Toward Adaptive Disk Failure Prediction via Stream Mining

Shujie Han^1, Patrick P. C. Lee^1, Zhirong Shen^2, Cheng He^3, Yi Liu^3, and Tao Huang^3
^{1}The Chinese University of Hong Kong ^{2}Xiamen University ^{3}Alibaba Group

Abstract—**We explore machine learning for accurately predicting imminent disk failures and hence providing proactive fault tolerance for modern storage systems. Current disk failure prediction approaches are mostly offline and assume that the disk logs required for training learning models are available a priori. However, in large-scale disk deployment, disk logs are often continuously generated as an evolving data stream, in which the statistical patterns vary over time (also known as concept drift). Such a challenge motivates the need of online techniques that perform training and prediction on the incoming stream of disk logs in real time, while being adaptive to concept drift.**

We present STREAMDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. We start with a measurement study and demonstrate the existence of concept drift on various disk models based on the datasets from Backblaze and Alibaba Cloud. Motivated by our study, we design STREAMDFP with three key techniques, namely (i) online labeling, (ii) concept-drift-aware training, and (iii) general prediction, with a primary objective of making STREAMDFP support various machine learning algorithms as a general framework. Our evaluation shows that STREAMDFP improves the prediction accuracy significantly compared to without concept-drift adaptation under various settings, and achieves reasonably high stream processing performance.

I. INTRODUCTION

Maintaining storage reliability is a critical yet challenging requirement for modern cloud-scale data centers, typically composed of thousands to millions of disks [32], [44]. In large-scale disk deployment, disk failures are prevalent [11], [42]; more severely, sector error bursts [6], [41] and correlated disk failures (e.g., cluster-wide power outages) [5], [14] can cause the simultaneous crashes of all replicas of a data chunk, leading to data loss and unavailable services. Although traditional redundancy mechanisms, such as replication and RAID, are widely used for data protection, they are no longer sufficient for providing strong reliability guarantees in the face of prevalent and correlated failures [32].

To complement existing redundancy mechanisms, we explore the prediction of imminent disk failures based on *machine learning* as a proactive fault tolerance mechanism to pinpoint and replace soon-to-fail disks, before the actual disk failures happen. In particular, various machine learning algorithms (e.g., [11], [29]–[31], [33], [43], [44], [46]) are shown to achieve highly accurate prediction. Such algorithms capture disk logs with performance and reliability statistics (e.g., SMART (Self-Monitoring, Analysis and Reporting Technology)) as the training data from a set of disks with known labels (i.e., healthy or failed). They train a prediction model using the training data, and use the trained prediction model to predict if any unknown disk (i.e., no label) will remain healthy or become failed in near future. Evaluation on production workloads (e.g., SMART logs from Backblaze [1]) justifies the effectiveness of machine learning; for example, over 95% of failed disks can be predicted in advance with a very small false positive rate [11], [29], [33], [46].

Existing disk failure prediction schemes are mostly *offline*, meaning that all training data must be available in advance before training any prediction model. On the other hand, in practice, disk logs are continuously generated from disks over time. With the enormous scale of the disk population in production environments, it is infeasible to keep all past data for training, rendering offline approaches inadequate for long-term use. Recent work [43] explores *online* disk failure prediction based on the Online Random Forests (ORF) algorithm, by labeling the healthy and failed disk samples and updating the prediction model on the fly. We believe that online prediction is essential for large-scale disk deployment.

However, how to generalize online disk failure prediction for various machine learning algorithms remains unexplored and non-trivial. As different disk models are subject to reliability heterogeneity [28], it is impractical to identify the "best" learning algorithm that applies to all disk models. More importantly, the statistical patterns of disk logs are varying over time (e.g., due to the aging of disks, or the additions/removals of disks in production). Such a phenomenon, known as *concept drift* [19], implies that we must carefully identify the proper window of samples for training: if we choose too few samples, we do not have sufficient samples to build an accurate prediction model; if we choose too many samples, the prediction model may be disturbed by the old samples that no longer correctly capture the failure characteristics of the current pool of disks due to concept drift.

This motivates us to regard disk failure prediction as a *stream mining problem*. By viewing disk logs as an evolving stream of time-series samples, we process the samples through the following steps in real time: (i) train the prediction model incrementally over the stream of samples, (ii) detect concept drift and adapt the prediction model using a properly tuned number of samples, and (iii) predict the failure status of any unknown disk. Prior work has proposed algorithms on adapting machine learning to concept drift in stream mining (Section II). An open question is to support and customize various machine learning algorithms with concept-drift adaptation for a diverse mix of disk models in disk failure prediction.

We propose STREAMDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. STREAMDFP is designed to support a variety of machine learning algorithms (rather than specific algorithms), based on three key techniques: (i) *online labeling*, which labels the samples for a disk on the fly; (ii) *concept-drift-aware training*, which incorporates concept-drift adaptation when training a prediction model; and (iii) *general prediction*, which supports both classification (i.e., whether a disk will fail in near future) and regression (i.e., how likely a disk will fail in near future). To summarize, this paper makes the following contributions:

- We motivate our work via an extensive measurement study on five SMART datasets, four from the public Backblaze dataset [1] and one from Alibaba Cloud; the latter has a much larger disk population than the total of the four Backblaze datasets. We demonstrate not only the existence of concept drift in all five datasets, but also the variation of concept-drift existence across healthy/failed disks, disk models, and SMART attributes.
- We present the complete design of STREAMDFP as a general stream mining framework that extracts features, labels samples, and trains a prediction model, all in real-time.
- We implement a complete prototype of STREAMDFP based on Massive Online Analysis (MOA) [10].
- We evaluate both prediction accuracy and stream processing performance of nine decision-tree-based algorithms on our five datasets. STREAMDFP increases the precision, recall, and F1-score by 27.5-71.8%, 15.7-37.4%, and 26.8-53.2%, respectively, through concept-drift adaptation. It also supports fast stream processing: it performs training and prediction in 13.5 seconds on the daily SMART data of 37 K disks.

We now open-source our STREAMDFP prototype that can be used for the validation on the public Backblaze dataset at **http://adslab.cse.cuhk.edu.hk/software/streamdfp**.

II. BACKGROUND

In this section, we provide background details on disk failure prediction and stream mining.

A. Disk Failure Prediction

Our goal is to predict imminent disk failures over a collection of disks in production based on SMART statistics. SMART is a widely adopted disk monitoring tool for collecting performance and reliability statistics of a disk. Modern disks include SMART in their firmware. With SMART enabled, a disk periodically reports various numerical values (called *attributes*) on operational status and error information. Some SMART attributes provide useful indicators for soon-to-fail disks. For example, RAIDShield [32] suggests to proactively replace a disk in production whose reallocated sector count (i.e., the attribute SMART-5) exceeds 200. However, checking SMART attributes against thresholds for disk failure prediction is highly error-prone, as its accuracy heavily depends on how the thresholds are configured. In this work, we explore the use of machine learning in disk failure prediction.

Specifically, we formulate our disk failure prediction as a *stream mining problem*, by viewing the SMART attributes emitted by disks as a stream of *samples* over a time series. For each disk i, where i is a unique disk identifier, we denote the SMART attributes emitted by disk i at time t as a vector $\mathbf{x}_t^i$ (called the *input variable*), and denote the failure status of disk i at time t as a scalar variable y_t^i (called the *target variable*). We assume that the SMART attributes are generated at the granularity of *days*, so each time t refers to a particular day. We feed $\mathbf{x}_t^i$ into a *prediction model* (denoted by $\mathcal{M}$) to predict the future failure status of disk i (denoted by $\hat{y}_t^i$) (e.g., within the next 30 days). As the true output for y_t^i is known, we update $\mathcal{M}$ over time with $(\mathbf{x}_t^i, y_t^i)$ (called a *labeled sample*). We also refer to the samples that correspond to the failed disks and healthy disks as *positive samples* and *negative samples*, respectively.

In practice, we collect SMART attributes and predict failures over a collection of disks simultaneously at each time point. For brevity of discussion, we omit the superscript i and use $\mathbf{x}_t$ and y_t to refer to the SMART attributes and the failure status of the *whole* collection of disks, respectively.

