Dazhong Wu[1]
Department of Mechanical and Aerospace Engineering,
University of Central Florida,
Orlando, FL 32816
e-mail: Dazhong.Wu@ucf.edu

Connor Jennings
Department of Industrial and Manufacturing Engineering,
Pennsylvania State University,
University Park, PA 16802
e-mail: connor@psu.edu

Janis Terpenny
Department of Industrial and Manufacturing Engineering,
Pennsylvania State University,
University Park, PA 16802
e-mail: jpt5311@psu.edu

Soundar Kumara
Department of Industrial and Manufacturing Engineering,
Pennsylvania State University,
University Park, PA 16802
e-mail: skumara@psu.edu

Robert X. Gao
Department of Mechanical and Aerospace Engineering,
Case Western Reserve University,
Cleveland, OH 44106
e-mail: robert.gao@case.edu

Cloud-Based Parallel Machine Learning for Tool Wear Prediction

The emergence of cloud computing, industrial internet of things (IIoT), and new machine learning techniques have shown the potential to advance prognostics and health management (PHM) in smart manufacturing. While model-based PHM techniques provide insight into the progression of faults in mechanical components, certain assumptions on the underlying physical mechanisms for fault development are required to develop predictive models. In situations where there is a lack of adequate prior knowledge of the underlying physics, data-driven PHM techniques have been increasingly applied in the field of smart manufacturing. One of the limitations of current data-driven methods is that large volumes of training data are required to make accurate predictions. Consequently, computational efficiency remains a primary challenge, especially when large volumes of sensor-generated data need to be processed in real-time applications. The objective of this research is to introduce a cloud-based parallel machine learning algorithm that is capable of training large-scale predictive models more efficiently. The random forests (RFs) algorithm is parallelized using the MapReduce data processing scheme. The MapReduce-based parallel random forests (PRFs) algorithm is implemented on a scalable cloud computing system with varying combinations of processors and memories. The effectiveness of this new method is demonstrated using condition monitoring data collected from milling experiments. By implementing RFs in parallel on the cloud, a significant increase in the processing speed (14.7 times in terms of increase in training time) has been achieved, with a high prediction accuracy of tool wear (eight times in terms of reduction in mean squared error (MSE)). [DOI: 10.1115/1.4038002]

Keywords: data-driven prognostics, cloud computing, machine learning, prognostics and health management, tool wear prediction, random forests

1 Introduction

Mechanical components such as shafts, bearings, and gears are subject to failures resulting from excessive load, corrosion, overheating, vibration, fracture, and material fatigue [1,2]. Prognostics and health management (PHM) is a discipline that predicts health condition and remaining useful life based on previous and current operating conditions. PHM techniques fall into two categories: model-based and data-driven prognostics [3–5]. Model-based prognostics refers to approaches based on analytical models of system behaviors derived from physical laws or probability distributions. Model-based prognostics includes methods based on Wiener and Gamma processes [6], hidden Markov models (HMMs) [7], Kalman filter [8], and particle filter [9]. Model-based PHM methods are generally effective for applications where the underlying physics is well understood. However, in-depth physical knowledge of system failures is not always available or sometimes too expensive to acquire. Another limitation of model-based PHM methods is that oftentimes certain distributional assumptions such as Gaussian distributions have to be made in order to develop close-form representations. However, these assumptions may not hold true in practice, particularly for complex systems.

Data-driven prognostics trains predictive models using statistical methods or machine learning algorithms. Some of the well-known data-driven prognostic methods include autoregressive model, multivariate adaptive regression, fuzzy set theory, artificial neural networks (ANNs) with variants such as adaptive neuro-fuzzy inference system, decision trees, logistic regression, Bayesian belief network, support vector machines (SVMs), and principal component analysis (PCA). While data-driven approaches may require large volumes of quality training data, an in-depth understanding of system physical behaviors is not a prerequisite [10]. In addition, data-driven prognostics is typically more effective than model-based prognostics for complex systems where underlying physics is not well known. While classical data-driven approaches such as ANNs and SVMs have been widely investigated in applications such as tool wear prediction, little research has been reported on the capabilities of cloud-based parallel machine learning algorithms for predictive analytics in smart manufacturing [11–14].

To complement traditional model-based and data-driven PHM techniques, a cloud-based parallel machine learning algorithm based on random forests (RFs) is introduced. According to Refs. [15] and [16], some of the desirable characteristics of RFs are: (1) RFs can process a large number of input variables without overfitting because RFs use bootstrap aggregating to generate replicate data sets and (2) RFs do not require cross validation because each tree is built using a different bootstrap sample from the original data. More importantly, because the tree construction process in RFs is independent of each other, RFs are relatively

[1]Corresponding author.

Manuscript received May 30, 2017; final manuscript received September 17, 2017; published online February 12, 2018. Assoc. Editor: Laine Mears.

easy to be parallelized. Due to this unique characteristic, RFs have the potential to process large volumes of measurement data generated from machine condition monitoring systems on multi-core processors or clusters [17–21]. In this paper, a new scalable data-driven approach to PHM is introduced to generate big data analytics more effectively and efficiently [20,22].

The main contributions of this paper include:

- A MapReduce-based parallel random forests (PRFs) algorithm is developed to predict tool wear in milling operations. This new algorithm is implemented on a scalable, high performance computing cloud platform.
- The efficiency of the cloud-based PRFs algorithm is evaluated using training time and relative speedup ratio. The performance of cloud-based PRFs algorithm is compared with that of the original RFs algorithm that is implemented sequentially.

The remainder of the paper is organized as follows: Sec. 2 reviews the related work on cloud computing, industrial internet of things (IIoT)-enabled smart manufacturing, and prognostics for machining processes. Section 3 introduces RFs and a parallel programming framework (i.e., MapReduce). Section 4 presents a new data-driven PHM approach for tool wear prediction using the cloud-based PRFs algorithm. Section 5 presents an experimental setup and experimental data acquired from multiple sensors (e.g., acoustic emission (AE) sensor, vibration sensor, and current sensor). Section 6 discusses experimental results, demonstrates the effectiveness of RFs, and evaluates the performance of the cloud-based PRFs algorithm. Section 7 provides conclusions that include a discussion of research contribution and future work.

2 Data-Driven Prognostics for Smart Manufacturing.

2.1 Cloud Computing and IIoT. Cloud-based manufacturing was introduced to increase manufacturing productivity and reduce costs by utilizing big data analytics, machine learning, high performance computing, sensor networks, automation, and control technologies [23–25]. Tao et al. [20] presented an overview of applications of IIoT and cloud computing in smart manufacturing. A conceptual framework of cloud computing and IIoT-based manufacturing systems was introduced to connect manufacturing machines as well as monitor manufacturing processes. Bi et al. [26] presented the key requirements of IIoT infrastructures for next-generation manufacturing systems. Some of the enabling technologies for the development of IIoT infrastructures include cloud computing, wireless sensor networks, radio-frequency identification, remote monitoring, data analytics, and communication standards. Lee et al. [27] proposed a cyber-physical system architecture for IIoT-based manufacturing systems. The cyber-physical system architecture consists of several layers such as digital connection, data-to-information conversion, high-performance computing infrastructure, cognition, and configuration layers. Wang et al. [28] developed an assembly system for complex products using IIoT and cloud computing. Experimental results have shown that the cloud-based assembly system is capable of performing assembly planning for aircraft engines very effectively. Mourtzis et al. [29] introduced a cloud-based approach to process monitoring and process planning in distributed manufacturing environments. Machine conditions and availability were monitored in real time to identify optimal manufacturing process plans using cloud storage and wireless sensor networks. To demonstrate the effectiveness of the cloud-based monitoring system, three five-axis CNC machines were equipped with angular velocity and current sensors that measure spindle speed and energy consumption. Sensor-generated streaming data were collected through a data-acquisition system and transmitted to a remote cloud storage for data mining. Mai et al. [30] presented a cloud-based additive manufacturing system that enables users to create digital models and fabricate parts via cloud-based CAD/CAM applications. By connecting three-dimensional printers via the Internet and sensors, additive manufacturing processes can be monitored and controlled remotely. Wu et al. [2] presented a fog computing-based data-driven PHM system that is capable of collecting real-time machine condition data and monitoring the vibrations and energy consumption of pumps. Fog computing reduces network latency by moving computing infrastructure geographically closer to clusters. Several classical machine learning algorithms were implemented on a highly scalable public cloud to generate predictive data analytics. In addition, wireless gateway devices were developed to provide connectivity between a factory floor and the cloud.

