CCF Transactions on High Performance Computing (2022) 4:233–264
https://doi.org/10.1007/s42514-022-00101-3

REGULAR PAPER

A survey on AI for storage

Yu Liu[1] · Hua Wang[1] · Ke Zhou[1] · ChunHua Li[1] · Rengeng Wu[1]

Received: 16 December 2021 / Accepted: 8 April 2022 / Published online: 23 May 2022

Abstract
Storage, as a core function and fundamental component of computers, provides services for saving and reading digital data. The increasing complexity of data operations and storage architectures is challenging the performance and reliability of storage services. Artificial intelligence (AI) advancements in the field of intelligent algorithms show significant promise for resolving storage issues. A hot topic of current studies is how to marry AI and storage. In this paper, we present a comprehensive survey of "AI for Storage" and categorize storage research employing intelligent algorithms according to public literature in recent years into Architecture-oriented, Data-specific, and Operation & Maintenance. Based on this classification, we fine-categorize all of the studies by application environment and elaborate on their development history in order to provide guidelines for future research on how to employ AI technologies based on this history. Finally, we present a discussion and future work.

Keywords Survey · Storage · AI · Machine learning

1 Introduction

The scenarios and transactions that storage systems must deal with are becoming more complicated with the development of cloud services and Artificial Intelligence (AI) applications. Although the storage devices are being upgraded and iterated, there is still a need to extend new architectures and explore new approaches to respond to data storage and management difficulties. Due to the ability of feature-awareness and empirical data learning, AI technologies, especially intelligent algorithms, have been a classic topic in data awareness and cognition. Meanwhile, the advancements in machine learning (ML), especially in deep learning (DL), have elevated it to become a popular attempt for tricky problems. "Marrying AI and Storage" has been proposed as the first issue in the proceedings of NSF Visioning Workshop at FAST'19. "AI for Storage" is gradually forming its own concept and system. We define it as using AI technologies in storage devices or systems, learning from application workload, running status, and user data, achieving efficient detection, prediction, and decision, so as to realize intelligent optimization.

In fact, there has been a lot of research trying to apply AI technologies, specifically intelligent algorithms, for storage, where the intelligent algorithms include heuristic-based simulation algorithms (e.g., reinforcement algorithms), planning algorithms (e.g., quadratic programming), knowledge representation algorithms (e.g., knowledge graphs), learning-based algorithms (e.g., ML), probabilistic reasoning algorithms (e.g., Bayesian), and their fusion algorithms. Based on this, we retrieval the papers since 2009 through keywords about AI (e.g. learning, smart, swift, intelligent, sage, etc.) combined with conventional storage issues. Note that we chose 2009 as a watershed year since the model using convolutional neural networks (CNN) won the championship of image classification competitions in the year, which sparks the current AI development. The results returned from

✉ Hua Wang
hwang@hust.edu.cn

Yu Liu
liu_yu@hust.edu.cn

Ke Zhou
zhke@hust.edu.cn

ChunHua Li
li.chunhua@hust.edu.cn

Rengeng Wu
rengengwu@hust.edu.cn

[1] Huazhong University of Science and Technology, Luoyu Road, Wuhan 430074, Hubei, China

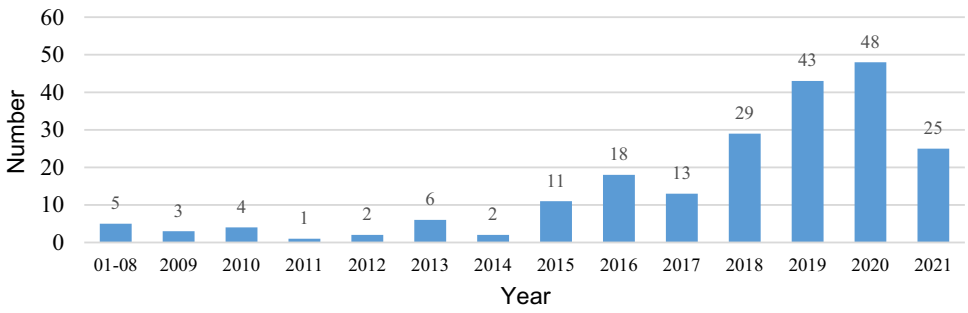

Fig. 1 The number of research on storage using intelligent algorithms from 2001 to 2021. The amount of such studies has increased by year. The drop back in data in 2021 may be due to some studies not being public before we counted them