We consider two types of prediction: (i) *classification*, in which we predict if disk i is either failed or healthy in the future, and y_t^i is equal to either 1 or 0, respectively; and (ii) *regression*, in which we predict the likelihood that disk i is failed, and set y_t^i as some continuous value between 0 and 1.

B. Concept Drift

Concept drift [19] describes the phenomenon that the relationship between the input variables and the target variable continuously changes over time. Mathematically, let t_0 and t_1 be two time points in a stream (assuming $t_0 < t_1$), and $p(\mathbf{x}_t, y_t)$ be the joint probability of $\mathbf{x}_t$ and y_t at time t. We say that concept drift occurs if $p(\mathbf{x}_{t_0}, y_{t_0}) \neq p(\mathbf{x}_{t_1}, y_{t_1})$; in this case, the prediction model $\mathcal{M}$ can no longer accurately map $\mathbf{x}_{t_1}$ to y_{t_1}.

In our problem setting, we focus on detecting the concept drift in $p(y_t|\mathbf{x}_t)$ (i.e., the posterior probability of the target variable y_t given the input variable $\mathbf{x}_t$), as it describes the change of our prediction results. Based on the Bayesian decision theory, we can express $p(y_t|\mathbf{x}_t) = \frac{p(y_t)p(\mathbf{x}_t|y_t)}{p(\mathbf{x}_t)}$, where $p(\mathbf{x}_t)$ is the marginal probability of $\mathbf{x}_t$, $p(y_t)$ is the prior probability of y_t, and $p(\mathbf{x}_t|y_t)$ is the conditional probability of $\mathbf{x}_t$ given y_t. Thus, the change of $p(y_t|\mathbf{x}_t)$ can be characterized as the changes in the components $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$. We measure the changes in such components (Section III-B). Our goal is to adapt the prediction model $\mathcal{M}$ with $p(y_t|\mathbf{x}_t)$.

C. Change Detection

We perform change detection in stream mining to identify the existence of concept drift in $p(y_t|\mathbf{x}_t)$. Specifically, we define the absolute error, denoted by ϵ_t^i, at time t as $\epsilon_t^i = |\hat{y}_t^i - y_t^i|$, where $\hat{y}_t^i$ and y_t^i denote the predicted and true output for disk i at time t, respectively. We take a stream of ϵ_t^i's over a time window as input in change detection.

We can apply different change detectors. For example, ADaptive sliding WINdow (ADWIN) [8] keeps a variable-size sliding detection window of the most recent samples. It partitions the detection window into two sub-windows and monitors each of their average values. If the two sub-windows have significantly different average values (based on the Hoeffding bound [23]), then it implies that a change

Category	**Algorithm**	**Change detector**
Classification tree	Hoeffding tree (HT) [16]	None
	Hoeffding adaptive tree (HAT) [9]	ADWIN [8]
Regression tree	Fast incremental model trees with drift detection (FIMT-DD) [26]	PH test [35]
Ensemble learning	Oza's bagging (Bag) [37]	None
	Oza's boosting (Boost) [37]	None
	Online random forests (RF) [20]	None
	Bagging with ADWIN (BA) [9]	ADWIN [8]
	Boosting-like online ensemble (BOLE) [15]	DDM [18]
	Adaptive random forests (ARF) [20]	ADWIN [8]

TABLE I: Overview of incremental learning algorithms.

happens, and ADWIN drops the older sub-window and replaces the detection window with the newer sub-window. Other change detectors include the Page-Hinckley (PH) test [38] and the Drift Detection Model (DDM) [18].

D. Incremental Learning Algorithms

To support adaptive disk failure prediction, we consider several state-of-the-art *incremental learning* algorithms that perform prediction on an input data stream and continuously update the prediction model using the labeled samples. Table I summarizes the incremental learning algorithms that we consider in the paper. Such algorithms all build on *decision trees* to train the prediction model for classification or regression. We can classify them into two categories: *single decision trees* and *ensemble learning*. Instead of advocating a specific incremental learning algorithm for prediction, whose effectiveness highly varies across disk brands and models (Section I), we focus on supporting general incremental learning algorithms for disk failure prediction.

Single decision trees. Several incremental learning algorithms maintain a single decision tree for prediction. The Hoeffding Tree (HT) [16] recursively updates the tree structure using a small subset of labeled samples and decides how many labeled samples are modeled by each tree node using the Hoeffding bound [23]. The Hoeffding Adaptive Tree (HAT) [9] builds on HT and associates ADWIN with each tree node. If ADWIN detects concept drift at a tree node, HAT creates an alternate tree rooted at the tree node and trains it separately. If the original tree has a larger error than the alternate tree, it will be replaced by the alternate tree. Both HT and HAT are designed for classification. On the other hand, FIMT-DD [26] is a regression tree that uses the PH test [35] as the change detector at each tree node. It has similar operations of creating and managing alternate trees like HT and HAT.

Ensemble learning. Single decision trees are limited in both diversity and lookahead ability for large amounts of data [39]. Ensemble learning is proposed to combine multiple decision trees as base learners in prediction. Classical (offline) ensemble learning methods include *bagging* [12], which draws random samples with replacements during training to improve the overall accuracy; *boosting* [17], which trains prediction models iteratively by increasing the weights for less accurate learners to improve the overall accuracy; and *random forests* [13], which train multiple base learners on re-sampled data (similar to bagging) and randomly select subsets of attributes for tree updates. To support online ensemble learning, we adopt Oza's online versions of bagging and boosting [37] and the online random forests in [20], such that they update the prediction models based on incoming labeled samples; however, these online methods do not address concept drift.

Bagging with ADWIN (BA) [9], Boosting-like online ensemble (BOLE) [15], and Adaptive Random Forests (ARF) [20] add concept-drift adaptation to the online versions of bagging, boosting, and random forests, respectively. Their key idea is to associate a change detector with each decision tree in an ensemble of trees. If a tree has concept drift detected, it will be removed and substituted by a new tree root (e.g., in BA and BOLE) or a newly trained tree (e.g., in ARF).

III. Measurement Analysis

In this section, we present a measurement study of the existence of concept drift on production datasets.

A. Datasets

Our analysis builds on five SMART datasets collected from two independent sources, as shown in Table II. Our datasets are diverse, covering different disk models, manufacturers, and production environments. Thus, they allow us to validate the generality of our findings.

The first group of datasets is collected and made publicly available by Backblaze [1], which has released SMART datasets for various hard disk drive models in its data centers since 2013. Here, we select the datasets namely D1, D2, D3, and D4, on four disk models that are among the largest disk populations with the highest disk failure rates. Note that the disk models are also selected and evaluated by prior studies [11], [33], [43].

The remaining dataset is a private SMART dataset collected in Alibaba Cloud. The dataset, namely D5, belongs to a specific hard disk drive model with around 250 K disks, which are at least 6× as many as D2 and D3 and nearly 55× as many as D1 and D4. However, it only has around 1,000 failures (even fewer than those of D1 and D2), implying that the dataset is highly imbalanced as the failure rate is extremely low.

Table III provides an overview of the collected SMART attributes. The datasets span 29 SMART attributes in total. Each collected SMART attribute includes both the raw and normalized values.

B. Measurement of Concept Drift

We now study the existence of concept drift in each dataset. Recall from Section II-B that the change of $p(y_t|\mathbf{x}_t)$ can be characterized through the three components $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$. In the following, we measure the changes of each component to motivate the need of adapting to the change of $p(y_t|\mathbf{x}_t)$ in our disk failure prediction problem[1]. Here, we focus on binary classification (i.e., a disk is healthy or failed).

[1]Note that the changes of all three components do not necessarily imply the change of $p(y_t|\mathbf{x}_t)$. However, we claim that this is unlikely, and our evaluation in Section V shows that adapting to the change of $p(y_t|\mathbf{x}_t)$ is critical to improve the prediction accuracy.

Dataset ID	Disk model	Capacity	Disk count	# failures	Period	Duration (months)
D1	Seagate ST3000DM001	3 TB	4,516	1,269	2014-01-31 to 2015-10-31	21
D2	Seagate ST4000DM000	4 TB	37,015	3,275	2013-05-10 to 2018-12-31	68
D3	Seagate ST12000NM0007	12 TB	35,462	740	2017-09-06 to 2019-06-30	22
D4	Hitachi HDS722020ALA330	2 TB	4,601	226	2013-04-10 to 2016-12-31	45
D5	Private disk model of Alibaba Cloud	6 TB	$\sim$250 K	$\sim$1,000	2019-01-01 to 2019-05-31	5

TABLE II: Overview of datasets.

