Scalable Approach to Failure Analysis of High-Performance Computing Systems

Doaa Shawky

Failure analysis is necessary to clarify the root cause of a failure, predict the next time a failure may occur, and improve the performance and reliability of a system. However, it is not an easy task to analyze and interpret failure data, especially for complex systems. Usually, these data are represented using many attributes, and sometimes they are inconsistent and ambiguous. In this paper, we present a scalable approach for the analysis and interpretation of failure data of high-performance computing systems. The approach employs rough sets theory (RST) for this task. The application of RST to a large publicly available set of failure data highlights the main attributes responsible for the root cause of a failure. In addition, it is used to analyze other failure characteristics, such as time between failures, repair times, workload running on a failed node, and failure category. Experimental results show the scalability of the presented approach and its ability to reveal dependencies among different failure characteristics.

Keywords: Failure analysis, high-performance computing, rough sets theory.

I. Introduction

Failure analysis is the process of analyzing failure data to determine the root cause of a failure and predict new failures. It is an important step in all engineering processes; one that aims at improving system reliability and performance. The process of effectively analyzing and understanding failure data is a non-trivial one, especially for large complex systems where failures are more likely to occur. Moreover, sometimes the collected failure data are inconsistent and ambiguous making them even more difficult to be interpreted. High-performance computing (HPC) systems are notable examples of such systems. They are usually used for running advanced application programs efficiently and reliably. Thus, it is very important to analyze the failures of such systems to minimize the probability of a failure in the future. Los Alamos National Laboratory (LANL) has released a large set of failure data of HPC systems [1]. The data cover 23 HPC systems with a total of 23,740 records. Each record contains the start time of the failure; its end time; the system and node affected by the failure; the application type that runs on the node; and the root cause of the failure. Many approaches were proposed in the literature for the failure analysis of HPC systems, especially after the release of LANL's large data set. However, the presented approaches in the literature depend mainly on the statistical properties of the measured attributes and employ statistical modeling techniques to analyze the failure data. For example, in [2]–[3], the authors studied the failure data that were released by LANL using statistical modeling. The studied attributes include the root cause of failures, the mean time between failures, and the mean time to repair. The authors concluded that average failure rates differed widely across systems. They were able to model the time between failures by a Weibull distribution with decreasing

Manuscript received Nov. 12, 2013; revised Apr. 19, 2014; accepted May 7, 2014.

Doaa Shawky (doaashawky@staff.cu.edu.eg) is with the Department of Engineering Mathematics, Cairo University, Cairo, Egypt.

hazard rate. In addition, they modeled the repair times by a log-normal distribution. Moreover, in [4], error logs of 395 nodes were studied. The authors analyzed the time between failures using distribution fitting. They concluded that the time between failures follows a Weibull distribution with increasing hazard rates. In addition, they reported that higher failure rates were more likely to occur during the day. Furthermore, Yawei and others [5] suggested a strategy for spare-node allocation and job scheduling in large-scale parallel systems (in addition to failure prediction). They showed that their approach improves a system's productivity. In [6], the authors used decision tree classifiers to predict failures in a HPC system. They also used the data that were collected on supercomputing clusters at LANL. They analyzed the root causes to predict if a failure, would occur within an hour. The prediction precision of their proposed system is 73%, with a recall of about 80%. They employed the usage data, along with the failure data, to improve the accuracy of prediction. Also, the same set of data is studied in [7]. The authors performed statistical analysis to determine the root cause of a failure, the time between failures and the repair times. Moreover, they studied the effect of failures on standard checkpoint/restart fault-tolerance strategies. They concluded that the efficacy of peta-scale machines running complex applications will fall off. Thus, they recommended increasing the number of cycles that compress checkpoints. Michalak and others studied the failure of the Advanced Simulation and Computing Q supercomputer [8]. They found a high rate of node failures. This was hypothesized to be caused primarily by soft errors (that is, board-level cache and tag parity errors). Log errors were analyzed, and errors were modeled using Poisson, gamma, and exponential distributions. These models were used to predict the rate at which fatal soft errors occur. Also, in [9], the authors studied how the growing complexity and size of HPCs can lead to frequent job failures. An empirical study on the job failures of ten public workload data sets collected from eight large-scale HPCs is presented. Their results show that job failure rates are significant in most HPCs, and on average, a failed job often consumes more computational resources than a successful job. They also observed that the submission inter-arrival time of failed jobs is better fit by generalized Pareto and log-normal distributions. Moreover, in [10], the authors proposed an analytical approach to quantifying the capabilities of the Message Passing Interface and MapReduce programming models to tolerate failures. The impact of different parameters on fault tolerance has been studied to predict the scale at which the MapReduce programming model has a better performance under the presence of failures. A survey on the failure analysis methods of HPC systems can be found in [11].