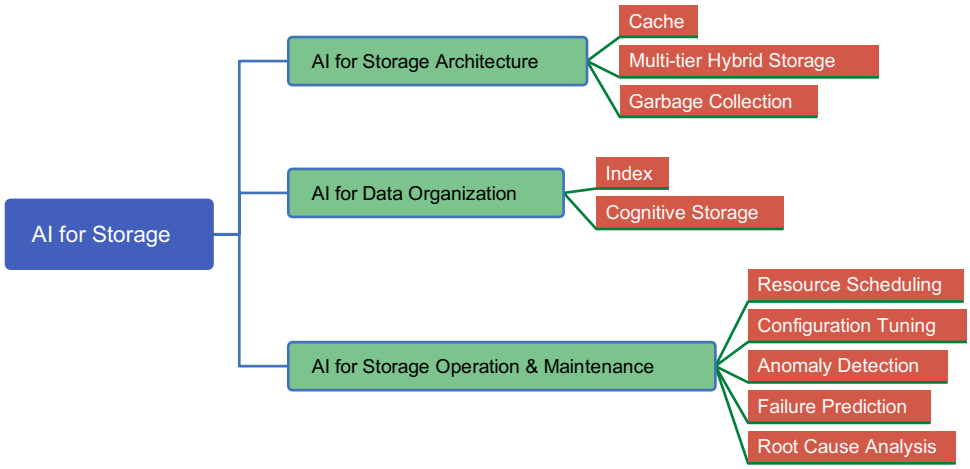

Fig. 2 The category of research on "AI for storage"

GOOGLE[1] and DBLP[2] are numerous and varied. Since we focus on "AI for Storage", we discard many papers referring "Storage for AI" (e.g., system design for AI accelerator), "AI for Database", and "Retrieval". Finally, the numbers of relevant papers in different years are shown in Fig. 1.

As shown in Fig. 1, there have been 210 papers we gathered for storage research employing intelligent algorithms since 2001. The phenomenon clearly illustrates the proliferation of relevant studies from 2015 to 2020, when the incorporation of AI technologies and storage has received a lot of attention. Note that storage research applying ML accounted for 85.8% ($\frac{139}{162}$) in this period. Meanwhile, we also note a pullback in numbers at 2017 and 2021. In addition to some unpublished related work for 2021, this reveals that researchers remain discreet in applying AI technologies to storage. We think that albeit there is a compelling success of ML in computer vision (CV) tasks, the usage of AI algorithms in the storage seems a tough job caused by limited resources, e.g., limited annotation data in general storage devices, limited arithmetic power in edge storage devices, limited data access, etc. However, AI algorithms that are tailored to the storage environment have been evolving (e.g., few-shot and zero-shot learning, federal learning, etc.), resulting in more relevant studies. Nevertheless, these studies are still amorphous, and we hope to have a real breakthrough in storage, as the CNN model did for CV.

To review how AI algorithms can empower storage, we categorizes the related studies in Fig. 2. These studies have focused on the three issues consisting of system, data organization, and operation and maintenance. In this survey, we summarize the AI algorithms utilized in these studies, as well as the motives for using them, in the hopes of generating new ideas for future research. In the following, we will describe each research by the logical sequence of Fig. 2.

2 AI for storage architecture

Architecture-oriented research demonstrates how AI technologies play a role in, the key components (e.g., cache) that determine storage performance, the multi-tier hybrid storage that affects system performance and utilization, and the garbage collection that automatically manages the memory, from the perspective of the infrastructure that provides storage services.

[1] https://scholar.google.com/.

[2] https://dblp.org/.