ID	SMART attribute name	D1	D2	D3	D4	D5
S1	Read error rate	r|n	r|n	r|n	r|n	r|n
S2	Throughput performance	–	–	–	r|n	–
S3	Spin-up time	r|n	r|n	r|n	r|n	r|n
S4	Start/stop count	r|n	r|n	r|n	r|n	r|n
S5	Reallocated sector count	r|n	r|n	r|n	r|n	r|n
S7	Seek error rate	r|n	r|n	r|n	r|n	r|n
S8	Seek time performance	–	–	–	r|n	–
S9	Power-on hours	r|n	r|n	r|n	r|n	r|n
S10	Spin retry count	r|n	r|n	r|n	r|n	r|n
S12	Power cycle count	r|n	r|n	r|n	r|n	r|n
S183	SATA downshift error count	r|n	r|n	–	–	–
S184	End-to-end error	r|n	r|n	–	–	r|n
S187	Reported uncorrectable errors	r|n	r|n	r|n	–	r|n
S188	Command timeout	r|n	r|n	r|n	–	r|n
S189	High fly writes	r|n	r|n	–	–	r|n
S190	Airflow temperature	r|n	r|n	r|n	–	r|n
S191	G-sense error rate	r|n	r|n	–	–	r|n
S192	Power-off retract count	r|n	r|n	r|n	r|n	r|n
S193	Load cycle count	r|n	r|n	r|n	r|n	r|n
S194	Temperature celsius	r|n	r|n	r|n	r|n	r|n
S195	Hardware ECC recovered	–	–	r|n	–	r|n
S196	Reallocation event count	–	–	–	r|n	–
S197	Current pending sector	r|n	r|n	r|n	r|n	r|n
S198	Uncorrectable sector count	r|n	r|n	r|n	r|n	r|n
S199	UltraDMA CRC error count	r|n	r|n	r|n	r|n	r|n
S200	Write error rate	–	–	r|n	–	–
S240	Head flying hours	r|n	r|n	r|n	–	r|n
S241	Total LBAs written	r|n	r|n	r|n	–	r|n
S242	Total LBAs read	r|n	r|n	r|n	–	r|n

"r": Raw value; "n": Normalized value.
– The SMART attribute is not collected or the value is not provided.

TABLE III: Overview of collected SMART attributes.

Measurement of $p(y_t)$. To understand the change of $p(y_t)$, we measure the percentage of failed disks over time (i.e., the percentage of $y_t^i = 1$ over the whole collection of disks) for each dataset. Given the long duration of each Backblaze dataset, we conduct the measurement on D1, D2, D3, and D4 on a monthly basis; on the other hand, we conduct the measurement on D5 on a daily basis.

Figure 1 shows the results. The percentage of failed disks highly oscillates over time. For example, the percentages of failed disks of D1 and D4 can reach as high as 9.7% and 9.1%, respectively; for D5, its daily percentage of failed disks ranges from 0 to 0.09%. One main reason for the highly varying failure rates is that new disks are kept being added, or old disks are kept being retired, over the entire measurement span, so the number of healthy disks varies significantly.

Measurement of $p(\mathbf{x}_t)$. We now measure the change of $p(\mathbf{x}_t)$. Here, we use the two-sample Kolmogorov-Smirnov (KS) test [34] to measure the change of each SMART attribute (based on its raw values) in $p(\mathbf{x}_t)$. Specifically, we compare the samples in two time periods, denoted by t_0 and t_1 under the null hypothesis that the samples of t_0 and t_1 are both drawn from the same probability distribution. We measure the p-value, such that a p-value that is smaller than a threshold (currently set as 5%) will reject the null hypothesis. Here, we measure how many SMART attributes have changed distributions (i.e., their null hypotheses are rejected).

We set the granularity of time periods for the Backblaze datasets as years, while that for the dataset D5 as months. Note that our datasets are imbalanced, with much fewer failed disks than healthy disks. Thus, we *downsample* the healthy disks to prevent them from dominating the overall distributions (we also apply downsampling in our prediction; see Section IV-D). Specifically, for failed disks, we take all their (positive) samples over a time period, while for healthy disks, we only take their (negative) samples at the last day of a time period.

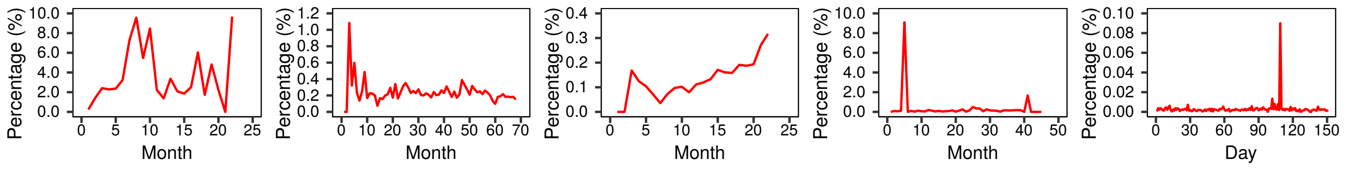
Table IV shows the number of SMART attributes with changed distributions in $p(\mathbf{x}_t)$ over the total number of SMART attributes being collected (we will discuss the changes of $p(\mathbf{x}_t|\text{failed})$ and $p(\mathbf{x}_t|\text{healthy})$ later). We see that a significant fraction of SMART attributes has changed distributions. For example, D2 has more than half of the SMART attributes with changed distributions.

We further study the changes of several *critical* SMART attributes defined by Backblaze, which provide strong indicators for disk failures [3]. Table V shows the presence of changed distributions for each critical SMART attribute based on the KS test. We observe the change of $p(\mathbf{x}_t)$ for D1 and D2 in S187 and S188 for most time periods. However, D3, D4, and D5 do not show a change of $p(\mathbf{x}_t)$ in all critical SMART attributes, meaning that the change mainly appears in other non-critical SMART attributes.

Measurement of $p(\mathbf{x}_t|y_t)$. Finally, we study the change in $p(\mathbf{x}_t|y_t)$. We consider two conditional probability distributions, $p(\mathbf{x}_t|\text{healthy})$ and $p(\mathbf{x}_t|\text{failed})$, for healthy and failed disks, respectively. We revisit Tables IV and V on the changed distributions across the SMART attributes.

From Table IV, a significant fraction of SMART attributes (e.g., at least one-fourth for D2) has changed distributions for healthy and failed disks. However, in D2 and D4, failed disks generally have more SMART attributes with changed distributions than healthy disks, but in D1, D3, and D5, it is opposite. Thus, the effects of changed distributions vary across disk models.

From Table V, failed disks generally show changed distributions in some of the critical SMART attributes (and in all critical SMART attributes for D2 from 2014 to 2015). One reason is that failed disks tend to show various failure symptoms on different critical SMART attributes (which measure the error counts), so the distributions of the critical SMART attributes also have high variations. However, the changed behaviors



(a) D1 (b) D2 (c) D3 (d) D4 (e) D5

Fig. 1: Percentage of failed disks of each dataset in each month (for D1, D2, D3, and D4) or each day (for D5) in the whole duration.

D1	Total	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2014 vs. 2015	24	15	10	8
D2	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2013 vs. 2014	5	2	2	2
2014 vs. 2015	24	15	8	16
2015 vs. 2016	24	15	7	10
2016 vs. 2017	24	14	6	8
2017 vs. 2018	24	14	7	8
D3	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2017 vs. 2018	22	13	13	8
2018 vs. 2019	22	14	13	11
D4	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
2013 vs. 2014	5	2	1	1
2014 vs. 2015	17	6	4	5
2015 vs. 2016	17	6	4	3
D5	**Total**	$p(\mathbf{x}_t)$	$p(\mathbf{x}_t\|\text{healthy})$	$p(\mathbf{x}_t\|\text{failed})$
Jan. vs. Feb.	24	13	13	1
Feb. vs. Mar.	24	13	13	4
Mar. vs. Apr.	24	13	11	15
Apr. vs. May	24	13	13	13

"Total": total number of collected SMART attributes; "$p(\mathbf{x}_t)$", "$p(\mathbf{x}_t|\text{healthy})$", "$p(\mathbf{x}_t|\text{failed})$": numbers of SMART attributes with changed distributions. Note that ST4 and HD2 only collect five SMART attributes (S1, S5, S9, S194, and S197) in 2013.

