# A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms

Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop
School of Computer Engineering, Suranaree University of Technology (SUT), Thailand
Email: banjongkan@gmail.com, watthana.p@sskru.ac.th, {nittaya, kerdpras}@sut.ac.th

***Abstract*—In High-Performance Computing (HPC) system, job failure is a major problem because it means the losses in computation time, resources, and power. Job failure also degrades significantly overall efficiency of the HPC system. In this paper, we propose two sets of models to predict job failure at two points of submission: job submit-state and job start-state. The models can be used as guiding tools for HPC-user to make efficient decision on managing their job submisison on the HPC system. The tools are thus for improving the efficiency of the HPC system at the job level. In the evaluation stage, we conduct a comparative study in order to compare performance of the job failure predictive models developed based on the decision-tree induction techniques including C5.0, Classification and Regression Tree (CART), and Chi-square Automatic Interaction Detector (CHAID). The datasets used for training and testing the models are the two workload logs collected from the HPC system at the National Electronics and Computer Technology Center (NECTEC), Thailand, and the Los Alamos National Laboratory (LANL), USA. To predict failure at the job submit-state and at the job start-state, the results show that the models built from C5.0 algorithm provide the highest accuracy of prediction (around 85% for the NECTEC dataset and 87% for the LANL dataset). The experimental results regarding prediction at different job states reveal that failure forecasting at the job start-state is slightly more accurate than making prediction at the job submit-state (accuracy improvement is around 1.45% for the NECTEC dataset and 0.46% for the LANL dataset). However, when considering both criteria of the performance of the models and the overhead of job waiting time, job failure prediction modeling at the job submit-state provides the best efficiency.**

***Index Terms*—decision tree, high-performance computing, job failure prediction, workload log**

## I. Introduction

High-Performance Computing (HPC) is a computer system that combines many computers working together as a single system. Each of computer machines is referred to as a computing node. To achieve a single system characteristic, a group of computers communicate with each other via the internal network system. The HPC system has a special software named a scheduler to prioritize and manage the jobs in the queue to appropriately and efficiently be processed in the computing nodes. The HPC system uses central storage to guarantee correctness by making all computing nodes to access the same data. The two important aspects of most HPC systems are reliability and scalability.

To deliver reliable service, the HPC system operates with the redundancy principle such that it can resist the system failure. The second feature of HPC system, named scalability, refers to the ability to expand the system hardware to handle jobs that need extremely large number of computing nodes. The fundamental idea of combining many computers to work together as a single cluster makes the HPC system naturally effective in expanding its computing resources such that its computing power is unlimited. Based on these two important features, HPC systems are thus installed and operated in many computational laboratories worldwide.

Most of the jobs processed in the HPC system are related to advanced computational science and engineering tasks [1]. These jobs have common characteristics of high-memory consumption and complex calculation, such as jobs from the astronomical computing group, particle and high energy physics, the forecasting of climate, and so on. Most HPC systems are established as either the organization's computing resource center or open to public as a computing service with hourly service fees. Based on the vast areas of service, HPC systems must be able to support a variety of tasks. Variety issue also includes the different number of users, the variance of computational domains, and the variety of computational applications. To handle efficiently diverse tasks in the HPC systems, there are three levels of efficiency to be considered (as shown in Fig. 1).

The efficiency development at the system level is the maintenance of computing resources to be able to work or provide computational services at all time [2], [3]. At scheduling level, the HPC system must manage the submitted jobs as much as possible according to the full capacity of computing resources [4], [5]. The last one is at the job level which makes the jobs processed in the HPC

Manuscript received July 20, 2020; revised January 19, 2021.


doi: 10.12720/jait.12.2.84-92

system run through the finish-state with a high success rate [6], [7]. For the efficiency development of the HPC system at the system level and the scheduling level, it is the responsibility of the system administrator. Meanwhile, efficiency at the job level is the direct responsibility of users of the HPC system.

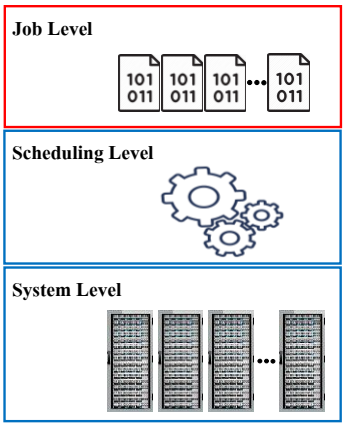



Figure 1. The level of efficiency development of the HPC system.

This research aims to improve the computing efficiency of the HPC system at the job level. Unsuccessful processing or job failure while processing in the HPC system is the most important problem for users because most jobs submitted to the HPC system is a large and complex job requiring long processing time and high usage of computing resources. If the job processing fails at running state, that means the waste of time, computing resources, and electrical power [8]. This research proposes a job failure predictive model while the job is processing at the job submit-state and the job start-state in the HPC system. The model can be further developed as a guiding tool for the HPC-users in a soft recommendation system or as a notifier in a monitoring system. We expect that such tool would be very useful for many HPC-users. For example, it is a decision-making helper for new users who lack experience of setting parameters while submitting a job to be processed in the HPC system, or it can issue notification when an error occurs by providing some helpful message for users to correctly handle the erroneous situation.