2.2 Data-Driven Prognostics for Machining Processes. In addition to cloud computing and IIoT, one of the most important applications in smart manufacturing is diagnostics and prognostics. Tool wear is the most commonly observed and unavoidable phenomenon in manufacturing processes such as drilling, milling, and turning [31,32]. The rate of tool wear is typically affected by process parameters (e.g., cutting speed and feed rate), cutting tool geometry, and properties of workpiece and tool materials. The Taylor's equation for tool life expectancy provides an approximation of tool wear [33]. However, with the rapid advancement of sensing technology and increasing number of sensors equipped on modern CNC machines, it is possible to predict tool wear more accurately using various measurement data. This section presents a review of model-based and data-driven prognostic approaches for tool wear prediction.

The objective of model-based prognostics is to predict system performance based on probability distributions. The limitation of model-based prognostic methods is that certain stochastic or random processes such as Wiener processes and Gamma processes are assumed. The most popular model-based approaches to prognostics include HMMs, Kalman filter, and particle filter. Ertunc et al. [34] developed an online tool wear condition monitoring system for drilling processes. Based on torque, force, and spindle power signals collected during drilling operations, the health status of cutting tools was predicted using HMMs. Atlas et al. [35] developed a tool condition monitoring system for milling processes using HMMs. Zhu et al. [36] proposed a multicategory classification approach for tool flank wear state identification in micromilling using continuous HMMs. Niaki et al. [21] developed an approach for online tool flank wear estimation in milling operations using a Kalman filter. The Kalman filter can predict tool flank wear with a maximum average error of 10%. Wang et al. [22,37] proposed an enhanced particle filter method to predict tool wear by integrating autoregressive models and the particle filtering algorithm. The enhanced particle filter based on Bayesian theory yields more accurate prediction of the state of tool wear as well as quantifies the uncertainty of remaining useful life of cutting tools. Wang and Gao [9] developed an adaptive resampling-based particle filtering approach to predict tool wear in a dry-milling operation on a high-speed CNC machine.

Some of the most commonly used data-driven approaches to PHM include ANNs, decision trees, and support vector machines. Elangovan et al. [38] presented a decision tree-based method for tool wear prediction using the data generated from vibration sensors. Ten-fold cross-validation was conducted to evaluate the performance of the predictive model trained by the decision tree algorithm. Experimental results have shown that the accuracy of the decision tree-based method was 87.5%. Shi and Gindy [39] developed a tool wear predictive model by combining least squares support vector machines (LS-SVM) and PCA. PCA was used to extract statistical features from multiple sensor signals. The LS-SVM algorithm was used to predict tool wear using the extracted features. A set of experiments was conducted on a vertical broaching machine with high speed steel broaching tool. Experimental results have shown that the LS-SVM algorithm can predict the flank wear accurately. Wu et al. [40] evaluated the performance of three machine learning algorithms, including ANNs, SVM, and RFs, for tool wear prediction using the data collected from 315 milling tests. A set of statistical features were extracted from the cutting force, vibration, and AE signal channels. Experimental results have shown that the predictive model trained by RFs outperforms ANNs and SVM.

In summary, while earlier work introduced data-driven prognostic methods based on classical machine learning algorithms as well as developed cloud-based data acquisition and process monitoring systems, little research has been reported on how machine learning algorithms can be parallelized to process big data as well as generate big data analytics for smart manufacturing. This paper introduces a cloud-based parallel machine learning approach that is capable of predicting tool wear more effectively and efficiently.

3 Random Forests and MapReduce

The objective of machine learning is to build a predictive model from training data and make predictions on test data. Machine learning techniques are typically classified into three broad categories, including supervised learning, unsupervised learning, and collaborative filtering that uses both supervised and unsupervised learning. Specifically, a supervised machine learning algorithm is used to construct an estimator, which is able to predict the label of an object given a set of features. Supervised machine learning algorithms can be used for regression and classification. Regression focuses on predicting a continuous-valued attribute associated with an object. Classification focuses on identifying to which category an object belongs. An unsupervised learning algorithm is used to identify similarities between objects given input data without labeled responses. One of the unsupervised machine learning techniques is clustering. Clustering is concerned with grouping together objects that are similar to each other and dissimilar to the objects belonging to other clusters.

Figure 1 illustrates a typical supervised machine learning process. Similar to unsupervised learning, supervised learning algorithms require two types of input data: training and testing data sets. A set of features or variables are extracted as input to a learning algorithm based on the training data sets and labeled data. A training set is a set of data used to discover predictive relationships. A test set is a set of data used to evaluate a predictive model. Once a machine learning model is evaluated, it can be used to predict a target or response variable based on independent variables.

3.1 Random Forests. The RFs algorithm [15] is an ensemble learning method for regression and classification by building a large number of decision trees. Some of the important concepts associated with RFs, including bootstrap aggregating or bagging, splitting and stopping criterion, are presented as follows.

Bootstrap aggregating, also known as bagging, is a method that improves the accuracy of RFs as well as helps avoid overfitting. Given a training data set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, bagging generates B new training data sets D_i of size N by sampling from the original training data set D with replacement. The new training data set D_i is referred to as a bootstrap sample. By sampling with replacement, some observations may be repeated in each training data set D_i. The number of regression trees B is a tuning parameter. In general, as the number of regression trees increases, the accuracy of the predictive model trained by RFs will increase. Once a bootstrap sample is generated, a regression tree is constructed based on the bootstrap sample. To split a parent node into two children nodes, a variable or feature space $\theta_b = \{x_1, x_2, \ldots x_m, \ldots x_N\}$ needs to be defined. At each node, m variables are randomly selected from the variable space θ_b. The best split is chosen among m variables instead of all the N variables. The reason why a random subset of the variables or features is selected is because the correlation of the regression trees can be reduced. In RFs, the number of variables or features is a tuning parameter. For regression, the number of features is typically one third of the total number of predictors. In general, as the number of variables increases, the bias will decrease, meanwhile, the correlation of regression trees will increase.

To determine the best split for each node, a splitting variable j and a cutting point s are defined. Two regions, $R_1(j,s)$ and $R_2(j,s)$, are defined based on the splitting variable and cutting point

$$R_1(j,s) = \{X|X_j \le s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j \ge s\} \tag{3.1}$$

The splitting criterion of RFs is to determine the splitting variable j and the split point s that solve the following objective function:

$$\min_{j,s}\left[\min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c1)^2 + \min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c2)^2\right] \tag{3.2}$$

The inner minimization can be solved by

$$\hat{c_1} = \text{ave}(y_i|x_i \epsilon R_1(j,s)) \text{ and } \hat{c_2} = \text{ave}(y_i|x_i \epsilon R_2(j,s)) \tag{3.3}$$

Once the best split is determined, the training data can be partitioned into two subregions. Each resulting subregion represents a children node. This splitting process is conducted recursively until a predefined stopping criterion is satisfied. One of the most commonly used stopping criteria is that the splitting process proceeds until the number of examples in D_i falls below a threshold. An alternative stopping criterion is to terminate the splitting process until the depth of a regression tree reaches a threshold.