2.1 Cache

Cache as a vital component of storage is usually used to reduce latency, bandwidth usage and alleviate congestion for reading by duplicating and storing contents in devices. The caching task can be thought of as a decision-making process. For example, the cache is a system with two levels of memory: a slow memory of size m, and a fast memory of size k. A caching algorithm is confronted with a series of element requests. A cache hit happens when the requested element resides in the fast memory, and the algorithm satisfies the request at negligible cost. A cache miss happens when the requested item is not in the fast memory; the algorithm then pulls the item from the slow memory and stores it in the fast memory before satisfying the request. If the fast memory is full, one of the objects will have to be evicted. The core of the caching online algorithmic challenge is the eviction approach. The goal of eviction policy is to cause the fewest cache misses. Consequently, the goal of intelligent caching algorithms is to predict future requests, then decides which objects to be cached and evicted according to maximize the cache hit.

In practice, the three types of caching algorithms are replacement policy, admission policy, and prefetching policy. In addition, they can be classified into the web cache, on-chip cache, and generic cache based on the cache's environment. According to these classifications, we list most of intelligent caching algorithms in Table 1. In the following, we detail these algorithms by environments.

On the web The main focus of research for web cache is content distribution networks (CDN). A CDN is a large distributed system of servers that caches and delivers content to users. The first-level cache in a CDN server is the memory-resident Hot Object Cache (HOC). A major goal of a CDN is to maximize the object hit ratio (OHR) of its HOCs. However, the small size of the HOC, the huge variance in the requested object sizes, and the diversity of request patterns make this goal challenging.

Berger et al. (2017) observed that the size of requests and objects in CDN caches changes over time and proposed AdaptSize, a size-adaptive CDN cache admission policy based on a Markov model that uses object size and request rate information for admission. In practice, AdaptSize enhances the object hit rate. After that, they Berger (2018) also observed the efficiency problem of feedback using reinforcement learning (RL). They suggested solving this problem by modeling the optimal caching decisions (OPT) and proposed the Learning From OPT (LFO) algorithm for cache policy. Experiments show that using the LightGBM algorithm for inference, LFO can improve the prediction accuracy. Wang et al. (2018, 2020) observed that 61% of images under social network workloads were accessed only once, and that denying these images to the cache could improve solid-state drives' (SSD) lifetime. Therefore, they proposed a "one-time-access criteria" strategy based on decision trees (DTs), which can improve the prediction accuracy by extracting socially relevant information. Guan et al. (2019) proposed a Content-feature Aware Cache Admission (CACA) policy to admit video objects to cache by video features, not by request patterns. CACA employs a tree-structure RL algorithm to extract features of preferred content. Kirilin et al. (2019, 2020) proposed an admission policy, i.e., RL-Cache, for CDN caching by RL using features such as object size, repetitiveness, and access frequency. Zhang et al. (2020) proposed a caching algorithm ML-WP to cope with the large amount of write-only traffic in cloud block storage systems. ML-WP uses an random forest (RF) algorithm to split data into write-only and normal data, writing write-only data directly to back-end storage without entering the cache. ML-WP reduces SSD cache writes and improves the hit rate compared to the industry's widely deployed write-back policy. Vietri et al. (2018) proposed a RL approach LeCaR by loss minimization to dynamically select LRU and LFU algorithms. When the cache size is 1/1000 of the workload, LeCaR achieves an 18x performance advantage over ARC in a production scenario. Narayanan et al. (2018) proposed the DEEPCACHE content caching framework for replacement. The codec model implemented by long short-term memory (LSTM) may anticipate future properties of objects, such as popularity, and enhance the cache hit ratio. DEEPCACHE can greatly enhance the hit rate of classical LRU, k-LRU, and other algorithms under the experimentally developed streaming service workload. Wang et al. (2019) proposed a sub-sampling technique to reduce cache space and decrease the time of reward feedback by accessing the hash of objects. The method reduces the cache space by a factor of 100 and makes the cache replacement problem easier to learn using RL models. Song et al. (2020) proposed the Learning Relaxed Belady algorithm (LRB) for replacement, which simplifies the Belady's goal from the object with the longest reuse distance to a Belady boundary containing multiple objects whose reuse distances are beyond the boundary. LRB addresses the necessary system challenges in building an end-to-end ML caching prototype, including how to gather training data, limit memory overhead, and explore lightweight training and inference paths. Yan and Li (2020) proposed an algorithm RL-Belady, which uses an ensemble learning model, i.e., XGBM, to learn the features of videos and provide the prediction for the admission policy and eviction policy. Ye et al. (2021) proposed a distribution-guided RL framework, i.e., JEANA, to learn the joint cache size scaling and strategy adaptation policy, achieving the balance of performance and utility for small content providers. We believe it is an active cache or the "cache as the service" scheme for cloud-edge-end environment. Rodriguez et al. (2021) proposed a new class of lightweight, adaptive RL