Machine Learning (ML) technique was used to create the job failure predictive model to predict a job failure while processing in the HPC system. We select three Decision Tree (DT) algorithms: C5.0, Classification and Regression Tree (CART), and Chi-square Automatic Interaction Detector (CHAID) to create the models based on the workload logs or job logs of the HPC systems. The first dataset is a workload log from the National Electronics and Computer Technology Center (NECTEC), Thailand. The second dataset belongs to the HPC system at the Los Alamos National Laboratory (LANL), USA, which is publicly available at the Parallel Workloads Archive [9].

The contributions of this research are as follows.

- This research proposes a framework to improve the computing efficiency of the HPC system at the job level by creating the model to predict job failure while the job is processed in the HPC system. The model can help users to submits a job to the HPC system in an effective way.
- We evaluate and compare the performance of several models to find the best one. The predictive tree-based models were created by using the algorithms C5.0, CART, and CHAID trained on the workload logs of the HPC systems.
- We perform a comparative study to assess model's performance at the job submit-state and the job start-state to find a suitable position to create the job failure predictive model.

The rest of this research is organized as follows. Section II presents a literature review. Section III describes theory and algorithms used in this research. Section IV explains details of the dataset, experimentation setting and results. Sections V and VI are discussion and conclusion, respectively.

## II. Literature Review

The improve in computing efficiency of the HPC system can be done at 3 levels: the system level, the scheduling level, and the job level. Many research works propose the development of computing efficiency of the HPC system at the system level by preventing system from failure using analytical method to find patterns of system defects. Such patterns are in a form of a predictive model to forecast system failure in advance.

In 2006, Schroeder and Gibson [10] proposed the idea to investigate the root causes of the problem that occur on the HPC system with the objective to estimate the Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR). They used static tools to analyze the defect. The dataset used in their research was collected from 20 HPC systems at the Los Alamos National Laboratory, USA. The analysis results showed that the number of faults in the HPC system in one year was in a very wide range from 20 times to 1,000 times and defects mostly occurred in the large HPC system.

In 2016, Chuah *et al.* [11] also studied patterns of defects or system failure on the HPC system called Ranger Supercomputer. They introduced the CRUMEL (Correlating Resource Usage data and MEssage Logs) framework, which used data analysis principles based on correlation relationships between the system log and the workload dataset. The data from two sources were connected via timestamp. They reported the results that the CRUMEL tool could identify the pattern of defects. Their framework could also show the relationship of fault events on the system.

In 2007, Liang *et al.* [12] analyzed a system log of the IBM BlueGene/L system to predict system failures. The dataset was recorded in a period of 142 days. They tested and compared the efficiency of the models for predicting the system failure of the HPC system. The models were created by Rule-based Method, Support Vector Machine (SVM) and k-Nearest Neighbors (kNN). The results showed that the best performance in terms of time complexity and accuracy was from the model of the kNN algorithm.

In the same year, Gujrati *et al.* [13] also used Rule-based Method and statistical tools to create models to predict abnormal events in the IBM Blue Gene/L system. Their tools consisted of three processes: event processing, based prediction, and the Meta-learning prediction. The dataset was collected from the system log of two IBM Blue Gene/L systems. They reported that experimental results provided by the proposed tool could increase performance of system failures prediction by up to 3 times compared to their previous research.

In 2018, Soysal *et al.* [14] proposed a method to predict how long the job is to be processed in the HPC system (called wall-time). Their modeling method was based on the automated machine learning using 15 algorithms trained with the workload dataset of the HPC system that was collected from the Parallel Workloads Archive. The results showed that the models built from automated machine learning apprpach provided better performance than human predictions up to 7 times.

In 2012, Zhang *et al.* [15] proposed a descriptive analysis method using statistical tools and clustering algorithm to analyze the workload log data of the HPC system with the objective to find a suitable form of resource usage for the jobs to be processed in the HPC system. Their analysis scheme also employed information obtained from the experiment to create a tool for suggesting job submission to the system, called a knowledge-based recommendation system. The system is for users who lack experience in submitting job to the HPC system. Users were satisfied with the system that showed accuracy as high as 64.2%.

In 2012, Yuan *et al.* [16] studied the nature of a job that had not been successfully processed in the HPC system. They used statistical method to analyze the workload log dataset of the HPC system. The data were collected from 10 public datasets compiled from 8 HPC systems. The results showed that the unsuccessful jobs had much effect toward the overall efficiency of the HPC system in terms of service quality as well as wasted computation time.

From the literature review, we found that researchers focused on developing methods to improve the computing efficiency of the HPC system at two levels: the system level and the job level. The majority of research work concerned improvement the computing efficiency of the HPC system at the system level. Some researchers performed log analysis at the system level to find the root cause of the problems in the HPC system. Many recent works aimed at creating the predictive model to forecast the system failure of the HPC system. The predictive model was created by using statistical tools and machine learning techniques. There was some research work aiming at improving the computing efficiency of the HPC system at the job level by trying to find the appropriate form of requesting computing resources for the job to be processed in the HPC system. However, there are some existing limitations such as the model could not be applied to the real world HPC system because the model was built from the jobs at the finish-state and such state is not so useful for applying the model to the actual situation that job finish-state has not been reached yet. Therefore, we propose a new study framework by performing a comparative study of the job failure predictive model where jobs are at the submit-state and the start-state.

