On the provenance extraction techniques from large scale log files

Alper Tufek, Mehmet S. Aktas

Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey

Correspondence
Mehmet S. Aktas, Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey.
Email: aktas@yildiz.edu.tr

Abstract
Numerical weather prediction (NWP) models are the most important instruments to predict future weather. Provenance information is of central importance for detecting unexpected events that may develop during the long course of model execution. Besides, the need to share scientific data and results between researchers also highlights the importance of data quality and reliability. The weather research and forecasting (WRF) Model is an open-source NWP model. In this study, we propose a methodology for tracking the WRF model and for generating, storing, and analyzing provenance. We implement the proposed methodology—with a machine learning-based parser, which utilizes classification algorithms to extract provenance information. The proposed approach enables easy management and understanding of numerical weather forecast workflows by providing provenance graphs. By analyzing these graphs, potential faulty situations that may occur during the execution of WRF can be traced to their root causes. Our proposed approach has been evaluated and has been shown to perform well even in a high-frequency provenance information flow.

KEYWORDS
machine learning-based provenance extraction, numerical weather prediction models, provenance, provenance analysis

1 | INTRODUCTION

The need for accurate and fast weather forecasts is on a constant rise. Mainly due to global warming, the number and impact of extreme weather events are gradually increasing. Today, the de facto method of weather forecasting for the near future is numerical models. These models simulate the atmosphere according to fluid dynamics and physics laws and try to calculate the future states at which the atmosphere will be. Numerical weather prediction (NWP) models calculate such parameters as temperature, pressure, wind speed, and so forth, by processing data primarily used for meteorological purposes, such as radar/satellite data and observation data gathered from weather observation stations. These models are usually run more than once every day at regular intervals. Therefore, the management of data quality, reusability, and reliability become more complex and challenging. In this respect, the need for systematic provenance is gaining importance, especially in scientific studies.¹

NWP models take input parameters such as the boundary information of the prediction domain and the resolution at which the predicted values are calculated. When the model starts to run, it usually takes hours to produce its results, depending on the input parameters. Most of the time, it is not possible to intervene in the course of model execution. To evaluate the correctness of the model outputs after its completion, it is of great importance to track the processing steps that took place during the generation process.

W3C consortium defines provenance in its PROV specification.² All of the entities, events, and persons that impact the process of generating a data product. In turn, this can be used to assess the quality and reliability of the data. Modifications to the data, the methods used in the production process, and metadata for reproducing the same data can be included in the definition of provenance.
In general terms, our research investigates the software architecture designs that can be used for collecting, analyzing, and storing the structured provenance information about an execution life cycle of an NWP model in real-time. In this study, the log files produced during model execution are selected as the real-time provenance source. Log files are usually free-form text files that may contain partially recurring patterns. In more technical terms, the problem can be defined as investigating the design of a software system that can identify log entries containing provenance information within these log files and parsing them into a structured provenance data structure.

In this work, the weather research and forecasting (WRF) model is preferred as the NWP model for the case study since it is open-source and has large community support. The WRF model is actively used for weather forecasts by meteorological organizations in many countries across the world. The WRF model software comprises several executable programs, each of which generates some particular log outputs. Other than that, there is no structured provenance generation or storage in any phase of a complete execution cycle. These raw log outputs are just free-form text lines containing various levels of information about the execution details.

This study proposes a distributed system software, an intelligent log parsing methodology, utilizing ML learning algorithms to mine provenance data from the log files. To illustrate the testing of the proposed software, we provide a prototype implementation. With the prototype implementation, we integrate the system with different ML algorithms. We argue that the distributed software architecture proposed in this study can be utilized in other e-science domains. The main contribution of this study is to address the lack of capability for provenance support in WRF model software and provide methodologies for machine learning-based ways of provenance collection. In this research, we analyze the WRF log files generated in the WRF model’s execution. We develop a machine learning-based parser, which utilizes classification algorithms for line filtering.

The article is organized as follows. Section 2 gives information about the fundamental concepts and provides a literature review. A brief overview of the WRF model and the PROV specification can be found in subsections 2.1.1 and 2.1.2, respectively. Section 3 explains detailed information about the proposed methodology for machine learning-based provenance parser. In Section 4, the implementation details of the prototype system are explained and the evaluation of system performance is discussed. Finally, in Section 5, the results obtained in the study are summarized.

2 | FUNDAMENTAL CONCEPTS AND LITERATURE REVIEW

In this section, we explain some fundamental concepts first. Next we give some background literature related to the subject.

2.1 | Fundamental concepts

In this work, we focus on provenance extraction from numerical weather forecast models with a special case study for WRF model. Before diving into technical details, it would be useful to give a preliminary information on the WRF model and provenance concept.

