Cloud-Based Machine Learning for Predictive Analytics: Tool Wear Prediction in Milling

Dazhong Wu, Connor Jennings, Janis Terpenny, Soundar Kumara
Department of Industrial and Manufacturing Engineering
The Pennsylvania State University
University Park, PA 16802, USA
{dxw279, connor, jpt5311, skumara}@psu.edu

Abstract—**The proliferation of real-time monitoring systems and the advent of Industrial Internet of Things (IIoT) over the past few years necessitates the development of scalable and parallel algorithms that help predict mechanical failures and remaining useful life of a manufacturing system or system components. Classical model-based prognostics require an in-depth physical understanding of the system of interest and oftentimes assume certain stochastic or random processes. To overcome the limitations of model-based methods, data-driven methods such as machine learning have been increasingly applied to prognostics and health management (PHM). While machine learning algorithms are able to build accurate predictive models, large volumes of training data are required. Consequently, machine learning techniques are not computationally efficient for data-driven PHM. The objective of this research is to create a novel approach for machinery prognostics using a cloud-based parallel machine learning algorithm. Specifically, one of the most popular machine learning algorithms (i.e., random forest) is applied to predict tool wear in dry milling operations. In addition, a parallel random forest algorithm is developed using the MapReduce framework and then implemented on the Amazon Elastic Compute Cloud. Experimental results have shown that the random forest algorithm can generate very accurate predictions. Moreover, significant speedup can be achieved by implementing the parallel random forest algorithm.**

Keywords-prognostics and health management; machine learning; cloud computing; tool wear prediction

I. INTRODUCTION

Almost all engineering systems (e.g., aerospace systems, nuclear power plants, and machine tools) are subject to mechanical failures resulting from deterioration with usage and age or abnormal operating conditions [1-3]. Some of the abnormal operating conditions include wear, corrosion, high temperature, high pressure, vibration, buckling, and fatigue. The degradation and failures of engineering systems or system components will often incur higher costs and lower productivity due to unexpected machine down time. In order to increase manufacturing productivity while reducing maintenance costs, it is crucial to perform a maintenance strategy that allows manufacturers to schedule production shutdowns for repairs, inspection, and maintenance.

Conventional maintenance strategies include reactive, preventive, and predictive maintenance [4-6]. The most basic approach to maintenance is reactive, also known as run-to-failure maintenance planning. In the reactive maintenance strategy, assets are deliberately allowed to operate until failures actually occur. The assets are maintained on an as-needed basis. One of the disadvantages of reactive maintenance is that it is difficult to anticipate maintenance resources (e.g., manpower, tools and replacement parts) will be needed for repairs. In preventive maintenance, systems or components are replaced based on a conservative schedule to prevent commonly occurring failures. Although preventive maintenance allows for more consistent and predictable maintenance schedules, it is expensive to implement preventive maintenance because of frequent replacement of components or parts before their end-of-life. To reduce the high costs of preventive maintenance, predictive maintenance is an alternative strategy in which maintenance actions are scheduled based on equipment performance or conditions instead of time. The objective of predictive maintenance is to determine the condition of in-service equipment, and ultimately to predict the time at which a system or a component will no longer meet desired functional requirements.

The discipline that predicts health condition and remaining useful life based on previous and current operating conditions is often referred to as PHM. Classical prognostic approaches fall into two categories: model-based and data-driven prognostics [7-12]. Model-based prognostics refers to approaches based on mathematical models of system behavior derived from physical laws or probability distribution. For example, conventional model-based prognostics include methods based on Wiener and Gamma processes [13], hidden markov models [14], Kalman filter [15], and particle filter [16]. One of the disadvantages of model-based prognostics is that an in-depth understanding of the underlying physical processes that lead to system failures is required. Another disadvantage is that it is assumed that underlying processes follow certain probability distribution such as gamma or normal distributions.

In comparison with model-based prognostics, data-driven prognostics refers to approaches that build a predictive model using a learning algorithm and large volumes of historical data. For example, classical data-driven prognostics include approaches based on autoregressive model, multivariate adaptive regression, fuzzy set theory, and artificial neural networks (ANNs). The unique benefit of data-driven methods is that an in-depth understanding of system physical behaviors is not required. In addition, data-driven methods do not assume any underlying probability distributions. While a few machine learning algorithms such as ANNs and decision trees have been applied in the area of tool wear prediction, little research has been reported on the parallel implementation of machine learning algorithms on the cloud in the context of manufacturing [17]. To address the research gap, we developed a cloud-based parallel random forest algorithm to predict tool wear using two experimental data sets. The performance of the random forest algorithm is measured using accuracy and training time. The advantages of random forests [18] are: it is one of the most accurate machine learning algorithms; it runs efficiently on large datasets; it handles a large number of input variables (i.e., predictors) without variable selection; feature importance is estimated during training; and cross validation is not required because random forests generate an internal unbiased estimate of the generalization error as the forest building progresses.