## III. Background Theory and Algorithms

### A. The Job State in the HPC System

The job state is the various statuses of job that has been processed through the HPC system (as shown in Fig. 2). There are three possible states; the first one is the state at which the job has been sent or submitted to the HPC system (called job submit-state), the second state is the state at which the job begins to be processed on the HPC system (called job start-state), and the last one is the state at which the job is processed completly and successfully (called job finish-state).

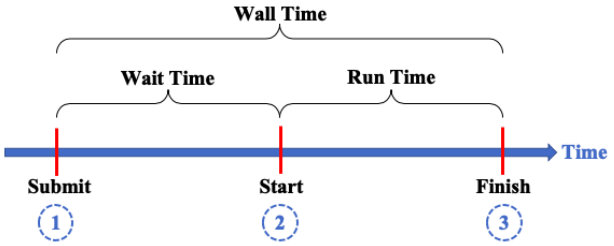



Figure 2. The three job states in the HPC system.

The time period from the job submit-state to the job start-state is called the wait-time. During this time, the job is waiting for its processing in the queue system. The next period from the job start-state to the job finish-stae, which is the end of job processing in the HPC system, is called the run-time or execute-time of the job. The whole duration of the job in the HPC system is called the wall-time (as in equation 1). It is a time frame from the point that users submitting their job until they receive the result.

$$\textit{Wall Time} = \textit{Wait Time} + \textit{Run Time} \quad (1)$$

We are interested in evaluating and comparing the performance of models to predict job failure while processing in the HPC system. We analysis job failure where the job is at the submit-state and the start-state with the main purpose of finding the most efficient model. The job failure predictive model can help the HPC-user making good decision on submitting job with the most efficient configuration of job processing in the HPC system. In some cases, if users know in advance at the submit-state that the job to be sent to the HPC system is likely to fail, they can avoid the useless computation and not to lose the job wall-time. Also, in the situation of the job that is in the start-state, the model can help users saving time for job processing (job run-time).

### B. Decision Tree

Usually, the purpose of analyzing data from log files [17], [18] whether it is system log, network log, or workload log data, is to find the cause of problems (root cause) or to find defect that causes suboptimal performance of the system. Such analysis has be done regularly using the manual style that requires the

knowledge of experts to make decision about which part of data should be used, which analysis tool should be employed, and how to interpret the analysis results. So, it can be seen that it is an inefficient method of data analysis because it takes a long time for getting the correct result and it is limited by experience of the expert. Therefore, in this research we propose to use machine learning technique to create a model to predict job failure while processing in the HPC system. Machine learning technique is more efficient than manual analysis because it works in an automatic way; thus, limitations in time and expertise of the human analyst are eliminated.

This research adopts the decision tree algorithm as a modeling method to predict job failures while processing in the HPC system. The advantagees of a decision tree algorithm are that it has a simple work process, accuracy of the model is high, and the model is easy for interpretation, which is especially useful for root cause analysis. The decision tree model can classify or predict the target attribute with the logic-based concept, which is like the reasoning generally made by humans [19]. The structure of the decision tree is in the form of an upside-down tree (as illustrated in Fig. 3). The top node of the decision tree is the root node which is the node that has only the branching out lines. The node that has both the input output lines is called the internal node. The node at the end of the tree structure, in which there are only the input lines, is the leaf node. The leaf node is responsible for showing the final result of the classification or prediction of the decision tree model.

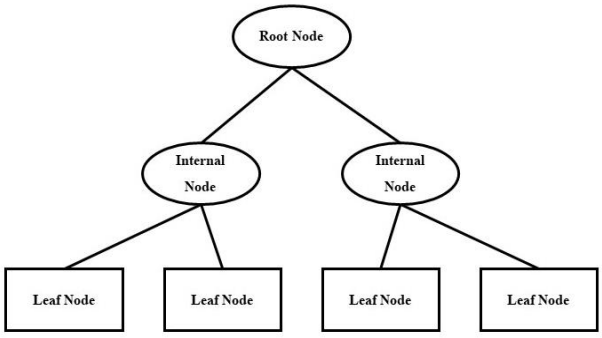



Figure 3. The decision tree structure.

This research builds the tree-based model to predict job failure while processing in the HPC system using three algorithms: C5.0, CART, and CHAID. The reasons for choosing theses three algorithms are due to their efficiency and successful application in many domains.