Table 1 Intelligent caching algorithms using intelligent algorithms

Author	Algorithm	Environment	Type	Used Intelligent Algorithm	Improvement
Berger et al. (2017)	AdaptSize	Web	A	MDP	Object hit ratio
Berger (2018)	LFO	Web	AR	LightGBM	Prediction accuracy
Wang et al. (2018) Wang et al. (2020)	OTAE	Web	A	DT	Prediction accuracy
Guan et al. (2019)	CACA	Web	A	Tree-structure RL	Hit ratio, back-to-origin, memory
Kirilin et al. (2019) Kirilin et al. (2020)	RL-Cache	Web	A	RL	Hit rate
Zhang et al. (2020)	ML-WP	Web	A	ML	Write traffic, hit ratio, read latency
Vietri et al. (2018)	LeCaR	Web	R	RL, MAB	Select algorithms
Narayanan et al. (2018)	DEEPCACHE	Web	R	LSTM	Hit rate
Wang et al. (2019)	–	Web	R	RL	Cache size
Song et al. (2020)	LRB	Web	R	GBM	WAN traffic
Yan and Li (2020)	RL-Belady	Web	AR	XGBM	Hit rate
Ye et al. (2021)	JEANA	Web	R	RL	Hit ratio, rental cost
Rodriguez et al. (2021)	CACHEUS	Web	R	RL	Workload-aware
Tsai et al. (2018)	–	Web	L	RF	Dynamic allocation
Liu et al. (2020)	MLCache	Web	L	DL	Predict allocation
Zhang et al. (2020)	OSCA	Web	L	Quadratic programming	–
Teran et al. (2016)	–	Chips	R	Perceptron	Accuracy
Jiménez and Teran (2017)	Multi-perspective reuse prediction	Chips	R	Perceptron	Accuracy
Shi et al. (2019)	Glider cache replacement policy	Chips	R	LSTM, SVM	Miss rate
Sethumurugan et al. (2021)	RLR	Chips	R	RL	Performance, overhead
Peled et al. (2015)	Context-based prefetcher	Chips	P	Contextual bandits	Speedup
Peled et al. (2020)	Context-based NN prefetcher	Chips	P	NN	Speedup
Rahman et al. (2015)	–	Chips	P	LR DT	Speedup
Bhatia et al. (2019)	PPF	Chips	P	Perceptron	Coverage, accuracy
Jain et al. (2017)	MLC	Chips	L	RL	Dynamical adaption
Kim et al. (2019)	–	Chips	L	tree-Boosting	Performance
Hashemi et al. (2018)	RNN prefetcher	Chips	P	LSTM, Clustering	Precision, recall
Zeng and Guo (2017)	LSTM prefetcher	Chips	P	LSTM	Accuracy, coverage
Shi et al. (2021)	Voyager	Chips	P	LSTM	Coverage, accuracy
Braun and Litz (2019)	–	Chips	P	LSTM	Accuracy
Srivastava et al. (2019)	ML-based prefetchers	Chips	P	LSTM	Recall, accuracy
Xu et al. (2018)	–	Devices	P	MDP	Memory cost, hit ratio, mis-prefetching rate
Lykouris and Vassilvitskii (2018)	Predictive Marker	Devices	R	ML	Competitive ratios
Paschos et al. (2019)	OGA, BSA	Devices	R	ML	Hit rate
Chakrabortti and Litz (2020)	–	Devices	P	LSTM	Speedup
Chledowski et al. (2021)	–	Devices	R	Learning-Augmented	Overhead, insurance
Cohn and Singh (1996)	–	Devices	L	DT	Dynamic allocation
Maas et al. (2020)	L1AMA	Devices	L	LSTM	Prediction error, latency

In the column of Type, P, R, L and A represent the prefetching policy, replacement policy, allocation policy and admission policy respectively. LSTM represents the long short-term memory unit. MDP denotes the Markov decision process. DT is the decision tree model. MAB represents Multi-Armed Bandit. NN denotes the neural network

cache replacement algorithms called CACHEUS, which uses a combination of replacement algorithms to deal with different workloads. CACHEUS trains the model for switching between SR-LRU and CR-LFU based on the reward of MR changes. It is an updated version of LeCaR.

