Detecting Failures in HPC Storage Nodes

Anas A. Hadi*, Fadi F. Fouz**

Abstract— Future High-Performance Computing (HPC) systems are expected to include more components than current HPC systems. This increase in components, decreases the reliability of the system. Recovering from failure is one of the hardest problem in future HPC. In this research, storage node failures will be considered, as they are the least reliable hardware components due to their mechanical aspects. Ensemble Learning was proposed as a prediction algorithm to predict the failure in these nodes. According to our evaluation, acceptable prediction could be obtained with sufficient lead time window.

Keywords: Future High-Performance Computing (HPC), Ensemble Learning, Prediction, Storage Nodes.

1 Introduction

Future High-Performance Computing (HPC) systems will need concurrent use and control of hundreds of thousands of processing, storage, and networking nodes. With this large number of elements, failures will no longer be rare event, it will become a normal event [1]. The success of future HPC will depend on the ability to provide high Reliability, Availability and Serviceability (RSA).

Reliability can be defined as the probability that a system will produce correct outputs up to some given time t [2]. Reliability is enhanced by avoiding, detecting and repairing faults. A reliable system does not silently continue and deliver results that include uncorrected corrupted data. Instead, it detects and, if possible, corrects the corruption. Reliability can be characterized in terms of Mean Time Between Failures (MTBF) [2].

Availability represents the amount of time a device is actually operating. It is given as a percentage of total time it should be operating. Availability features allow the system to stay operational even when faults do occur.

Serviceability is the speed with which a system can be repaired or maintained. If the time to repair a failed system increases, then availability will decrease. Serviceability includes various methods which easily diagnose the system when problems arise. Early detection of faults can decrease or avoid system downtime. Thus detection and prediction methods are very useful for increasing RSA.

*Anas A. Hadi is currently pursuing PhD degree program in computer Science in King Abdulaziz University, Saudi Arabia. E-mail: Anas1401@gmail.com
**Fadi F. Fouz is currently professor in computer science in King Abdulaziz University, Saudi Arabia. E-mail: ffouz@kau.edu.sa

Analyzing current HPC, using Top500 HPC systems[3], it is clear that the number of HPC components are steadily increasing. On the other hand, the overall system MTBF is reduced to just a few hours [4]. For example, the IBM Blue Gene/L was built with 131,000 processors. If the MTBF of each processor is 876,000 hours (100 years), a cluster of 131,000 processors has an MTBF of 876,000/131,000 = 6.68 hours [5].

In general, if we assume that in a system of m components, the MTBF of any component i is independent of all other components, the reliability R of the system will be:

$$R = \frac{component\ MTBF}{m}$$

We can conclude that, in HPC systems if the number of components is increased, then the system reliability will be decrease.

In this research, storage node failures will be considered. The reason is the potential severity of storage failures, which can not only cause temporary system unavailability, but in the worst case lead to permanent data loss. Moreover, disks are the least reliable hardware component, due to the mechanical aspects of a disk [6, 17].

Nowadays, large scale storage systems usually deploy massive hard disk drives as primary data storage device. To provide high reliability in such systems, failure avoidance is done by taking a preventive action [7]. When a part of an application running on a node that seems likely to fail (which may lead to failure of the whole application), fault-tolerant techniques, such as replication and erasure code are often used.

The main issue here is that, these techniques rely primarily on accurate prediction of node failures that will occur. In order to predict the failure of a storage node, we need some historical data about the health of these nodes.

Currently, almost all hard drive manufacturers have implemented Self-Monitoring, Analysis and Reporting

Technology (SMART) [8] in their products, which monitor internal attributes of individual drives and raise an alarm if any attribute exceeds a pre-defined threshold. However, it has been estimated that the threshold algorithm can only reach a failure detection rate of 3 ~ 10% at 0.1% False Alarm Rate (FAR) [8]. Some statistical and machine learning methods have been proposed to build better prediction models based on the SMART attributes [9, 10, 11, 12, and 13]. However, their failure detection rates are only up to 50% ~ 60% with low FARs. Thus there is a need for more research in order to improve the prediction accuracy for storage nodes in HPC.

