPetStore's key features is the ability to scale from a single machine to a large cluster, making it easy to develop projects on a local machine and transfer the application to a large cluster for production testing.

To achieve the project's goals of providing high-quality, realistic examples, BigPetStore requires pattern-rich, complex data. At its core, BigPetStore relies on a generative data model for producing synthetic transaction data. Despite the growing number of real data sets now publically available, synthetic data generators have a number of advantages for BigPetStore over real data sets. The synthetic data generator can be packaged with BigPetStore, avoiding the cost, time, and infrastructure needed to host, transfer, and store large data sets. Synthetic data generators are scalable, allowing the user to choose how much data to generate – a requirement for supporting BigPetStore's goal of running on both single machines and large clusters. Licensing and privacy issues are avoided with the use of synthetic data sets. And lastly, the generated data can be customized by the user, allowing the user to generate data with specific criteria amenable to testing. For example, a user may generate transactions from a small number of purchasable items so that clustering results can be easily visualized.

A variety of approaches for generating synthetic data sets exist. TeraGen and the Intel Hadoop Benchmark Suite [7] are popular tools that can generate data sets quickly and at any scale, but the resulting data is semantically-void and not useful for much more than simple benchmarks. Many frameworks [8]–[18] for generating data that satisfy user-defined schemas exist. Such frameworks allow for reproducing the time-indepenent, structural properties of real data, but the frameworks are not expressive enough to describe the dynamic generation processes and latent variables that would be needed to embed the desired informational complexity and rich patterns needed for BigPetStore's analytics examples.

Until such purely generic methods reach maturity, BigPetStore's needs are best met by a customized, domain-driven model. In [19], the authors described a new, domain-driven model combining the features of the TeraGen (scalability) and MovieLens [20] (rich patterns) data sets for generating data. *Ab initio* probabilistic models of stores and customers and stochastic dynamical models of purchasing customer behaviors parameterized by real data from the U.S. Census were used to generate data with geospatial and temporal patterns. The approach was validated by analyzing geospatial patterns and customer purchasing preferences in data generated by Python prototype of the model.

Influenced by observations that weather patterns affect customers' daily shopping patterns [21] and retail sales [22], we describe a new stochastic model for simulating local weather features such as temperature and precipitation and updates to the data generator model that incorporate the effects of

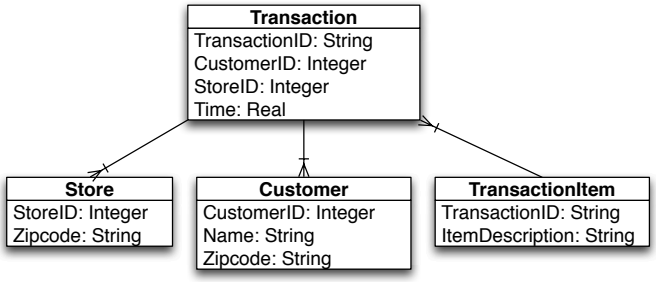


Fig. 1: Relational Data Model for Generated Data

weather on transaction times. With the addition of the weather model, generated data are now more realistic with geospatial and temporal patterns that vary by region and season.

Additionally, a new Java library implementation of the data generator is described. Compatibility with existing, JVM-based big data tools and support for parallelization are demonstrated through an Apache Spark [23], [24] driver for the library. The implementation is available under the Apache Public License v2.

II. Overview of Previous Model

Our generative model combines various well-known mathematical modeling techniques to simulate the factors affecting customers' purchasing habits. Each part of the model is derived from *ab initio* assumptions. In several cases, real data is used to parameterize parts of the model. We described the initial version of the model in [19], which we summarize here.

The model generates data four types of records: stores, customers, transactions, and transaction items (Figure 1). Store records contain both a unique identifier and a location in the form of a 5-digit zip code. Customer records consist of a unique identifier, a name, and a location given as a 5-digit zip code. Transaction records consist of a transaction identifier, a customer identifier, a store identifier, and a time given in days since the beginning of the simulation. Transaction item records contain a transaction identifier and an item description. The item description is a list of key-value pairs stored as a string. Key-value pairs are used since item characteristics differ depending on the type of item. For example, dog food has a brand, a flavor, a size (in lbs), and a per-unit cost, while poop bags have a brand, a color, a count, and a per-unit cost.