2.1.1 | The WRF model

The WRF model is an open-source software package consisting of various submodules divided into four groups:

1. WRF preprocessing system
2. WRF Model Core
3. WRF data assimilation (WRFDA)
4. Postprocessing

Depending on the scenario in which the model is to be used, different submodules must be run. The WRF model is primarily designed for two different scenarios. In the first scenario, called idealized cases, the aim is to perform various atmosphere simulations based on the state of the atmosphere at a given moment and some desired conditions. In the second type of scenario, which is called real cases, the aim is to make real-time weather predictions by using real data representing the current state of the atmosphere. In idealized cases, only the WRF Model Core and the required post-processing tools are required to be run, and the preprocessing component does not need to be executed. But if the WRF is to be used for real-time weather forecasting, it is necessary to first run the preprocessing and data assimilation (WRFDA) components if needed, then the Model Core and the necessary postprocessing tools accordingly.

Figure 1 illustrates the main components of the WRF model and the overall architecture of the provenance collection and presentation modules. The most commonly used scenario is to run the WRF model for real-time weather forecasting. For this purpose, the forecast domain must first be determined. Following this in the preprocessing stage, the static geographical data (GEOG_DATA) are interpolated to the model grid with the help of the geogrid.exe program. Meteorological data representing the current state of the atmosphere are then preprocessed by the ungrib.exe program.
The meteorological data are usually an output of a previous run of WRF or another prediction model and are mainly stored in GRIB format. The ungrib.exe program converts these data into the intermediate format that the WRF Model Core can understand. Next in the preprocessing stage, the metgrid.exe program is run, which interpolates the meteorological data to the model grid on the horizontal plane. The real.exe program is run at the end of the preprocessing. In this stage, vertical interpolation is performed according to the model grid, and the initial and boundary conditions are initialized. The WRF prediction engine (wrf.exe) is now ready to run to produce the final predictions.

2.1.2 PROV-O provenance standard

The PROV specification is a family of general-purpose documents recommended by the W3C consortium for modeling, representing, storing, and transferring of provenance data in a standard way independent of the discipline. While PROV-DM defines a basic data model for provenance data, PROV-N defines a provenance notation that people can understand. Besides, PROV-XML defines the framework of an XML schema so that provenance data can be stored and transferred in accordance with the PROV-DM data model, while PROV-O provides the necessary definitions to be able to create provenance ontologies by expressing the PROV data model with the help of OWL 2 Web Ontology Language.

Since the PROV specification is intended to provide a common provenance framework that is independent of a specific discipline, there are only three basic concepts and basic relationships that can be established between those concepts: Activity, Entity, and Agent.

According to the PROV specification, an entity can be anything physical, digital or conceptual, or a real or virtual thing. An activity is defined as anything that takes place in a given period of time and that carries out certain operations on entities. Operations such as processing, transforming, changing, using, or generating an entity are examples of activities. An agent is generally defined as anything that has certain responsibilities concerning entities or activities. An agent may be an entity or an activity.

2.2 Literature review

The scientific programs developed within the scope of scientific studies are generally developed by scientists from their own disciplines, so the priority of the developers is to produce algorithmically correct solutions to scientific problems. For this reason, scientific programs generally do not have an integrated provenance infrastructure.

Log analysis is one of the commonly used methods to extract provenance information. However, the quality of the provenance information produced in this method is both highly dependent on the level of detail of the log files and on what percentage of log lines containing provenance it can capture. In our earlier work, we proposed a rule-based log parser to extract provenance information from WRF log files. The parser utilized a rule database that consists of a list of special keywords to distinguish lines containing provenance information. These keywords were predetermined manually by examining various sample log files and the debug statements in the source code. In this study, we introduce a novel approach to filtering log files where line filtering is achieved by machine-learning methods. We make no modifications to the scientific source code. Our approach does not require a workflow orchestrator. We analyze log outputs and make inferences about the internal steps of the execution.
Simmhan et al. propose a general-purpose provenance collection framework in their work in 2006, which allows provenance information to be compiled from data-driven scientific workflows. They try to define the requirements for systems that collect data and workflow provenance. They also develop a standalone tool, Karma, as a prototype for the collection, representation, and storage of provenance data. Karma then evolves into the PROV compliant Komadu framework that is used in this study as the provenance storage backend. This provenance framework is tested on the linked environments for atmospheric discovery (LEAD) project by Droegemeier et al. LEAD is a meteorological research and training project that is Service-Oriented Architecture-SOA based and designed to enable operations such as access, preprocessing, assimilation, management, analysis, data mining, visualization, and so forth to be easily applied, independent of the format and the location of the data. Karma is workflow-oriented and needs a workflow orchestrator. Therefore, it needs each discrete event (workflow step) to be defined and implemented as an SOA-service. SOA-based architectures have been studied in detail in different studies such as References 9–11. In our study, we focus on numerical weather forecast models, particularly WRF, and make no modifications to the scientific source code. Our approach does not require a workflow orchestrator. We analyze log outputs and make inferences about the internal steps of the execution.