2 Proposed Methods

Fault prediction algorithms help the user to get a warning of the faults that are going to occur in the system. They are used in order to increase the migration performance. The main purpose of this research is to improve storage node failure prediction by applying Ensemble Learning as prediction algorithm. SMART data will be used to implement the predictor.

2.1 Ensemble learning

In this research classification trees will be used as weak classifiers, and will be combined using Ensemble learning concept. Ensemble learning is the process by which multiple weak classifiers are combined to solve a particular computational intelligence problem [14]. In order to increase the accuracy, the outputs of these classifiers or decisions are combined using weighted or unweighted voting. Bagging is one of the earliest, most intuitive and perhaps the simplest ensemble based algorithms, with a surprisingly good performance [14]. Classification trees use a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

2.2 Used Metrics

		Predicted Drive Statues: Positive	Predicted Drive Statues: Negative
Actual Drive Statues	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

considering the above illustration, the our experiment will be evaluated using three key metrics:

Accuracy: This represents the percentage of correctly predicted drive statues (Good or failed).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Detection Rate: This represents the percentage of failed drives that are predicted correctly as failed.

$$Detection\ Rate = \frac{TP}{TP + FN}$$

False Alarm Rate (FAR): is another important metric which represents the fraction of good drives that are miss-classified as failed. High FAR implies too many false-alarmed drives and results in heavy processing cost. False Alarm is also called False Positive Rate (FPR) in the terminology of machine learning.

$$FAR = \frac{FP}{FP + TN}$$

Using both Detection Rate and False Alarm Rate, Receiver Operating Characteristic (ROC) curve could be used to evaluate the results. We need to increase the Detection Rate and reduce the False Alarm Rate in the same time. Thus the optimal results should be as near as possible to upper left corner as we can see in Figure 1

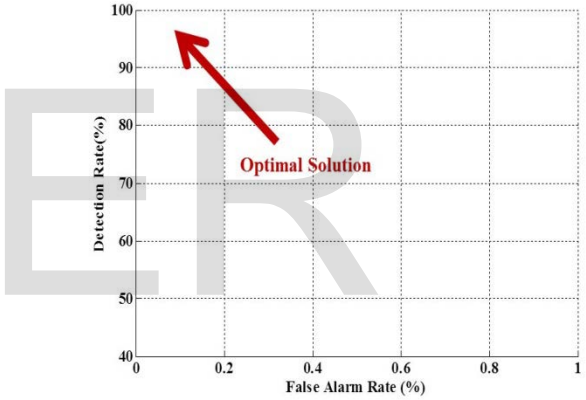

Fig.1 Optimal solution using Detection Rate & False Alarm Rate

Beyond the predictor performance, sometimes there is not enough time to take proactive actions such as migration. Thus Lead time window size is important as shown in Figure 2

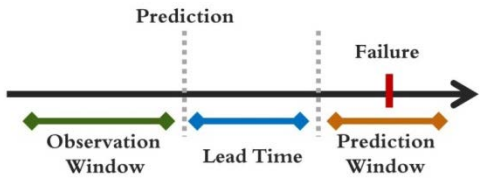

Fig.2 Observation window, Lead Time, and Prediction window for failure prediction

Thus, the following three issues will be investigated:

- ✓ Can we increase the accuracy of storage node failure prediction by applying Ensemble Learning?
- ✓ Can we decrease the False Alarm Rate (FAR) of storage node failure prediction by applying Ensemble Learning?
- ✓ What is the effect of changing the size of Lead time window, and how far we can extend this size?

3 Experiment Evaluation

3.1 Used Datasets

The first used dataset consists of 396 drives with the same model. The sampling rate was one read per two hours. Each drive is labeled good or failed, with 191 drives in the failed class and 178 in the good class. For both drives, samples for the last25 days (600 hours) are kept in the dataset [12].

