Scalable and Efficient Data Analytics and Mining with Lemonade

Walter dos Santos Gustavo P. Avelar Manoel Horta Ribeiro
Dorgival Guedes Wagner Meira Jr.
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
{waltersf,gustavopaula,manoelribeiro,dorgival,meira }@dcc.ufmg.br

ABSTRACT

Professionals outside of the area of Computer Science have an increasing need to analyze large bodies of data. This analysis often demands high level of security and has to be done in the cloud. However, current data analysis tools that demand little proficiency in systems programming struggle to deliver solutions which are scalable and safe. In this context we present Lemonade, a platform which focuses on creating data analysis and mining flows in the cloud, with authentication, authorization and accounting (AAA) guarantees. Lemonade provides an interface for the visual construction of flows, and encapsulates storage and data processing environment details, providing higher-level abstractions for data source access and algorithms. We illustrate its usage through a demo, where a data processing flow builds a classification model for detecting fake-news, also extracting some insights along the way.

PVLDB Reference Format:
Walter dos Santos, Gustavo de P. Avelar, Manoel Horta Ribeiro, Dorgival Guedes and Wagner Meira Jr.. Scalable and Efficient Data Analytics and Mining with Lemonade. *PVLDB*, 11 (12): 2070-2073, 2018.
DOI: https://doi.org/10.14778/3229863.3236262

1. INTRODUCTION

Analyzing large bodies of data is an increasingly important task in the most diverse fields: journalists have to dig through terabytes of documents leaked by governments and corporations to find scoops [14] marketing specialists have to process extensive user logs to get insight on their behavior and preferences [12] businessmen are pressured to take "data-driven" decisions rather than trusting their guts or rules-of-thumb [13]. This has led to the flourishing of easy-to-use programming languages such as *Python* and also to the popularization of coding as an interdisciplinary practice.

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236262

This paradigm, however, has its shortcomings: *(i)* professionals without coding expertise are left out of data analysis tasks, which is undesirable, as their domain expertise would provide valuable insights; *(ii)* the volume and sensitivity of data demands scalable and efficient processing, which often has to be performed on the cloud while simultaneously requiring high levels of security.

Most existing solutions fail to address these issues. Parallel and distributed processing environments, such as Apache Hadoop [15], Apache Spark [16] and COMPSs [11], offer a level of abstraction that hides many details of parallel and distributed systems programming. However, the developer still needs a reasonable amount of programming knowledge, as well as mastering the technology to effectively use them [1]. Visual data flows tools, such as RapidMiner [7], Orange [3] and KNIME [2] are designed to be used locally, making it unfeasible to process large data sets that exceed the capabilities of a single machine. Some of these tools also provide web-based access, but charge expensive license fees. Other platforms, such as Microsoft Azure Machine Learning Studio (MAML) [6] and ClowdFlows [5] allow the use of multiple machines in clusters, but still have their limitations. MAML is not open-source and requires expensive licenses, whereas CrowdFlows does not support *QoS* (Quality of Service) and *AAA* (Authentication, Authorization & Accounting) aspects.

In this context we present Lemonade (Live Exploration and Mining Of a Non-trivial Amount of Data from Everywhere), a platform which focuses on creating data analysis and mining flows in the cloud, with authentication, authorization and accounting (AAA) guarantees. Using an interface for the visual construction of flows, Lemonade enables the creation and execution of flows by encapsulating the details of storage, coding, security and distributed processing, allowing them to be used in cloud environments by domain experts. The preliminary version of Lemonade was described in [9] and we now present and demonstrate its full operational version, which enhances and extends several aspects of the preliminary one. We consider the scenario of a fact-checking agency which has too many news to check. Overwhelmed with the sheer volume of news, they develop a simple classification model to detect the most likely false news-pieces, and thus being able to work more efficiently. Our demo is available online. Log in into `http://demovldb.lemonade.org.br` using the user and password, demo@lemonade.org.br and 123456 respectively.

2. LEMONADE

The Lemonade architecture consists of seven individual components, which function as *micro services* focused on tasks related to interface, security/privacy, execution, execution monitoring, data management, algorithms, and data visualization. Its structure supports the scalability of components individually, since they can be encapsulated in a *Docker* container or in standalone applications running in physical machines. The next subsections briefly describe the behavior of each of these components.

Limonero — Data Sources metadata

Limonero stores metadata about data sources and provides them as a service. It maintains information about the location, access permissions, storage details (name, data type, precision, size, data format), data characteristics such as distribution, missing values, mean and maximum values for each data source available. In the advent of databases with sensitive (or protected) attributes, the platform implements sophisticated control policies, such as anonymization techniques (e.g. k-anonimity [10]) to ensure the protection of the data stored and processed by the platform.