Suppose that the training data are partitioned into M regions $R_1, R_2, \ldots, R_m$. The response of the regression tree can be modeled as follows:

$$T_b(x, \theta_b) = \sum_{m=1}^{M} c_m I(x \epsilon R_m) \tag{3.4}$$

where $I(.)$ is an indicator function. If its argument is true, then the indicator function returns 1; otherwise 0. After B such trees $\{T_b(x, \theta_b)\}_1^B$ are constructed, a prediction at a new data point x can be made by averaging the responses from all the B regression trees on x

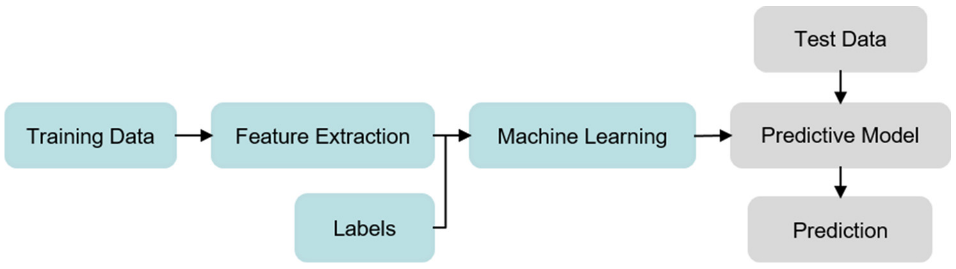

Fig. 1 Supervised machine learning

$$\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x, \theta_b) \quad (3.5)$$

The importance of a variable x_i for predicting the response variable Y is evaluated by averaging the sum of the weighted reduction in residual sum of squares for all nodes t where x_i is used over the number of regression trees

$$\text{VarImp}(x_i) = \frac{1}{N_T}\sum_{T}\sum_{t\in T: v(s_t)=x_i} p(t)\Delta i(s_t, t) \quad (3.6)$$

where $p(t)\Delta i(s_t, t)$ denotes the weighted reduction in residual sum of squares by splitting an internal node t into two children nodes. $p(t) = N_t/N$ denotes the proportion of the data points/samples at node t. N_t denotes the number of samples at node t. N denotes the total number of samples that is drawn to build a regression tree. N_T denotes the total number of regression trees in a random forest. T denotes a regression tree structure. s_t denotes a split at node t. $v(s_t)$ denotes the splitting variable that is selected for the split s_t. $\Delta i(s_t, t) = i(t) - p_1 i(t_1) - p_2 i(t_2)$. $i(t)$ denotes the residual sum of squares at node t. t_1 and t_2 denote the two children nodes of node t.

With the RFs algorithm described earlier, a single point estimate is calculated for a new instance. However, the single point estimate does not estimate the dispersion of future observations. To measure how reliable a prediction for a new instance is, it is important to quantify uncertainty in predictions for RFs [41]. A prediction interval is used to estimate an interval in which future observations will fall with a certain probability. The conditional distribution of Y, given $X = x$, is defined by $F(y|X = x) = P(Y \leq y|X = x)$. For a continuous variable such as tool wear, the α-quantile $Q_\alpha(x)$ is defined such that the probability of Y being less than $Q_\alpha(x)$ is equal to α. The α-quantile function is defined as $Q_\alpha(x) = \inf\{y : F(y|X = x) \geq \alpha\}$. A 95% prediction interval for the value of Y is given by $I(x) = [Q_{0.025}(x), Q_{0.975}(x)]$. To compute prediction intervals for RFs, all the responses in the leaf nodes of a random forest are recorded instead of the mean of the responses. Based on these responses produced by the regression trees, the conditional distribution of the response variable Y can be calculated. Once the conditional distribution is calculated, the α-quantile $Q_\alpha(x)$ can be calculated.

3.2 MapReduce Programming Framework. Parallel computing is a means of improving the performance and efficiency of algorithms by using multiple compute resources simultaneously to solve a large-scale computational problem. In a parallel program, a problem can be partitioned into a series of smaller subproblems, and then solved concurrently. To develop a parallel machine learning algorithm, it is important to identify a set of tasks that can be processed concurrently. In the context of RFs, the tasks in which multiple regression trees are constructed can execute concurrently because the regression trees are de-correlated. MapReduce is a programming framework that processes parallelizable problems on a single machine with multicore CPUs or large clusters of commodity hardware [42,43]. Figure 2 illustrates a high-level view of the MapReduce architecture [44]. A master splits the input data into multiple independent chunks according to the number of cores. Each chunk is assigned to a mapper. A *Map* function maps the input data to a set of intermediate <*key*, *value*> pairs. 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 subsequently grouped together and passed to a *Reduce* function. The *Reduce* function reduces all the intermediate pairs with the same key to a smaller set of values.

4 MapReduce-Based Parallel Random Forests

A MapReduce-based PRFs algorithm is developed to predict tool wear in milling operations. The input of PRFs is training data $D = (x_i, y_i)$, where $i = 1$ to N, $x_i = (F_X, F_Y, F_Z, V_X, V_Y, V_Z, \text{AE}_X, \text{AE}_Y, \text{AE}_Z) \in \mathbb{R}^{p=28}$, y_i denotes the value of tool wear

$$F_X = \{F_X^{\max}, F_X^{\text{median}}, F_X^{\text{mean}}, F_X^{\text{sd}}\}$$

$$F_Y = \{F_Y^{\max}, F_Y^{\text{median}}, F_Y^{\text{mean}}, F_Y^{\text{sd}}\}$$

$$F_Z = \{F_Z^{\max}, F_Z^{\text{median}}, F_Z^{\text{mean}}, F_Z^{\text{sd}}\}$$

$$V_X = \{V_X^{\max}, V_X^{\text{median}}, V_X^{\text{mean}}, V_X^{\text{sd}}\}$$

$$V_Y = \{V_Y^{\max}, V_Y^{\text{median}}, V_Y^{\text{mean}}, V_Y^{\text{sd}}\}$$

$$V_Z = \{V_Z^{\max}, V_Z^{\text{median}}, V_Z^{\text{mean}}, V_Z^{\text{sd}}\}$$

$$\text{AE}_X = \{\text{AE}_X^{\max}, \text{AE}_X^{\text{median}}, \text{AE}_X^{\text{mean}}, \text{AE}_X^{\text{sd}}\}$$

$$\text{AE}_Y = \{\text{AE}_Y^{\max}, \text{AE}_Y^{\text{median}}, \text{AE}_Y^{\text{mean}}, \text{AE}_Y^{\text{sd}}\}$$

$$\text{AE}_Z = \{\text{AE}_Z^{\max}, \text{AE}_Z^{\text{median}}, \text{AE}_Z^{\text{mean}}, \text{AE}_Z^{\text{sd}}\}$$

$F_X^{\max}, F_X^{\text{median}}$, F_X^{mean}, and F_X^{sd} denote the maximum, median, mean, and standard deviation of the cutting forces in the X direction. $F_Y^{\max}, F_Y^{\text{median}}$, F_Y^{mean}, and F_Y^{sd} denote the maximum, median, mean, and standard deviation of the cutting forces in the Y direction. $F_Z^{\max}, F_Z^{\text{median}}$, F_Z^{mean}, and F_Z^{sd} denote the maximum, median, mean, and standard deviation of the cutting forces in the Z direction. $V_X^{\max}, V_X^{\text{median}}$, V_X^{mean}, and V_X^{sd} denote the maximum, median, mean, and standard deviation of vibrations in the X direction. $V_Y^{\max}, V_Y^{\text{median}}$, V_Y^{mean}, and V_Y^{sd} denote the maximum, median, mean, and standard deviation of vibrations in the Y direction. $V_Z^{\max}, V_Z^{\text{median}}$, V_Z^{mean}, and V_Z^{sd} denote the maximum, median, mean, and standard deviation of vibrations in the Z direction. $\text{AE}_X^{\max}, \text{AE}_X^{\text{median}}$, $\text{AE}_X^{\text{mean}}$, and AE_X^{sd} denote the maximum, median, mean, and standard deviation of acoustic emissions in the X direction. $\text{AE}_Y^{\max}, \text{AE}_Y^{\text{median}}$, $\text{AE}_Y^{\text{mean}}$, and AE_Y^{sd} denote the maximum, median, mean, and standard deviation of acoustic emissions in the Y direction. $\text{AE}_Z^{\max}, \text{AE}_Z^{\text{median}}$, $\text{AE}_Z^{\text{mean}}$, and AE_Z^{sd} denote the maximum, median, mean, and standard deviation of acoustic emissions in the Z direction. A random forest was constructed using $B=$ 10,000 regression trees. Given the training data set $D = (x_i, y_i)$, 10,000 bootstrap samples were generated from the