In 2013, Jensen et al. proposed a provenance framework to be used in the processing of satellite data. NOAA and NASA instrument data from satellites are beamed down to locations where they are gathered and then sent for processing. Jensen et al. used the Karma tool as the backend provenance storage and retrieval in their proposed framework and developed an adaptor to extract provenance-related activities from application log files. The Karma provenance system uses an extension of version 1.1 of the open provenance model (OPM) for its data model for external communication. Shu et al. conducted a similar case study on the modeling and analysis of provenance data in hydrological models. They present a provenance model for the representation of provenance information in streamflow forecasting. For this purpose, they extend the OPM to satisfy the requirements for their case. There exist various other provenance-based systems utilizing the Karma tool. In our study, the provenance representation and data model are fully compatible with the W3C consortium’s PROV specification, which defines a common provenance framework that is independent of a specific domain.

However, to the best of our knowledge, weather prediction/atmosphere modeling systems that are run either on a global or a regional scale by meteorological organizations or by universities or research institutions within the scope of scientific research or weather forecasting are not capable of producing, storing and analyzing systematic provenance records. The global forecast system (GFS) (link: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs) is a nonopen source NWP system that includes a global model run by the United States’ National Weather Service. It is workflow-based and composed of multiple workflow components (data assimilation, forecast model, postprocessing, and so forth). Bernardet al. proposed an infrastructure, NWP Information Technology Environment—NITE, for scientists to configure, launch, and track experiments with various NWP models including GFS. The main goal is to record the provenance of codes, scripts, and configuration files, and inputs related to an experiment, so that it can be reviewed and reproduced. ECMWF’s Integrated Forecasting System (https://www.ecmwf.int/en/research/modelling-and-prediction) has its own workflow management system, ecFlow. Each workflow must be defined as task suites.

In this study, we have designed a provenance/trackingsystem for the open-source WRF model that is used by most meteorological organizations in different countries across the world. The log files produced during the execution of the WRF model are analyzed, and lines containing provenance information are filtered in the first stage. In the second stage, the corresponding provenance notifications are generated and recorded in a provenance database in the background according to the information in the filtered lines. In our earlier work, we proposed a rule-based log parser to extract provenance information from these log files. The parser utilized a rule database that consists of a list of special keywords to distinguish lines containing provenance information. In this study, we introduce a machine learning based log parsing methodology to extract provenance from the log files.

There are studies on automatic and intelligent log analysis in the literature. Still, these studies mainly focus on the early detection of certain types of events by utilizing various classification or clustering algorithms. For example, by analyzing network logs, new features (number of logins per hour, number of failed logins per hour, and so forth) can be generated. A machine-learning model based on these features can be trained to classify an activity as harmful or usual. These studies only focus on detecting certain types of events or predicting certain activities in the log files. They do not mine the relationships between these events and construct causal relationships. Unlike these studies, we propose a distributed system-based software architecture, designed for mining provenance notifications that can be used to build provenance graphs.

Text classification by machine-learning algorithms has been used in countless areas such as search engines, social media platforms, indexing, and emotion analysis in texts. We utilize various text classification algorithms inside the machine learning-based log parser. This way, our tracking and provenance analysis tool can run on different log files when filtering the lines containing provenance in WRF log files without the need for a rule base.

3 | PROPOSED METHODOLOGY

There are various approaches to obtaining provenance information. The first is the manual labeling approach which is not effective since it requires a high amount of labor and time. It is also error-prone because it is human-handled. The second approach is to modify the source code to make it produce provenance records automatically. However, the disadvantages of this method are the lack of access to the source code of the software at
hand, the need to recompile the code after the changes to the source code, as well as the additional errors this may cause. A third method sometimes referred to as scavenging, is to examine sources such as various log files that are generated during the execution of programs and to extract provenance information from these sources. Even if it may lack a configurable debug level setting or enough information for a complete provenance, this approach is more applicable to most use cases than the other two methods.