C5.0 algorithm [20], [21] was developed from the C4.5 algorithm. The main extension from the previous version is that C5.0 algorithm has some new features such as Boosting and Cost-sensitive tree. The C5.0 algorithm uses the node splitting criteria from calculating a value called Information Gain. One prominent advantage of C5.0 algorithm is its robustness against missing values in the dataset. The C5.0 algorithm only supports the categorical target variables. While, the CART algorithm [21], [22] is an outstanding algorithm that be able to support both data classification and data prediction as the target data can be either categorical or continuous. For classification, branching criteria for decision tree construction of the CART algorithm is the Gini index value, whereas in prediction, CART uses variance reduction value. The CHAID algorithm [23], [24] works like CART, but it has the advantage of being able to support branching of decision trees in more than two subgroups. It uses the statistical criteria based on the chi-square value for branching.

### *C. Assessment*

This research generates job failure models based on three different algorithms. Performance of the obtained models are to be evaluated and compared using the four assessment matrices: accuracy, recall, precision, and $F_1$ score. The computation of each assessment metric can be done by observing values from a confusion matrix. Structure of confusion matrix is shown in Table I, while computation of various assesments are summarized in Table II.

TABLE I. THE CONFUSION MATRIX OF BINARY LABEL CLASSIFICATION

| | Predict as Positive | Predict as Negative |
|---|---|---|
| Actual Positive | True Positive (TP) | False Negative (FN) |
| Actual Negative | False Positive (FP) | True Negative (TN) |

True Positive (TP) = The number of data that their actual class is “true” and the model predicts correctly as “true”.

False Negative (FN) = The number of data that their actual class is “true”, but the model predicted incorrectly as “false”.

False Positive (FP) = The number of data that their actual class is “false”, but the model predicted incorrectly as “true”.

True Negative (TN) = The number of data that their actual class is “false” and the model predicts correctly as “false”.

TABLE II. THE MODEL ASSESSMENT METRICS

| No. | Metric Name | Formula |
|---|---|---|
| 1 | Accuracy | $\frac{TP+TN}{TP+FP+TN+FN}$ |
| 2 | Recall | $\frac{TP}{TP+FN}$ |
| 3 | Precision | $\frac{TP}{TP+FP}$ |
| 4 | $F_1$ score | $2\times\frac{Recall\times Precision}{Recall+Precision}$ |

Accuracy is an assessment metric that considers the overall classification accuracy (for both “true” and “false” classes) of the model. It may not be a good metric is the dataset is imbalance in that number of data in one class significantly outnumbers data in other class. For such imbalance cases, we can use other assessment metrics such as recall (or sensitivity), precision, and $F_1$ score for a specific class of interest (which is normally called a positive class). The recall is used to evaluate the model performance from a perspective of the power to predict correctly as much as possible the data from the class of interest. While precision is the assessment of the model

from the aspect of correctness that a good model should not incorrectly predict data in negative class (those that are out of interest) to be a positive class. The $F_1$ score is the harmonic mean of the values from recall and precision. The range of these values are from 0 to 1. The value 1 is the most desirable measurement.

## IV. Experimentation and Results

### A. Dataset

The data used in this research are the workload log or job log of the HPC system. It is the result of recording the activity related to the jobs processed on the HPC system, which are recorded by the scheduler. The dataset used in this research is the HPC-workload logs from two HPC systems. The first one is the public dataset obtained from the Parallel Workloads Archive which is the workload log of the large cluster computers of the Los Alamos National Laboratory (LANL) in the United States. The cluster computers consist of the Origin 2000 computers, a total of 2,048 nodes and use the LSF Scheduler. The dataset of LANL contains the data from December 1999 to April 2000, with a data size of 122,233 elements. The second dataset is a workload log of a small cluster computer that has approximately 500 processing units of the X86_64 computer. This cluster computer belongs to the National Electronic and Computer Technology Center (NECTEC) of Thailand. This dataset was recorded by PBS/Torque scheduler with a total data size of 87,046 elements.

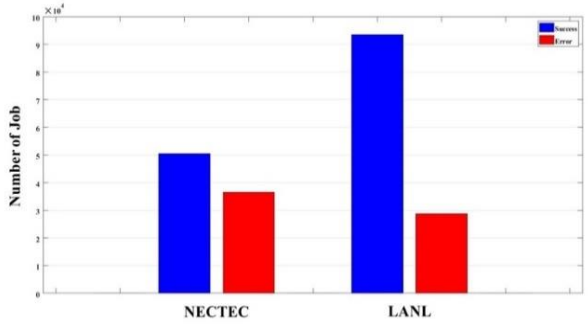



Figure 4. Histogram of the job success (blue) and error (red) in the dataset.

Fig. 4 shows the proportion between number of job completed with success and those completed with error status in the two datasets. The job success per job failure of the LANL dataset is around 76.5% to 23.5%. This dataset shows imbalance between the majority class of job success and the minority class of job failure. The NECTEC dataset has proportion between job success and job failure at 58% to 42%. The amount of data in the two classes are quite balance.