TABLE IV: Number of SMART attributes with changed distributions.

D1	S5	S10	S184	S187	S188	S197	S198
2014 vs. 2015				○ ‡	○ † ‡	‡	‡
D2	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2013 vs. 2014		–	–	–	–	‡	–
2014 vs. 2015	‡	‡	‡	‡	○ † ‡	‡	‡
2015 vs. 2016	‡			‡	○ †	‡	‡
2016 vs. 2017	‡			○ ‡		‡	‡
2017 vs. 2018	‡			‡	○ †	‡	‡
D3	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2017 vs. 2018	‡		–				
2018 vs. 2019	‡		–			‡	‡
D4	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
2013 vs. 2014		–	–	–	–		–
2014 vs. 2015	‡		–	–	–	‡	
2015 vs. 2016			–	–	–		
D5	**S5**	**S10**	**S184**	**S187**	**S188**	**S197**	**S198**
Jan. vs. Feb.							
Feb. vs. Mar.							
Mar. vs. Apr.	‡			‡		‡	‡
Apr. vs. May	‡			‡		‡	‡

○ $p(\mathbf{x}_t)$ shows a changed distribution.
† $p(\mathbf{x}_t|\text{healthy})$ shows a changed distribution.
‡ $p(\mathbf{x}_t|\text{failed})$ shows a changed distribution.
– The SMART attribute is not collected.

TABLE V: Changed distributions for critical SMART attributes.

across different disk models are highly varying for different critical SMART attributes.

Summary. Our measurement study shows two major observations. First, we observe the presence of changed distributions in $p(y_t)$, $p(\mathbf{x}_t)$, and $p(\mathbf{x}_t|y_t)$, indicating that the change of $p(y_t|\mathbf{x}_t)$ (i.e., concept drift) also likely exists. Second, the changed behaviors cannot be readily predicted, as they vary across healthy and failed disks, disk models, as well as SMART attributes. Thus, the mechanism for adapting to concept drift needs to be generally applicable for various changed behaviors.

IV. Design

We present the design of StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. Specifically, StreamDFP aims to address the following challenges:

- **Online labeling.** Unlike offline learning, StreamDFP accesses a stream of samples from a collection of disks and labels the samples on the fly. It should accurately label the samples as either positive (for failed disks) or negative (for healthy disks) based on the current disk failure patterns.
- **Concept-drift-aware training.** StreamDFP builds on a number of incremental learning algorithms with concept-drift adaptation (Section II-D). It should accurately detect and adapt to concept drift in training a prediction model specifically for disk failure prediction.
- **General prediction.** StreamDFP treats disk failure prediction as both classification and regression problems. For classification, StreamDFP directly answers if an unknown disk will remain healthy or will be failed in near future. For regression, StreamDFP should determine the likelihood that an unknown disk will fail.

A. Architectural Overview

Figure 2 shows the architecture of StreamDFP. Specifically, StreamDFP takes a stream of samples on each day as input. It extracts SMART attributes as learning features (Section IV-B). It configures a sliding window to buffer the recent samples and disk failure status, and then labels the disks on the fly (Section IV-C). It downsamples the negative samples and feeds the labeled samples into the prediction model for training. With change detection enabled, StreamDFP detects concept drift explicitly during training and adapts the prediction model to concept drift. (Section IV-D). In the prediction phase, it uses the prediction model to output the prediction results (for both classification and regression) for an unknown disk (Section IV-E).

B. Feature Extraction

Given an input stream of samples, StreamDFP extracts the SMART attributes of each sample as the learning features for prediction model training. Here, we use *all* collected SMART

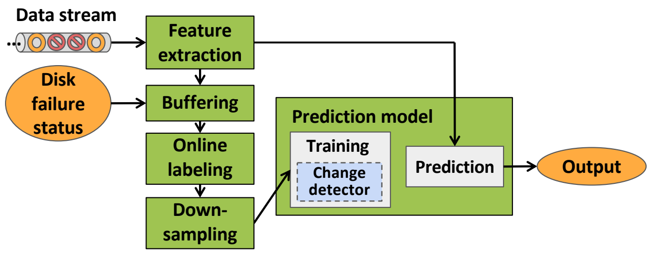


Fig. 2: Architecture of STREAMDFP.

Algorithm 1 Online Labeling

```
1: procedure MAIN($t$, $\mathcal{W}$, $D_L$)
2:     for each failed disk $i$ on day $t$ do
3:         Find $y^i$ = time-series failure status of disk $i$ in $\mathcal{W}$
4:         if classification then
5:             Set $y^i_{t'} = 1$ for all $t' \in [t - D_L, t]$
6:         else if regression then
7:             for each day $t' = t - D_L$ to $t$ do
8:                 Set $y^i_{t'} = 1 - \frac{t-t'}{D_L+1}$
9:             end for
10:        end if
11:    end for
12: end procedure
```

attributes in Table III (including raw and normalized values) as learning features, instead of choosing a subset of SMART attributes based on historical disk logs like [43]. In practice, we may have no or only few historical disk logs for feature selection. Even though historical disk logs are available for us to identify the representative SMART attributes for failure characterization, the selected attributes may vary over time due to the changing distributions of the SMART attributes (Section III-B). Thus, for simplicity, we take all collected SMART attributes as learning features. Note that we can also leverage the statistical distributions of the SMART attributes for feature selection [44]; we pose this as future work.

C. Buffering and Online Labeling

STREAMDFP needs to label the samples on the fly before feeding them into training. A straightforward approach is to label a disk sample as positive once the corresponding disk is diagnosed as failed, or as negative otherwise. However, a disk often does not fail immediately; instead, a soon-to-fail disk has actually shown failure symptoms (e.g., a sharp increase in the reallocated sector count [32]). Thus, STREAMDFP also labels the samples of soon-to-fail disks as positive (in addition to the disk samples of actual failed disks), so as to better reflect the disk failure characteristics. A side benefit is that the proportion of positive samples also increases, which mitigates the well-known *data imbalance* issue in disk failure prediction [11], [33] as failed disks often account for a very small fraction over the entire disk population. A key challenge is how to label the samples of soon-to-fail disks on the fly.

STREAMDFP buffers the recently received samples for online labeling. Specifically, it configures a sliding time window, denoted by $\mathcal{W}$, to buffer the samples of a sufficiently long recent period (30 days in our case), as well as the number of extra labeled days before the disk failure occurs, denoted by D_L. If a failed disk is found, STREAMDFP labels the samples within D_L before the failure as positive (by default, all samples before failures are negative). Note that the number of samples within D_L may be less than the length of D_L when $\mathcal{W}$ is not full or limited samples are collected before a disk failure. By default, we set $D_L = 20$ days, while we evaluate the sensitivity of D_L to the prediction accuracy in Section V. Note that D_L is a configurable parameter.

Algorithm details. Algorithm 1 shows the pseudo-code on online labeling (while buffering is done before the algorithm). It takes the inputs of the current day t, $\mathcal{W}$, and D_L.

We update the labels of the samples of all failed disks within D_L. We label the samples of soon-to-fail disks for classification and regression separately (Lines 2-11). For classification, we set the target variable $y^i_{t'} = 1$ for all $t' \in [t-D_L, t]$ (Lines 4-5). For regression, we update the labels as the failure probabilities. Here, we take a simple approach by modeling the failure probability as a linear function that increases over time, starting from $y^i_{t'} = \frac{1}{D_L+1}$ at $t' = t - D_L$ to $y^i_{t'} = 1$ at $t' = t$ (i.e., when the failure occurs) (Lines 6-10).

D. Downsampling and Training

STREAMDFP trains a prediction model based on the labeled samples. Before training, it is critical to mitigate the data imbalance issue [11], [33] for accurate prediction. In addition to labeling more samples as positive for soon-to-fail disks (Section IV-C), STREAMDFP downsamples the negative samples to increase the proportion of positive samples. After downsampling, STREAMDFP attaches a change detector to the prediction model to adapt to concept drift.