A. Stores

Store records consist of unqiue integer identifiers and locations given as zip codes. The stores' zip codes are sampled from a probability mass function (PMF) where probabilities are linearly proportional to the zip code's population and exponentialy proportional to the zip code's median household income. The population and household income data for the zip codes were taken from from the U.S. Census American Community Survey [25].

B. Customers

Customer records consist of uniquer integer identifers, first names, last names, and locations given as zipcodes. Names are generated using data from the Name Database [26]. Each record in the database gives a name, a weight, and flag indicatings if the name can be used as a first name, a last name, or both. PMFs generated for the first and last names using the weights. The customer's name is generated by sampling a first name and a last name from each PMF respectively. Zip codes are sampled from a PMF where the probabilities are exponentially distributed according to the distance (in miles) to the nearest store's location. Distances are calculated using latitude and longitude coordinates taken from the Zip Code Database Project [27]).

C. Transactions

Transactions records consist of customer, transaction, and store identifiers, the date and time of the transaction (as a real number indicating the number of days since the start of the simulation), and a list of purchased products. The customer and transaction identifiers form a unique composite key for each transaction record.

Transactions are simulated individually for each customer. Transaction times are sampled using a Monte Carlo process. Transaction times are proposed by sampling offsets before the exhaustion of products from an exponential distribution, with the average time parameterized separately for each customer by sampling from a uniform interval. Acceptance probabilities are given by a separate PDF which can be used to model the effect of with factors such as store hours and customer work schedules. Currently, the acceptance PDF only ensures that a new transaction time occurs after the previous transaction. The proposed transaction time is accepted if a random number sampled uniformally from $[0, 1)$ is less than the acceptance probability. Otherwise, a new transaction time is proposed.

Product purchases are simulated by a separate process. First, a state is chosen from the set of product categories and a stop state which indicates the end of the transaction. Each product category's weight is exponentially proportional to the elapsed time between the transaction and exhaustion of the product category. The weight of the stop state is given by a constant which is zero until the first product is chosen.

Once a product category has been chosen, the process of choosing a product within the category is modelled by a Markov Model. The transition probabilites are determined by .... A product is chosen by advancing the state of the model by simulate one transition. The state of the Markov Model is preserved between transactions.

Once a product has been chosen, the customer's inventory is updated and the exhaustion time is determined by simulating the usage of the products by numerically integrating a stochastic differential equation (SDE). The purchasing process is restarted, using the updated exhaustion time. If the stop state is chosen, the next transaction is generated.

III. Improvements to the Data Generator

A. Association of Customers and Transactions with Stores

In the data generator prototype, the store for each each transaction was picked uniformally from the set of all stores. To improve upon the realism and support the incorporation of geospatial patterns, each customer is now associated with a

specific store and all of a customer's transactions occur at that store.

In the new model, a store is sampled for each customer from a PMF with weights proportionally to population of the stores' zip codes. The customers' locations are sampled from a PMF exponentially-distributed weights for each zip code z according to the distance between the zip code s of the customers' associated store:

$$w(s,z) = \lambda \exp(-\lambda d(s,z)) \tag{1}$$

where λ is the inverse of the desired average distance.

B. Weather Model

TODO why?

Weather data is generated using a local weather model – i.e., weather is not explicitly correlated between regions – so that each region can be simulated independently for computational efficiency. The model was derived and evaluated by comparing to local weather data downloaded from the NOAA QCLCD cite. The goal of the model was not to accurately predict day-to-day weather patterns but to reproduce temporal and static statistical properties. The model is similar in concept to other models [28]–[31] available in the literature.