The first step prior to the modeling process is the attribute selection. This step corresponds to the main aim of this research to model HPC log at the job submit-state and job start-state. For creating a model to predict success/failure of a job at the job submit-state, we selcet three attributes to be used as predictors to predict the target value of job status as either success or failure. The three predictor attributes are "User ID", "CPU Request" and "Queue Type". This selection is based on the real situation that they are basic information that users need to specify before submitting the job into the HPC system. To create a model to predict at the job start-state the final job status, we additional attribute from the workload log regarding a job being processed on the HPC system. Therefore, on modeling at the job start-state, four attributes are used as predictors to predict success/failure of a job. These predictor attributes are "User ID", "CPU Request", "Queue Type" and "Wait Time". The target attribute for both job at submit-state and job at start-state modeling is "Finished Status". Summary of data attributes is presented in Table III.

The meaning of each attribute is as follows. "User ID" is the unique id of HPC-user. "CPU Request" is the number of processor elements that the user requires to use for the job. "Queue Type" is the queue system in the HPC system, which relates to the limitation of job run time. "Wait Time" is the period of a job waiting in the queue. "Finished Status" is the job status at the job finish-state.

TABLE III. The Attributes Used in This Research

| No. | Attribute at Job Submit-state | Attribute at Job Start-state | Data Type | Attribute Type |
|---|---|---|---|---|
| 1 | User ID | User ID | Nominal | Predictor |
| 2 | CPU Request | CPU Request | Numeric | Predictor |
| 3 | Queue Type | Queue Type | Nominal | Predictor |
| 4 | n/a | Wait Time | Numeric | Predictor |
| 5 | Finished Status | Finished Status | Binary | Target |

### B. Experimentation

The experimentation steps in this research are shown in Fig. 5. The data collection is the procedure for collecting the HPC-workload logs from data sources. After that, it is the data pre-processing step. At this step, the data cleaning was performed on the HPC-workload log from NECTEC. Data cleaning is unnecessary for the LANL dataset as it is already in the Standard Workload Format (SWF). After that, we split randomly the cleased data into three sub-datasets of each HPC-workload log. Each sub-dataset consists of 1,000 elements with five attributes: "User ID", "CPU Request", "Queue Type", "Wait Time", and "Finished Status". This research sets the target attribute to be "Finished Status". There are two distinct values in the target attribute representing the job status (either success or error) after the completion of HPC processing. The reason to use different sub-dataset to create the model because we would like to observe the model robustness when the dataset is changed.

The next steps are modeling and evaluating performance of each model on predicting final status of the job while it is at the processing stage in the HPC system. The three decision tree learning algorithms (C5.0, CART, and CHAID) are applied to create the job failure predictive model. In this experiment, we use 70% of the dataset to create the model, while the remaining 30% is for testing the model's performance. The last step of our experimentation is the part of model evaluation and comparison. The total scenarioes to test the model are 32 cases (2 workload log x 3 sub-datasets x 3 DT algorithms x 2 states of the job).

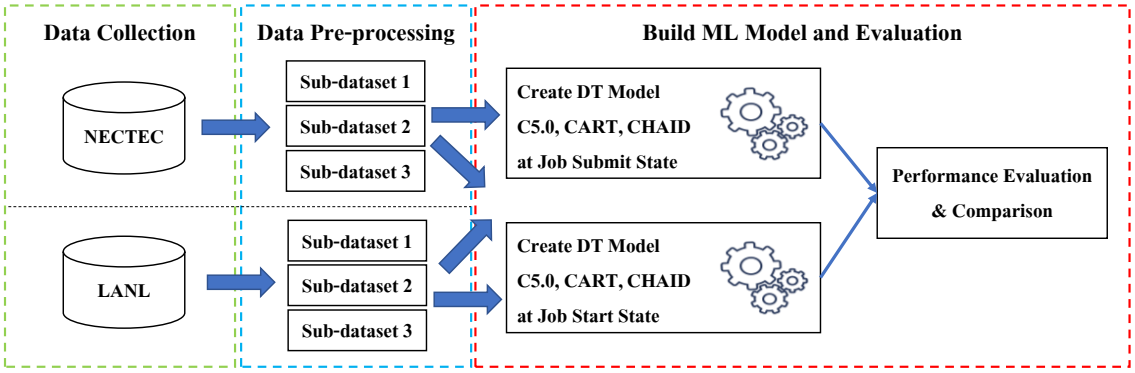
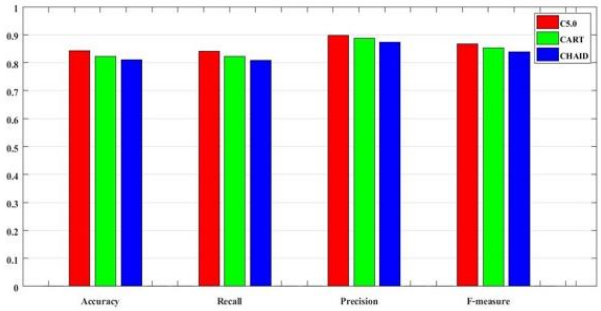



Figure 5. The research workflow.