In our previous work, we introduced an alternative approach that utilized both the scavenging method and the instrumentation of the shell script files. These are external shell scripts that are not part of the WRF software. They just invoke the required WRF components and insert the related provenance information into the log file. We implemented a rule-based provenance extractor that analyzes raw log records by scanning each line and checking whether it contains any of the predefined keywords inside the rules stored in a rule configuration file. These keywords were predetermined by manually examining sample log files and the source code of the WRF scientific program modules. In Table 1, we illustrate this with an example. In the source code, wrf_debug procedure calls give useful information, such as one procedure invokes another one or opens a file for input/output. Source comments are also useful for interpreting the raw logs. The main drawback of this method is that the rules must be maintained manually by the programmer to adapt the parser to different log files. The rule-based provenance extraction systems rely on the rules that contain predetermined keywords. These systems rely on the fact these predetermined keywords must appear as-is in the log entries. Rule-based provenance extraction systems assume that the log files always have the correctly recorded log entries, or there could be no typos in the log entries, and there could be no improperly functioning logging tool. However, when analyzing the log files, we observe that there exist such cases. With the proposed ML-based provenance mining software, we intend to create training data that includes properly constructed log entries and improperly constructed log entries. We manually label both correctly constructed log entries and log entries that are missing letters or that have spelling mistakes. Then we build ML models that learn from a variety of log entries. We use these models to predict the labels of new log entries for provenance notification mining. The rule-based provenance extraction systems do not detect the class labels for those improperly constructed log entries.

In this study, we investigate the use of supervised learning algorithms to model the provenance data and predict the type of provenance notifications. Here, machine-learning algorithms are employed during the analysis of log lines produced by the WRF model. In other words, lines containing provenance information are predicted by using text classification methods. Figure 2 shows the component diagram of the machine learning-based provenance collection methodology that we have developed. To illustrate the testing of the machine learning algorithms, the following supervised learning algorithms are used to classify lines containing provenance information: Logistic Regression, Naive Bayes, Random Forest, and Multilayer Perceptron.

### TABLE 1 Summary statistics of log files used in experiments

| Rule Id | Formal representation | Description |
|---------|-----------------------|-------------|
| Rule1   | ON PATTERN (→ Action on (“wrf_debug” and “open”) | “wrf_debug” open procedure call gives information, such as one procedure opens a file for input/output |
|         | DO ACTION (Label as Notification#2)         |             |
| Rule2   | ON PATTERN (→ Action on (“wrf_debug” and “invoke”) | “wrf_debug” invoke procedure call gives useful information, such as one procedure invokes another one. |
|         | DO ACTION (Label as Notification#3)         |             |

![Machine learning-based approach to provenance collection from WRF modules](image)
Data preprocessing: Using N-gram frequency profiles, one can provide a simple data representation to categorize text files for a wide range of classification tasks. N-gram frequency profiles are a commonly used approach in text classification. An N-gram is usually referred to as an N-character slice of a longer string. To illustrate testing of the text classification for provenance data, in this study, we simply use N-word frequency profiles and take into account 1-word string for data representation.

Feature indexes in "feature_index:value" pairs indicate the index numbers that are automatically assigned, starting from 1, to each different word in the log file in the order they are encountered. The value number in "feature_index:value" pairs indicates the frequency of the word in the log line. In this study, the value part is assigned as 1 in all samples, since only the term existence is considered rather than the term frequency in the scope of the study.

Training dataset: Note that, classification algorithms require a training dataset to construct classification models. In this study, we created a labeled dataset for WRF log files. We created a training dataset by scanning each line and checking whether it contains any of the provenance data and determine the type of provenance relationship. The training dataset is constructed by manually examining sample log files obtained from the WRF scientific program modules.

The class labels start from zero, indicating an irrelevant line, and will go up to the total number of different types of provenance relationships, incrementing by one. We use the following possible provenance relationships as labels according to the PROV-O Specification: used, wasGeneratedBy, wasAssociatedWith, wasInformedBy, wasAttributedTo, wasDrivenFrom, actedOnBehalfOf.

Model construction: Model construction is performed with a training set of log files before the system is deployed. The log file is given as input to the machine learning algorithms to train classification models. In the scope of the study, we constructed various machine-learning models by using Logistic Regression, Naive Bayes, Random Forest, Multilayer Perceptron, Decision Tree, and SVM algorithms. We should note that, in this study, we introduce an architecture of distributed machine-learning-based log parsing software, which is designed to mine provenance from log files (which may contain both proper and improper log entries) by utilizing big data processing technologies. To facilitate the usability of the architecture, we implement a prototype, which can employ and run various ML algorithms using big data processing libraries. We should note that we are not particularly investigating the best possible ML algorithm for provenance mining in this study. We are interested in showing the proposed system can employ different ML algorithms in a scalable way. Hence, we use different ML algorithms on the map-reduce model-based big data processing libraries.

Provenance notification prediction: The classification process for the new WRF log lines is performed based on the constructed models. Each model predicts one class label from the available multiclass labels. After the prediction phase, the Adaptor constructs a provenance notification with the appropriate provenance relationship and sends it to the provenance repository. We discuss the evaluation of the prediction tasks in Section 4.

The proposed approach can be used in the same way to analyze the log files of different NWP models other than the WRF model, without requiring software development.