Moreover, because machine condition monitoring systems generate large volumes of measurement data, it is extremely challenging to design and implement efficient and scalable data-driven approaches that are capable of processing large volumes of historical data or high speed streaming data on a multi-core processor and/or a cluster. In order to benefit from multi-core processors and high performance computing clusters, it is important to parallelize the data-driven algorithms. To address this research gap, a novel PRF machine learning algorithm is implemented on a public cloud based on the MapReduce framework. It should be noted that the objective of this paper is to investigate the performance of the random forest algorithm and its parallel implementation using the MapReduce paradigm. Due to this reason, the comparison of random forests with other machine learning algorithms such as ANNs is not conducted.

The main contributions of this paper include:

- A parallel random forest (PRF) algorithm is developed based on the MapReduce framework and implemented on a single machine with multiple cores in a high performance computing cloud.
- The performance of the PRF algorithm is compared with that of the random forest algorithm implemented in serial. The speedup and scalability of the PRF are evaluated using two training data sets.

The remainder of the paper is organized as follows: Section 2 reviews the related literature on data-driven prognostics. Section 3 introduces the theoretical background of the random forest algorithm and a PRF implementation based on the MapReduce framework. Section 4 presents the methodology for data-driven prognostics for tool wear prediction using the MapReduce-based PRF algorithm. Section 5 presents an experimental setup, an experimental data set acquired from different types of sensors on a CNC milling machine, and experimental results. Section 6 provides conclusions that include a discussion of research contribution and future work.

II. DATA-DRIVEN PROGNOSTICS

Schwabacher and Goebel [19] conducted a review of data-driven methods for prognostics. The most popular data-driven approaches to prognostics include ANNs and decision trees in the context of systems health management. ANNs are a family of computational models based on biological neural networks which are used to estimate complex relationships between inputs and outputs. Chungchoo and Saini [20] developed an online fuzzy neural network (FNN) algorithm that estimates the average width of flank wear and maximum depth of crater wear. A modified least-square backpropagation neural network was built to estimate flank and crater wear based on cutting force and acoustic emission signals. Chen and Chen [21] developed an in-process tool wear prediction system using ANNs for milling operations. A total of 100 experimental data were used for training the ANN model. The input variables include feed rate, depth of cut, and average peak cutting forces. The ANN model can predict tool wear with an error of 0.037mm on average. Ozel and Karpat [22] developed a predictive model for tool flank wear and surface roughness in finish dry and turning operations using feedforward neural networks and regression. Based on experimental results, predictive neural network models provided more accurate predictions than regression models. Bukkapatnam *et al.* [23-25] developed effective tool wear monitoring techniques using ANNs based on features extracted from the principles of nonlinear dynamics. The disadvantages of ANNs include (1) the training outcome depends significantly on the choice of initial parameters such as number of layers and number of neurons in each layer and (2) training is too computationally expensive to solve large problems.

Another data-driven method for prognostics is based on decision trees, which is a non-parametric supervised learning method used for classification and regression. The goal of decision tree learning is to create a model that predicts the value of a target variable by learning decision rules inferred from data features. A decision tree is a tree structure in which each internal node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf node holds a class label. Jiaa and Dornfeld [26] proposed a decision tree-based method for the prediction of tool flank wear in a turning operation using acoustic emission and cutting force signals. The features characterizing the AE RMS and cutting force signals were extracted from both time and frequency domains. The decision tree approach was demonstrated to be able to make reliable inferences and decisions on tool wear classification. Elangovan *et al.* [27] developed a decision tree-based algorithm for tool wear prediction using vibration signals. Ten-fold cross-validation was used to evaluate the accuracy of the predictive model created by the decision tree algorithm. The maximum classification accuracy was 87.5%. While the advantage of decision trees is interpretability, decision trees can be very sensitive to small variations in training data.

III. Machine Learning

A. Random Forests

To address the research gap, the random forest algorithm is introduced to predict tool wear. A comprehensive tutorial on random forests can be found in Friedman *et al.* [28]. The random forest algorithm, developed by Leo Breiman [18,29], is an ensemble learning method that constructs a forest of decision trees from bootstrap samples of a training data set. Each decision tree produces a response, given a set of predictor values. In a decision tree, each internal node represents a test on an attribute, each branch represents the outcome of the test, each leaf node represents a class label for classification or a response for regression. A decision tree in which the response is continuous is also referred to as a regression tree. In the context of tool wear prediction, each individual decision tree in a random forest is a regression tree because tool wear describes the gradual failure of cutting tools. The pseudo code of the random forest algorithm for regression is shown in Table 1.