In addition, there is a portion of the allocation policy applied to cache devices. Tsai et al. (2018) proposed the concept of effective memory demand and constructed a ML model for predicting effective memory demand using RF to solve the memory allocation problem of big data analytic programs in java virtual machine (JVM) in Spark environment, so that they can be informed of the memory usage of applications in distributed cluster environment and allocate appropriate memory. Liu et al. (2020) proposed MLCache for efficient multi-queue cache allocation and management of NVMe SSDs, which uses a deep learning model to predict the optimal future cache allocation using the reuse distance distribution and relative frequency of multiple streams. Zhang et al. (2020) proposed an Online-Model based Scheme for Cache Allocation(OSCA) for shared cache servers among cloud block storage devices. OSAC uses the re-access traffic ratio to obtain data reuse distances and gets the miss ratio curves(MRC) with low overhead. Then, it searches for a near-optimal configuration using a dynamic programming method according to the MRCs. It can reduce IO traffic to the back-end storage server efficiently.

Cohn and Singh (1996) use DTs to predict the life cycle of the memory blocks to be allocated into two categories: short-term, long-term, and then use appropriate memory allocation strategies based on the life cycle of the memory blocks to implement dynamic memory allocation optimally. Maas et al. (2020) proposed a new memory manager, LlAMA, to manage large pages and life cycle classes in memory based on the construction of LSTM models to predict the life cycle of objects, and to track the actual life cycle of objects for correcting the prediction error of the model. Instead of memory allocation based on the size of the object, the method used in this model is based on the life cycle of the object, and deals with both the prediction error problem of ML methods and the latency problem when incorporated into the underlying system.

On the Chips The performance difference between CPU and memory has inspired the research of cache management on the chips. There are two main optimization directions: replacement and prefetching policy.

For the replacement policy, Teran et al. (2016) proposed a perceptron model to predict LLC reuse, which considerably improves accuracy and provides an option for additional optimization. Then, they've proposed multiperspective reuse prediction Jiménez and Teran (2017), which employs a variety of factors to predict future cache block reuse, hence enhancing cache efficiency. Shi et al. (2019) trained an attention-based LSTM model offline, and then studied and interpreted the LSTM model to gain a significant insight, which they used to create an online support vector machine (SVM) model with comparable accuracy and lower costs. Although PC-based replacement policies (such as SHiP, SHiP++, Hawkeye, etc.) are generally better than non-PC-based replacement policies (such as LRU and DRRIP). Unfortunately, a PC-based replacement policy necessitates not just greater storage, but also considerable changes to the processor's control and data paths. Chip-makers have thus far been hesitant to adopt a PC-based replacement scheme. Based on this challenge, Sethumurugan et al. (2021) use RL to learn the replacement policy of last-level cache. They use domain knowledge analysis to drive a new non-PC-based cache replacement policy called Reinforcement Learned Replacement (RLR) after successfully learning the replacement method with good performance. The overall performance of RLR is superior than that of non-PC-based replacement policies, and it achieves equivalent performance to existing PC-based replacement strategies, according to experimental results.