In addition to the previous approaches for failure prediction and analysis, the literature includes other approaches that measure the quality of failure prediction. For example, in [12], the authors proposed a new metric for measuring the failure prediction error. Instead of using the mean square error, precision, or recall, they used a metric called "lost computing time." Also, in [13], the authors described context-relevant methodologies for determining the accuracy and the cost-benefit of predictors.

In this paper, we present an approach that employs rough sets theory (RST) [14] for modeling and analyzing the failure data of HPC systems. RST has been used by many researchers and in many interesting applications. It seems to be of fundamental importance to artificial intelligence and cognitive sciences, especially in the areas of machine learning, knowledge acquisition, decision analysis, and knowledge discovery from databases [15]. In addition, we previously used RST for the analysis of dynamically collected data in the context of software feature location [16]. RST-based approaches have several advantages [17]. They do not need any additional information about the analyzed data, in contrast to similar approaches such as the fuzzy set theory. In addition, they can find hidden patterns in the data in an efficient and easy-to-understand way, and they are suitable for distributed processing, which makes them scalable.

Using RST as an engine for failure data analysis, we can find the minimum set of attributes responsible for a failure. The approach is scalable because it removes any unnecessary attributes and highlights only ones that are important. Thus, we can analyze the failure data over the complete lifetime of a system. Moreover, we can generate a set of rules that can be used in the new predictions of time between failures, repair times, and root causes of a failure.

The paper is organized as follows. Section II introduces some RST concepts. Section III describes the used data. In Section IV, we present the experimental results with respect to the root cause, the time between failures, the time to repair, and for some systems, the type of the application running on the failed node. Finally, Section V draws conclusions and limitations, as well as outlining ideas for future work.

II. RST Background

In this section, the basic concepts of RST are presented. A detailed review on RST is found in [15]. The rough sets philosophy is based on the assumption that with every object of the universe, there is a certain amount of information expressed by means of some attributes. Objects having the same description are indiscernible with respect to the available information. The indiscernibility relation constitutes a mathematical basis of RST [16]. It induces a partition of the universe into blocks of indiscernible objects, called elementary

sets or decision classes. These sets can be used to build knowledge about a real or abstract world. Any subset of the universe, say *x*, may be expressed in terms of these blocks either precisely or approximately. In the latter case, *X* may be characterized by two ordinary sets, called lower and upper approximations. A rough set is defined by means of these two approximations. The lower approximation of *X* is composed of all the elementary sets included in *X* while the upper approximation of *X* consists of all the elementary sets that have a non-empty intersection with *X*. The difference between the upper and lower approximations constitutes the boundary region of the rough set, whose elements cannot be characterized with certainty as belonging to *X*, using the available information [18]–[20]. Information about objects in the boundary region is inconsistent or ambiguous. Using the rough sets approach, one can deal with two major problems in the analysis of an information system [18]–[20]: (a) reducing unnecessary objects and attributes so as to get the minimum subset of attributes (which ensures a good approximation of classes and an acceptable quality of classification) and (b) representing the information system in a decision table that shows dependencies between the minimum subset of attributes and particular attributes (called decisions) [20].

1. Information Systems

In RST, information systems are used to represent knowledge. The notion of an information system that is presented here is described in [20]–[21].

An information system $S = (U, \Omega, V_q, f_q)$ consists of a non-empty finite set, called the universe (U) and a non-empty finite set of attributes (Ω), where $\Omega = C \bigcup D$ (in which C is a finite set of condition attributes and D is a finite set of decisions). For each $q \in \Omega$, there exists the domain of q (V_q) and an information function f_q, where f_q: $u \rightarrow V_q$. Objects can be interpreted as cases, states, processes, or observations. Attributes can be interpreted as features, variables, or characteristic conditions. A special case of information systems is called a decision table. In a decision table, the rows and columns correspond to observations and attributes, respectively.