Table 1. Pseudo Code of the Random Forest Algorithm

```
Random Forests for Regression [28]
Input: Training data
Output: Prediction at a new data point
1. for b = 1 to $B$ do % $B$ is the number of trees %
1.1 Draw a bootstrap sample $Z$ of size $N$ from the training data
1.2 Grow a random-forest tree $T_b$ to the bootstrapped data
(1.2.1) Select $m$ variables at random from $p$ variables
(1.2.2) Pick the best split-point among the $m$ variables
(1.2.3) Split the node into two children nodes
2. Output the ensemble of trees $\{T_b\}_1^B$
3. Make a prediction at a new point $x$ by aggregating the
predictions of the $B$ trees
    $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
```

Bootstrap aggregating or bagging: Given a training data set

$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$,

bootstrap aggregating or bagging generates B new training data sets D_i of size N by sampling from the original training data set D with replacement. D_i is referred to as a bootstrap sample. By sampling with replacement or bootstrapping, some observations may be repeated in each D_i. Bagging helps reduce variance and avoid overfitting. The number of regression trees B is a parameter specified by users. Typically, a few hundred to several thousand trees are used in the random forest algorithm.

Choosing variables to split on: For each of the bootstrap samples, grow an un-pruned regression tree with the following procedure: At each node, randomly sample m variables and choose the best split among those variables rather than choosing the best split among all predictors. This process is sometimes called "feature bagging". The reason why a random subset of the predictors or features is selected is because the correlation of the trees in an ordinary bootstrap sample can be reduced. For regression, the default $m = p/3$.

Splitting criterion: Suppose that the training data is partitioned into M regions $R_1, R_2, \ldots, R_m$. A regression tree can be modeled as follows:

$$f(x) = \sum_{m=1}^{M} c_m I(x \epsilon R_m) \tag{3.1}$$

where $I(.)$ is an indicator function; If its argument is true, then the indicator function returns 1; otherwise 0; and the response is modeled as a constant c_m in each region. The splitting criterion at each node is to minimize the sum of squares. Therefore, the best $\widehat{c_m}$ is the average of y_i in region R_m:

$$\widehat{c_m} = ave(y_i | x_i \epsilon R_m) \tag{3.2}$$

Consider a splitting variable j and split point s, and define the pair of half-planes

$$R_1(j, s) = \{X | X_j \le s\}, R_2(j, s) = \{X | X_j \ge s\}. \tag{3.3}$$

Then we seek the splitting variable j and split point s that solve

$$\min_{j,s}\left[\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c2)^2\right]. \tag{3.4}$$

For any choice j and s, the inner minimization is solved by

$$\hat{c_1} = ave\left(y_i \middle| x_i \epsilon R_1(j, s)\right)$$
$$\hat{c_2} = ave\left(y_i \middle| x_i \epsilon R_2(j, s)\right) \tag{3.5}$$

Having found the best split, we partition the data into two resulting regions and repeat the splitting process on each of the two regions. This splitting process is repeated until a predefined stopping criterion is satisfied.

Stopping criterion: Tree size is a tuning parameter governing the complexity of a model. The stopping criterion is that the splitting process proceeds until the number of records in D_i falls below a threshold. The default threshold is five ($n_{min} = 5$). Alternatively, the maximum depth to which a decision tree should be constructed can be specified.

After B such trees $\{T_b\}_1^B$ are constructed, a prediction at a new point x can be made by averaging the predictions from all the individual B regression trees on x:

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{3.6}$$

B. MapReduce-based Parallel Random Forests

Because the tree growth step of the random forest machine learning algorithm is parallelizable, the MapReduce framework is used to parallelize the random forest algorithm. MapReduce is a programming model for processing large data sets with a parallel algorithm on a single machine with multi-core CPUs and a cluster [30].

Fig. 1 illustrates a high-level view of the MapReduce architecture [31,32]. In step 0, input data (i.e., training data sets) are fed into an algorithm. In step 1, the algorithm is executed on a single machine with multiple CPU cores or a cluster. In step 2, a master is created to split the input data into multiple pieces. Each piece is assigned to a mapper. In step 3, a Map function parses the input data and generates a list of intermediate <key, value> pairs. In step 4, the master collects the intermediate data from the mappers and sorts the intermediate data by the keys. All the intermediate data with the same key are grouped together. After sorting, a Reduce function is called. In step 5, the Reduce function aggregates all the intermediate pairs with the same key generated by the Map function. Finally, in step 6, the reducer returns the final results.