Models were derived for daily average temperatures ($T(t)$, °F), daily total precipitation ($P(t)$, in), and daily average wind speed ($V(t)$, mph). The temperature for each location is modeled as a first-order Fourier series plus noise generated by an Ornstein-Uhlenbeck process $Z(t)$ [32]:

$$T(t) = a_0 + a_1 \sin\left(\frac{-2\pi t}{365}\right) + a_2 \cos\left(\frac{-2\pi t}{365}\right) + Z(t) \tag{2}$$

$$dZ(t) = -\gamma Z(t)dt + \sigma dW_t \tag{3}$$

The stochastic differential equation (SDE) for the Ornstein-Uhlenbeck process is integrated numerically using the Euler-Maruyama method [33]:

$$\begin{aligned} Z_{t+1} &= -\gamma Z_t \Delta t + \sigma\sqrt{\Delta t}\Delta W_{t+1} \\ Z_0 &= 0 \end{aligned} \tag{4}$$

The daily total precipitation for each region is assumed to be independently distributed from day-to-day and sampled from an exponential distribution:

$$P(t) \sim Exp(\lambda) \tag{5}$$

where λ is inverse of the daily average precipitation.

The daily average wind speed for each region is modeled using a first-order Fourier series and values sampled independently day-to-day from an Erlang distribution:

$$V(t) = b_0 + b_1 \sin\left(\frac{-2\pi t}{365}\right) + b_2 \cos\left(\frac{-2\pi t}{365}\right) + X(t) \tag{6}$$

$$X(t) \sim Erlang(k,\theta) \tag{7}$$

where k and θ are the shape and scale parameters, respectively.

From the temperature, precipitation and wind speed, the snowfall ($S(t)$, in), rainfall ($R(T)$, in), and wind chill ($WC(T)$, °F) cite are calculated as follows:

$$r(t) = \frac{1}{1+\exp(-a(T-b))} \tag{8}$$

$$S(t) = 10(1-r(t))P(T) \tag{9}$$

$$R(T) = r(t)P(T) \tag{10}$$

$$\begin{aligned} WC(T) = 35.74 + 0.6215T(t) - 35.75V^{0.16}(t) + \\ 0.4275T(t)V^{0.16}(t) \end{aligned} \tag{11}$$

Model parameters were estimated separately for each weather station in the NOAA QCLCD data from October 2011 to September 2014. Weather stations missing more than 5% of the data were excluded. The Fourier coefficients a_0, a_1, a_2, b_0, b_1, and b_2 were found from Fourier transforms of the daily average temperature and daily average wind speed data. The distribution parameters σ, λ, k, and θ were computed directly from the data using the Maximum Likelihood Estimaters (MLE). The damping coefficient γ was determined empirically and the same value was used for every weather station.

Local weather simulations are run for each store using model parameters from the closest weather stations. The closest weather station is found for each store by computing the distances from the GPS coordinates of the stores' zip codes and weather station from the NOAA QCLCD data.

C. Incorporating the Effect of Weather on Transaction Times

The transaction time model was updated to incorporate the effect of the daily weather on customers' shopping times.

Separate conditional probability density functions (PDFs) giving the probability of a transaction occuring on day d_i is given for the daily wind chill and snow fall. The PDFs are modeled using sigmoidal functions:

$$P(\text{trans}_{d_i} = \text{YES}|x_{d_i}) = \frac{A}{1+\exp(-B(x-C))} + D \tag{12}$$

TODO table of parameters

The probability of a transaction occurring on a day d_i is given by the minimum probability of the individual PDFs:

$$\begin{aligned} P(\text{trans}_{d_i} = \text{YES}|S_{d_i}, WC_{d_i}) = \min\{&P(\text{trans}_{d_i} = \text{YES}|S_{d_i}), \\ &P(\text{trans}_{d_i} = \text{YES}|WC_{d_i})\} \end{aligned} \tag{13}$$

The transaction time model was modified by adding the weather PDF to the transaction time acceptance probability PDF for transaction time t_i:

$$P(\text{trans}_{t_i} = \text{YES} | t_{i-1}, S_{d_i}, WC_{d_i}) = Z^{-1} P(\text{trans}_{t_i} = \text{YES} | t_{i-1}) P(\text{trans}_{d_i} = \text{YES} | S_{d_i}, WC_{d_i}) \quad (14)$$

where Z is the normalization factor.