The second one was collected from the data center of Baidu Inc.[15]. There are 23,395 drives in this dataset and all of them are with the same model. SMART attribute values were sampled from each working drive at every hour. Each drive is labeled good or failed, with 433 drives in the failed class and the rest (22,962 drives) in the good class. For good drives, the samples in a week are kept in the dataset. For failed drives, samples in a longer time period (20 days before actual failure) are saved. Every attribute value has been scaled to the same interval [-1, 1] and their exact values withhold. The serial-number of the disk is replaced by a number ranging from 1 to 23,395 [7]. Table 1 lists the differences between the two used datasets.

TABLE 1 COMPARING USED SMART DATASETS

	SMART dataset #1	SMART dataset #2
Published Year	2005	2013
# All Drives	369	23,395
# Good Drives	178	22,962
# Failed Drives	191	433
SMART attributes samples	Per 2 hour for each disk	Per one hour for each disk
Reading period (Good Drives)	600 h (25 days)	168 h (7 days)
Reading period (Failed Drives)	600 h (25 days)	480 h (20 days)
SMART attributes count	64	14
# All samples in dataset	68,411	4,006,453

For the second dataset, each line in the dataset contains 14 columns which are separated by commas. The meaning of each column is listed as follows:

1) Index of the disk: representing its serial-number, ranging from 1 to 23,395.
2) Class label of the disk: -1 for failed and +1 for good.
3) Read Error Rate: Frequency of errors during read operations.
4) Spin Up Time: Time required a spindle to spin up to operational speed.
5) Normalized value of Reallocated Sectors Count: The number of the unused sectors. When encountering a read/write/check error, a device remaps a bad sector to a healthy one.
6) Seek Error Rate: Frequency of the errors during disk head positioning.
7) Power on Hours: The Raw value shows the actual powered-on time, usually in hours.
8) Reported Uncorrectable Errors: The number of UNC errors, i.e. read errors which Error Correction Code (ECC) failed to recover.
9) High Fly Writes: The number of write errors caused by the fact that a write head was outside normal range of height above disk platter.
10) Temperature: Temperature monitored by a sensor somewhere inside the drive.
11) Hardware ECC Recovered: The number of errors which were corrected using Error Correction Code.
12) Normalized value of Current Pending Sector Count: The number of unstable sectors which are waiting to be re-tested and possibly remapped.
13) Raw value of Reallocated Sectors Count: The number of the unused sectors. When encountering a read/write/check error, a device remaps a bad sector to a healthy one.
14) Raw value of Current Pending Sector Count: The number of unstable sectors which are waiting to be re-tested and possibly remapped.

3.2 Experiment setup

In order to conduct the experiment, MATLAB R2014a was used as programing tool [16]. The two datasets were processed and imported to MATLAB.

The first step was to prepare the data for the experiment, in this step a new set of the data was generated for different lead time window. 16 sets were generated for the lead window from 6h up to 96h (4days) with step of 6h between each set. Observation window was fixed to 24h during all the experiment. This step is essential for studying the effect of changing the size of Lead time window.

The data was divided into 10 sets in order to apply the cross validation as the following. The first set is used as

testing, while the rest sets were merged to provide the training data. Using this training data, ensemble learning of three classification trees was learned. Then for the test set, the true drives labels were used to evaluate the performance. This step (training and testing) was repeated 10 times, one time for each set. And the whole experiment was repeated 10 times. Both of the Accuracy and False Alarm Rate metrics were calculated. And the mean values over all experiment were considered. Figure.2 shows the flowchart for the evaluation experiment.

The previous steps were done on samples level. As it mentioned above that for each drive the observation window was fixed to 24h. This means that we have 24 samples for each drive, thus 24 predicted values for each drive. In order to get the predicted statues of the drive voting methodology was used. If we have the majority predicted values as failed samples, then the predicted drive status will be failed and vice versa. If they are equal then the predicted drive status will be failed will be failed as a precaution.