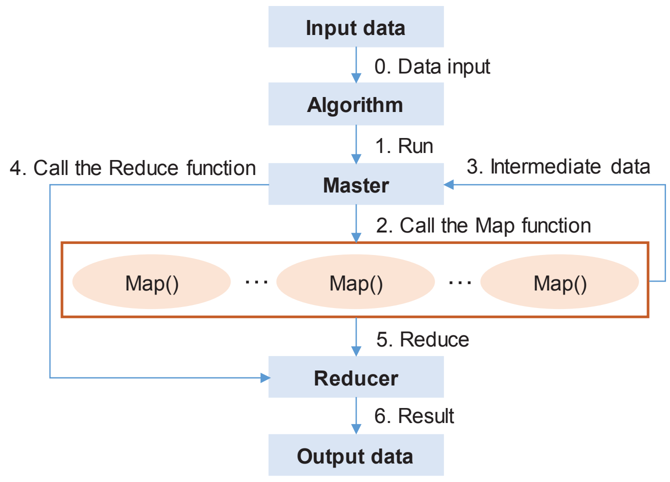

Figure 1. MapReduce Framework

IV. Methodology

This section presents the methodology for data-driven prognostics for tool wear prediction with the MapReduce-based PRF algorithm. The input of the PRF is the training data $D = (x_i, y_i)$ where x_i denote the cutting forces, vibrations, and acoustic emissions, y_i denotes the magnitude of tool wear. A random forest is constructed using $B = 10{,}000$ regression trees. Given the labeled training dataset $D = (x_i, y_i)$, we drew a bootstrap sample of size N from the training dataset (Step 1.1). For each regression tree, we select $m = 3$ $(m = \frac{p}{3}$ $rounded\ down, p = 9)$ variables at random from the 9 variables (Step 1.2.1). The best variable/split-point is selected among the 3 variables (Step 1.2.2). A regression tree progressively splits the training dataset into two child nodes, left node (with samples < z) and right node (with samples >= z). A splitting variable and split point are selected by solving Equations 3.4 and 3.5. The process is applied recursively on the dataset in each child node. The splitting process stops if the number of records in a node is less than 5. An individual regression tree is built by starting at the root node of the tree, performing a sequence of tests about the predictors, and organizing the tests in a hierarchical binary tree structure as illustrated in Fig. 2.

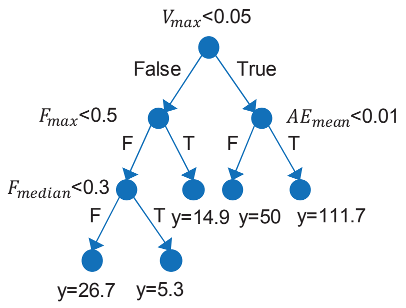

Figure 2. Binary Regression Tree Growing Process

After 10,000 trees are constructed, a prediction at a new point can be made by averaging the predictions from all the individual binary regression trees on this point. Because the random forest algorithm can be decomposed into a large number of independent computations, also known as perfectly parallel, the MapReduce framework performs optimally.

V. Experiment and Results

A. Experimental Setup

The dataset used in this paper was obtained from Li *et al.* [33]. The details of the experiment are presented in this section. The experimental setup is shown in Fig. 3.

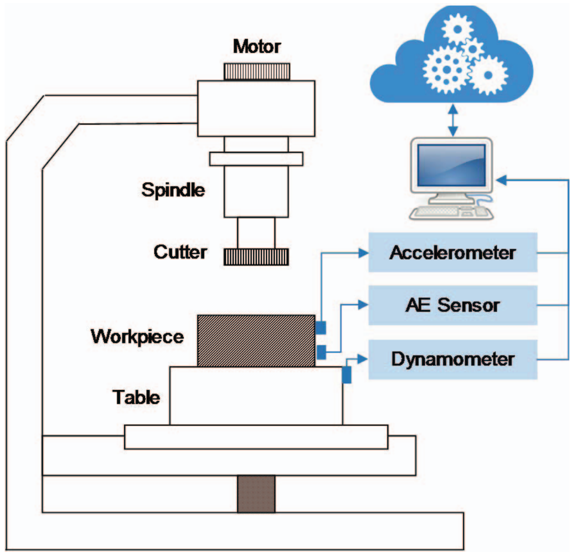

Figure 3. Binary Regression Tree Growing Process

The experiment was conducted on a three-axis high speed CNC machine (Röders Tech RFM 760). The workpiece material used in the dry milling experiment was stainless steel. The detailed description of the operating conditions in the dry milling operation can be found in Table 2. The spindle speed of the cutter was 10,400 RPM. The feed rate was 1,555 mm/min. The Y depth of cut (radial) was 0.125 mm. The Z depth of cut (axial) was 0.2 mm. The sampling rate was 50 KHz/channel.