D. Java Library Implementation

The data generator was originally prototyped in Python [**?**]. Once the utility of the generator was validated, the generator was implemented as a Java library for production usage. Java was chosen for compatibility with popular big data processing systems such as Apache Hadoop [1] and Apache Spark [23], [24] and other JVM languages such as Scala and Clojure. The Java implementation has 5,500 lines of code, 110 classes, and 90 unit tests.

The library's API is relatively simple with one class for loading input data, five classes for generating data, and six data model classes. Broad support for multiple parallelization paradigms was a primary design requirement. Generators interact only through standard Java collections containing serializable objects and are designed to allow multiple instantiated copies to support parallelization. As such, parallelization is as simple as creating as many independent instances as needed and combining the resulting Java collections before each stage. An example driver implemented with Apache Spark with support for parallel and distributed computing is presented in Section IV. For development purposes, the library includes a single-threaded command-line driver.

The source code is available at https://github.com/rnowling/bigpetstore-data-generator under the Apache Public License v2. Jars are published via bintray for easy integration with build systems that support automatic dependency resolution.

IV. Parallelization with Spark Driver

TODO code listing

V. Evaluation of the Model

To evaluate the changes in the model, we analyzed data for 10 stores, 10,000 customers, and five years of transactions generated from the model.

A. Comparison of Real and Simulated Weather Data

Three years of observed daily average temperature, total precipitation, and average wind speed from the NOAA QCLCD data cite for South Bend, IN from October 2011 to September 2014 are shown in Figure 3 alongside simulated data generated from the model. The simulated data is able to produce temporal and stationary patterns similar to those of the real data.

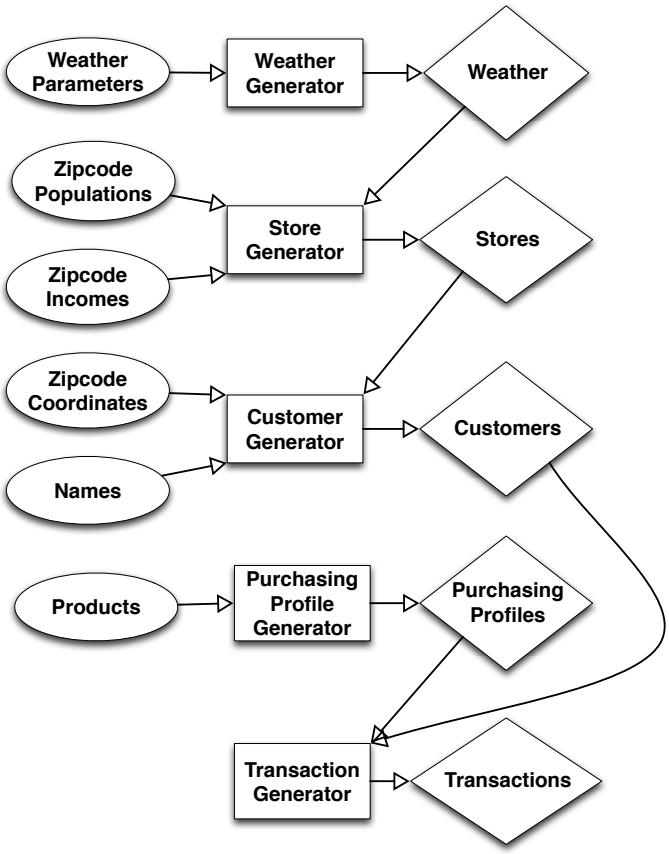


Fig. 2: BigPetStore Flowchart. Ovals represent input data, rectangles represent data generators, and diamonds represent generated data.

B. Weekly Transaction Counts by Store

The weekly transaction counts over five years of simulation for six stores are plotted in Figure 4. The variation in the number of transactions by store validates that the model is assigning customers to stores in proportion to the population of the stores' zip codes. Large fluctuations in the weekly transaction counts occur during winter months, validating that the weather transaction model is capturing the effect of weather on transaction times.