2. Lower and Upper Approximations

Due to imprecisions that exist in the real-world data, there are always conflicting observations contained within a decision table. In RST, approximations of sets are introduced to deal with such inconsistencies. If $S = (U, \Omega, V_q, f_q)$ is a decision table, $B \subseteq \Omega$, and $X \subseteq U$, then the *B*-lower ($\underline{B}(X)$) and *B*-upper ($\overline{B}(X)$) approximations of *X* are defined, respectively, as follows [22]:

$$\underline{B}(X) = \bigcup \{Y \in U / \mathrm{IND}(B) : Y \subseteq X\}, \quad (1)$$

$$\overline{B}(X) = \bigcup \{Y \in U / \mathrm{IND}(B) : Y \bigcap X \neq \phi\}, \quad (2)$$

where U/IND(B) denotes the family of all equivalence classes of B (classification of U). In addition, IND(B), which is called the B-indiscernibility relation, is defined as follows:

$$\mathrm{IND}(B) = \{(x, y) \in U^2, \ a(x) = a(y)\}. \quad (3)$$

The set $\mathrm{BNB}(X) = \overline{B}(X) - \underline{B}(X)$ is called the *B*-boundary of *X*. The set of all elements of U that can be certainly classified as elements of *X* employing the set of attributes *B*, is denoted by $\underline{B}(X)$. The set of elements of U that can be possibly classified as elements of *X* using the set of attributes *B*, is denoted by $\overline{B}(X)$.

3. Quality of Approximations

A rough set can be also characterized numerically by the following coefficient:

$$\alpha_B(X) = \frac{\left|\underline{B}(X)\right|}{\left|\overline{B}(X)\right|}. \quad (4)$$

This coefficient is called the accuracy of the approximation, where $|X|$ denotes the cardinality of *X*. Obviously, $0 \le \alpha_B(X) \le 1$. If $\alpha_B(X) = 1$, then *X* is crisp with respect to *B*; otherwise, if $\alpha_B(X) < 1$, then *X* is rough with respect to *B*. This provides a measure of how closely the rough set is approximating the target set.

4. Dependency of Attributes

An important issue in data analysis is discovering dependencies between attributes. Intuitively, a set of attributes D depends totally on a set of attributes C, denoted by $C \Rightarrow D$, if all the values of the attributes from D are uniquely determined by the values of the attributes from C. Let D and C be subsets of A, then we say that D depends on C to a certain degree, say k, such that $0 \le k \le 1$. This is denoted by

$$C \Rightarrow D_k \quad \text{if} \quad k = \gamma(C, D) = |POS_c(D)|/|U|, \quad (5)$$

where $POS_C(D) = \bigcup_{X \in U/D} \underline{C}(X)$, called a positive region of the partition U/D with respect to C, is the set of all elements of U that can be uniquely classified into blocks of the partition U/D by means of C.

In addition,

$$\gamma(C, D) = \sum_{X \in U/D} \frac{\left|\underline{C}(X)\right|}{\left|U\right|}. \quad (6)$$

If $k = 1$, then we say that D depends totally on C; and if $k < 1$, then we say that D depends partially (to the degree k) on C.

The coefficient γ expresses the ratio of all elements of the universe that can be properly classified into blocks of the partition U/D employing attributes C. This coefficient represents the accuracy of the classification and can be used for characterizing a rough set.

5. Reducts and Core

An interesting question is whether there are attributes in the information system that are more important to the knowledge represented in the decision classes than other attributes. Usually, we need to find a subset of attributes that can fully characterize the knowledge in the decision table. Such an attribute set is called a reduct. Formally, let $S = (U, A)$ be an information system, $B \subseteq A$, and let $b \in B$. We say that b is dispensable in B if $\mathrm{IND}(B) = \mathrm{IND}(B-\{b\})$; otherwise b is indispensable in B. A set B is independent if all its attributes are indispensible. Any subset B' of B is called a reduct of B, if B' is independent and $\mathrm{IND}(B) = \mathrm{IND}(B')$. The core of B is the set of all indispensable attributes of B. The following property connects the notion of the core and reducts [18]–[20]:

$$\mathrm{Core}(B) = \bigcap \mathrm{Red}(B), \tag{7}$$

where $\mathrm{Red}(B)$ is the set of all reducts of B.

6. Decision Rules

A unique feature of the RST method is its generation of rules that play an important role in predicting the output. A decision rule can be expressed as a logical statement:

IF conjunction of elementary conditions;
THEN disjunction of elementary decisions.

Several numerical factors can be associated with a synthesized rule; for example, the rule support, accuracy, and level of discrimination [19].