Algorithm details. Algorithm 2 shows the pseudo-code on downsampling and training. It takes the inputs of the current day t, $\mathcal{W}$, and $\mathcal{M}$.

STREAMDFP downsamples the negative samples in a two-phase process. It first selects a subset of samples in $\mathcal{W}$, including all positive samples and the negative samples in the recent days (currently set as seven days to preserve sufficient negative samples), into $\mathcal{W}'$ (Line 2). It further downsamples the negative samples via *Poisson sampling*, which is commonly used in ensemble learning algorithms [20], [37] and online disk failure prediction [43]. We borrow the approach in [43] by customizing the hyper-parameters of the Poisson distribution, denoted by λ_n and λ_p, for the negative and positive samples, respectively. For each decision tree $T \in \mathcal{M}$, we generate a weight k from the respective Poisson distribution of either negative or positive samples, where k specifies the frequency that the sample is updated in the prediction model (Lines 5-9). Since a larger hyper-parameter implies a larger weight, we ensure that $\lambda_p > \lambda_n$ to ensure that the positive samples weigh more in the prediction model. More precisely, our evaluation varies λ_p and λ_n based on the given false positive rate for each incremental learning algorithm (Section V). Note that we do not claim the novelty of this approach.

Algorithm 2 Downsampling and Training

```
1: procedure MAIN($t$, $\mathcal{W}$, $\mathcal{M}$)
2:     Select all positive samples and the negative samples in the
       recent seven days from $\mathcal{W}$ into $\mathcal{W}'$
3:     for each $(\mathbf{x}_t^i, y_t^i) \in \mathcal{W}'$ do
4:         for each decision tree $T \in \mathcal{M}$ do
5:             if $y_t^i$ == 0 then                ▷ negative samples
6:                 Set $k = Poisson(\lambda_n)$
7:             else                                ▷ positive samples
8:                 Set $k = Poisson(\lambda_p)$
9:             end if
10:            if $k > 0$ then
11:                TRAIN($T$, $\mathbf{x}_t^i$, $y_t^i$, $k$)
12:                Set $\hat{y}_t^i$ = PREDICT($\mathbf{x}_t^i$)
13:                if DETECTCHANGE($T$, $\hat{y}_t^i$, $y_t^i$) then
14:                    Update $T$
15:                end if
16:            end if
17:        end for
18:    end for
19: end procedure
```

For each incremental learning algorithm, STREAMDFP extends the corresponding prediction model $\mathcal{M}$ with a change detector (e.g., ADWIN [8], PH test [35], and DDM [18]) (Section II-C). STREAMDFP associates the change detector with each decision tree $T \in \mathcal{M}$ (recall that $\mathcal{M}$ is composed of either a single decision tree or multiple decision trees in ensemble learning (Section II-D)). Specifically, if $k > 0$, STREAMDFP first trains T with a labeled sample $(\mathbf{x}_t^i, y_t^i)$ and weight k (Line 11) and then performs the prediction to obtain $\hat{y}_t^i$ (Line 12). It compares the predicted output $\hat{y}_t^i$ and the labeled target variable y_t^i to detect if concept drift exists (Line 13); if so, T is updated accordingly based on the incremental learning algorithm (e.g., ARF [20] replaces T with a new decision tree trained by recent samples).

E. Prediction

STREAMDFP supports both classification and regression in prediction. For classification, it predicts whether a disk failure will occur, while for regression, it returns the failure probability of a disk within the near future. In particular, for regression, based on how we label a sample as positive (Section IV-C), the product $(1 - \hat{y}_t^i)(D_L + 1)$ can be viewed as the predicted residual lifetime of the disk.

Note that during training (Section IV-D), we call prediction once to detect concept drift (see Line 12 of Algorithm 2). However, we still need to call the prediction again after training, so that the prediction output is based on the updated prediction model due to concept drift.

F. Putting It All Together

Algorithm 3 shows the entire workflow of STREAMDFP. We first initialize $\mathcal{W}$ and $\mathcal{M}$ (Lines 1-2). For each day t, if $\mathcal{W}$ is full, we slide $\mathcal{W}$ by one day (Lines 4-6). We then buffer the samples for online labeling as follows. We extract the learning features to $\mathbf{x}_t$ from all disks and buffer ($\mathbf{x}_t$, y_t) into $\mathcal{W}$ (Lines 7-8). We call Algorithm 1 to label samples in $\mathcal{W}$

Algorithm 3 STREAMDFP

```
1: Initialize $\mathcal{W}$ = empty sliding time window
2: Initialize $\mathcal{M}$ = prediction model
3: for each day $t$ do
4:     if $\mathcal{W}$ is full then
5:         Slide one day for $\mathcal{W}$
6:     end if
7:     Extract learning features to $\mathbf{x}_t$ from all disks
8:     Buffer ($\mathbf{x}_t$, $y_t$) into $\mathcal{W}$
9:     Call Algorithm 1 (online labeling) to label samples in $\mathcal{W}$
10:    Call Algorithm 2 (downsampling and training) to train $\mathcal{M}$
       using $\mathcal{W}$
11:    if $\mathcal{W}$ is full then
12:        Set $\hat{y}_t = \mathcal{M}(\mathbf{x}_t)$
13:    end if
14: end for
```

and Algorithm 2 to train $\mathcal{M}$ using $\mathcal{W}$ (Lines 9-10). Finally, if $\mathcal{W}$ is full, it implies that we have buffered enough labeled samples for training and $\mathcal{M}$ is warmed up, and hence we use $\mathcal{M}$ with $\mathbf{x}_t$ to output the prediction results $\hat{y}_t$ (Lines 11-13).

G. Implementation Details

We implement a STREAMDFP prototype in two parts. The first part is implemented in Python (with around 750 LoC), in which STREAMDFP performs feature extraction, buffering, online labeling, the first phase of downsampling (i.e., selecting a subset of samples from $\mathcal{W}$), and finally writes the processed data into a local file system. The second part is written in Java (with around 900 LoC), in which STREAMDFP reads the processed data from the local file system and feeds each sample into the second phase of downsampling (i.e., Poisson sampling) and training. We realize all incremental learning algorithms and change detectors in Table I using Massive Online Analysis (MOA) [10]. The complete STREAMDFP prototype forms a stream processing pipeline.

V. EVALUATION

We present trace-driven evaluation results on the prediction accuracy and stream processing performance of STREAMDFP. We summarize our findings as follows.

- Enabling concept-drift adaptation in STREAMDFP increases the classification accuracy for different incremental learning algorithms, although the highest accuracy among the algorithms varies across datasets. STREAMDFP also makes earlier prediction of disk failures. (Exp#1)
- Online labeling with extra labeled days improves the overall prediction accuracy in most cases, while the number of extra labeled days can be flexibly tuned via STREAMDFP. (Exp#2)
- STREAMDFP can accurately predict the likelihood of disk failures under regression. (Exp#3)
- STREAMDFP maintains the accuracy gains through concept-drift adaptation for various thresholds of false positive rates. (Exp#4)
- STREAMDFP takes within 13.5 seconds per day for processing 37 K disks in D2, making it viable for practical stream processing usage. (Exp#5)

A. Methodology

Recall that we use all SMART attributes in Table III as learning features for each dataset (Section IV-B). However, for the Backblaze datasets (i.e., D1 to D4), we find that the SMART attributes are not all available at the first day of the collection period, and the datasets have very different collection durations, ranging from 21 to 68 months (Table II)). For consistent comparisons, we select the same 460 days of samples from each Backblaze dataset for evaluation. To ensure that all SMART attributes are available, we set the start date of each dataset as 2014-02-15 for D1, 2015-01-01 for D2, 2018-02-01 for D3, and 2014-09-01 for D4. For the Alibaba Cloud dataset (i.e., D5), we select all 150 days of samples in our evaluation.

For each dataset, we first warm-up the prediction model from scratch using the first 30 days of samples, same as the length of $\mathcal{W}$ (Section IV-C). We then predict disk failures in the next 30 days on a daily basis, and evaluate the accuracy for each day of prediction. Thus, the total durations of our evaluation last for 400 days for D1 to D4 and 90 days for D5.