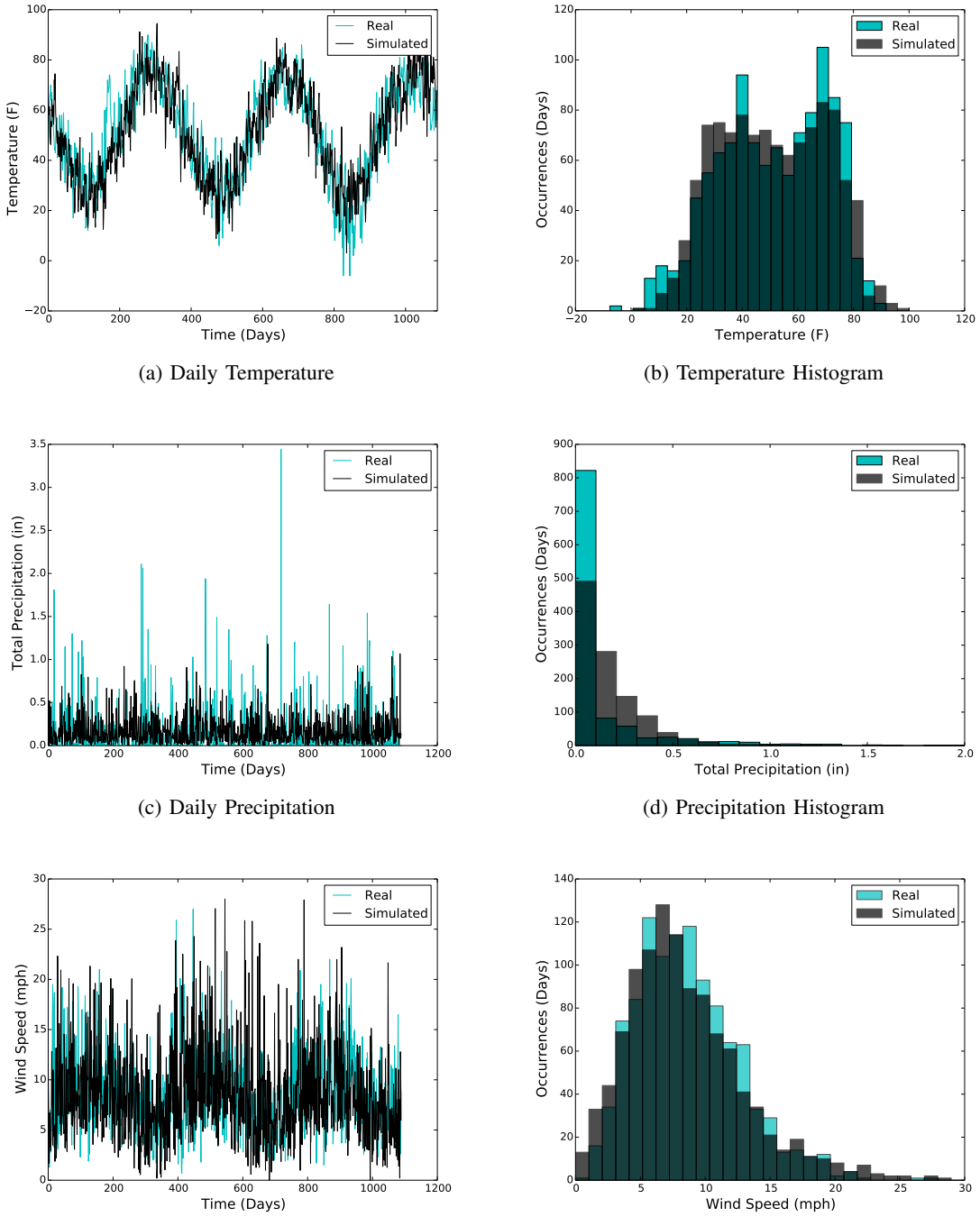
C. Impact of Weather on Shopping Patterns by Region

TODO: effect of weather, variations by region

VI. Discussion and Conclusion

The BigPetStore data generator is a domain-driven mathematical model and simulation for generating data with rich patterns. A new model for local weather simulation which enabled the incorporation of additional geospatial and temporal correlations was described. A new JVM library implementation was described. Support for parallelization and interopability with tools in the big data ecosystem was demonstrated using an Apache Spark driver.