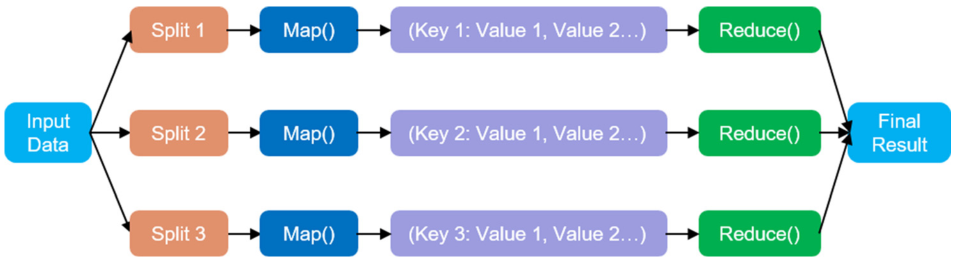

Fig. 2 MapReduce programming framework

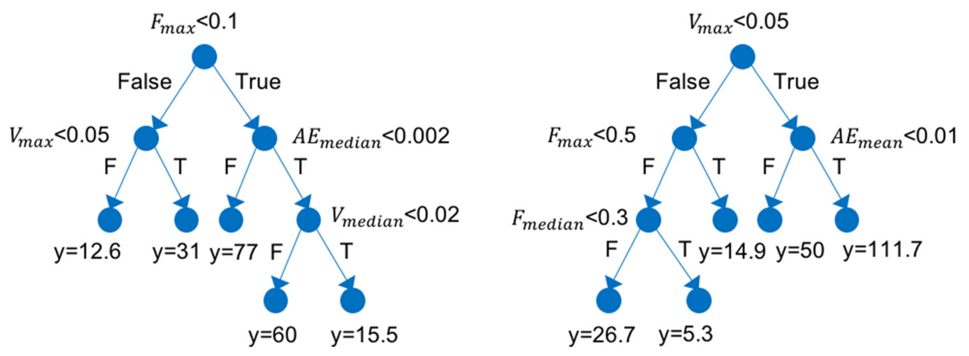

Fig. 3 Regression tree growing process

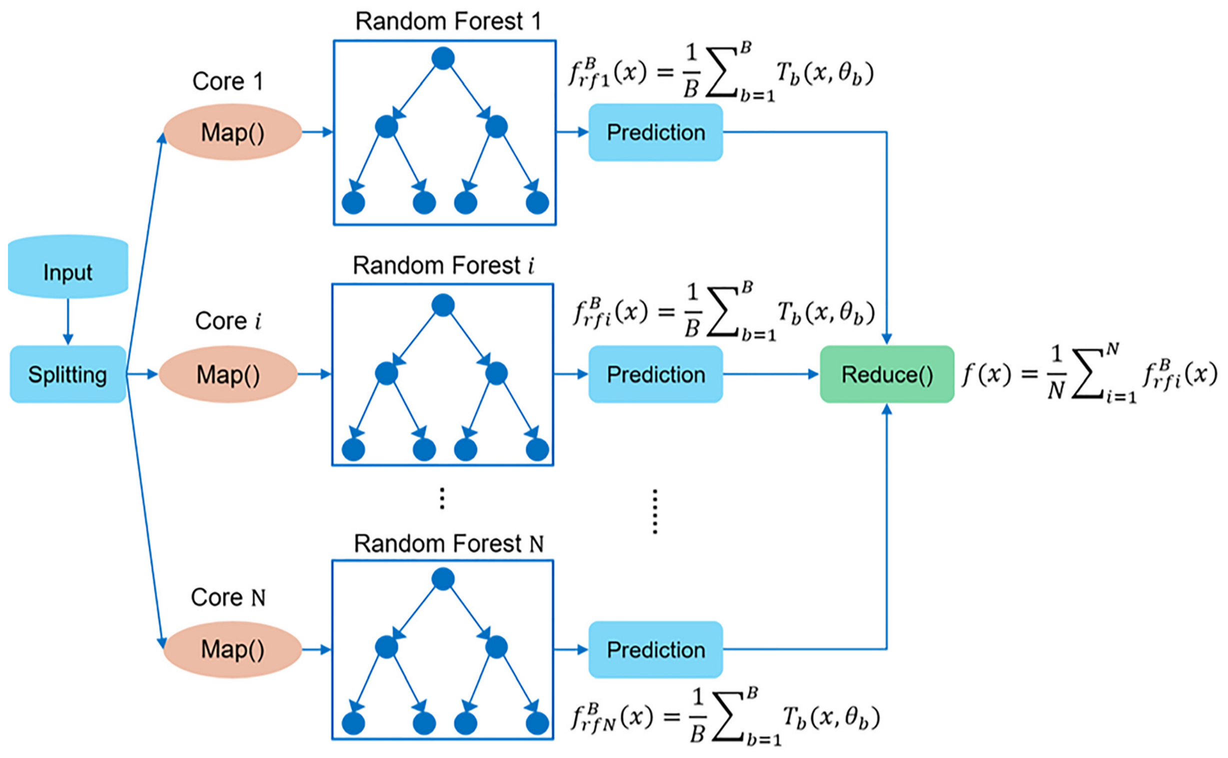
Random Forest 1
Core 1
Map()
$f_{rf1}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x,\theta_b)$
Prediction
Input
Splitting
Random Forest i
Core i
Map()
$f_{rfi}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x,\theta_b)$
Prediction
Reduce()
$f(x) = \frac{1}{N}\sum_{i=1}^{N} f_{rfi}^{B}(x)$
Random Forest N
Core N
Map()
Prediction
$f_{rfN}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x,\theta_b)$

Fig. 4 MapReduce-based PRF

training data set. Based on the method presented in Sec. 3, nine features ($m = (p/3), p = 28$) were randomly selected from 28 statistical features when splitting a parent node into two children nodes. For each split, a splitting variable and a cutting point were determined by solving Eq. (3.2). This splitting process will proceed until the stopping criterion (number of instances in a node is less than 5) is satisfied. Figure 3 illustrates the typical process of building binary regression trees. Figure 3 shows two simple illustrative examples of regression trees with nine nodes. The root nodes have 315 instances (cases). $F_{\max}$ and $V_{\max}$ were selected as the splitting variables, respectively. 0.1 and 0.05 were selected as the cutting points, respectively. Take the regression tree on the left as an example. The root node is split into two children nodes (left and right nodes). The instances with $F_{\max} < 0.1$ are in the right children node; The instances with $F_{\max} \geq 0.1$ are in the left children node. After building 10,000 regression trees of the RFs, the predicted value of tool wear on a new data point of the test data set can be calculated by taking the mean of all the predicted values from the regression trees.

Because constructing regression trees in RFs can be performed concurrently, the MapReduce programming model enables automatic parallelization and distribution of training predictive models using RFs on a single computing node with multiple cores or large clusters of commodity hardware.

Figure 4 shows a flowchart that illustrates the MapReduce-based PRFs algorithm. The detailed steps are as follows:

Step 1: Assign an ID to an individual bootstrap sample. This ID number is the key value associated with each bootstrap sample. The ID number ranges from 1 to N where N denotes the number of cores.

Step 2: Partition the bootstrap samples into N segments based on their key values. Assign the bootstrap samples with different key values to different CPU cores.

Step 3: Construct regression trees using the bootstrap samples on each core and generate a list of <key, value> pair (<sample ID, the value of tool wear>).

Step 4: Aggregate the <key, value> pairs with the same key value.

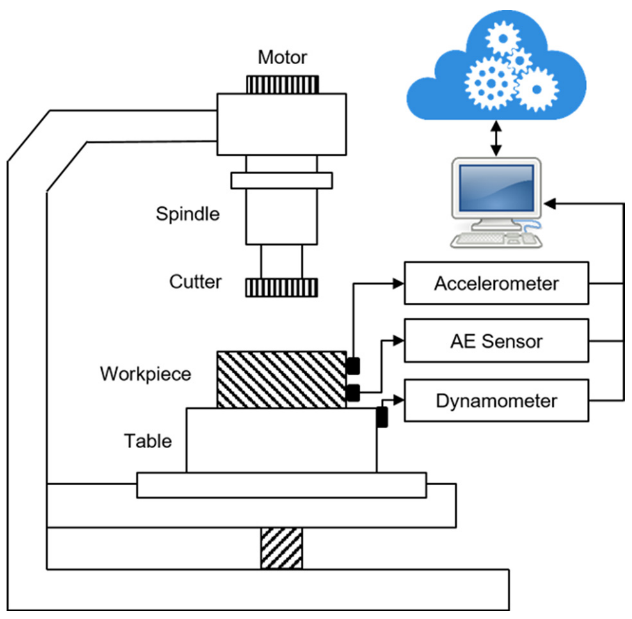

Fig. 5 Experimental setup

Table 1 Signal channels and measurement data

Signal channel	Measurement data
Channel 1	F_X: cutting force (N) in the X-axis
Channel 2	F_Y : cutting force (N) in the Y-axis
Channel 3	F_Z: cutting force (N) in the Z-axis
Channel 4	V_X: vibration (g) in the X-axis
Channel 5	V_Y: vibration (g) in the Y-axis
Channel 6	V_Z: vibration (g) in the Z-axis
Channel 7	AE: acoustic emission (V)