4 PROTOTYPE IMPLEMENTATION AND PERFORMANCE EVALUATION

In this work, we propose a machine learning-based log parser and provenance extractor for the WRF NWP software. This component is positioned between WRF and a backend provenance repository. For the repository, we used a PROV-O compatible provenance storage technology, Komadu Service. To illustrate the testing of the proposed system, we developed a prototype. In the following subsections, we give some preliminary information, investigate the performance of the prototype, and discuss the results. For this study, the Turkish State Meteorological Service provided us atmospheric data input and the computational facilities for running the WRF model.

4.1 Training datasets

Our proposed parser and provenance extractor reads WRF execution logs and generates structured provenance notifications according to a pre-trained machine-learning model. For testing purposes, we collected five different log files having an increasing number of lines. The first file is generated by the WRF model executed with debug_level 100 while the second and the third files are generated with debug_level 150. The fourth and fifth files are produced by replicating the contents of the third file three and fifteen times, respectively. The last two files are used only for evaluating training times. The summary statistics of these log files are given in Table 2. 'Dictionary size' in the table refers to the total number of distinct words in the corresponding file. Since the contents of the fourth and fifth files are the replication of the third one, values of the average number of words per line and dictionary size are the same for all three.

[1]https://www.mgm.gov.tr
TABLE 2  Summary statistics of log files used in experiments

| File name  | lvl_100 | lvl_150 | lvl_150_v2 | lvl_150_v3 | lvl_150_v4 |
|------------|---------|---------|------------|------------|------------|
| Debug level|         |         |            |            |            |
| 100        | 150     | 150     | 150        | 150        | 150        |
| Total number of rows | 28,159 | 83,444 | 141,831    | 425,493    | 2,127,465  |
| Total number of words | 121,246 | 898,786 | 1,716,024  | 5,148,072  | 25,740,360 |
| The average number of words per line | 4.30 | 10.77  | 12.10      | 12.10      | 12.10      |
| Dictionary size | 842 | 3235   | 3283       | 3283       | 3283       |

4.2 | Apache Spark framework

For the scalability tests of the system, a working prototype is deployed on an Apache Spark cluster. Apache Spark is an open-source parallel processing framework that is written in Scala programming language and it also provides APIs for Java, Python, and R languages. At its core, Spark provides a data structure called resilient distributed dataset which is an immutable set of elements distributed over a cluster of nodes and can be processed in parallel. The main advantage of Spark is that it provides a native in-memory computation facility that makes it a faster and more-suited platform for iterative processing such as machine-learning tasks. On top of the core, Apache Spark has the MLlib (Machine Learning Library) library which contains the implementations of well-known machine-learning algorithms from various categories. The classification algorithms in the MLlib library accept files of LIBSVM format as input. Hence raw log files must be converted to LIBSVM format first. LIBSVM is a sparse feature vector notation, a file format with rows representing a feature vector that is composed of “attribute_index:value” pairs separated by a space character. To convert raw log lines to feature vectors, we use the Apache Spark machine-learning library’s CountVectorizer and CountVectorizerModel classes. Then, static class labels are determined as the ground truth for multiclass classification by using the ruleset created for our previous rule-based parser.

4.3 | Evaluated classification algorithms

The proposed system simply utilizes various classification algorithms to train a model for filtering out log lines containing provenance information. To evaluate the effect of algorithm selection, we picked up a total of six representative algorithms from different categories. We referred to a recent work of Brownlee’s for the categorization of classification algorithms and chose six different categories to pick up a representative algorithm from each: regression algorithms, instance-based algorithms, decision tree algorithms, Bayesian algorithms, artificial neural network algorithms, and ensemble algorithms.

Regression algorithms are basically statistical methods for estimating the relationships between a dependent variable (class label) and one or more independent variables (also called “predictors,” or “features”). Among regression algorithms, we picked up the Logistic Regression algorithm for its simplicity and widespread use. There are two different versions of Logistic Regression implemented in Apache Spark’s MLlib library: LogisticRegressionWithLBFGS, and LogisticRegressionWithSGD. The latter is a binary Logistic Regression algorithm using stochastic gradient descent (SGD) whereas the former can be used for both multilabel and binary classification. We preferred LogisticRegressionWithLBFGS implementation since it supports multiclass classification. Besides, LogisticRegressionWithLBFGS uses Limited-memory BFGS optimization which is especially suitable for optimization problems with many variables (features).

Instance-based algorithms represent each example (observation) as a point in a d-dimensional space, where d is the number of features. Algorithms in this group use a proximity measure (either similarity or dissimilarity) such as Euclidean distance, Jaccard, cosine similarity, and so forth. From this group of algorithms, we picked support vector machines (SVM) which simply find support vectors to define the boundaries between classes. Like regression algorithms, SVM also has its roots in statistical learning and works very well with high-dimensional data since it avoids the curse of dimensionality problem. In the MLlib library, SVM is implemented with the name SVMWithSGD which uses the SGD algorithm for optimizing the margins (distances between the lines/hyperplanes and the nearest points) such that the nearest point in either class is as far away from the separating line/hyperplane as possible.