The flowchart of the evaluation experiment is illustrated in Figure 3

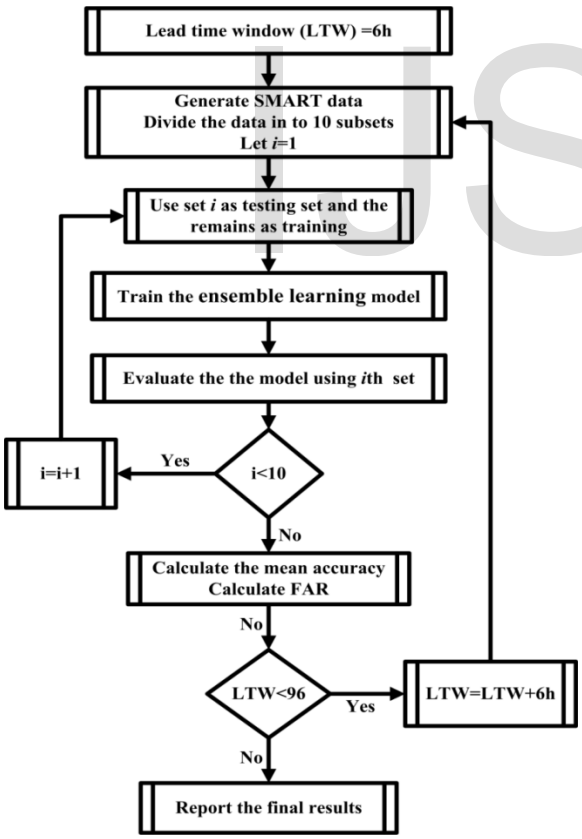

Fig.3 Evaluation Experiment

4 Results and Discussion

Table 2 illustrates the results of predicting drive failure using ensemble learning. The mean accuracy rate were 91.89% and 99.32% for dataset one and two respectively. The second dataset has a huge number of drives thus our ensemble learner was trained well to get this accuracy. The mean detection rate for the first dataset was 92.45% which is better than the mean detection rate of the second dataset (81.40%). That is due to the fact that the second dataset was unbalanced towered the good drives and the failure on one drive effects the detection rate since the number of failed drives are relatively small. The mean false alarm rate for the second dataset was 0. 37% which is much better than the mean false alarm rate of the first dataset (9.88%). This difference can be explained again with that fact of the unbalanced data of dataset one. The number of good drives was very high such that the model as trained well for it.

TABLE 2 RATING VECTORS SAMPLE FOR 5 USERS AND 4 CONTEXTS

	SMART dataset #1	SMART dataset #2
Accuracy Rate (%)	91.89±1.9318	99.32±0.0408
Detection Rate (%)	92.45±1.74	81.40±1.14
False Alarm Rate (%)	9.88±4.06	0. 37±0.408

In order to study the effect of changing the lead time window, detection rate versus the lead time were calculated. Figure 4 shows this relationship for the first dataset. We ca see that when we increase the lead time the detection rate is reduced. In the same time we can see that the least achieved the detection rate was 89% which is relatively high. Figure 5 shows the effect of changing the lead time window for the second dataset. We can see that the detection rate was not affected much by the changing in the lead time. The least achieved the detection rate was 79%.

Figure 6 and Figure 7 illustrate the effect of changing the lead time window on the false alarm rate. For both of them it is clear that increasing lead time window will also increase the false alarm rate. We can see also that the effect is higher for the first dataset.