III. Failure Data

As previously mentioned, we used the large set of failure data released by LANL. The data were collected over a period of nine years (1996–2005), and they consist of 23 systems that include a total of 5,006 nodes and 25,116 processors. The systems vary in the number of nodes and processors. In addition, the nodes within a system can differ in accordance with the number of processors and the amount of main memory. Most of these systems are large clusters of either NUMA nodes or 2-way and 4-way SMP nodes. The majority of the workloads are large-scale scientific simulations that perform long periods of CPU computation, interrupted every few hours by a few minutes of I/O for check-pointing. Simulation workloads are often accompanied by scientific visualization of large-scale data, which are also CPU-intensive but exhibit more reading of data from storage than compute workloads. Finally, some nodes are used purely as front-end nodes, and others run more than one type of workload; for instance, graphics nodes often run compute workloads. Thus, the workloads can be classified into either graphics, computation, or front-end workloads. In addition, some nodes run more than one type of workload; for example, graphics and computational workloads. An important characteristic of a failure is its root cause. The root causes of a failure are categorized as one of the following: human error, network failure, environment, hardware error, software error, or undetermined sometimes. More information on the data, the collection methodology, and the environment can be found in [1]–[2].

IV. Results and Discussion

In this section, we will present the results of the application of RST on the failure data presented in the previous section.

The RST modeling and analysis were performed using ROSE2 [23]. A discretization step is performed for the continuous attributes; the time between failures and the down times. Then, decision classes using defined decisions were found. The quality of the approximations and the accuracy of the classifications were also determined. Then, the core attributes and the set of reducts for each experiment were specified. Finally, we generated a set of rules that can represent the given data. Using only the set of strong rules, we were able to build classifiers that can be used for the prediction of new failure characteristics. In the experimental study, we investigated the basic characteristics of a failure; that is, the root cause (failure type), the time between failures, the down (repair) time, and the node number. Thus, we performed four experiments for each system; each considers one of these characteristics as the decision. Moreover, Systems 2 and 16 contain nodes that run different workloads. Therefore, for these two systems, we added one more experiment that considers the type of the workload as a decision. Table 1 presents the abbreviations that are used in the results to address the different failure characteristics. Also, it should be mentioned that the size of the internal memory (MM) has large variations in System 18 only. Thus, only for this system, we performed an experiment that considers MM as a decision. Moreover, the attribute NP has constant values in each system, so it is meaningless to use it as a decision. Although it appeared in some rules, as will be shown later. Table 2 presents the coding that was used for the failure categories. Meanwhile, Table 3 shows the coding that was used for the workload types.

Table 1. Symbols given to attributes.

Attribute name	Symbol
Node number	NN
Down time	DT
Failure type	FT
Time between failures	TBF
Memory	MM
# of processors in a node	PN
Workload type	WT

Table 2. Coding for failure categories.

Failure type	Code
Environment	1
Hardware	2
Human	3
Network	4
Undetermined	5
Software	6

Table 3. Coding for workload types.

Workload type	Code
Unspecified	0
Graphics and front end	1
Compute	2
Graphics and compute	3
Graphics	4
Front end	5

1. Decision Classes

The decision classes, quality, and accuracy for Systems 2 and 16, are presented in Table 4. In addition, those results for the remaining systems are presented in Table 5. It should be noted that, on average, the quality and the accuracy values are acceptable. The average quality is 0.86, 0.69, 0.97, and 0.86 for the decisions TBF, DT, FT, and NN, respectively. Meanwhile, the average accuracy is 0.76, 0.55, 0.94, and 0.73 for the decisions TBF, DT, FT, and NN, respectively. These results can be interpreted as follows. Firstly, the data are actually inconsistent and contain ambiguity. Secondly, RST is an effective candidate for analyzing these data. It should be noted that the lowest obtained results were for System 19, which uses TBF and DT as decisions. For nearly all systems, making TBF or DT as decisions yielded the lowest values. This is due to their nature, as they are the only continuous attributes in the data. Thus, their values have to be discretized first. Also, for some systems (for example, 7, 15, 17, 22, and 24), using the NN as a decision has no meaning as it has only a single value.

Table 4. Decision classes for Systems 2 and 16, using WT as a decision.

# of decision classes	Quality	Accuracy
3	0.99	0.98
4	1.00	1.00

2. Core Attributes

The core attributes for the 23 systems, together with the quality and the accuracy of the classifications using the four failure characteristics, are presented in Table 6.