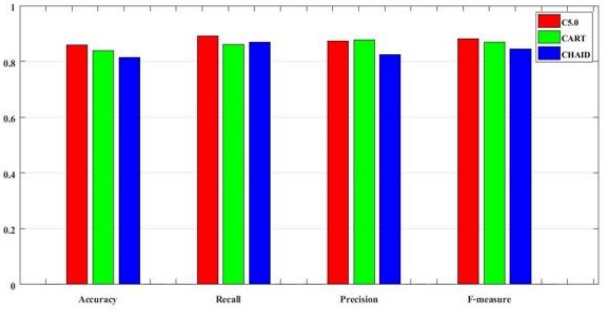
### C. Results

The modeling evaluation results for the models built from the NECTEC dataset where the jobs are at the submit-state are as follows. The results of the job failure predictive model of C5.0 algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8435, 0.8399, 0.8984, and 0.8670, respectively, with the variance of the $F_1$ score at 0.65e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8234, 0.8218, 0.8870, and 0.8526, respectively, with the variance of the $F_1$ score at 0.26e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8100, 0.8078, 0.8728, and 0.8388, respectively, with the variance of the $F_1$ score at 0.011e-3.

The modeling results for the case of building predictive models with the NECTEC dataset while the jobs are at the start state are as follows, The results of the job failure predictive model of C5.0 algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8580, 0.8924, 0.8734, and 0.8817, respectively, with the variance of the $F_1$ score at 0.33e-3. Next, the results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8381, 0.8604, 0.8771, and 0.8686, respectively, with the variance of the $F_1$ score at 0.25e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8153, 0.8697, 0.8237, and 0.8446, respectively, with the variance of the $F_1$ score at 0.88e-3.




(a) job submit-state




(b) job start-state

Figure 6. Performance of models built from the NECTEC dataset at different HPC processing stages.

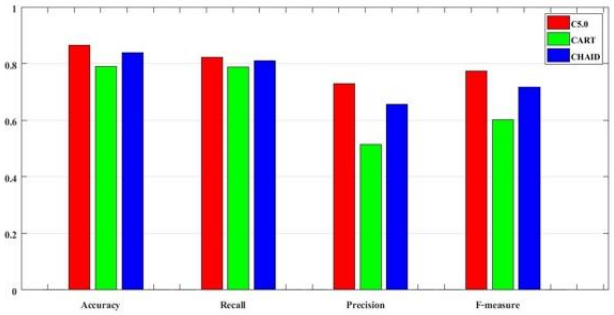
The performances of models built from the C5.0, CART, and CHAID algorithms using the NECTEC dataset are demonstrated in Fig. 6. The models built from jobs at submit-state and jobs at start-state are shown in the upper and lower graphs, respectively.

The results of the experiment using the LANL dataset at the job submit state show that the job failure predictive model of C5.0 algorithm having the average accuracy, average recall, average precision, and average $F_1$ score at 0.8654, 0.8230, 0.7298, and 0.7734, respectively, with the variance of the $F_1$ score at 0.69e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.7893, 0.7876, 0.5151, and 0.6010, respectively, with the variance of the $F_1$ score at0.061e-3. Lastly, the results of the job failure predictive model of the CHAID algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8390, 0.8096, 0.6551, and 0.7166, respectively, with the variance of the $F_1$ score at 0.022e-3.

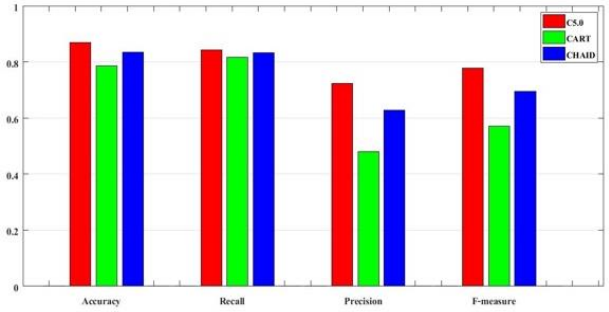
The results of the experiment in the case of LANL dataset at the job start state are as follows. The job failure predictive model of C5.0 algorithm shows the average accuracy, average recall, average precision, and average $F_1$ score at 0.8700, 0.8422, 0.7227, and 0.7777, respectively, with the variance of the F1 score at 0.87e-3. The results of the job failure predictive model of the CART algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.7861, 0.8171, 0.4796, and 0.5712, respectively, with the variance of the $F_1$ score at 0.0189. Lastly, the results of the job failure predictive

model of the CHAID algorithm show the average accuracy, average recall, average precision, and average $F_1$ score at 0.8345, 0.8332, 0.6282, and 0.6955, respectively, with the variance of the $F_1$ score at 0.094e-3.

The performance comparison of the predictive models of C5.0, CART, CHAID algorithms built from the LANL dataset are shown in Fig. 7. The upper graph correspomds to models built from jobs at submit-state, whereas the lower graph shows performance of models built from jobs at start-state.




(a) job submit-state




(b) job start-state

Figure 7. Performance of models built from the LANL dataset at different HPC processing stages.

## V. Discussion

The overall performance of all the predictive models to forecast job failure when processing in the HPC system gives the average accuracy of the model not less than 78%. The performance of the predictive model of the C5.0 algorithm is better than the predictive model of other algorithms in almost every test case. The C5.0 achieves the best average accuracy of 85.8% and 87% for the NECTEC and the LANL datasets, respectively. While, the predictive model of CART shows the lowest performance with 83.81% and 78.93% of average accuracy for the NECTEC dataset and the LANL dataset, respectively.