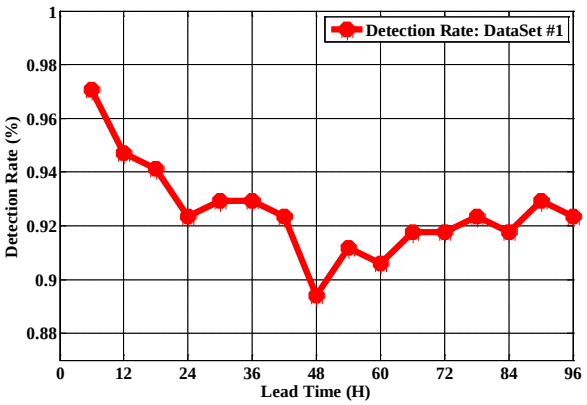

Fig.4 Detection rate VS. Lead time for the first dataset

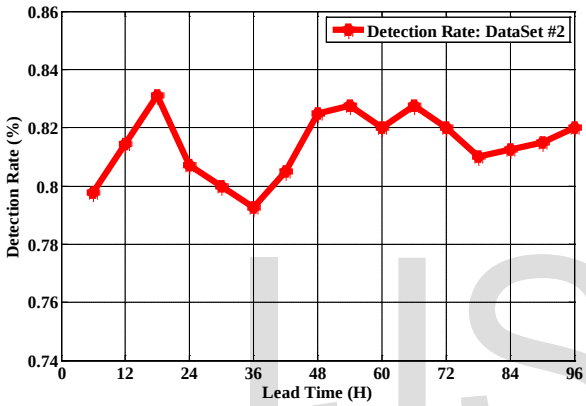

Fig.5 Detection rate VS. Lead time for the second dataset

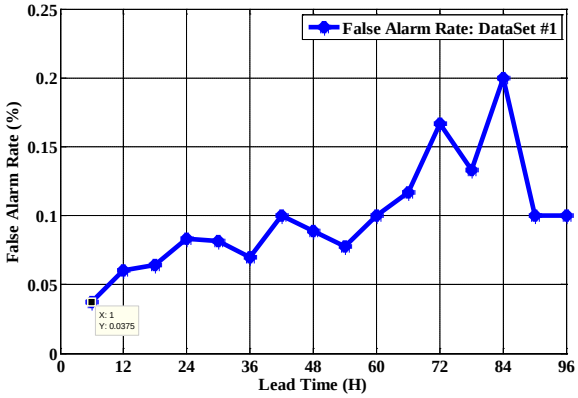

Fig.6 FAR VS. Lead time for the first dataset

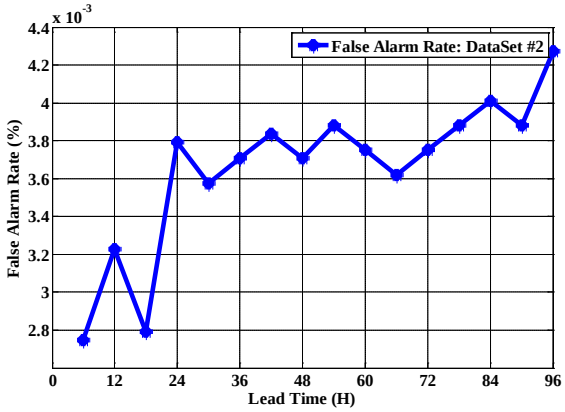

Fig.7 FAR VS. Lead time for the second dataset

In order to compare our results with the results achieved in previous studies [7,15], we compare the results using ROC plot. As we mentioned before the optimal solution should have both high detection rate and low false alarm rare in the same time. Figure 8 shows the achieved results compared with previous study for the first dataset. We can see that our result is much higher in detection rate than the ones achieved using support vector machines (SVM). But using SVM they could achieve 0.00% false alarm rate with detection rate of 51%. Figure 9 shows the comparison for the second dataset. We can see that our method outperforms SVM. But the results of Neural Networks are clearly better than our method.