It should be mentioned that using the workload type as a decision gives no core attributes for the 23 systems. Thus, we report the core for the four other characteristics only. This suggests that the data cannot be used for predicting the work load type on a failed node with high accuracy. Also, the quality of approximations using only the attributes in the core is acceptable. It is, on average, 0.84, 0.69, 0.94, and 0.85 for the decision attributes TBF, DT, FT, and NN, respectively. This suggests a strong dependency of the failure characteristics on these attributes; hence, they can be used effectively (with high accuracy) in the prediction of future decision attributes. Moreover, the lowest quality of approximations using only the core is reported for System 19, which makes use of TBF and DT as decisions. This is due to the low quality of the RST representation of this system. Figure 1 presents how the frequency of the core attributes for the four failure characteristics vary across systems. As shown in Fig. 1, the attributes with the highest frequencies are DT and TBF, which suggests that these two attributes are the most important failure characteristics. In addition, DT is the most frequent attribute in the set of core attributes using TBF as a decision; and the converse is true. This suggests a strong dependency between these two attributes. Moreover, there is a strong dependency between FT and the two decisions DT and TBF.

3. Reducts

For most systems, the reducts were the same as the core, except for the systems presented in Table 7. Although the core is different from the reducts, we note that the three main failure characteristics (that is, DT, TBF, and FT) contribute to the

Table 5. Decision classes.

System ID	Decision											
	TBF			DT			FT			NN		
	# of decision classes	Quality	Accuracy	# of decision classes	Quality	Accuracy	# of decision classes	Quality	Accuracy	# of decision classes	Quality	Accuracy
2	193	0.96	0.92	180	0.26	0.54	5	0.94	0.92	49	0.84	0.77
3	42	0.98	0.97	150	0.90	0.76	6	0.99	0.98	115	0.92	0.86
4	38	0.99	0.98	162	0.86	0.75	6	1.00	1.00	108	0.91	0.84
5	43	0.99	0.98	169	0.83	0.67	6	1.00	1.00	109	0.93	0.87
6	20	0.96	0.93	51	0.84	0.65	6	1.00	1.00	27	1.00	1.00
7	80	0.69	0.52	84	0.62	0.23	7	0.98	0.96	1	-	-
8	56	0.97	0.95	352	0.79	0.60	7	0.99	0.99	119	0.96	0.93
9	63	0.98	0.92	120	0.91	0.82	7	0.99	0.98	127	0.73	0.51
10	30	0.97	0.95	119	0.92	0.85	6	0.94	0.91	127	0.80	0.56
11	38	0.95	0.91	129	0.94	0.90	7	0.99	0.98	126	0.78	0.54
12	33	0.92	0.85	129	0.89	0.79	7	1.00	1.00	134	0.78	0.60
13	26	0.94	0.85	117	0.92	0.86	7	0.99	0.98	112	0.80	0.61
14	25	0.88	0.73	68	0.84	0.73	7	1.00	1.00	58	0.64	0.29
15	20	0.75	0.58	38	0.38	0.15	5	1.00	1.00	1	-	-
16	167	0.60	0.30	414	0.38	0.08	7	0.92	0.85	16	0.84	0.58
17	9	1.00	1.00	18	0.44	0.13	7	0.80	0.66	1	-	-
18	23	1.00	1.00	47	1.00	1.00	7	1.00	1.00	40	1.00	1.00
19	65	0.14	0.01	555	0.01	0.00	7	0.75	0.60	670	0.82	0.74
20	75	0.96	0.92	448	0.73	0.51	7	0.99	0.99	400	0.72	0.50
21	43	1.00	0.98	86	1.00	1.00	7	0.98	0.96	48	1.00	1.00
22	120	0.59	0.32	131	0.55	0.25	7	0.98	0.96	1	-	-
23	73	0.65	0.37	194	0.28	0.05	7	0.92	0.83	5	0.96	0.93
24	71	0.73	0.47	107	0.49	0.24	7	1.00	1.00	1	-	-

reducts.