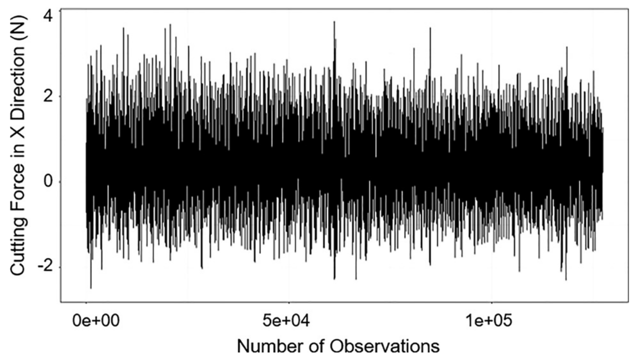

Fig. 6 Cutting force in X direction

5 Experimental Setup

Figure 5 illustrates the schematic diagram of the experimental setup. The experiment was conducted on a Röders Tech RFM 760 three-axis high-speed vertical CNC machine [45]. Seven signal channels, including cutting force, vibration, and acoustic emission, were monitored using a Kistler piezoelectric dynamometer, three Kistler piezoelectric accelerometers, and a Kistler acoustic emission sensor. The piezoelectric dynamometer was mounted on the table of the CNC to collect cutting force data in X, Y, Z dimensions. The piezoelectric accelerometers were mounted on a workpiece to collect vibration data in X, Y, Z dimensions. The AE sensor was also mounted on the workpiece to collect AE data during the milling experiment. AE occurs when a material undergoes irreversible changes (e.g., crack formation or plastic deformation) in its internal structure. Table 1 summarizes the signal channels and measurement data.

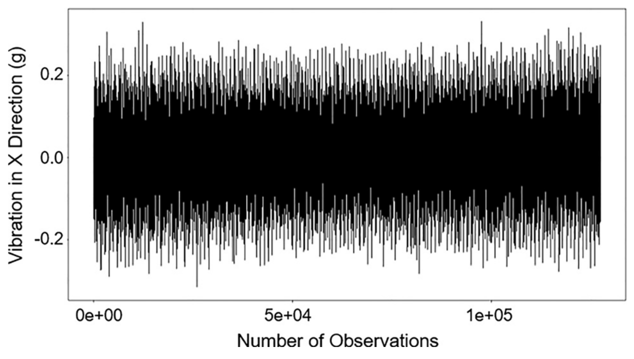

Fig. 7 Vibration in X direction

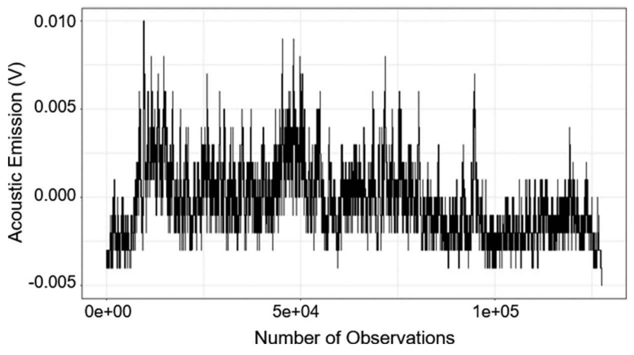

Fig. 8 Acoustic emission

Table 2 Operating conditions of the milling tests

Parameter	Value
Spindle speed	10,400 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

The example cutting force, vibration, and AE signals collected from the dynamometer, accelerometer, and AE sensors are shown in Figs. 6–8, respectively. Figures 6–8 show a total of 127,399 sampling signals collected from one cutting test.

The material of the workpiece used in the milling experiment was stainless steel. 315 cutting tests were conducted by the following two steps:

(1) Remove material from the workpiece using a predefined tool path;
(2) Measure the amount of tool wear using a LEICA MZ12 high-performance stereomicroscope.

Table 2 summarizes the operating conditions of the milling experiment. The total size of the condition monitoring data collected from 315 cutting tests is 9 GB.

6 Results and Discussions

6.1 Feature Generation and Extraction. In this section, feature generation and extraction are presented. Feature generation involves the process of defining statistical features or variables based on raw data collected from sensors. In this study, a set of

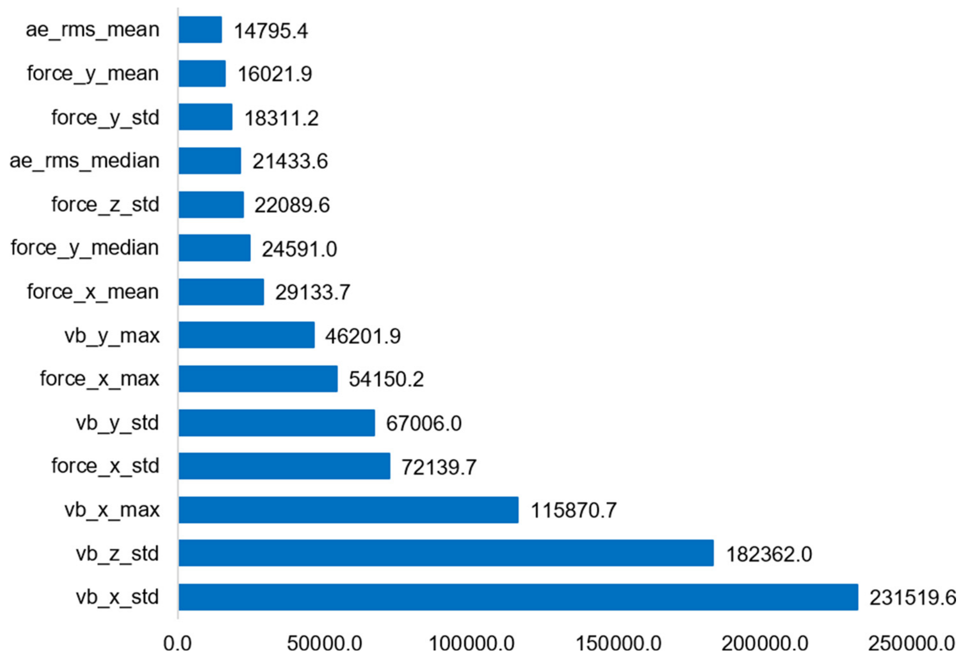

Fig. 9 Mean decrease in residual sum of squares/variable importance

statistical features (28 features), including maximum, median, mean, and standard deviation, was generated from the cutting force, vibration, and acoustic emission raw data. The importance of these features for predicting tool wear was evaluated using the variable importance metric expressed in Eq. (3.6). Figure 9 shows the variable importance scores for the 14 most important features. The statistical features with greater variable importance scores are more significant. For example, the standard deviation of vibration in the X direction (vb_x_std) with a feature importance score of 231,519.6 is the most significant feature.

6.2 Prediction of Tool Wear Based on Random Forests. After generating the statistical features, these feature data are fed into the RFs algorithm. A predictive model for tool wear prediction was trained using 10,000 regression trees. A total of 315 instances in the input data set was divided into training and validation data sets, respectively. To train the predictive model, two thirds of the 315 instances were used for the development of the predictive model. The remainder of the 315 instances was used for model validation. The tool wear prediction results are shown in Fig. 10. The data points in Fig. 10 represent observed (true) and predicted tool wear. If all of the data points fall on the straight line with a slope of 1, the accuracy of the predictive model is 100%. Figure 10 suggests that the predictive model trained by RFs can estimate tool wear with reasonably good prediction accuracy.