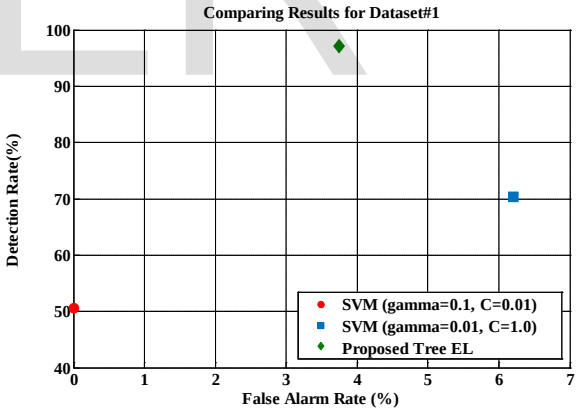

Fig.8 Comparing our results with previous studies for the first dataset

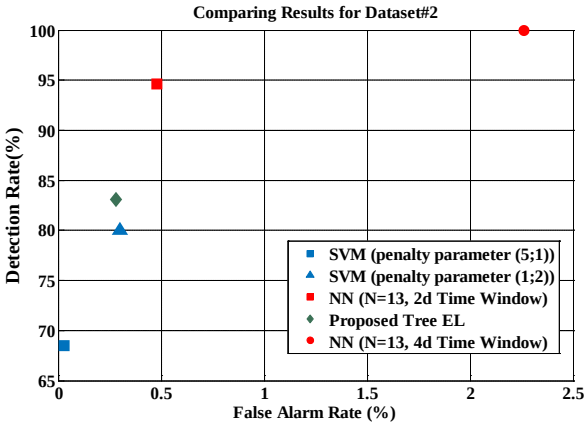

Figure 9 Comparing our results with previous studies for the Second dataset

5 Conclusions and Future Work

In this research, we have illustrates the effects of failure prediction of HPC storage nodes. The main aim was to increase the reliability of HPC. Ensemble Learning was used as prediction algorithm in order to predict these failures. Our proposed method was applied using two SMART datasets. The results show that acceptable prediction could be obtained even with lead time windows (up to 4 days). This widow is sufficient to perform any precaution actions.

As a future work we can include the effect of changing the costs of learning. Using the costs could enhance the false alarm rate. Other direction is to use the neural networks in the ensemble learning model. Even though the learning process may take longer time, the classification of new samples will still very short. A third direction is to integrate this method (with little modification) in online detection system. This step may have considerable effect in the performance of online failure prediction of HPC.

References

[1] Ferreira, K. B. (2011). Keeping Checkpointing Viable for Exascale Systems (Doctoral dissertation, THE UNIVERSITY OF NEW MEXICO).

[2] McCluskey, E. J., & Mitra, S. (2004). Fault tolerance. Computer science handbook, 2nd edn. Chapman and Hall/CRC Press, London.

[3] Top500 (2012) [Online]. Available: http://www.top500.org

[4] Cappello, F., Geist, A., Gropp, B., Kale, L., Kramer, B., & Snir, M. (2009). Toward exascale resilience. International Journal of High Performance Computing Applications.

[5] Egwutuoha, I. P., Levy, D., Selic, B., & Chen, S. (2013). A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. The Journal of Supercomputing, 65(3), 1302-1326.

[6] Schroeder, B., & Gibson, G. A. (2007, July). Understanding failures in petascale computers. In Journal of Physics: Conference Series (Vol. 78, No. 1, p. 012022). IOP Publishing.

[7] Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., & Ma, J. (2013, May). Proactive drive failure prediction for large scale storage systems. In Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on (pp. 1-5). IEEE.

[8] Allen, B. (2004). Monitoring hard disks with smart. Linux Journal, (117), 74-77.

[9] Hamerly, G., & Elkan, C. (2001, June). Bayesian approaches to failure prediction for disk drives. In ICML (pp. 202-209).

[10] Hughes, G. F., Murray, J. F., Kreutz-Delgado, K., & Elkan, C. (2002). Improved disk-drive failure warnings. Reliability, IEEE Transactions on, 51(3), 350-357.

[11] Murray, J. F., Hughes, G. F., & Kreutz-Delgado, K. (2003, June). Hard drive failure prediction using non-parametric statistical methods. In Proceedings of ICANN/ICONIP.

[12] Murray, J. F., Hughes, G. F., & Kreutz-Delgado, K. (2005). Machine learning methods for predicting failures in hard drives: A multiple-instance application. In Journal of Machine Learning Research (pp. 783-816).

[13] Zhao, Y., Liu, X., Gan, S., & Zheng, W. (2010). Predicting disk failures with HMM-and HSMM-based approaches. In Advances in Data Mining. Applications and Theoretical Aspects (pp. 390-404). Springer Berlin Heidelberg.

[14] Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

[15] Baidu, Inc (2000) [Online]. Available: https://www.baidu.com/

[16] MATLAB Toolbox Release R2014a, The MathWorks, Inc., Natick, Massachusetts, United States.

[17] Schroeder, B., & Gibson, G. A. (2007, March). The computer failure data repository (CFDR). In Workshop on Reliability Analysis of System Failure Data (RAF'07), MSR Cambridge, UK.