4. Rules

Using the RST modeling tool, a large number of rules for each system are generated. The following criteria were used for the generation of such rules. The maximum rule length is set equal to three, the minimum relative strength is varied between 50% and 80%, and the minimum discrimination level is set equal to 100%. This results in three strong rules, on average, for each system. For example, for System 2, some of the generated rules (with the highest coverage) include the following:

(NN = 7) & (DT = 42) $\Rightarrow$ (TBF = 0.81)

(NN = 1) & (TBF = 0.59) $\Rightarrow$ (DT = 239)

(DT = 30) $\Rightarrow$ (FT = 6)

(FT = 6) $\Rightarrow$ (NN = 21)

Due to the space limitations, the complete set of generated rules is omitted. Also, for System 16, some rules using workload type as a decision were generated. Examples of these rules include:

(NN = 0) $\Rightarrow$ (WT = 3)

(MM = 32) $\Rightarrow$ (WT = 2)

However, these rules are of no importance to failure analysis. Also for System 18, some rules that show the dependency between some failure characteristics and the size of the memory are generated. For example,

(FT = 6) & (TBF = 0.06) $\Rightarrow$ (MM = 32)

(DT = 49) $\Rightarrow$ (MM = 32).

In addition, for System 22, some rules revealed a dependency between the number of processors in a node and the time between failures, which suggests that the hardware

http://dx.doi.org/10.4218/etrij.14.0113.1133

Table 6. Core attributes and quality of classification.

System ID	TBF		DT		FT		NN	
	Core	Quality	Core	Quality	Core	Quality	Core	Quality
2	NN, DT, FT	0.67	NN, FT, TBF	0.26	NN, DT, TBF	0.65	TBF, DT, FT	0.76
3	NN, DT, FT	0.98	NN, FT, TBF	0.90	NN, DT, TBF	0.99	TBF, DT, FT	0.92
4	NN, DT, FT	0.93	NN, FT, TBF	0.86	NN, DT, TBF	1.00	TBF, DT, FT	0.91
5	NN, DT, FT	0.99	NN, FT, TBF	0.83	NN, DT, TBF	1.00	TBF, DT, FT	0.93
6	NN, DT	0.96	NN, FT, TBF	0.84	DT	0.96	DT, TBF	1.00
7	DT, FT	0.69	FT, TBF	0.62	DT, TBF	0.98	DT, TBF	1.00
8	NN, DT, FT	0.97	NN, FT, TBF	0.79	NN, DT, TBF	0.99	DT, FT, TBF	0.96
9	NN, DT, FT	0.92	NN, FT, TBF	0.91	NN, DT, TBF	0.99	TBF, DT, FT	0.73
10	NN, DT, FT	0.97	NN, FT, TBF	0.92	NN, DT, TBF	0.93	TBF, DT, FT	0.80
11	NN, DT, FT	0.95	NN, FT, TBF	0.94	NN, DT, TBF	0.99	TBF, DT, FT	0.78
12	NN, DT, FT	0.92	NN, FT, TBF	0.89	NN, DT, TBF	1.00	TBF, DT, FT	0.78
13	NN, DT, FT	0.94	NN, FT, TBF	0.92	NN, DT, TBF	0.99	TBF, DT, FT	0.80
14	NN, DT, FT	0.88	NN, FT, TBF	0.84	NN, DT, TBF	1.00	DT, TBF	0.64
15	DT, FT	0.75	FT, TBF	0.38	DT, TBF	1.00	TBF, DT, FT	0.63
16	NN, DT, FT	0.60	NN, FT, TBF	0.38	NN, DT, TBF	0.92	TBF, DT, FT	0.79
17	DT, FT	1.00	FT, TBF	0.44	DT, TBF	0.80	TBF, DT, FT	0.74
18	NN, DT	1.00	NN, TBF	1.00	None	-	DT, TBF	1.00
19	DT, FT	0.14	FT, TBF	0.01	DT, TBF	0.75	DT, TBF	0.77
20	NN, DT, FT	0.96	NN, FT, TBF	0.73	NN, DT, TBF	0.99	TBF, DT, FT	0.72
21	NN, DT	0.98	NN, FT, TBF	1.00	DT	0.79	DT, TBF	1.00
22	DT, FT	0.59	FT, TBF	0.55	DT, TBF	0.98	TBF, DT, FT	0.93
23	NN, DT, FT	0.65	NN, FT, TBF	0.28	DT, TBF	0.88	MM, DT, TBF	0.96
24	DT, FT	0.73	FT, TBF	0.49	DT, TBF	1.00	None	-