Bayesian algorithms apply the Bayes’ rule for solving classification problems by modeling probabilistic relationships between the features and the class variable. We preferred the Naive Bayes algorithm which is implemented in the MLlib library with the same name and it supports both multinomial naive Bayes (multiclass) and Bernoulli naive Bayes (binary).

Decision tree algorithms construct a model represented as a tree, which is a hierarchical structure consisting of internal (decision) nodes and leaf nodes. Each leaf node is assigned a class label and nonleaf nodes contain attribute test conditions to separate records by their different characteristics. Starting from the root node, the test conditions in internal nodes are applied to the test records and the appropriate branches are followed based on the outcomes of the tests until reaching a leaf node where the class label associated with that leaf is then assigned to the test record. The
MLlib library’s decision tree implementation uses the Iterative Dichotomiser 3 (ID3) algorithm with classification and regression tree (CART). ID3 can only handle categorical variables whereas CART can handle continuous variables. We used the Gini index as the impurity measure to calculate information gain.

Ensemble algorithms simply learn many weaker models that are independently trained and the final prediction is made by combining these weaker models’ predictions in some way to reduce the possibility of overfitting. Among the algorithms in this group, we chose the Random Forest which is an ensemble of decision trees. The MLlib library contains an implementation of the Random Forest classifier with the Gini index as the default impurity measure.

The final category of algorithms is artificial neural networks. We picked up the multilayer perceptron implementation in the MLlib library as the representative algorithm of the category since it is suitable for multiclass classification. We configured a simple network with two hidden layers consisting of five and four neurons, respectively. The number of neurons in the input layer is equal to the dictionary size for each log file whereas the number of neurons in the output layer is six for all log files where six is the total number of possible provenance relationships.

4.4 The scalability tests

All scalability tests are run on an Apache Spark cluster with a various number of active worker nodes, ranging from just a single node up to three worker nodes. The Spark cluster consists of a total of four identical virtual machines, one of which is the master node and the other three are worker nodes. Each virtual machine is configured to have an Ubuntu 18.04.3 operating system installed with 1 CPU core and 2 GB RAM. The physical machine hosting those virtual machines has 16 GB of system memory and an Intel Core i7-8750H processor consisting of six physical cores with a 2.20 GHz base frequency (12 virtual CPUs with hyperthreading enabled).

The most time-consuming part of the deployment is model training. To see the scalability of the proposed system, we evaluated the training times as the size of the log files grows. For each algorithm-log file-worker nodes combination, a test pipeline is submitted to the Spark cluster which is repeated 100 times, and the average training times are calculated along with the SDs. The results are summarized in Figure 3.

We also investigated inference performance for each algorithm. We run the inference capability of each algorithm for varying sizes of test data. For this purpose, a model was trained for each algorithm with lvl_150_v3 log file, then using this model, the entire lvl_150_v4 file was classified, and the total inference times were measured. For each algorithm-log file combination, we measure the performance of the inference on a single node, two and three nodes. We repeat the tests 100 times and record the average times and SDs. The results are summarized in Figure 4.

4.4.1 Results for scalability experiment

When analyzing the results in Figure 4, the first thing in the test results that attract the attention is that the algorithm with the longest training time is the ANN-based Perceptron algorithm. The average training time of the algorithm on a single node for the log file lvl_150_v4 containing more than 2M lines is around 3913 s. This corresponds to a little more than 1 h. For the same algorithm and log file, when the number of nodes is increased to 2 and 3, training times decrease to 2318 (38 min) and 1816 (30 min) s, respectively. On the other hand, the algorithm with the shortest training time is Naive Bayes. The average training time of the algorithm for the largest log file on a single node is 34.5 s. This time is reduced to 20.8 and 16.2 s for two and three nodes, respectively.

The test results for inference capability performance analysis are shown in the graph in Figure 4. The first thing that draws attention to the graph is that the shortest inference time belongs to the Perceptron algorithm. Although its training time is the longest, it can make inferences quickly. For instance, it can classify the largest log file, which contains more than two million lines in only 7.3 s on average, even on a single node. On the other hand, the longest inference time belongs to the Naive Bayes algorithm, which can classify the entire log file on a single node in an average of 47.5 s. In other words, it can classify a single line in approximately 22.3 μs on average, which is more than six times of Perceptron algorithm.

4.4.2 Discussion

Figure 3 shows that with map-reduce programming model-based implementations of the machine learning algorithms, the proposed system can scale to processing large-scale log files for both the training ML models and the inference functionalities. The results show that adding more nodes reduces the average training execution time. We observed the same behavior for all algorithms. To this end, we can argue that the proposed system is designed to mine provenance from large-scale log files.