Table 2. Operating Conditions

Parameter	**Value**
Spindle Speed	10400 RPM
Feed Rate	1555 mm/min
Y Depth of Cut	0.125 mm
Z Depth of Cut	0.2 mm
Sampling Rate	50 KHz/channel
Material	Stainless steel

As shown in Table 3, seven signal channels, including cutting force, vibration, and acoustic emission data, were monitored. A stationary dynamometer, mounted on the table of the CNC machine, was used to measure cutting forces in three, mutually perpendicular axes (x, y, and z dimensions). Three piezo accelerometers, mounted on the workpiece, were used to measure vibration in three, mutually perpendicular axes (x, y, and z dimensions). An acoustic emission (AE) sensor, mounted on the workpiece, was used to monitor a high frequency oscillation that occurs spontaneously within metals due to crack formation or plastic deformation. Acoustic emission is caused by the release of strain energy as the micro structure of the material is rearranged. Three datasets were generated. Each dataset contains 315 individual data acquisition files in the csv format. The size of each dataset is about 2.89 GB.

Table 3. Signal Channel and Data Description

Signal Channel	Data Description
Channel 1	Force (N) in X dimension
Channel 2	Force (N) in Y dimension
Channel 3	Force (N) in Z dimension
Channel 4	Vibration (g) in X dimension
Channel 5	Vibration (g) in Y dimension
Channel 6	Vibration (g) in Z dimension
Channel 7	Acoustic Emission (V)

B. *Tool Wear Prediction with Random Forests*

Feature extraction is an essential preprocessing step in which raw data collected from various signal channels is converted into a set of statistical features in a format supported by machine learning algorithms. The statistical features are then given as an input to a machine learning algorithm. In the first experiment, the raw data was collected from (1) cutting force, (2) vibration, and (3) acoustic emission signal channels. A set of statistical features (28 features) extracted from these signals include Maximum, Median, Mean, and Standard Deviation as listed in Table 4.

Table 4. Extracted Features

Cutting Force	Vibration	Acoustic Emission
Max	Max	Max
Median	Median	Median
Mean	Mean	Mean
Standard Deviation	Standard Deviation	Standard Deviation

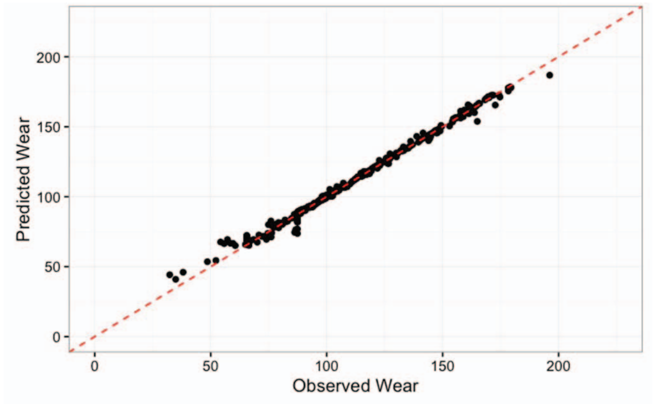

Figure 4. Tool Wear Prediction

A predictive model was developed using the random forest algorithm. Two thirds (2/3) of the input data was used for model development (training). The remainder (1/3) of the input data was used for model validation (testing). Fig. 4 shows the predicted against observed tool wear values using the experimental data set.

The performance of the random forest algorithm is evaluated using accuracy and training time. The accuracy of the random forest algorithm is measured using the R^2 statistic, also referred to as the coefficient of determination, and mean squared error (MSE). In statistics, the coefficient of determination is defined as $R^2 = 1 - \frac{SSE}{SST}$ where SSE is the sum of the squares of residuals, SST is the total sum of squares. The coefficient of determination is a measure that indicates the percentage of the response variable variation that is explained by a regression model. A higher R-squared indicates that more variability is explained by the regression model. For example, an R^2 of 100% indicates that the regression model explains all the variability of the response data around its mean. In general, the higher the R-squared, the better the regression model fits the data. The MSE of an estimator measures the average of the squares of the errors. The MSE is defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y_i} - Y_i)^2$ where $\widehat{Y_i}$ is a predicted value, Y_i is an observed value, and n is the sample size. The random forest algorithm uses between 50% and 90% of the input data for model development (training) and uses the remainder for model validation (testing). Table 5 lists the MSE, R^2, and training time.

Table 5. Accuracy and Training Time

	Random forest (10,000 Trees)		
Training size (%)	MSE	R^2	Training time (Second)
50	14.242	0.986	20.876
60	11.466	0.989	26.562
70	10.469	0.990	33.230
80	8.195	0.992	38.995
90	8.295	0.992	45.224

C. *Performance Evaluation for Parallel Random Forests*

This section presents the performance evaluation for the parallel implementation of the random forest algorithm. Specifically, the speedup and efficiency of the MapReduce-based PRF algorithm are discussed. The PRF algorithm was implemented on one of the most popular public cloud computing platforms, Amazon Elastic Compute Cloud (Amazon EC2). A variety of instance types with varying combinations of CPU, memory, and storage are provided for evaluating the speedup, efficiency, and scalability of the PRF algorithm. The cloud computing service on the Amazon EC2 can be accessed by an online user interface, called the AWS Management Console. A user can configure, launch, stop, restart, and terminate an instance (i.e., a virtual server in Amazon EC2) to run application programs in the cloud computing environment via a web browser. Amazon EC2 provides a variety of instance types which comprise varying combination of virtual CPU (vCPU), memory, and storage. A C3 instance on the Amazon EC2 is one of the compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance. Table 6 lists the hardware specifications of the C3 instance. The C3 large instance equips with 32 virtual cores, 60 GB memory, and two disks of 320 GB storage and runs on the Linux operating system.