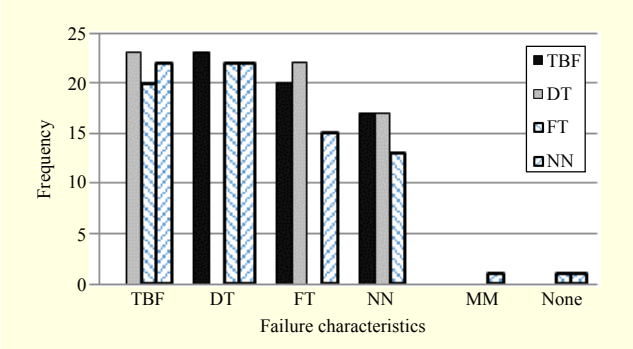

Fig. 1. Frequency of attributes in the core using each of the four basic characteristics as decisions.

specifications have an influence on the failure of some systems. Moreover, using FT as a decision, we had to relax the criteria for the generation of rules to have 30% relative strength and 80% discrimination level for Systems 15, 17, and 18. This indicates less correlation between FT and the other attributes in these systems. For some systems (for example, 10), even when we relaxed the rule-generation criteria, no decent rules could be generated. The rules with the highest coverage were obtained for System 8 using FT as decision (about 88%). On the other hand, the rules with the lowest coverage were obtained for System 2 using TBF as a decision. On average, the obtained rules have a coverage of about 70%.

Table 7. Systems with reducts that are different from the core.

System ID	Decision	Reducts
16	NN	{MM, DT, FT, TBF}, {WT, DT, FT, TBF}
16	WT	{NN}, {MM}
6	FT	{NN, DT}, {DT, TBF}
18	MM	{NN}, {DT, TBF}
23	FT	{NN, DT, TBF}, {MM, DT, TBF}

5. Failure Prediction

We used the set of rules that were generated in the previous section in building a predictor of failures. The prediction is based on classification techniques. The RST tool employs the L-metrics classifier [21]. As for any classification technique, the set of data for each system is divided into two sets: the first is used for learning (70% of the data), while the second is for testing. However, the obtained accuracy highly depends on the system, the type of the validation test (leaving-one-out or *k*-fold cross validation), and the decision attribute. The highest obtained accuracy was 83.6% for System 6, using *k*-fold cross validation with $k = 10$ and the failure type as a decision attribute. On the other hand, the lowest classification accuracy was 54% for System 16, using TBF as a decision with 10-fold cross validation. Other experiments yielded accuracies that varied between 56% and 77%.

V. Conclusion and Future Work

Failure analysis of HPC systems is necessary to improve system reliability. Thus, we need an intelligent, scalable, and efficient method for this task. In this paper, we proposed an RST-based approach for the failure analysis of the first ever release of large-size failure data of HPC systems. Using the capabilities of RST as a powerful analysis tool, the minimum sets of factors related to each failure characteristic can be found. In addition, the complete set of data for each system can be analyzed without facing scalability problems. Obtained results show strong dependency between the time between failures and the down times. Also, in contrast to some previous works (for example, [4]), results show a very weak dependency between the time between failures and the workload type running on a failed node. In addition, new relationships were revealed among the attributes of the failure data. For example, the results show that, for some systems, there is a dependency between the hardware specification of a node (the size of the internal memory) and the failure type.

It should be mentioned that the sets of rules that were generated for each system could not be used for generalization since the support (coverage) of most rules was relatively low. However, they can be used for revealing some sort of dependencies between the attributes of a failure and its characteristics.

Additional work is needed to extract the optimal set of rules describing each system and to utilize these rules in the building of a predictor of failures with high accuracy. Another important point, in need of extra investigation, is the effect of the discretization of the time between failures and the down time on the quality of the obtained rules. We think that if these attributes are properly discretized, the accuracy of the predictor can be highly improved.

References

[1] *The Raw Failure Data*. Los Alamos National Laboratory. Accessed May 14, 2012. http://www.lanl.gov/projects/computersciencedata/, 2012.

[2] B. Schroeder and G.A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 4, Oct.–Dec. 2010, pp. 337–350.

[3] B. Schroeder and G.A. Gibson, "A Large-Scale Study of Failures in High-Performance-Computing Systems," *Proc. Dependable Syst. Netw.*, Pennsylvania, PA, USA, June 25–28, 2006, pp. 249–258.

[4] R.K. Sahoo et al., "Failure Data Analysis of a Large-Scale Heterogeneous Server Environment," *Proc. Dependable Syst. Netw.*, Florence, Italy, June 28– July 1, 2004, pp. 772–781.

[5] L. Yawei et al., "Fault-Aware Runtime Strategies for High-Performance Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 4, Apr. 2009, pp. 460–473.