When the increases in the training times compared with the growth rates of the log files for all algorithms are examined, it can be seen that there is a linear relationship in general. When the number of records increases by three or five times, the training time also increases at approximately the same rates. As the number of nodes increases, the slope of the increase in training times decreases.
Figure 3: Training times for various classification algorithms

Figure 4: Inference timings for tested algorithms
Even though the Naive Bayes algorithm has the shortest model training time, it performs worst when making inferences with that model. There is a tradeoff in terms of inference time versus training time. Figure 3 also shows that adding additional nodes reduces inference times almost linearly.

4.5 | The classification test

The proposed system’s classification performance in terms of accuracy and precision/recall metrics is also evaluated. Each test procedure is repeated 100 times for each algorithm-log file combination, and the average accuracy and precision/recall metrics are calculated together with their SDs. A 10-fold cross-validation approach is chosen for the evaluation methodology. Each log file is randomly split into 10 subsets of approximately equal size. Each time, a different subset is selected as the test set, and the remaining subsets are used for training. The overall performance metric for a specific log file is calculated by taking the average of the results calculated for each one of the 10 subsets.

4.5.1 | Results for the classification tests

We evaluated all the algorithms’ multiclass classification performance in terms of accuracy, precision, and recall metrics, and the results can be seen in Figure 5. We notice that all algorithms seem to have achieved very high classification accuracies and performed with high precision and high recall in general.

4.5.2 | Discussion

The size and the contents of the log files produced by the WRF model may show some variations depending on the parameters, such as the size of the region to be predicted or the length of the prediction period. However, when different log files are examined, it can be seen that they generally have

---

**FIGURE 5** | Accuracy, precision, and recall results for multiclassification
a common pattern and a high degree of similarity. For this reason, it is observed that the machine-learning models achieve very high performance on various log files obtained after running the WRF model with different initial parameters, such as time periods or prediction regions.

Among all six algorithms, we can say that the SVM algorithm has the lowest classification performance (in terms of all metrics). Although Naive Bayes performs better than SVM, it performs relatively worse than other algorithms. Decision tree, logistic regression, perceptron, and random forest algorithms have close performances to each other’s and they all perform well in terms of accuracy, precision, and recall.

5 | CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This study proposes a distributed system, an intelligent log parser utilizing ML learning algorithms to mine provenance data from the log files. To facilitate testing of the system, we developed a prototype implementation that uses Apache Spark’s MLlib library. With the prototype implementation, we integrate the system with different ML algorithms. We picked a representative algorithm from each of six different supervised machine learning categories: regression algorithms, instance-based algorithms, decision tree algorithms, Bayesian algorithms, artificial neural network algorithms, and ensemble algorithms. The trained models are run on the sample log files, and it is observed that they perform well even on log files containing a large number of lines. The results show that the proposed approach can scale to large-scale log files for provenance mining. We argue that the distributed software architecture proposed in this study can be utilized in other e-science domains.

In proposed approach, supervised learning algorithms are used to model the provenance data and predict the type of provenance relationships. The supervised machine-learning algorithms are employed during the analysis of log lines produced by the WRF model. To obtain different provenance relationships from the log lines, we utilize multilabel classification.

The results indicate that successful provenance extraction can also be conducted by utilizing machine-learning algorithms without the need for a ruleset. Hence, the use of machine-learning algorithms for log parsing for provenance can eliminate the need for a rule database.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Alper Tufek © https://orcid.org/0000-0003-1279-9318
Mehmet S. Aktas © https://orcid.org/0000-0001-7908-5067

REFERENCES

1. Simmhan YL, Plale B, Gannon D. A survey of data provenance in E-science. SIGMOD Rec. 2005;34(3):31-36. https://doi.org/10.1145/1084805.1084812

2. Missier P, Belhajjame K, Cheney J. The W3C PROV family of specifications for modelling provenance metadata. Paper presented at: Proceedings of the 16th International Conference on Extending Database Technology, Genoa, Italy; 2013:773-776; Association for Computing Machinery.

3. Groth P, Moreau L. PROV-overview: an overview of the PROV family of documents; 2021. Accessed March 01, 2021. https://www.w3.org/TR/prov-overview

4. Tufek A, Gurbuz A, Ekuklu OF, Aktas MS. Provenance collection platform for the weather research and forecasting model. Proceedings of the 14th International Conference on Semantics, Knowledge and Grids (SKG) SKG ‘18. IEEE; 2018: 17-24.

5. Tufek A, Aktas MS. On the provenance extraction techniques from large scale log files. Proceedings of the Euro-Par 2020: Parallel Processing Workshops Euro-Par. Springer; 2020.