The math model was validated using ...



(a) Daily Temperature

(b) Temperature Histogram

(c) Daily Precipitation

(d) Precipitation Histogram

(e) Daily Wind Speed

(f) Wind Speed Histogram

Fig. 3: NOAA QCLCD weather data alongside simulated data for South Bend, IN for October 2011 to September 2014.

We see multiple opportunities for building on the work presented. ...

We intend to expand the model to incorporate additional factors. Deeper integration of existing fields such as location and purchasing profiles (e.g., to model regional purchasing preferences) will increase the variety of the semantic information encoded in the generated data. The incorporation of weather and climate data can be used to influence when customers shop, the types of products they buy, and the amount of products purchased. For example, customers would be less likely to shop during snow storms, more likely to buy items such as winter apparel for their pets, and more likely to purchase bulk quantities to reduce the number of transactions. Modeling of time-dependent events such as sales, evolution of customer purchasing profiles over time, and "life events," such

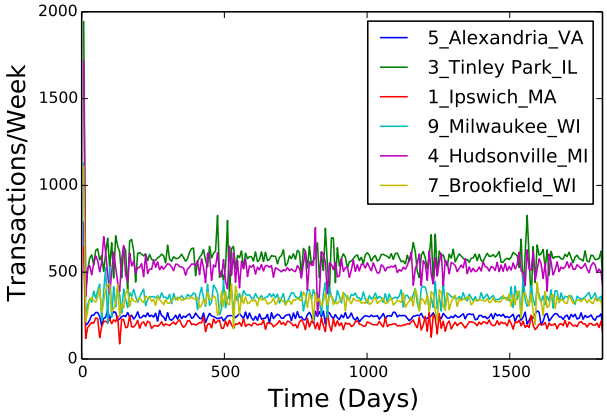


Fig. 4: Stores' Weekly Transaction Counts

as the birth or passing of pets, will enable more interesting time-series analysis of the data. We would also like to expand the scope of the model to incorporate business processes (such as inventory management, customer complaints, employees, etc.), thus enabling queries about the relationship between internal business process and customer behavior.

Acknowledgment

The authors would like to thank Brian P. Clare and Casey Robinson for useful and interesting discussion. Will Benton, Trevor M. Cickovski, Erik Erlandson, Scott A. Hale, Hank Jakiela, Casey Robinson, and Douglas Thain provided invaluable feedback on the manuscript. JV would like to thank the Apache BigTop community for their interest in and support of BigPetStore. The authors would also like to thank Matt Fenwick, Nigel Savage, and Bhashit Parikh for their contributions to BigPetStore. The authors would like to thank Red Hat, Inc. and the University of Notre Dame for their support of this work.

References

[1] "Apache Hadoop," http://hadoop.apache.org/.

[2] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop," in *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. Ieee, 2010, pp. 996–1005.

[3] C. Olston, B. Reed, and U. Srivastava, "Pig latin: a not-so-foreign language for data processing," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of Data*, 2008, pp. 1099–1110.

[4] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1414–1425, 2009.

[5] "Apache Mahout," https://mahout.apache.org/.

[6] "Apache BigTop," http://bigtop.apache.org/.

[7] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," in *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. IEEE, 2010, pp. 41–51.

[8] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolette, and H.-A. Jacobsen, "Bigbench: Towards an industry standard benchmark for big data analytics," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 1197–1208.

[9] T. Rabl and M. Poess, "Parallel Data Generation for Performance Analysis of Large, Complex RDBMS," in *Proceedings of the Fourth International Workshop on Testing Database Systems*, 2011, pp. 1–6.