For the prefetching policy, researchers focus on using ML algorithms to overcome the multiple prefetcher problem posed by multiple data structures and access patterns. Peled et al. (2015) proposed the concept of semantic locality and implemented the context-based prefetcher using the contextual bandits model. This RL model approximates semantic locality to identify access patterns and provide hints for memory access. After that, Peled et al. (2020) regarded the data prefetch as a regression problem and proposed a neural network (NN) prefetch based on context. Prefetchers use a variety of workload cues to learn these patterns, including semantic program information, traditional architecture information (e.g., PC, miss history), and so on. The prefetcher employs a small NN implemented using a systolic array. The NN performs buffered inference over program-state context vectors, and backpropagation over their associated prefetch candidates, thereby finding correlation and using it for prefetching. Rahman et al. (2015) proposed a ML technique based on program characteristics to dynamically adjust prefetch configurations for optimal performance. On average, it delivers prefetching speedup on the PARSEC benchmark suite and two additional programs. Bhatia et al. (2019) proposed a perceptron based prefetch filtering (PPF) method that can filter out inaccurate prefetches requested by the underlying prefetchers, thereby improving the coverage of the underlying prefetchers and overall performance without affecting accuracy. Experimental results show that PPF improves the performance of the underlying prefetcher. Note that PPF is a robust and adaptable technology that can be used to enhance any existing prefetchers. Jain et al. (2017) proposed a coordinated multi-agent RL approach to multi-level cache co-partitioning (MLC). They train multiple agents to learn the sensitivity of different applications to

different cache levels and then use the Hill-Climbing algorithm to find the optimal cache allocation method within the limited cache space, resulting in the dynamical adaption for the changing application scenarios using the appropriate multi-level cache allocation strategy. The experimental results on Sniper show that the MLC algorithm improves the throughput and energy-delay-product of the system compared with the latest research results. Kim et al. (2019) proposed a novel approach to automatically build a prediction model, i.e., TMAM, which accurately predicts program performance in the case of CAT changing cache resource allocation. TMAM characterizes workload characteristics with the model trained by the tree-Boosting machine learning technique. Furthermore, the prediction model creates a dynamic cache management strategy that can intelligently assign cache resources and boost application throughput.

Hashemi et al. (2018) transformed the data prefetch problem into a classification problem, which used LSTM to solve this problem and predicted memory access patterns. The experimental results show that it has excellent performance in precision and recall. This work represents the first step in the use of NNs for hardware prefetching. Zeng and Guo (2017) proposed an LSTM prefetcher, i.e., a NN based hardware prefetcher, which can accurately predict complex memory access patterns and achieve higher accuracy and coverage. Compared with previous work, LSTM prefetchers have better memory ability for long access sequences and higher noise resistance. However, it also has some disadvantages such as long warm-up time, prediction latency and relatively larger storage overheads. Shi et al. (2021) proposed a novel NN Voyager for data prefetching, which can learn not only delta correlations but also address correlations, which is very important for prefetching irregular memory access patterns. The key to Voyager is using two separate attention-based LSTM layers, i.e., a page LSTM layer and an offset LSTM layer, to predict the page and offset, respectively. Voyager shows a huge cost reduction in all dimensions when compared to earlier NN models, and it's a big step toward practical neural prefetchers. To better understand what types of memory access patterns with LSTM NNs can learn, Braun and Litz (2019) investigated the effect of hyperparameters on performance of LSTM-based prefetchers under workloads of different memory access patterns by training individual models on MicroBenchmarks with well-characterized memory access patterns. It was found that there was a strong relationship between the size of the look-back size(access history window) and the ability of the model to learn the pattern, and that increasing the model size was important as the number of distinct streams to learn is increased. Although LSTM can achieve high precision and recall in learning memory access, it is not practical to solve hardware prefetch problems due to high storage resource consumption, high latency, offline training and online testing. Srivastava et al. (2019) proposed an LSTM-based approach to accurately predict the next access and a compression technique that reduce the number of parameters by a theoretical factor of $\frac{n}{log_2(n)}$ times. Experiments show that the compressed LSTM can achieve similar accuracy to the original LSTM, and can be trained quickly in a short time.

On the Devices The cache algorithm on the devices is concerned with the relationship between hit rate and device performance, and the theory of caching algorithms.