[6] N. Nakka, A. Agrawal, and A. Choudhary, "Predicting Node Failure in High Performance Computing Systems from Failure and Usage Logs," *IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, Shanghai, China, May 16–20, 2011, pp. 1557–1566.

[7] G. Gibson, B. Schroeder, and J. Digney, "Failure Tolerance in Petascale Computers," *CTWatch Quarterly*, vol. 3, no. 4, Nov. 2007, pp. 4–10.

[8] S.E. Michalak et al., "Predicting the Number of Fatal Soft Errors in Los Alamos National Laboratory's ASC Q Supercomputer," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, Sept. 2005, pp. 329–335.

[9] Y. Yuan et al., "Job Failures in High Performance Computing Systems: A Large-Scale Empirical Study," *Comput. Math. Appl.*, vol. 63, no. 2, Jan. 2012, pp. 365–377.

[10] H. Jin and X.-H. Sun, "Performance Comparison under Failures of MPI and MapReduce: An Analytical Approach," *Future Generation Comput. Syst.*, vol. 29, no. 7, Sept. 2013, pp. 1808–1815.

[11] M. Sharifi and S.A. Hamedi, "Failure Prediction Mechanisms in Cluster Systems," *Int. Conf. Biocomput., Bioinformat. Biomed. Technol.*, Bucharest, Romania, June 29–July 5, 2008, pp. 23–28.

[12] N. Taerat et al., "Proficiency Metrics for Failure Prediction in High Performance Computing," *Int. Symp. Parallel Distrib. Process. Appl.*, Taipei, Taiwan, Sept. 6–9, 2010, pp. 491–498.

[13] J. Brandt et al., "Quantifying Effectiveness of Failure Prediction and Response in HPC Systems: Methodology and Example," *Int. Conf. Dependable Syst. Netw. Workshops*, Chicago, IL, USA,

June 28–July 1, 2010, pp. 2–7.
[14] Z. Pawlak, "Rough Sets," *Int. J. Comput. Inf. Sci*., vol. 11, no. 5, Oct. 1982, pp. 341–356.
[15] G. Alfredo et al., "A New Proposal for Multi-objective Optimization Using Differential Evolution and Rough Sets Theory," *Genetic Evol. Comput. Conf*., Seattle, WA, USA, July 8–12, 2006, pp. 675–682.
[16] D. Shawky, " The Application of Rough Sets Theory as a Tool for Analyzing Dynamically Collected Data," *J. Eng. Appl. Sci*., Cairo University, vol. 55, no. 6, Nov. 2008, pp. 473–490.
[17] B. Suraj, "*Rough Set Methods for the Synthesis and Analysis of Concurrent Processes*," in *Studies Fuzziness Soft Comput*., Heidelberg, Germany: Springer, 2000, pp. 379–488.
[18] J. Komorowski et al., "Rough Sets: A Tutorial," in *Rough Fuzzy Hybridization: A New Trend in Decision Making*, Singapore: Springer, 1999, pp. 3–98.
[19] J. Liang, Z. Shi, and D. Li, "Applications of Inclusion Degree in Rough Set Theory," *Int. J. Comput. Cognition,* vol. 1, no. 2, June 2003, pp. 67–78.
[20] Z. Pawlak, "Rough Sets" in *Rough Sets Data Mining*, Dordrecht, Netherlands: Kluwer Academic Publisher, 1997, pp. 3–7.
[21] J. Hampton, "Rough Set Theory: The Basics (Part 1)," *J. Comput. Intell. Finance,* vol. 5, no. 6, Jan.–Feb. 1997, pp. 25–29.
[22] X. Hu, T. Lin, and J. Han, "A New Rough Sets Model Based on Database Systems," *Fundam. Informat*., vol. 59, no. 2–3, Apr. 2004, pp. 125−152.
[23] *ROSE2*, Rough Sets Data Explorer. Laboratory of Intelligent Decision Support Systems. Poznan University of Technology, Poland. Accessed Jan. 22, 2012. http://idss.cs.put.poznan.pl/site/

Doaa Shawky received her BS degree in electronics and communications engineering in 1996, her MS degree in computer engineering in 2000, and her PhD degree in engineering mathematics in 2005. All degrees were received from Cairo University, Faculty of Engineering (CUFE), Giza, Egypt. She has been working for CUFE since 1998. She has been working as an associate professor with the Engineering Mathematics Department at CUFE since 2012. She is a member of the IEEE computer society. Her main research interests include data analysis, software engineering, cloud computing, and evolutionary computation techniques.