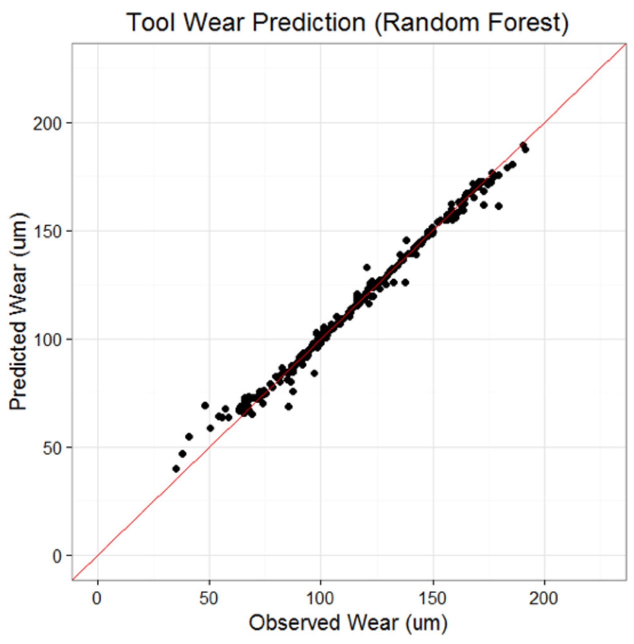

Fig. 10 Comparison of observed and predicted tool wear

Figure 11 shows a series of data points (observed and predicted tool wear data) indexed in time order. In comparison with Figs. 10 and 11 shows the trend of tool wear over time. The data points in green represent predicted tool wear. The data points in red represent observed tool wear. 95% prediction intervals were also calculated. A prediction interval estimates an interval within which the predicted tool wear is expected to lie with a specified probability. The results have shown that the tool life in milling operations include three stages, including break-in region with rapid initial wear rate, steady-state wear region with uniform wear rate, and failure region with accelerating wear rate.

To measure the performance of the predictive model trained by RFs, several common performance metrics, including mean squared error (MSE), coefficient of determination (R-squared), and training time, were used in this study. The MSE is defined as $\mathrm{MSE} = (1/n)\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2$ where $\widehat{Y}_i$ is a predicted value, Y_i is an observed value, and n is the sample size. The MSE measures the average of the squares of the errors. The coefficient of determination is defined as $R^2 = 1 - (\mathrm{SSE}/\mathrm{SST})$ where SSE is the sum of the squares of residuals and SST is the total sum of squares. The coefficient of determination is interpreted as the proportion of the variance in the dependent variable that can be predicted from the independent variable. If the R-squared value is equal to 1, all of the data points fall perfectly on the fitted regression line. If the R-squared value is equal to 0, the model explains none of the variability of the response data around its mean. The R-squared metric provides an indication of the goodness of fit of a set of predictions to the actual values. Table 3 summarizes the

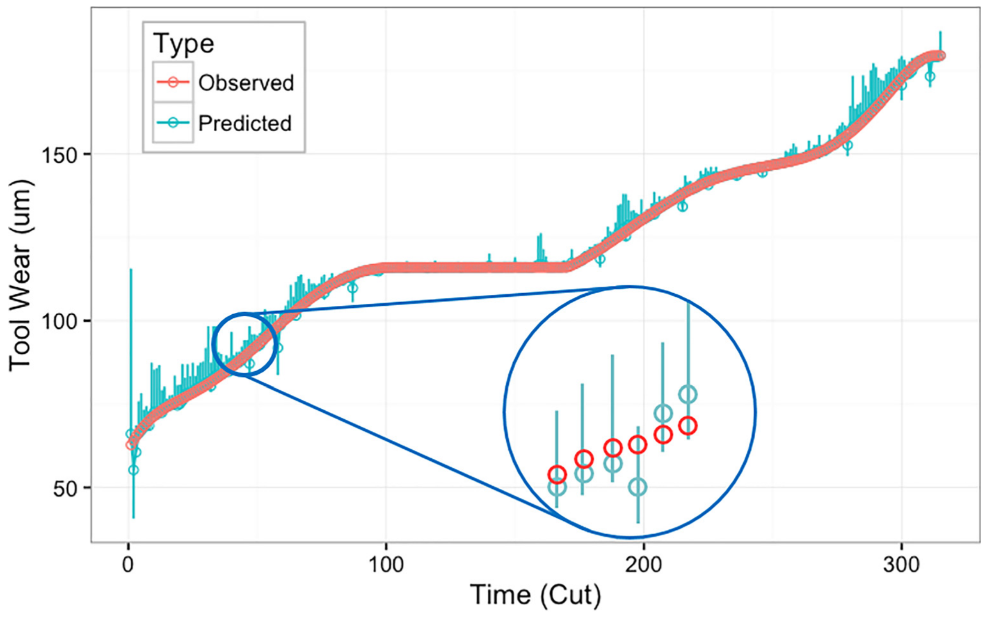

Fig. 11 Tool wear progression with prediction intervals

Table 3 MSE and R-squared values on test data and training time

	Random forests (10,000 trees)		
Training size (%)	MSE	R^2	Training time (s)
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

MSE, R-squared values, and training time when randomly sampling 50–90% of the total data as training data.

6.3 Performance Evaluation for Cloud-Based Parallel Random Forests. The MapReduce-based parallel RFs algorithm was implemented on the Amazon Elastic Compute Cloud (Amazon EC2). Amazon EC2 is a web service that provides scalable high performance computing capacity on the Amazon Web Services cloud. In comparison with traditional clusters or supercomputers, Amazon EC2 runs instances on its physical infrastructure using the open-source virtualization middleware Xen. Various configurations of CPU cores, memory, storage volumes, and operating systems, also known as instance types, are provided on the Amazon EC2 cloud platform. In this study, two instance types were selected to evaluate the performance of the MapReduce-based parallel RFs. Table 4 summarizes the detailed hardware configurations of the C3.8 and R3.8 instances. The C3.8 $\times$ large instance type has an Intel Xeon E5-2680V2 processor, 32 virtual cores, 60 GB of memory, and 640 GB of solid state drive storage. C3 instances are optimized for compute-intensive applications. The R3.8 $\times$ large instance type has an Intel Xeon E5-2670V2 processor, 32 virtual cores, 244 GB of memory, and 640 GB of solid state drive storage. R3 instances are optimized for memory-intensive applications.

To evaluate the performance of the MapReduce-based parallel RFs, two performance metrics, including training time and relative speedup ratio, were used. The time to train a predictive model varies depending on the amount of training data and computing

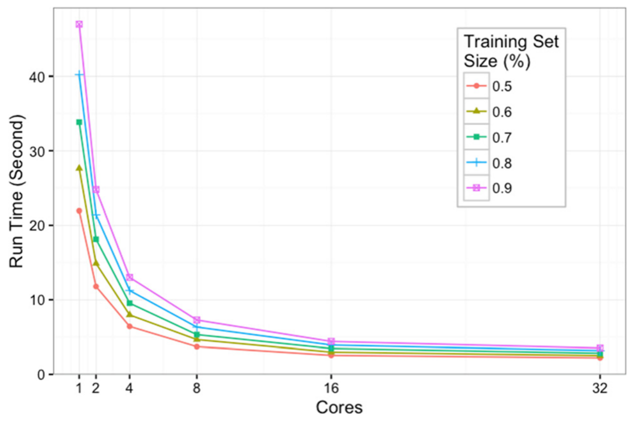

Fig. 12 Training time for C3 instances

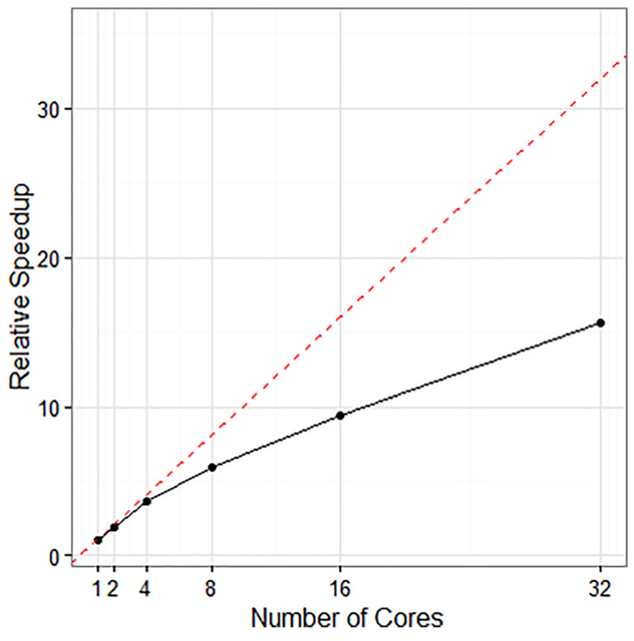

Fig. 13 Relative speedup ratio for C3 instances

Table 4 Hardware configurations for Amazon EC2 instances

Instance type	C3.8 × large	R3.8 × large
Operating system	Linux	Linux
Processor	Intel Xeon E5-2680 v2 (2.80 GHz)	Intel Xeon E5-2670 v2 (2.50 GHz)
Number of virtual CPU	32	32
Memory (GB)	60	244
Storage (GB)	640	640