Table 6. Amazon EC2 Infrastructures

Instance Type	C3.8×large	R3.8×large
Processor	Intel Xeon E5-2680 v2	Intel Xeon E5-2670 v2
Number of CPU	32	32
Memory (GB)	60	244
Storage (GB)	640 (2 × 320)	640 (2 × 320)

Fig. 5 shows the time spent training the predictive model using a C3 instance with varying number of cores and different sizes of training data sets. The algorithm was executed twenty times with the percentage of the training data sets and the number of cores ranging from 50% to 90% and from 1 to 32 cores, respectively. Execution time is calculated as an average over the twenty runs. Note that the time to construct the features from the initial signals is not included.

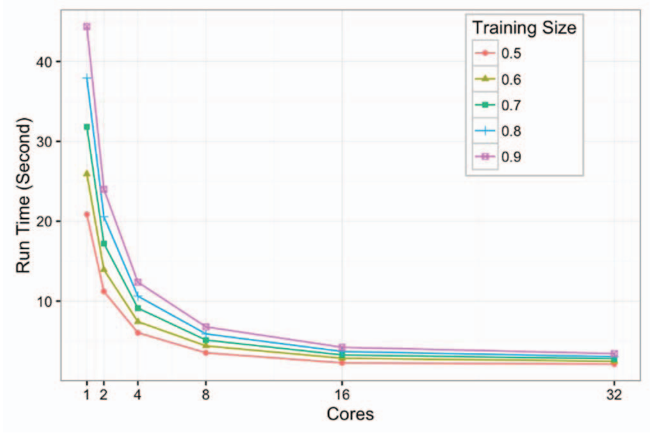

Figure 5 Runtime vs number of cores using C3 instances

As shown in Fig. 5 and Fig. 6, the PRF algorithm scales relatively well with the number of cores for different percentages of the training data sets. Take the run time with 90% of the training data set for example, near linear speedup is observed when the PRF algorithm runs on the number of cores ranging from 1 to 16 cores based on the speedup curve. However, the speedup falls rapidly for 32 cores. It should be noted that relative speedup is the ratio of the solution time for a problem with a parallel algorithm executed on a single processor to the solution time with the same algorithm when executed on multiple processors. Another metric to measure the performance of the PRF algorithm is efficiency, defined as the ratio of relative speedup to the number of processors. As shown in Fig. 5, the execution time using 1, 2, 4, 8, 16, and 32 cores are 44s, 23s, 12s, 7s, 4s, and 3s. For 1 to 16 cores, relative speedup is almost linear. Linear speedup in turn corresponds to efficiency of 1. When the number of cores continues to increase beyond 16 cores, the PRF cannot achieve linear speedup. This is because speedup is always limited by the serial part of the program according to Amdahl's law [34] of the theoretical speedup of the execution of a program.

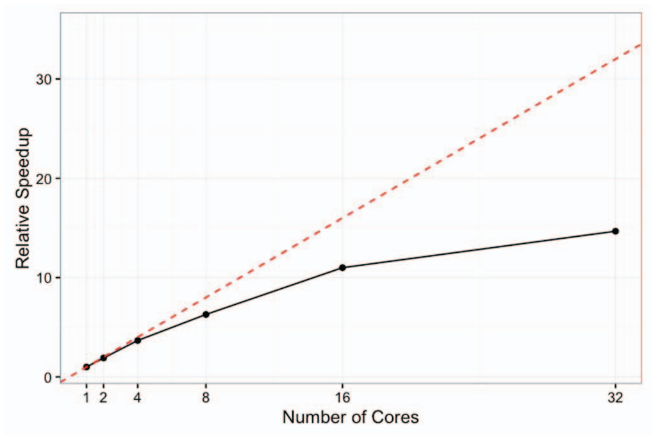

Figure 6 Speedup using C3 instances (90% of the training data set)

Because high performance computing applications are often limited by either computing speed or memory, it is worthwhile to evaluate whether this application is compute bound or memory bound. The PRF algorithm was executed on a R3 instance which is optimized for memory-intensive applications. Similar to the C3 instance, the R3 large instance equips with 32 virtual cores, two disks of 320 GB storage, and 244 GB memory instead of 60 GB memory. Table 6 lists the hardware specifications of the R3 large instance. As shown in Fig. 7, training time using the R3 instance with 244 GB memory is almost the same as that of the C3 instance with 60 GB memory. The results demonstrate that the PRF algorithm is compute bound instead of memory bound.