The models to predict job failure while processing in the HPC system built from decision tree algorithms can give high accuracy in every test case of the two HPC-workload datasets. But, the precision value of the model is quite low in the test case of the LANL dataset. The reason for this is that the LANL dataset has a high imbalance ratio between majority and minority classes. The low accuracy is from the predictive models built with the DT algorithms being unable to predict correctly the minority class.

The results regarding the performances of the predictive models at two different job states are the case of NECTEC workload log (Table IV) and LANL workload log (Table V) show that the predictive model being built while jobs are at start-state can predict the job failure at the end of the HPC processing slightly better than the model built from the jobs at job submit-state. This is because at the start-state there exist more useful information than while the jobs are at the submit-state. However, the predictive model to job failure prediction in the HPC system at job submit-state shows the best efficiency when considering the performance of the model together with the overhead of job waiting time. In both datasets, there are jobs waiting in the queue of the HPC system more than 80%.

TABLE IV. Performance of the Best Predictive Model (C5.0) Built from NECTEC Workload-Log at Different Stages: Job Submit-State vs Job Start-State

| NECTEC | Accuracy (avg) | Recall (avg) | Precision (avg) | $F_1$ Score (avg) |
|---|---|---|---|---|
| Submit-state | 0.8435 | 0.8339 | 0.8984 | 0.8670 |
| Start-state | 0.8580 | 0.8924 | 0.8771 | 0.8817 |

TABLE V. Performance of the Best Predictive Model (C5.0) Built from LANL Workload-Log at Different Stages: Job Submit-State VS Job Start-State

| LANL | Accuracy (avg) | Recall (avg) | Precision (avg) | $F_1$ Score (avg) |
|---|---|---|---|---|
| Submit-state | 0.8654 | 0.8230 | 0.7298 | 0.7734 |
| Start-state | 0.8700 | 0.8422 | 0.7227 | 0.7777 |

## VI. Conclusion

This research proposed the predictive modeling framework to create a model for forecasting job failure while processing in the HPC system. The model was created through the machine learning technique using three decision tree algorithms: C5.0, CART, and CHAID. The obtained model can be applied to help users make an efficient justification while their job is running in the HPC system. The datasets used in this work are the two HPC-workload logs that were collected from the operation of the HPC systems at NECTEC, Thailand, and LANL, USA. The results of the comparative study regarding the model performance show that the job failure predictive model built by the C5.0 algorithm has the best performance with an average accuracy of 85.8% and 87% using NECTEC dataset and LANL dataset, respectively. The C5.0 model also has robustness on data instability.

In the part of the comparative study when models are built at different job states, we found that performance of the model built from jobs while those jobs were at the start-state shows better accuracy than the model built from jobs running at the job submit-state. This finding is agree upon the two datasets. However, when considering the model accuracy with the tradeoff regarding the overhead of waiting time, we suggest that the position of the job at submit-state is more suitable to be applied for creating the job failure predictive model than the position of the job at start-state.

## Conflict of Interest

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

The first author is responsible for designing the research framework, organizing the experimentation steps and preparing the draft manuscript. The second author helps correcting the draft manuscript. The third author helps editing manuscript to be as appeared in the final version. The last author takes part in the conceptual design, experimentation setup and confirming correctness of the results.

## ACKNOWLEDGMENT

The authors would like to acknowledge the "National e-Science Infrastructure Consortium" of NECTEC for providing the HPC-workload log as a dataset that we use in this research (URL: http://www.escience.in.th). The first author has been supported by a scholarship from the Suranaree University of Technology (SUT). The second author has been supported by a scholarship from the Ministry of Science and Technology, Thailand. The third and fourth authors are researchers of the Data and Knowledge Engineering Research Unit, which has been fully supported by a research grant from SUT.

## REFERENCES

[1] P. Uthayopas, T. Angskun, and J. Maneesilp, "Building a parallel computer from cheap PCs: SMILE cluster experiences," in *Proc. the Second Annual National Symposium on Computational Science and Engineering*, 1998, p. 10.

[2] T. Pitakrat, D. Okanović, A. V. Hoorn, and L. Grunske, "Hora: Architecture-Aware online failure prediction," *Journal of System and Software*, vol. 137, pp. 669-685, Mar. 2018.

[3] B. Mohammed, I. Awan, H. Ugail, and M. Younas, "Failure prediction using machine learning in a virtualised HPC system and application," *Cluster Comput.*, vol. 22, no. 2, pp. 471-485, Jun. 2019.

[4] M. Stillwell, F. Vivien, and H. Casanova, "Dynamic fractional resource scheduling versus batch scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 521-529, Mar. 2012.

[5] A. Reuther, *et al.*, "Scalable system scheduling for HPC and big data," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 76-92, Jan. 2018.

[6] R. L. F. Cunha, E. R. Rodrigues, L. P. Tizzei, and M. A. S. Netto, "Job placement advisor based on turnaround predictions for HPC hybrid clouds," *Future Generation Computer System*, vol. 67, pp. 35-46, Feb. 2017.