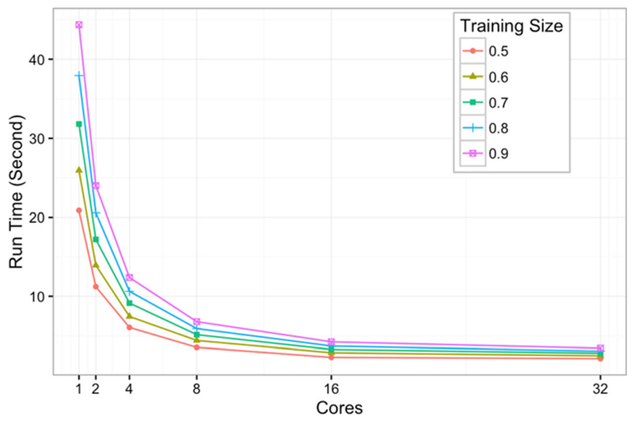

Fig. 14 Training time for R3 instances

capacity. Figure 12 shows the average training time with different amount of training data and cores. For example, the curve in red represents the average training time to train the predictive model with 50% of the total amount of data. Similarly, the curve in pink shows the average training time to train the predictive model with 90% of the total amount of data. The training times are 21, 28, 34, 40, and 47 s using one core, respectively. As expected, the training time increases as the training data increase.

In addition, to assess the performance of the MapReduce-based PRFs, the predictive model was trained with 1, 2, 4, 8, 16, and 32 cores on the different amount of training data. For example, it took 21, 11, 7, 4, 3, and 2 s to train the predictive model with 1, 2, 4, 8, 16, and 32 cores, respectively, when 50% of the total amount of data was used for training. It took 47, 25, 13, 8, 5, and 3 s to train the predictive model with 1, 2, 4, 8, 16, and 32 cores, respectively, when 90% of the total amount of data was used for training. Relative speedup ratio measures the relationship between the sequential execution time and the parallel execution time solving the same problem. Figure 13 shows the relative speedup ratios when 90% of the total amount of data was used as training data. The results show that the MapReduce-based PRFs achieved a near linear speedup for 1–8 cores and a sublinear speedup for 16–32 cores, respectively.

Because the C3 instance is optimized for compute-intensive applications, the MapReduce-based RPF was also executed on the R3 instance, which is optimized for memory-intensive applications. Figure 14 shows the training time. For example, it took 21, 11, 7, 4, 3, and 2 s to train the predictive model with 1, 2, 4, 8, 16, and 32 cores, respectively, when 50% of the total amount of data was used for training. It took 44, 23, 12, 7, 4, and 3 s to train the predictive model with 1, 2, 4, 8, 16, and 32 cores, respectively, when 90% of the total amount of data was used for training. Figure 15 shows the relative speedup ratios when 90% of the total amount of data was used as training data. The results show that the MapReduce-based PRFs achieved a near linear speedup for 1–8 cores and a sublinear speedup for 16–32 cores, respectively. The results show that the training time with the R3 instance is almost the same as that of the C3 instance.

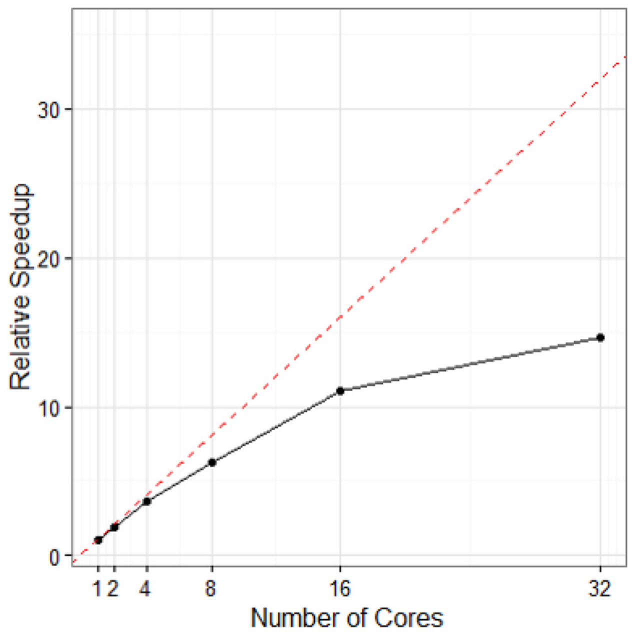

Fig. 15 Relative speedup ratio for R3 instances

7 Conclusions and Future Work

In this paper, prediction of flank tool wear in high-speed machining was conducted with RFs and MapReduce-based PRFs algorithms. The MapReduce-based PRFs algorithm was implemented on the Amazon EC2 cloud. The condition monitoring data, including cutting force, vibration, and acoustic emission, collected from 315 milling tests were used to evaluate performance of the algorithms. A set of statistical features was generated as the input of the machine learning algorithms. The performance metrics include MSE, R-squared, and training time. The experimental results have shown that RFs can predict tool wear very accurately with the condition monitoring data. The importance of the statistical features can be measured using RFs. In addition, the prediction intervals associated with tool wear predictions were computed to measure uncertainty in tool wear prediction. Moreover, the MapReduce-based PRFs algorithm was developed to increase the efficiency of the original RFs algorithm. The experimental results have shown that a significant increase in training time (15 times with 32 cores) has been achieved by parallelizing the original RFs with two Amazon EC2 instances. In the future, the MapReduce PRFs will be implemented on a cloud with multiple computing nodes to evaluate the scalability of the algorithm. Efforts will also be focused on evaluating the performance of the algorithm on large volumes of streaming data from multiple CNC machines.

Acknowledgment

The research reported in this paper is partially supported by the University of Central Florida and DMDII-15-14-01. 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 University of Central Florida and the Digital Manufacturing and Design Innovation Institute.

References

[1] Valdez-Flores, C., and Feldman, R. M., 1989, "A Survey of Preventive Maintenance Models for Stochastically Deteriorating Single-Unit Systems," Nav. Res. Logist., **36**(4), pp. 419–446.

[2] Wu, D., Terpenny, J., Zhang, L., Gao, R., and Kurfess, T., 2016, "Fog-Enabled Architecture for Data-Driven Cyber-Manufacturing Systems," ASME Paper No. MSEC2016-8559.

[3] Hu, C., Youn, B. D., and Kim, T., 2012, "Semi-Supervised Learning With Co-Training for Data-Driven Prognostics," IEEE Conference on Prognostics and Health Management (PHM), Denver, CO, June 18–21, pp. 1–10.

[4] Gao, R., Wang, L., Teti, R., Dornfeld, D., Kumara, S., Mori, M., and Helu, M., 2015, "Cloud-Enabled Prognosis for Manufacturing," CIRP Ann. Manuf. Technol., **64**(2), pp. 749–772.

[5] Daigle, M. J., and Goebel, K., 2013, "Model-Based Prognostics with Concurrent Damage Progression Processes," IEEE Trans. Syst. Man Cybern. Syst., **43**(3), pp. 535–546.

[6] Si, X.-S., Wang, W., Hu, C.-H., Chen, M.-Y., and Zhou, D.-H., 2013, "A Wiener-Process-Based Degradation Model With a Recursive Filter Algorithm for Remaining Useful Life Estimation," Mech. Syst. Signal Process., **35**(1), pp. 219–237.

[7] Dong, M., and He, D., 2007, "Hidden Semi-Markov Model-Based Methodology for Multi-Sensor Equipment Health Diagnosis and Prognosis," Eur. J. Oper. Res., **178**(3), pp. 858–878.