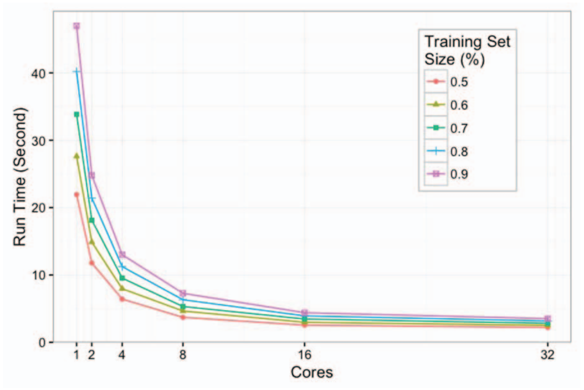

Figure 7 Runtime vs number of cores using R3 instances

VI. CONCLUSIONS

In this paper, the prediction of tool wear in milling operations was performed with the random forest and PRF algorithms. The PRF algorithm was developed using the MapReduce framework and then implemented on the Amazon Elastic Compute Cloud. The effectiveness and efficiency of the algorithms were demonstrated with two different data sets collected from two milling experiments under various operating conditions. Two sets of statistical features were extracted from cutting forces, vibrations, acoustic emissions and the electrical current of the spindle motor, vibrations at the table and spindle, acoustic emissions at the table and spindle, respectively. Two thirds of the input data were used for training. The remainder of the input data was used for testing. The performance of the random forest algorithm was evaluated using mean squared error, R-square, and training time. The experimental results have shown that random forests can generate very accurate predictions for the first data set. Due to the limited amount of training data, the random forest algorithm generated less accurate predictions for the second experiment. Moreover, the PRF algorithm was developed to scale up the random forest algorithm. The experimental results have shown that significant speedup can be achieved when building a large number of decision trees. Further, the PRF algorithm has been demonstrated to be compute bound by comparing the training time using two Amazon instances.

In the future, it will be worthwhile to predict tool wear with other machine learning algorithms such as support vector machines as well as compare the performance of these algorithms with that of random forests using accuracy and training time. In addition, our future work will focus on collecting large volumes of streaming data from a network of CNC machines and build predictive models for tool wear estimation with the PRF algorithm and a cluster on the cloud.

ACKNOWLEDGMENT

The research reported in this paper is partially supported by NSF under grant number IIP-1238335. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] L. Swanson, 2001, “Linking maintenance strategies to performance,” International Journal of Production Economics, 70(3), pp. 237-244.

[2] C. Valdez-Flores and R. M. Feldman, 1989, “A survey of preventive maintenance models for stochastically deteriorating single-unit systems,” Naval Research Logistics (NRL), 36(4), pp. 419-446.

[3] D. Wu, J. Terpenny, L. Zhang, R. Gao, and T. Kurfess, 2016, “Fog-enabled architecture for data-driven cyber-manufacturing systems,” Proceedings of the ASME 2016 International Manufacturing Science and Engineering Conference, Blacksburg, Virginia, U.S.

[4] J. Lee, 1995, “Machine performance monitoring and proactive maintenance in computer-integrated manufacturing: review and perspective,” International Journal of Computer Integrated Manufacturing, 8(5), pp. 370-380.

[5] M. Bevilacqua and M. Braglia, 2000, “The analytic hierarchy process applied to maintenance strategy selection," Reliab Eng Syst Safe, 70(1), pp. 71-83.

[6] J. H. Suh, S. R. Kumara, and S. P. Mysore, 1999, “Machinery fault diagnosis and prognosis: application of advanced signal processing techniques,” CIRP Ann-Manuf Techn, 48(1), pp. 317-320.

[7] C. Hu, B. D. Youn, and T. Kim, 2012, “Semi-supervised learning with co-training for data-driven prognostics,” Prognostics and Health Management (PHM), 2012 IEEE Conference on, IEEE, pp. 1-10.

[8] M. Schwabacher, 2005, “A Survey of Data-Driven Prognostics,” Proceedings of the AIAA Infotech@Aerospace Conference, Arlington, Virginia, pp. 1-5.

[9] G. Byrne, D. Dornfeld, I. Inasaki, G. Ketteler, W. König, and R. Teti, 1995, “Tool condition monitoring (TCM)—the status of research and industrial application,” CIRP Ann-Manuf Techn, 44(2), pp. 541-567.

[10] R. Teti, K. Jemielniak, G. O’Donnell, and D. Dornfeld, 2010, “Advanced monitoring of machining operations,” CIRP Ann-Manuf Techn, 59(2), pp. 717-739.