Xu et al. (2018) observes that designing complex prefetchers in the flash-translation layer (FTL) of SSDs introduces additional overhead. They proposed an efficient and resource-optimized Markov chain-based learning algorithm in FTL, capable of achieving high hits in complex access patterns. The algorithm is able to save access time on TPC-H. Lykouris and Vassilvitskii (2018) discussed the caching problem under ML prediction. He recommends treating the caching algorithm as a black box and using the Marker algorithm to build the Predictive Marker algorithm. The method can approximate the optimum algorithm as the predictor's accuracy improves and preserve the Marker algorithm's cache miss rate without relying on the predictor's prediction accuracy. Paschos et al. (2019) offered a fresh take on applying ML to cache design and performance analysis, as well as the first-ever used of online linear optimization (OLO) to establish optimal cache performance boundaries. They proposed the online gradient ascent algorithm (OGA) for single-machine caching, which was shown to be universally optimal, and the bipartite gradient algorithm (BSA) for online caching and routing. Chakrabortti and Litz (2020) proposed a NN-based prefetching method for SSD. This solution employs the LSTM seq2seq model to learn the pattern of complicated I/O accesses in a data center and ensures prefetching timeliness by predicting many accesses at once. Chledowski et al. (2021) proposed Learning-Augmented learning algorithms using both ML predictors and classical cache replacement algorithms. The method employs the conventional caching mechanism to ensure the stability of ML predictors, switching between them when one performs worse than the other.

2.2 Multi-tier hybrid storage

The current storage system is facing the bottleneck of performance due to the gap between fast CPU computing speed and the slow response time of hard disk. Recently a multi-tier hybrid storage system (MTHS) which uses fast flash devices like SSDs as the one of the high performance tiers has been proposed to boost the storage system performance. In order to maintain the overall performance of the MTHS, optimal storage assignment has to be designed so that the

Table 2 Intelligent data placement algorithms using intelligent algorithms

Author	Algorithm	Environment	Research content	Used Intelligent Algorithm	Improvement
Yuan et al. (2010)	–	Scientific Workflow	Data Placement	K-Means	Near data processing
Sun et al. (2015)	–	Scientific Workflow	Data Placement	N-dimensional bounding boxes; Greedy heuristic algorithm	Workload-ware prediction
Subedi et al. (2018)	Stacker	Scientific Workflow	Data Placement	n-gram	Pre-fetch
Tomes et al. (2018)	–	Cloud Storage	Data Placement	IFM; Graph coloring	Parallel access
Cheng et al. (2019)	ASL	Scientific Workflow	Data placement	CART	Overall performance
Shi et al. (2020)	WorkflowRL	Scientific Workflow	Data Placement	RL	Overall performance
Ren et al. (2021)	–	Clound storage	Data placement	MLP	Performance and overhead trade-off
Cao et al. (2019)	AdaM	Cloud storage	Metadata Placement	DDPG	Load balance and locality
Monjalet and Leibovici (2019)	–	File System	File Life Prediction	CNN/RF	Accuracy and underestimation
Thomas et al. (2021)	–	File System	File Life Prediction	CNN	higher accuracy and lower underestimation
Shi et al. (2012)	–	MTHS	Data Allocation	MCKP	Less data miss

data migrated to the high performance tier like SSD is the appropriate set of data.

Emerging applications produce massive files that show different properties in file size, lifetime, and read/write frequency. Existing MTHSs place these files onto different storage mediums assuming that the access patterns of files are fixed. However, we find that the access patterns of files are changeable during their lifetime. The key to improve the file access performance is to adaptively place the files on the MTHS using the run-time status and the properties of both files and the storage systems. Intelligent strategies that can select the optimal storage tier dynamically are the key to exploit the potentialities of hierarchical storage architecture (Table 2).

Data placement strategies on hierarchical storage can be divided into horizontal placement (how data are distributed inside a storage layer) and vertical placement (how data are distributed across different storage layers). Intelligent data placement strategies use ML techniques to mine the relationship between data placement strategies and I/O performance under varied workflow characteristics and system status, and use the learned models to choose the optimal storage tier. In practice, to solve the access patterns problem, an MTHS needs a self-learning method that can predict the data access pattern.