Tahiti — Operations metadata

Tahiti maintains metadata about individual operations and data flows created by users and provides them as a service. Operations are the smallest processing units in Lemonade, which currently supports reading and writing, ETL (Extraction, Transformation and Load), data mining and machine learning algorithms (and the evaluation of their results), as well as geo-referenced data operators.

Tahiti stores and provides information for each operation, such as its name, description, ports and parameters. We divide operation parameters in five categories: execution (e.g. number of iterations to be performed), privacy/security (e.g. access rights to the operation results), monitoring (e.g. granularity of event logging), appearance (colors and location of the boxes for operations), and quality of service requirements (e.g. thresholds for execution time).

Citron — User Interface

Citron is the web interface which allows users to create, execute, and monitor data flows. It can be used to choose predefined operations, drag and connect them through their ports to compose a data flow. Each operation's associated parameters may be inspected and flows can be instantiated by the user. Citron also allows users to track the execution of operations by showing their status in real time.

Juicer — execution and monitoring

Juicer is the module which runs the data flows and supports the monitoring of their execution. Upon receiving a dataflow, it generates the equivalent Spark source code or COMPSs source code, acting as a *transpiler* (source-to-source compiler), where each operation becomes a method.

The source code is then instantiated in the cloud execution environment, observing the user-defined QoS parameters to make sure operations execute with sufficient resources to meet user demands (*e.g.*, the user may indicate the number of compute nodes required). During execution, Juicer is also responsible for logging any new datasets produced as output, recording any runtime events, and reporting status changes.

Stand — Communication and Control

Stand coordinates the communication between Citron and Juicer, ensuring independence between the two components. When Citron requests an execution, it is enqueued by Stand to be consumed by Juicer. In the opposite direction, information about execution status and events produced in Juicer is provided. The current version implements asynchronous communication through *Redis* producer-consumer queues, for more information see about it: `https://redis.io/`.

Execution starts when a user requests to run a dataflow through the Citron interface. Citron then invokes Stand, which connects back to the first to provide feedback to the user. Stand, after receiving the request, adds it to the execution queue and begins to consume information about state changes of running tasks. Juicer consumes the execution queue and triggers executions in Spark clusters; it then reports the status of execution back to the Stand. Upon receiving a notification about the execution status of the tasks, Citron updates the user interface.

Thorn — Security and Privacy

Thorn is responsible for security, privacy, and access control (AAA) in Lemonade. When an user authenticates, he or she receives an access key to the platform. This key is used by other components to authorize the access to the platform's resources. At any time, other platform components may access Thorn through a secure SSL connection to guarantee the access key being provided by the user is valid.

Caipirinha — Results Visualization

Caipirinha provides data visualizations through different visual metaphors. Those include static data visualizations in well-defined formats, such as displaying samples from database records, time series graphs, histograms, maps, and exploratory views where users can parametrize the display according to their interests, such as zooming in on a particular region of a complex graph.

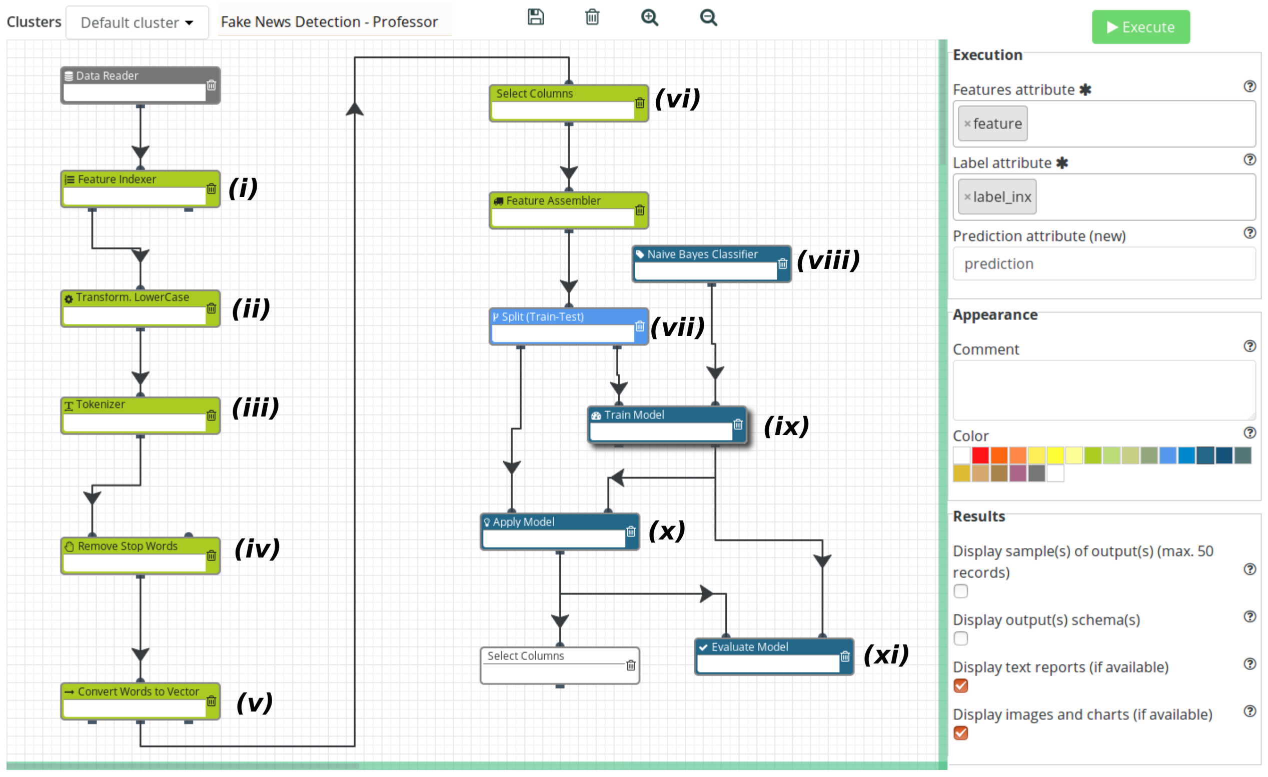
In addition to exploratory and static visualizations, Caipirinha also allows non-trivial visualization modeling, such as dynamic visualizations and the usage of visualizations to select new input data.