[11] R. Gao, L. Wang, R. Teti, D. Dornfeld, S. Kumara, M. Mori, and M. Helu, 2015, “Cloud-enabled prognosis for manufacturing,” CIRP Ann-Manuf Techn, 64(2), pp. 749-772.

[12] M. J. Daigle and K. Goebel, 2013, “Model-based prognostics with concurrent damage progression processes,” Systems, Man, and Cybernetics: Systems, IEEE Transactions on, 43(3), pp. 535-546.

[13] X.S. Si, W. Wang, C.H. Hu, M.Y. Chen, and D.H. Zhou, 2013, “A wiener-process-based degradation model with a recursive filter algorithm for remaining useful life estimation,” Mechanical Systems and Signal Processing, 35(1), pp. 219-237.

[14] M. Dong and D. He, 2007, “Hidden semi-Markov model-based methodology for multi-sensor equipment health diagnosis and prognosis,” European Journal of Operational Research, 178(3), pp. 858-878.

[15] B. Saha, K. Goebel, and J. Christophersen, 2009, “Comparison of prognostic algorithms for estimating remaining useful life of batteries,” T I Meas Control.

[16] M. E. Orchard and G. J. Vachtsevanos, 2009, “A particle-filtering approach for on-line fault diagnosis and failure prognosis,” T I Meas Control.

[17] B. Sick, 2002, “On-line and indirect tool wear monitoring in turning with artificial neural networks: a review of more than a decade of research,” Mechanical Systems and Signal Processing, 16(4), pp. 487-546.

[18] L. Breiman, 2001, “Random forests,” Machine Learning, 45(1), pp. 5-32.

[19] M. Schwabacher and K. Goebel, 2007, "A survey of artificial intelligence for prognostics," Proc. AAAI Fall Symposium, pp. 107-114.

[20] C. Chungchoo and D. Saini, 2002, “On-line tool wear estimation in CNC turning operations using fuzzy neural network model,” International Journal of Machine Tools and Manufacture, 42(1), pp. 29-40.

[21] J. C. Chen and J. C. Chen, 2005, “An artificial-neural-networks-based in-process tool wear prediction system in milling operations,” The International Journal of Advanced Manufacturing Technology, 25(5-6), pp. 427-434.

[22] T. Özel and Y. Karpat, 2005, “Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks,” International Journal of Machine Tools and Manufacture, 45(4), pp. 467-479.

[23] S. T. Bukkapatnam, A. Lakhtakia, and S. R. Kumara, 1995, “Analysis of sensor signals shows turning on a lathe exhibits low-dimensional chaos,” Phys Rev E, 52(3), p. 2375.

[24] S. T. Bukkapatnam, S. R. Kumara, and A. Lakhtakia, 2000, “Fractal estimation of flank wear in turning,” Journal of Dynamic Systems, Measurement, and Control, 122(1), pp. 89-94.

[25] S. Bukkapatnam, S. Kumara, and A. Lakhtakia, 1999, “Analysis of acoustic emission signals in machining,” Journal of Manufacturing Science and Engineering, 121(4), pp. 568-576.

[26] C. L. Jiaa and D. A. Dornfeld, 1998, “A self-organizing approach to the prediction and detection of tool wear,” ISA Transactions, 37(4), pp. 239-255.

[27] M. Elangovan, S. B. Devasenapati, N. Sakthivel, and K. Ramachandran, 2011, “Evaluation of expert system for condition monitoring of a single point cutting tool using principle component analysis and decision tree algorithm,” Expert Systems with Applications, 38(4), pp. 4450-4459.

[28] J. Friedman, T. Hastie, and R. Tibshirani, 2001, “The elements of statistical learning,” Springer Series in Statistics, Springer, Berlin.

[29] A. Liaw and M. Wiener, 2002, “Classification and regression by random forest,” R news, 2(3), pp. 18-22.

[30] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. E. Guestrin, and J. Hellerstein, 2014, “Graphlab: A new framework for parallel machine learning,” arXiv preprint arXiv:1408.2041.

[31] C. Chu, S. K. Kim, Y.A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, 2007, “Map-reduce for machine learning on multicore,” Advances in Neural Information Processing Systems, 19, p. 281.

[32] J. Dean and S. Ghemawat, 2008, “MapReduce: simplified data processing on large clusters,” Communications of the ACM, 51(1), pp. 107-113.

[33] X. Li, B. Lim, J. Zhou, S. Huang, S. Phua, K. Shaw, and M. Er, 2009, “Fuzzy neural network modelling for tool wear estimation in dry milling operation,” Annual Conference of the Prognostics and Health Management Society, pp. 1-11.

[34] Y. Censor and S. A. Zenios, 1997, “Parallel optimization: Theory, algorithms, and applications,” Oxford University Press on Demand.