[7] B. Silva, M. A. S. Netto, and R. L. F. Cunha, "JobPruner: A machine learning assistant for exploring parameter spaces in HPC applications," *Future Generation Computer System*, vol. 83, pp. 144-157, Jun. 2018.

[8] J. Kunkel and M. F. Dolz, "Understanding hardware and software metrics with respect to power consumption," *Sustainable Computing: Informatics and System*, vol. 17, pp. 43-54, Mar. 2018.

[9] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967-2982, Oct. 2014.

[10] B. Schroeder and G. A. Gibson, "A large-scale study of failures in high-performance computing system," in *Proc. International Conference on Dependable Systems and Networks*, 2006, p. 10.

[11] E. Chuah, A. Jhumka, J. C. Browne, N. Gurumdimma, S. Narasimhamurthy, and B. Barth, "Using message logs and resource use data for cluster failure diagnosis," in *Proc. IEEE 23rd International Conference on High Performance Computing*, Hyderabad, India, 2016, pp. 232-241.

[12] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in *Proc. Seventh IEEE International Conference on Data Mining*, Omaha, NE, USA, 2007, pp. 583-588.

[13] P. Gujrati, Y. Li, Z. Lan, R. Thakur, and J. White, "A Meta-learning failure predictor for blue Gene/L system," in *Proc. International Conference on Parallel Processing*, Xian, China, 2007.

[14] M. Soysal, M. Berghoff, and A. Streit, "Analysis of job metadata for enhanced wall time prediction," *Job Scheduling Strategies for Parallel Processing*, vol. 11332, 2018.

[15] H. Zhang, H. You, B. Hadri, and M. Fahey, "HPC usage behavior analysis and performance estimation with machine learning techniques," in *Proc. 18th International Conference on Parallel and Distributed Processing Techniques and Applications*, 2012, p. 7.

[16] Y. Yuan, Y. Wu, Q. Wang, G. Yang, and W. Zheng, "Job failures in high performance computing system: A large-scale empirical study," *Computers & Mathematics with Applications*, vol. 63, no. 2, pp. 365-377, Jan. 2012.

[17] S. Sabato, E. Yom-Tov, A. Tsherniak, and S. Rosset, "Analyzing system logs: A new view of what's important," in *Proc. the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, 2007, p. 7.

[18] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, no. 2, p. 55, Feb. 2012.

[19] A. Trabelsi, Z. Elouedi, and E. Lefevre, "Decision tree classifiers for evidential attribute values and class labels," *Fuzzy Sets and System*, vol. 366, pp. 46-62, Jul. 2019.

[20] S. Pang and J. Gong, "C5.0 classification algorithm and application on individual credit evaluation of banks," *System Engineering - Theory & Practice*, vol. 29, no. 12, pp. 94-104, Dec. 2009.

[21] M. Hassoon, M. S. Kouhi, M. Zomorodi-Moghadam, and M. Abdar, "Rule optimization of boosted C5.0 classification using Genetic algorithm for liver disease prediction," in *Proc. International Conference on Computer and Applications*, Doha, United Arab Emirates, 2017, pp. 299-305.

[22] S. Abdul Kareem, S. Raviraja, N. A Awadh, A. Kamaruzaman, and A. Kajindran, "Classification and regression tree in prediction of survival of AIDS patients," *MJCS*, vol. 23, no. 3, pp. 153-165, Dec. 2010.

[23] W. A. V. Clark, M. C. Deurloo, and F. M. Dieleman, "Modeling categorical data with chi square automatic interaction detection and correspondence analysis," *Geographical Analysis*, vol. 23, no. 4, pp. 332-345, Sep. 2010.

[24] M. Ramaswami and R. Bhaskaran, "A CHAID based performance prediction model in educational data mining," *International Journal of Computer Science Issues*, vol. 7, no. 1, p. 9, 2010.



**Anupong Banjongkan** is a Ph.D. student in a computer engineering program with the School of computer engineering, Suranaree University of Technology, Thailand. He graduated in B.S. of Computer Science and Master of Engineering in Electrical Engineering at King Mongkut's University of Technology North Bangkok, Thailand, in 2007 and 2011, respectively. His current research of interest includes High Performance Computing, Machine Learning, and Knowledge Discovery.

**Watthana Pongsana** is a Ph.D. student, School of Computer Engineering, Suranaree University of Technology (SUT), Thailand. He received his B.E. and M.E. in computer engineering from Suranaree University of Technology, Thailand, in 2008 and 2012. His research of interest includes Software Engineering, Data Mining, Artificial Intelligence, and Human-Computer Interaction.

**Nittaya Kerdprasop** is an associate professor and the head of the Data Engineering Research Unit, School of Computer Engineering, SUT, Thailand. She received her B.S. in radiation techniques from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. Her research of interest includes Data Mining, Logic and Constraint Programming.

**Kittisak Kerdprasop** is an associate professor at the School of Computer Engineering, SUT, and a Chair of the School. He received his bachelor's degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, MS in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, U.S.A., in 1999. His current research includes Machine Learning and Artificial Intelligence.