Metrics. Our evaluation addresses both classification and regression. For classification, we consider the following metrics:

- *Precision:* The fraction of actual failed disks being predicted over all (correctly or falsely) predicted failed disks.
- *Recall:* The fraction of actual failed disks being predicted over all actual failed disks.
- *F1-score:* $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.

For regression, we convert the disk failure likelihood reported by StreamDFP into the number of days of residual lifetime for more intuitive evaluation. We report the *average relative errors of the residual lifetime (ARE)*, defined as the average relative errors of the predicted residual lifetime with respect to the actual residual lifetime over all actual failed disks (we exclude the falsely predicted failed disks for this metric). If the ARE is positive, then it implies that the predicted likelihood of disk failures is larger than the actual likelihood and the residual lifetime is shorter than the actual one, and vice versa.

Default setup. We set the default extra labeled days D_L as 20 days, while we evaluate the impact of D_L in Exp#2. For fair comparisons, when we evaluate the classification accuracy metrics, we fix the threshold of the average false positive rate (FPR) over the evaluation period for each algorithm; on each day, we compute the FPR as the fraction of falsely predicted failed disks over the total number of healthy disks in the next 30 days. For D1 to D4, we set the default FPR threshold as 1% as in [43], while for D5, we set the default FPR threshold as 0.3%; we evaluate the impact on the accuracy for different FPR thresholds in Exp#4. For ensemble learning (Section II-D), we fix 30 decision trees, although including more decision trees (e.g., 100 trees) does not make significant improvements. For other parameters, we choose the default values as in MOA [10] (Section IV-G).

In our experiments, we plot the averaged results over five runs, including the error bars with the minimum and maximum results across all five runs.

B. Results

Exp#1 (Effectiveness of concept-drift adaptation). We first consider the eight classification algorithms in Table I (except the regression algorithm FIMT-DD, which is evaluated in Exp#3). We divide the algorithms into four pairs corresponding to before and after enabling concept-drift adaptation (i.e., HT vs. HAT, Bag vs. BA, Boost vs. BOLE, and RF vs. ARF). For D5, we focus on two pairs of ensemble learning algorithms, i.e., Bag vs. BA and RF vs. ARF, since they can parallelize the training of decision trees for fast execution.

Figure 3 shows the results of the precision, recall, and F1-score for different datasets. Concept-drift adaptation improves the overall F1-score for each pair of algorithms in all datasets, by up to 26.8%, 53.2%, 48.8%, 35.5%, and 49.9% for D1, D2, D3, D4, and D5, respectively, while the improvements of precision and recall are up to 27.5-71.8% and 15.7-37.4% across the datasets, respectively. We observe that concept-drift adaptation mainly improves the precision, while preserving the recall, in most cases; the only exceptions are the decreasing recall in BA (for D2 and D3) and ARF (for D2), as well as the decreasing precision in BOLE for D3, mainly because of the trade-off between the precision and recall.

Overall, ARF achieves the highest F1-score for D1, while BA achieves the highest F1-score for D2, D3, D4, and D5. Thus, it is difficult to identify the "best" algorithm for different datasets. This conforms to the main design goal of StreamDFP that it supports various incremental learning algorithms as a general framework rather than a specific algorithm.

Table VI further evaluates the average number of days ahead of a disk failure when the prediction is made (i.e., the duration from the day when a disk is predicted as failed to the day when the disk failure occurs). Here, we only consider the failed disks that are correctly predicted. The algorithms with concept-drift adaptation can predict disk failures earlier than those without concept-drift adaptation, by up to 2.9, 2.0, 1.8, 1.2, and 3.9 days for D1, D2, D3, D4, and D5, respectively. The reason is that concept-drift adaptation achieves better characterization of the failure patterns, thereby making earlier prediction.

Exp#2 (Sensitivity of extra labeled days). We study how the extra labeled days D_L affects the accuracy. We vary D_L from zero to 30 days; the zero days mean that we only label the samples as positive on the day when the failure occurs. Here, we focus on D2 and D4, which are derived from different disk manufacturers, and the three ensemble learning algorithms with concept-drift adaptation, i.e., BA, BOLE, and ARF.

Figure 4 shows the prediction accuracy versus D_L. In general, introducing extra labeled days (i.e., $D_L > 0$) increases the prediction accuracy especially for ARF in both D2 and D4 (by 23.4% and 19.9% F1-score, respectively) and BOLE in D2 (by 23.9% F1-score) when $D_L = 20$ days (our default setup). However, for D4, a smaller D_L for BOLE and BA achieves higher prediction accuracy. For example, the top two F1-scores are 63.4% and 61.8% for BA are on zero and five days, respectively. This implies that the optimal value of D_L

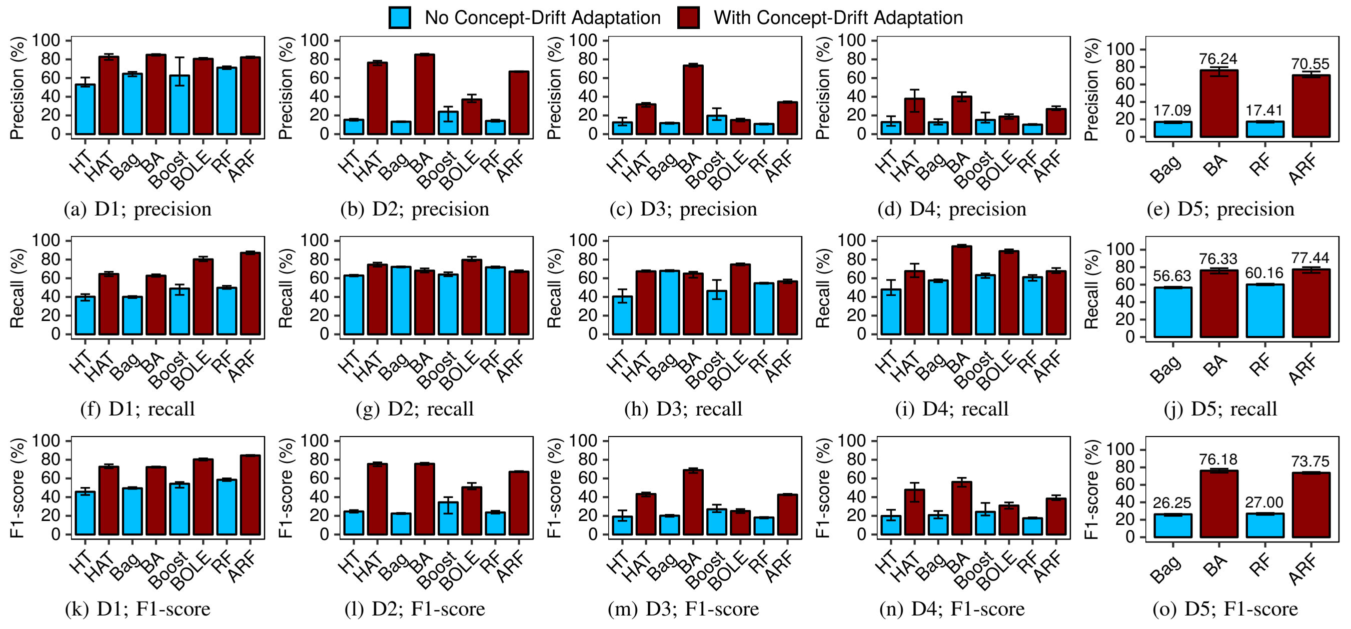


Fig. 3: Exp#1 (Effectiveness of concept-drift adaptation).

Algorithm	D1	D2	D3	D4	D5
HT	11.1	9.1	9.1	11.6	–
HAT	12.3	10.7	10.8	12.0	–
Bag	9.6	9.4	10.9	13.2	7.7
BA	12.2	11.3	11.2	13.9	11.6
Boost	10.5	9.1	9.4	12.3	–
BOLE	13.3	10.6	11.2	13.5	–
RF	11.0	9.7	9.2	12.5	8.0
ARF	13.9	11.3	9.7	12.6	11.5

TABLE VI: Days in advance when a failed disk is predicted for different incremental learning algorithms in Exp#1.

varies across algorithms and datasets. Thus, STREAMDFP opts to allow users to flexibly tune D_L based on production needs.