[8] Saha, B., Goebel, K., and Christophersen, J., 2009, "Comparison of Prognostic Algorithms for Estimating Remaining Useful Life of Batteries," Trans. Inst. Meas. Control, **31**(3–4), pp. 293–308.

[9] Wang, P., and Gao, R. X., 2015, "Adaptive Resampling-Based Particle Filtering for Tool Life Prediction," J. Manuf. Syst., **37**(Part 2), pp. 528–534.

[10] Malhi, A., Yan, R., and Gao, R. X., 2011, "Prognosis of Defect Propagation Based on Recurrent Neural Networks," IEEE Trans. Instrum. Meas., **60**(3), pp. 703–711.

[11] Sick, B., 2002, "On-Line and Indirect Tool Wear Monitoring in Turning With Artificial Neural Networks: A Review of More Than a Decade of Research," Mech. Syst. Signal Process., **16**(4), pp. 487–546.

[12] Siddhpura, A., and Paurobally, R., 2013, "A Review of Flank Wear Prediction Methods for Tool Condition Monitoring in a Turning Process," Int. J. Adv. Manuf. Technol., **65**(1–4), pp. 371–393.

[13] Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., and Siegel, D., 2014, "Prognostics and Health Management Design for Rotary Machinery Systems—Reviews, Methodology and Applications," Mech. Syst. Signal Process., **42**(1), pp. 314–334.

[14] Zhang, J., and Lee, J., 2011, "A Review on Prognostics and Health Monitoring of Li-Ion Battery," J. Power Sources, **196**(15), pp. 6007–6014.

[15] Breiman, L., 2001, "Random Forests," Mach. Learn., **45**(1), pp. 5–32.

[16] Biau, G., 2012, "Analysis of a Random Forests Model," J. Mach. Learn. Res., **13**(1), pp. 1063–1095.

[17] Wang, L., 2008, "Wise-Shopfloor: An Integrated Approach for Web-Based Collaborative Manufacturing," IEEE Trans. Syst. Man Cybern., Part C, **38**(4), pp. 562–573.

[18] Wang, L., 2013, "Machine Availability Monitoring and Machining Process Planning Towards Cloud Manufacturing," CIRP J. Manuf. Sci. Technol., **6**(4), pp. 263–273.

[19] Tao, F., Zuo, Y., Da, X. L., and Zhang, L., 2014, "IoT-Based Intelligent Perception and Access of Manufacturing Resource Toward Cloud Manufacturing," IEEE Trans. Ind. Inf., **10**(2), pp. 1547–1557.

[20] Tao, F., Cheng, Y., Da Xu, L., Zhang, L., and Li, B. H., 2014, "CCIoT-CMfg: Cloud Computing and Internet of Things-Based Cloud Manufacturing Service System," IEEE Trans. Ind. Inf., **10**(2), pp. 1435–1442.

[21] Niaki, F. A., Ulutan, D., and Mears, L., 2015, "In-Process Tool Flank Wear Estimation in Machining Gamma-Prime Strengthened Alloys Using Kalman Filter," Procedia Manuf., **1**, pp. 696–707.

[22] Wang, P., Gao, R. X., Wu, D., and Terpenny, J., 2016, "A Computational Framework for Cloud-Based Machine Prognosis," 49th CIRP Conference on Manufacturing Systems (CIRP-CMS), Stuttgart, Germany, May 25–27, pp. 309–314.

[23] Wu, D., Rosen, D. W., Wang, L., and Schaefer, D., 2015, "Cloud-Based Design and Manufacturing: A New Paradigm in Digital Manufacturing and Design Innovation," Comput. Aided Des., **59**, pp. 1–14.

[24] Wu, D., Greer, M. J., Rosen, D. W., and Schaefer, D., 2013, "Cloud Manufacturing: Strategic Vision and State-of-the-Art," J. Manuf. Syst., **32**(4), pp. 564–579.

[25] Xu, X., 2012, "From Cloud Computing to Cloud Manufacturing," Rob. Comput. Integr. Manuf., **28**(1), pp. 75–86.

[26] Bi, Z., Da, X. L., and Wang, C., 2014, "Internet of Things for Enterprise Systems of Modern Manufacturing," IEEE Trans. Ind. Inf., **10**(2), pp. 1537–1546.

[27] Lee, J., Bagheri, B., and Kao, H. A., 2015, "A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems," Manuf. Lett., **3**, pp. 18–23.

[28] Wang, C., Bi, Z., and Da Xu, L., 2014, "IoT and Cloud Computing in Automation of Assembly Modeling Systems," IEEE Trans. Ind. Inf., **10**(2), pp. 1426–1434.

[29] Mourtzis, D., Vlachou, E., Xanthopoulos, N., Givehchi, M., and Wang, L., 2016, "Cloud-Based Adaptive Process Planning Considering Availability and Capabilities of Machine Tools," J. Manuf. Syst., **39**, pp. 1–8.

[30] Mai, J., Zhang, L., Tao, F., and Ren, L., 2016, "Customized Production Based on Distributed 3D Printing Services in Cloud Manufacturing," Int. J. Adv. Manuf. Technol., **84**(1–4), pp. 71–83.

[31] Kamarthi, S., Kumara, S., and Cohen, P., 2000, "Flank Wear Estimation in Turning Through Wavelet Representation of Acoustic Emission Signals," ASME J. Manuf. Sci. Eng., **122**(1), pp. 12–19.

[32] Bukkapatnam, S. T., Kumara, S. R., and Lakhtakia, A., 2000, "Fractal Estimation of Flank Wear in Turning," ASME J. Dyn. Syst. Meas. Control, **122**(1), pp. 89–94.

[33] Taylor, F. W., 1907, *On the Art of Cutting Metals*, American Society of Mechanical Engineers, New York.

[34] Ertunc, H. M., Loparo, K. A., and Ocak, H., 2001, "Tool Wear Condition Monitoring in Drilling Operations Using Hidden Markov Models (HMMs)," Int. J. Mach. Tools Manuf., **41**(9), pp. 1363–1384.

[35] Atlas, L., Ostendorf, M., and Bernard, G. D., 2000, "Hidden Markov Models for Monitoring Machining Tool-Wear," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul, Turkey, June 5–9, pp. 3887–3890.

[36] Zhu, K., San Wong, Y., and Hong, G. S., 2009, "Multi-Category Micro-Milling Tool Wear Monitoring With Continuous Hidden Markov Models," Mech. Syst. Signal Process., **23**(2), pp. 547–560.

[37] Wang, J., Wang, P., and Gao, R. X., 2015, "Enhanced Particle Filter for Tool Wear Prediction," J. Manuf. Syst., **36**, pp. 35–45.

[38] Elangovan, M., Devasenapati, S. B., Sakthivel, N., and Ramachandran, K., 2011, "Evaluation of Expert System for Condition Monitoring of a Single Point Cutting Tool Using Principle Component Analysis and Decision Tree Algorithm," Expert Syst. Appl., **38**(4), pp. 4450–4459.

[39] Shi, D., and Gindy, N. N., 2007, "Tool Wear Predictive Model Based on Least Squares Support Vector Machines," Mech. Syst. Signal Process., **21**(4), pp. 1799–1814.

[40] Wu, D., Jennings, C., Terpenny, J., Gao, R., and Kumara, S., 2017, "A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests," ASME J. Manuf. Sci. Eng., **139**(7), p. 071018.

[41] Meinshausen, N., 2006, "Quantile Regression Forests," J. Mach. Learn. Res., **7**, pp. 983–999.

[42] Dean, J., and Ghemawat, S., 2008, "MapReduce: Simplified Data Processing on Large Clusters," Commun. ACM, **51**(1), pp. 107–113.

[43] Chu, C., Kim, S. K., Lin, Y.-A., Yu, Y., Bradski, G., Ng, A. Y., and Olukotun, K., 2007, "Map-Reduce for Machine Learning on Multicore," Adv. Neural Inf. Process. Syst., **19**, p. 281.

[44] Censor, Y., and Zenios, S. A., 1997, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, New York.

[45] Li, X., Lim, B., Zhou, J., Huang, S., Phua, S., Shaw, K., and Er, M., 2009, "Fuzzy Neural Network Modelling for Tool Wear Estimation in Dry Milling Operation," Annual Conference of the Prognostics and Health Management Society, San Diego, CA, Sept. 27-Oct. 1, pp. 1–11.