Integration

The aforementioned components communicate through well defined RESTful(Representational State Transfer) APIs. Figure 2 illustrates the various types of interaction.

In order to instantiate the user interface, Citron queries Tahiti to determine which operations, flows and forms (parameters) are defined (1). Citron interacts with Stand to feed it flow descriptions, and to receive informations about status changes and events about a given flow's execution (2). Stand then interacts with Juicer (3) to send the data flows descriptions with parameters, indicating the granularity of execution monitoring. On its turn, Juicer reports the events and the status of the execution back to the Stand, which will they relay them to Tahiti (2) for display.

While a flow is executing, Juicer reads and writes metadata about the data sources from/to Limonero (4). Limonero also provides Caipirinha which information needed to generate data views (5) and provides Citron with information about data sources to be displayed in the user interface (6). Citron also makes requests to Caipirinha to produce the



Execution

Features attribute ✱: × feature

Label attribute ✱: × label_inx

Prediction attribute (new): prediction

Appearance

Comment

Color

Results

Display sample(s) of output(s) (max. 50 records) ☐

Display output(s) schema(s) ☐

Display text reports (if available) ☑

Display images and charts (if available) ☑

Figure 1: Possible workflow to build a fake news classifier with Lemonade. Using lemonade, users process a dataset where some columns are raw text, create a classification model and use it in the processed data. Individual steps are explained in Section 3.

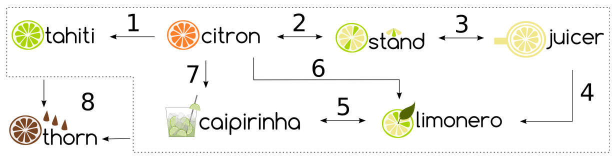


Figure 2: Interaction between Lemonade components.

views specified by users (7). All components interact with Thorn (8), which controls user access, provides API security keys for communication between modules and ensures privacy on data, flows and results.

Lemonade generates codes from the visual flows in the Python language, compatible with Spark version 2.2.0 and COMPSs 2.0. The Spark code is then executed by invoking `spark-submit` and each task in the flow generates intermediate results that are immediately consumed by subsequent tasks. The generated COMPs code is executed through a similar mechanism. At the end of an execution, Lemonade generates an execution report in JSON format.

3. DEMONSTRATION

3.1 Usage Scenario

The idea of the demo is to interactively create a data processing pipeline as the one depicted in Figure 1. Ideally, this would be done with a computer where an assistant would explain the system and allow others to interact and build a pipeline for a specific task which we selected.

The task proposed is: *given a dataset containing fake and normal newspieces, how can we build a model to detect those using Lemonade?* This is an interesting scenario, as it is an increasingly important task for fact-checking agencies: given a large volume of news-pieces that are created each day, how to assess which ones we should check? Notice that in a real world scenario, another aspect that would be evaluated is the reach of each one of these news, but we don't take this into consideration for our demo. In the upcoming subsections, we describe the pipeline users will be instructed to create, what users can do after creating this pipeline and briefly describe the dataset used in the demo.

3.2 Flow Description

We describe a possible pipeline which could be built interactively (with the aid of a supervisor). Each step corresponds to a data processing operation (a box) in Figure 1. For the sake of clarity we divide the explanation into two parts: data processing and model creation and validation.