6. Indiana University Data to insight center (D2I). Karma: Provenance collection and visualization tool; 2021. Accessed March 01, 2021. https://pti.iu.edu/impact/open-source/karma.html

7. Indiana University Data to insight center (D2I) Komadu: Provenance collection and visualization tool based on W3C PROV standard; 2021. Accessed March 01, 2021. https://github.com/Data-to-Insight-Center/komadu

8. Droegemeier K. Linked environments for atmospheric discovery (LEAD): architecture, technology roadmap and deployment strategy. Paper presented at: Proceedings of the 21st Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, San Diego, CA; 2005.

9. Aktas MS, Fox GF, Pierce M, Oh S. XML metadata services. Concurr Comput Pract Exper. 2008;20(7):801-823. https://doi.org/10.1002/cpe.1276

10. Aktas MS, Pierce M. High-performance hybrid information service architecture. Concurr Comput Pract Exper. 2010;22(15):2095-2123. https://doi.org/10.1002/cpe.1557

11. Aktas MS, Fox GC, Pierce M. Information services for dynamically assembled semantic grids. Paper presented at: Proceedings of the 1st International Conference on Semantics, Knowledge and Grid. Beijing, China; 2005. https://doi.org/10.1109/skg.2005.83.

12. Jensen S, Plale B, Aktas MS, Luo Y, Chen P, Conover H. Provenance capture and use in a satellite data processing pipeline. IEEE Trans Geosci Remote Sens. 2013;51(11):5090–5097.

13. Moreau L, Clifford B, Freire J, Futrelle J, Gil Y, Groth P, Kwasiowska N, Miles S, Missier P, Myers J, Plale B, Simmhan Y, Stephan E, Bussche J. The open provenance model core specification (v1.1). Future Gener Comput Syst. 2011;27(6):743–756.

14. Shu Y, Taylor K, Hapuarachchi P, Peters C. Modelling provenance in hydrologic science: a case study on streamflow forecasting. J Hydroinform. 2012;14(4):944–959.

15. Baeth M, Aktas MS. Detecting misinformation in social networks using provenance data. CCPE J. 2019;31(3):1–13.

16. Baeth M, Aktas MS. An approach to custom privacy policy violation detection problems using big social provenance data. CCPE J. 2018;30(21):1–9.
17. Riveni M, Nyugen T, Aktas MS, Dustar S. Application of provenance in social computing: a case study. CCPE J. 2019;31(3):1–14.
18. Barnardet L, Carson L, Tilapragada V. The design of a modern information technology infrastructure to facilitate research-to-operations transition for NCEP’s modeling suites. Bull Am Meteor Soc. 2017;98(5):899–904.
19. Yen S, Moh M. Intelligent log analysis using machine and deep learning. Machine Learning and Cognitive Science Applications in Cyber Security. IGI Global; 2019:154-189.
20. Sun Y, Guo S, Chen Z. Intelligent log analysis system for massive and multi-source security logs: MMSLAS design and implementation plan. Proceedings of the 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN). IEEE; 2019:416-421.
21. Zhu J, He S, Liu J, et al. Tools and benchmarks for automated log parsing. Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP); 2019:121-130.
22. He P, Zhu J, He S, Li J, Lyu MR. Towards automated log parsing for large-scale log data analysis. IEEE Trans Depend Secure Comput. 2018;15(6):931–944. https://doi.org/10.1109/TDSC.2017.2762673
23. McCallum A, Nigam K, Rennie J, Seymore K. Building domain-specific search engines with machine learning techniques. Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace; 1999 AAAI Spring Symposium; 1999:28-39.
24. Boyan J, Freitag D, Joachims T. A machine learning architecture for optimizing web search engines. Proceedings of the AAAI Workshop on Internet Based Information Systems; 1996:1-8.
25. Agichtein E, Castillo C, Donato D, Gionis A, Mishne G. Finding high-quality content in social media. Proceedings of the 2008 International Conference on Web Search and Data Mining. Association for Computing Machinery; 2008:183-194.
26. Neethu MS, Rajasree R. Sentiment analysis in twitter using machine learning techniques. Proceedings of the 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT); 2013:1-5.
27. Pang B, Lee L, Vaithyanathan S. Thumbs up? sentiment classification using machine learning techniques. CoRR. 2002;10cs.CL/0205070:79–86.
28. Brownlee J. A tour of machine learning algorithms, 2020. Accessed March 01, 2021. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/
29. Nocedal J. Updating quasi-Newton matrices with limited storage. Math Comput. 1980;35(151):773-782. https://doi.org/10.1090/S0025-5718-1980-0572855-7

How to cite this article: Tufek A, Aktas MS. On the provenance extraction techniques from large scale log files. Concurrency Computat Pract Exper. 2023;35(15):e6559. https://doi.org/10.1002/cpe.6559