Exp#3 (Accuracy of regression). We evaluate the accuracy of regression (in terms of ARE) with the regression tree FIMT-DD. Figure 5 shows the ARE results for all datasets. Throughout the evaluation period, the means of ARE are -0.0014, -0.27, 0.13, 0.29, and 0.58 (in days), while the standard deviations of ARE are 3.0, 2.4, 3.2, 5.6, and 2.7 (in days), for D1, D2, D3, D4, and D5, respectively. The results indicate that the predicted likelihood of disk failures is close to the actual likelihood. Also, the maximum absolute values of ARE for D1, D2, D3, and D5 are smaller than 9, 6, 8, and 6 days, respectively. On the other hand, the ARE for D4 is up to +20 days. The reason is that failure symptoms (e.g., sector errors) before failures for D4 last longer than those for other disk models due to the older average age [2], making the predicted residual lifetime for D4 much shorter than the actual one (i.e., the ARE is a large positive value).

Exp#4 (Impact of FPR thresholds). Machine learning models can be configured with a higher recall through increasing the FPR threshold (and vice versa). We study how different FPR thresholds affect the prediction accuracy, and examine if

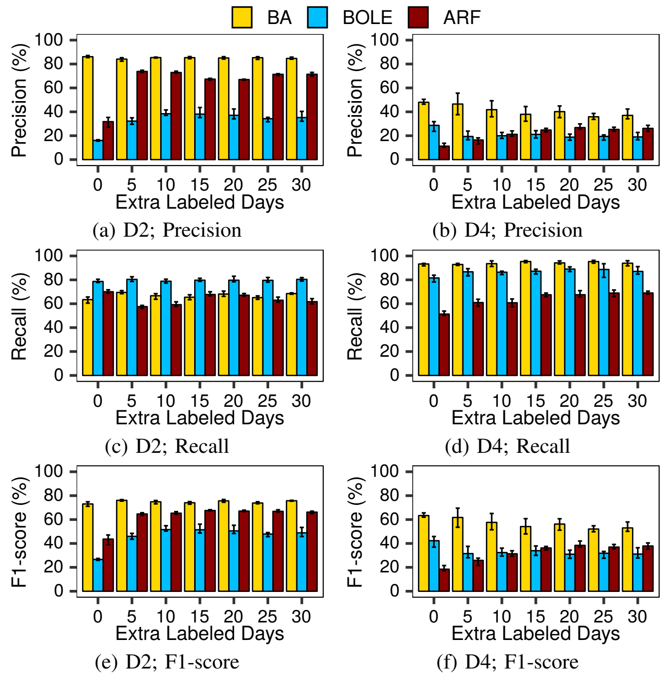
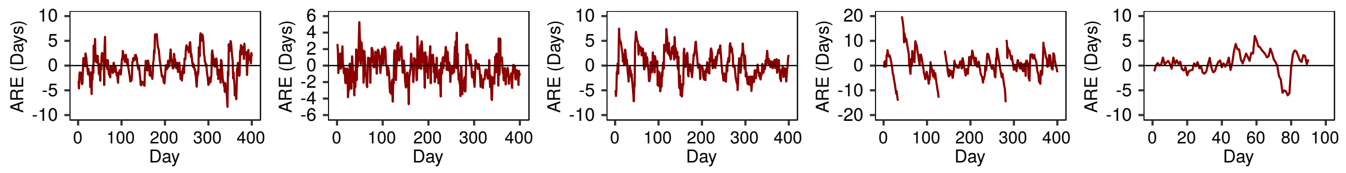


Fig. 4: Exp#2 (Sensitivity of extra label days). Here, we focus on D2 and D4.

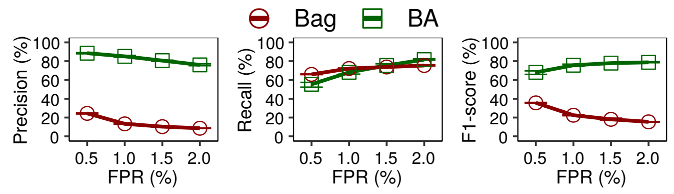
concept-drift adaptation still achieves accuracy gains. We focus on D2 as the representative dataset and bagging (including Bag and BA) as the representative algorithms.

Figure 6 shows the prediction accuracy of Bag and BA for D2 versus the FPR threshold (varied from 0.5% to 2.0%). BA improves the precision and F1-score of Bag by 64.0-71.8% and 32.5-63.3%, respectively, while its recall is less than Bag by 10.5% when the FPR is 0.5% but becomes higher than Bag by 6.3% when the FPR is 2.0%. It shows that BA improves



(a) D1 (b) D2 (c) D3 (d) D4 (e) D5

Fig. 5: Exp#3 (Accuracy of regression). Note that the gaps in D4 mean that no failed disks are observed in those periods.



(a) Precision (b) Recall (c) F1-score

Fig. 6: Exp#4 (Impact of FPR thresholds). Here, we focus on Bag and BA for D2. Note that the error bars are not visible when the FPR is at least 1%.

Step	Feature extraction	Buffer-ing	Online labeling	Down-sampling	Train-ing	Predic-tion
Time	0.0055 s (0.069 ms)	0.093 s (9.6 ms)	0.42 s (30.8 ms)	0.52 s (19.0 ms)	10.6 s (0.75 s)	1.9 s (0.14 s)

TABLE VII: Exp#5 (Execution performance of StreamDFP). Here, we show the average per-day execution time of StreamDFP for D2; the numbers in brackets are standard deviations of all five runs.

the precision and F1-score significantly, while preserving the recall. We also emphasize that the relative differences between Bag and BA are consistent with our findings in Exp#1. We make similar observations for other datasets and algorithms under different FPR thresholds.

Exp#5 (Execution performance of StreamDFP). We evaluate the stream processing performance of StreamDFP. We focus on D2, which has the largest disk population (with around 37 K disks) among all Backblaze datasets we consider. Note that for D5, since it is executed on a production server at Alibaba Cloud, we cannot reveal the hardware setting of the server for our performance evaluation of StreamDFP.

We measure the execution time of StreamDFP on a local server equipped with a quad-core 3.4 GHz Intel Core i5-7500, 32 GiB RAM, and a Toshiba DT01ACA100 7200 RPM 1 TiB SATA hard disk. We run StreamDFP to predict disk failures in 400 days using the BA algorithm (which achieves the highest F1-score), and report the average execution time of one day. We use single-thread execution for feature extraction, buffering, online labeling, and prediction for each arriving sample; for training, we enable multi-threading (with 4 threads) for BA in MOA [10] to parallelize training across all tree learners.

Table VII shows a breakdown of the per-day execution time of StreamDFP for D2, while the standard deviations of all five runs are in brackets. The most time-consuming step is training, while prediction is fast. This is expected, as training has complicated computation in growing multiple trees with the samples in recent days. Nevertheless, the execution time for training remains acceptable in practice. Overall, StreamDFP performs training and prediction within 13.5 seconds on the daily SMART data of 37 K disks. We believe that StreamDFP meets the performance need in large-scale disk deployment.

VI. Related Work

Field studies have analyzed the failure characteristics in production storage environments. Examples include disk replacement rates [40], [42], latent sector errors [6], [41], storage subsystem failures [27], data corruption [7], and disk failure degradations [24]. In particular, these studies characterize the statistical behaviors of disk failures, such as failure arrival rates [42], failure correlations [6], [7], [27], [40]–[42], and degradation signatures [24]. We complement the above field studies by showing the existence of concept drift in disk failures from a stream mining perspective. We validate our findings with the datasets from Backblaze [1] and Alibaba Cloud.

There have been extensive studies on disk failure prediction. Traditional prediction approaches are based on statistical techniques, such as Bayesian classifiers [21], hypothesis tests [25], support vector machines [36], Markov models [45], and rule-based methods [4], [32]. Recent studies improve the prediction accuracy via machine learning algorithms, such as back-propagation neural networks [46], decision trees [29], gradient boost regression trees [30], [31], and random forests [33]. RODMAN [22] studies data preprocessing to improve the prediction accuracy for general machine-learning-based disk failure prediction approaches. Some studies show how disk failure prediction facilitates disk replacements [11] and scrubbing [33], as well as improves cloud service availability [44]. All the above studies are based on offline prediction and assume that all training data is available in advance.