Data processing. Given a tabular dataset, the user would: *(i)* index the target, the row in the dataset to be predicted, *(ii)* make the whole text lower case for the features of interest, *(iii)* tokenize the text, removing spaces and commas, *(iv)* remove stop-words such as *and*, *or* and *of*, *(v)* vectorize the text, which will allow it to be inputted to the machine learning algorithm of interest *(vi)* project some columns of the processed data, ignoring those which are not needed;

Model Creation and Validation. After the column projection, is as if we had a new dataset with the fields of interest. We then may proceed into creating and validating a model. The user could *(vii)* split these columns into training set and test set, choosing the proportion of the data which would go to each *(viii)* define a classification model (e.g. Naïve Bayes), *(ix)* train this model with the training set, *(x)* apply this model in the test set, and *(xi)* evaluate the model with metrics such as accuracy and recall.

After building this model with supervision, users would be encouraged to try to plug other boxes into the pipeline and change the execution parameters of the existing boxes, assessing its impact on the model result. While they are building the boxes, they would be encouraged to peek at how the data looks like in the different steps of the data processing.

3.3 Demo Dataset

Our demonstration uses Kaggle's *Getting Real about Fake News* dataset mixed with sources from real news-pieces, available on `https://www.kaggle.com/mrisdal/fake-news`. It contains text and metadata scraped from 244 websites tagged as "bullshit" by the BS Detector Chrome Extension. The main features we use in the classifier are the text of the body of the newspiece and its title.

4. CONCLUSION

We presented the Lemonade open source platform for building and running data mining and analysis flows on top of Spark and COMPSs. The system's core idea is to be accessible to users with different skills in distributed processing (from learners to experts), allowing users who aren't experts to perform data analytics tasks in an scalable and secure setting. We also present a demo where users would interactively create a pipeline with the aforementioned system, after understanding its key aspects, they would then be able to modify the pipeline adding new operations and changing execution arguments.

Our goal is to provide a complete and intuitive platform which allows users to perform complex data analytics tasks. Towards this end, the platform is being used successfully on several projects in Brazilian institutions, confirming the expectations that it may indeed provide domain experts with a tool to extract insight from large volumes of often sensitive data.

Potential future work includes the incorporation of streaming capabilities using open source projects (for example, Apache Kafka [8]), improves safety and quality of service (QoS) policies, the integration with other platforms for multidimensional data analysis like Ophidia [4], implementing a dynamic resource controller to guarantee the usage in cloud environments, and implementing support for libraries such as the Scikit Learn library (for more information about this package, see `http://scikit-learn.org/stable/`.

Acknowledgments

This work was partially funded by Fapemig, CNPq, CAPES, and by projects InWeb, MASWeb, INCT-Cyber, EUBra-BIGSEA, and Atmosphere.

5. REFERENCES

[1] V. S. Agneeswaran, P. Tonpay, and J. Tiwary. Paradigms for Realizing Machine Learning Algorithm. *Big Data*, 1(4):207–214, 2013.

[2] M. R. Berthold et al. KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, Nov. 2009.

[3] J. Demšar et al. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[4] S. Fiore et al. Ophidia: a full software stack for scientific data analytics. In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pages 343–350. IEEE, 2014.

[5] J. Kranjc et al. ClowdFlows: Online workflows for distributed big data mining. *Future Generation Computer Systems*, 68:38–58, 2017.

[6] Microsoft. Microsoft Azure: Machine Learning. https://azure.microsoft.com/pt-pt/services/machine-learning/, 2016. Visited on: 2018-02-12.

[7] I. Mierswa et al. YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of the 12th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 935–940, New York, NY, USA, 2006. ACM.

[8] N. Narkhede, G. Shapira, and T. Palino. *Kafka: The Definitive Guide Real-Time Data and Stream Processing at Scale.* O'Reilly Media, Inc., 1st edition, 2017.

[9] W. d. Santos, L. F. Carvalho, G. de P Avelar, Á. Silva Jr, L. M. Ponce, D. Guedes, and W. Meira Jr. Lemonade: A scalable and efficient spark-based platform for data analytics. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 745–748. IEEE Press, 2017.

[10] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[11] E. Tejedor and R. M. Badia. COMP Superscalar: Bringing GRID Superscalar and GCM Together. In *Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on*, pages 185–193. IEEE, 2008.

[12] The Economist. A brand new game. Available online at `https://goo.gl/k8uUYR`, 2015.

[13] The Guardian. Why big data means big business for online retailers. Available online at `goo.gl/vHR2hf`, 2012.

[14] The London School of Economics and Political Science. Backstage to the panama papers: big data analytics and collaborative journalism. Available online at `https://goo.gl/7PLZN8`, 2017.

[15] T. White. *Hadoop: The Definitive Guide.* " O'Reilly Media, Inc.", 2012.

[16] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. *HotCloud*, 10:10–10, 2010.