[10] M. Frank, M. Poess, and T. Rabl, "Efficient update data generation for DBMS benchmarks," in *Proceedings of the Third Joint WOSP/SIPEW International Conference on Performance Engineering - ICPE '12*. Boston, Massachusetts, USA: ACM Press, 2012.

[11] T. Rabl, M. Frank, H. M. Sergieh, and H. Kosch, "A data generator for cloud-scale benchmarking," in *Performance Evaluation, Measurement and Characterization of Complex Systems*, R. Nambiar and M. Poess, Eds. Springer, 2011, pp. 41–56.

[12] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger, "Quickly generating billion-record synthetic databases," in *Proceedings of the 1994 ACM SIGMOD International Conference on Management of data*, 1994, pp. 243–252.

[13] N. Bruno and S. Chaudhuri, "Flexible Database Generators," in *VLDB '05 Proceedings of the 31st international conference on Very large data bases*, 2005, pp. 1097–1107.

[14] J. E. Hoag and C. W. Thompson, "A parallel general-purpose synthetic data generator," *ACM SIGMOD Record*, vol. 36, no. 1, pp. 19–24, Mar. 2007.

[15] "log-synth," https://github.com/tdunning/log-synth.

[16] A. Alexandrov, K. Tzoumas, and V. Markl, "Myriad: scalable and expressive data generation," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1890–1893, 2012.

[17] A. Alexandrov, C. Brücke, and V. Markl, "Issues in big data testing and benchmarking," in *Proceedings of the Sixth International Workshop on Testing Database Systems - DBTest '13*. New York, New York, USA: ACM Press, 2013, p. 1.

[18] A. Arasu, R. Kaushik, and J. Li, "Data generation using declarative constraints," in *Proceedings of the 2011 International Conference on Management of Data - SIGMOD '11*. New York, New York, USA: ACM Press, 2011, p. 685.

[19] R. J. Nowling and J. Vyas, "A Domain-Driven, Generative Data Model for BigPetStore," in *Proceedings of the Fourth IEEE International Conference in Big Data and Cloud Computing*, 2014.

[20] "Movielens," http://www.grouplens.org/datasets/movielens/.

[21] A. G. Parsons, "The Association Between Daily Weather and Daily Shopping Patterns," *Australasian Marketing Journal (AMJ)*, vol. 9, no. 2, pp. 78–84, Jan. 2001. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1441358201701772

[22] M. Starr-McCluer, *The effects of weather on retail sales*. Federal Reserve Board of Governors, 2000. [Online]. Available: http://www.federalreserve.gov/PubS/feds/2000/200008/200008pap.pdf

[23] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010, pp. 1–7.

[24] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012, pp. 1–14.

[25] "American Community Survey," http://www.census.gov/acs/www/.

[26] "https://www.drupal.org/project/namedb," http://incompetech.com/named/.

[27] "Zip Code Database Project," http://zips.sourceforge.net/.

[28] P. Racsko, L. Szeidl, and M. Semenov, "A serial approach to local stochastic weather models," *Ecological Modelling*, vol. 57, no. 1-2, pp. 27–41, Oct. 1991. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0304380091900534

[29] J. Chen, F. P. Brissette, and R. Leconte, "A daily stochastic weather generator for preserving low-frequency of climate variability," *Journal of Hydrology*, vol. 388, no. 3-4, pp. 480–490, Jul.

2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0022169410003082

[30] A. Birt, M. Valdez-Vivas, R. Feldman, C. Lafon, D. Cairns, R. Coulson, M. Tchakerian, W. Xi, and J. Guldin, "A simple stochastic weather generator for ecological modeling," *Environmental Modelling & Software*, vol. 25, no. 10, pp. 1252–1255, Oct. 2010. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1364815210000526

[31] S. Fatichi, V. Y. Ivanov, and E. Caporali, "Simulation of future climate scenarios with a weather generator," *Advances in Water Resources*, vol. 34, no. 4, pp. 448–467, Apr. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0309170811000042

[32] C. Gardiner, *Stochastic Methods: A Handbook for the Natural and Social Sciences*, 4th ed. Springer, 2009.

[33] P. E. Klouden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Springer, 2013.