The closest related work to ours is [43], which applies Online Random Forests (ORF) to disk failure classification and automatically updates labels based on incoming SMART attributes. However, StreamDFP considers an inherently different perspective from ORF-based classification [43]: while ORF-based prediction focuses on online learning and tackles the aging issue in the prediction model, StreamDFP focuses on stream mining and adapts the prediction model to concept drift in data streams. StreamDFP addresses the following issues that are not considered in [43]: (i) providing a general framework that supports various stream mining algorithms (instead of ORF only) and customizes them with concept-drift adaptation; (ii) considering both classification and regression (instead of classification only); and (iii) validating its correctness in a significantly larger production dataset in Alibaba Cloud in addition to the Backblaze dataset.

VII. Conclusion

We present StreamDFP, a general stream mining framework for disk failure prediction with concept-drift adaptation. StreamDFP is motivated by the existence of concept drift, backed by our measurement study on five SMART datasets from Backblaze and Alibaba Cloud. It also supports a variety of incremental learning algorithms. Our evaluation of nine decision-tree-based algorithms on the five SMART datasets shows that concept-drift adaptation improves the prediction accuracy significantly. StreamDFP also achieves high stream processing performance that is suitable for practical use.

Acknowledgements

This work was supported in part by Alibaba Group via the Alibaba Innovation Research (AIR) program and the National Science Foundation of China (NSFC) (Grant No.61602120).

References

[1] Backblaze. https://www.backblaze.com/b2/hard-drive-test-data.html.

[2] Backblaze Report in 2015. https://www.backblaze.com/blog/hard-drive-reliability-q3-2015/.

[3] Backblaze Rules. https://www.backblaze.com/blog/what-smart-stats-indicate-hard-drive-failures/.

[4] V. Agarwal, C. Bhattacharyya, T. Niranjan, and S. Susarla. Discovering Rules from Disk Events for Predicting Hard Drive Failures. In *Proc. of IEEE ICMLA*, 2009.

[5] R. Alagappan, A. Ganesan, Y. Patel, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Correlated Crash Vulnerabilities. In *Proc. of USENIX OSDI*, 2016.

[6] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An Analysis of Latent Sector Errors in Disk Drives. In *Proc. of ACM SIGMETRICS*, 2007.

[7] L. N. Bairavasundaram, G. R. Goodson, B. Schroeder, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dussea. An Analysis of Data Corruption in the Storage Stack. In *Proc. of USENIX FAST*, 2008.

[8] A. Bifet and R. Gavalda. Learning from Time-Changing Data with Adaptive Windowing. In *Proc. of SIAM SDM*, 2007.

[9] A. Bifet and R. Gavaldà. Adaptive Learning from Evolving Data Streams. In *Proc. of Springer IDA*, 2009.

[10] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive Online Analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[11] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann. Predicting Disk Replacement towards Reliable Data Centers. In *Proc. of ACM SIGKDD*, 2016.

[12] L. Breiman. Bagging Predictors. *Machine learning*, 24(2):123–140, 1996.

[13] L. Breiman. Random Forests. *Machine learning*, 45(1):5–32, 2001.

[14] A. Cidon, R. Escriva, S. Katti, M. Rosenblum, and E. G. Sirer. Tiered Replication: A Cost-effective Alternative to Full Cluster Geo-replication. In *Proc. of USENIX ATC*, 2015.

[15] R. S. M. de Barros, S. G. T. de Carvalho Santos, and P. M. G. Júnior. A boosting-like online learning ensemble. In *Proc. of IEEE IJCNN*, 2016.

[16] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In *Proc. of ACM SIGKDD*, 2000.

[17] Y. Freund, R. E. Schapire, et al. Experiments with a New Boosting Algorithm. In *Proc. of ACM ICML*, volume 96, pages 148–156, 1996.

[18] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with Drift Detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295, 2004.

[19] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A Survey on Concept Drift Adaptation. *ACM Computing Surveys*, 46(4):44, 2014.

[20] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem. Adaptive Random Forests for Evolving Data Stream Classification. *Machine Learning*, 106(9-10):1469–1495, 2017.

[21] G. Hamerly, C. Elkan, et al. Bayesian Approaches to Failure Prediction for Disk Drives. In *Proc. of ACM ICML*, 2001.

[22] S. Han, J. Wu, E. Xu, C. He, P. P. C. Lee, Y. Qiang, Q. Zheng, T. Huang, Z. Huang, and R. Li. Robust Data Preprocessing for Machine-Learning-Based Disk Failure Prediction in Cloud Production Environments. *arXiv preprint arXiv:1912.09722*, 2019.

[23] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[24] S. Huang, S. Fu, Q. Zhang, and W. Shi. Characterizing Disk Failures with Quantified Disk Degradation Signatures: An Early Experience. In *Proc. of IEEE IISWC*, 2015.

[25] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan. Improved Disk-Drive Failure Warnings. *IEEE Trans. on Reliability*, 51(3):350–357, 2002.

[26] E. Ikonomovska, J. Gama, and S. Džeroski. Learning Model Trees from Evolving Data Streams. *Data Mining and Knowledge Discovery*, 23(1):128–168, 2011.

[27] W. Jiang, C. Hu, Y. Zhou, and A. Kanevsky. Are Disks the Dominant Contributor for Storage Failures?: A Comprehensive Study of Storage Subsystem Failure Characteristics. In *Proc. of USENIX FAST*, 2008.

[28] S. Kadekodi, K. Rashmi, and G. R. Ganger. Cluster storage systems gotta have HeART: improving storage efficiency by exploiting disk-reliability heterogeneity. In *Proc. of USENIX FAST*, 2019.

[29] J. Li, X. Ji, Y. Jia, B. Zhu, G. Wang, Z. Li, and X. Liu. Hard Drive Failure Prediction Using Classification and Regression Trees. In *Proc. of IEEE/IFIP DSN*, 2014.

[30] J. Li, R. J. Stones, G. Wang, Z. Li, X. Liu, and K. Xiao. Being Accurate Is Not Enough: New Metrics for Disk Failure Prediction. In *Proc. of IEEE SRDS*, 2016.

[31] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu. Hard Drive Failure Prediction Using Decision Trees. *Reliability Engineering System Safety*, 164:55–65, 2017.

[32] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu. RAIDShield: Characterizing, Monitoring, and Proactively Protecting against Disk Failures. *ACM Trans. on Storage*, 11(4):17, 2015.

[33] F. Mahdisoltani, I. Stefanovici, and B. Schroeder. Proactive Error Prediction to Improve Storage System Reliability. In *Proc. of USENIX ATC*, 2017.

[34] F. J. Massey. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[35] H. Mouss, D. Mouss, N. Mouss, and L. Sefouhi. Test of Page-Hinckley, an Approach for Fault Detection in an Agro-Alimentary Production System. In *IEEE Asian Control Conference*, 2004.

[36] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *Journal of Machine Learning Research*, 6(May):783–816, 2005.

[37] N. C. Oza. Online Bagging and Boosting. In *Proc. of IEEE SMC*, 2005.

[38] E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41:100–115, 1954.

[39] B. Pfahringer, G. Holmes, and R. Kirkby. New Options for Hoeffding Trees. In *Proc. of Springer AI*, 2007.

[40] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In *Proc. of USENIX FAST*, 2007.

[41] B. Schroeder, S. Damouras, and P. Gill. Understanding Latent Sector Errors and How to Protect against Them. In *Proc. of USENIX FAST*, 2010.

[42] B. Schroeder and G. A. Gibson. Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *Proc. of USENIX FAST*, 2007.

[43] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu. Disk Failure Prediction in Data Centers via Online Learning. In *Proc. of ACM ICPP*, 2018.

[44] Y. Xu, K. Sui, R. Yao, H. Zhang, Q. Lin, Y. Dang, P. Li, K. Jiang, W. Zhang, J.-G. Lou, et al. Improving Service Availability of Cloud Systems by Predicting Disk Error. In *Proc. of USENIX ATC*, 2018.

[45] Y. Zhao, X. Liu, S. Gan, and W. Zheng. Predicting Disk Failures with HMM-and HSMM-based Approaches. In *Proc. of IEEE ICDM*, 2010.

[46] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma. Proactive Drive Failure Prediction for Large Scale Storage Systems. In *Proc. of IEEE MSST*, 2013.