Yuan et al. (2010) proposed a matrix-based *K*-Means clustering strategy for data placement in scientific cloud workflows. During the workflow build-time phase, two algorithms group existing datasets in k data centers, and during the run-time phase, the newly created datasets are dynamically clustered to the most relevant data centers based on dependencies. When there are no fixed-location datasets in the dataset, simulations in the cloud workflow system SwinDeW-C demonstrate a 50.8% reduction in data movement compared to random data placement in a cloud environment with limited storage space. As 10% of fixed location datasets are available in the system, data mobility is reduced by 47.4% when compared to random placement. However, workflows' complex and dynamic data exchange patterns, combined with a wide range of data access behaviors, make it difficult to efficiently put data within the staging area. Sun et al. (2015) described the data access pattern in an *N*-dimensional data domain as one or more *N*-dimensional bounding boxes, and then predict the next data access pattern. Based on this model, Subedi et al. (2018) investigated how data staging solutions can effectively use burst buffers. They designed Stacker that is a prefetching technique with ML to move data between storage tiers in a self-adaptive manner. Stacker uses an n-gram model to learn and predict the next data access request, implementing data prefetching in the dataspace of hierarchy service. Tomes et al. (2018) proposed a general framework for adaptive parallel storage systems. The framework monitors requests to blocks through a disk I/O monitoring tool, uses the Frequent Itemset Mining (FIM) algorithm to find associations between blocks, and transforms the data layout optimization problem into a graph coloring problem. Only a maximum capacity can be assigned to each color (disk) according to the disk capacity, and each vertex has a weight according to the block size. Thus, the problem is formulated as an MCBP problem, which can be solved using heuristic methods. Experimental results show that the framework is very successful in adjusting to the skewed parallel access patterns for both HDD-based traditional storage arrays and SSD-based

all-flash arrays. Cheng et al. (2019) proposed Adaptive Storage Learning (ASL), which uses a gradient enhancement algorithm and the classification and regression trees (CART) model to investigate the relationship between data placement policy and I/O performance to identify the best storage layer for workflows with varied features under different system status. Shi et al. (2020) proposed WorkflowRL, an intelligent data placement engine based on RL, managing data across multi-tier hierarchical storage systems. WorkflowRL extracts factors that affect I/O performance, including workflow features and system information used for feature learning, and can significantly improve I/O performance for different scientific workflows. Ren et al. (2021) proposed a data insertion technique aided by ML. By forecasting file access patterns, the technique adaptively stores files on appropriate storage media. A PMFS-based tracker is utilized in the mechanism to gather file access features for prediction, and two feature selection methods are employed to choose the key characteristics from the acquired features: RFE (Recursive feature Elimination) and MI (Mutual Information). The file access features are predicted using a sliding window of length N+1, and the access pattern of the (N+1)th file is defined by the access pattern of the previous N files.

Note that the directory is another management style of the hierarchical storage. In a distributed file system, how to reasonably distribute different directories/files created by different users to the appropriate nodes will make a difference to the system. Cao et al. (2019) proposed AdaM, an adaptive fine-grained metadata management scheme that collects the environment "state" at each time step, including access patterns, the structure of the namespace tree, and the current distribution of nodes on the multiple metadata servers (MDSs), to be able to trade-off the load balancing of the system and the localization of metadata faced in a distributed metadata management scheme. Based on the observed "state", an actor-critic network is trained to reassign hot metadata nodes to different servers. AdaM ensures load balancing while keeping metadata localized by automatically transferring hot metadata nodes between servers. Monjalet and Leibovici (2019) proposed to use RF and CNNs to predict the future lifetime of a file by its storage path, which can be used to build the storage hierarchy based on the prediction results. They use Accuracy and Underestimation to evaluate the performance of the prediction model. A file whose lifetime is underestimated will be migrated to lower tier before its final access, resulting in a performance drop. Nevertheless, this approach does not solve the scalability problem of the model under path change. By Learning previous work, Thomas et al. (2021) proposed a file lifetime prediction model entirely based on CNNs with the only input of the file path. This work is flexible because it allows the system administrator to adjust the weights of the loss function to balance the accuracy and underestimation.

The data allocation problem (DAP) is to find the optimal lists of data files for each storage tier in the multi-tier hybrid storage system to achieve maximal benefit values without exceeding the available size of each tier. In order to achieve the overall performance, it is necessary to keep the most frequently accessed data (hot data) to the high performance storage tier, while storing the least accessed data at low tiers. Shi et al. (2012) described DAP as a special multiple choice knapsack problem (MCKP) and proposed a multi-stage dynamic programming algorithm to find the optimal solution. Simulation results with different input data objects and different numbers of layers show that the algorithm improves 6 times over the greedy algorithms.

2.3 Garbage collection

Garbage collection (GC) is a type of automatic memory management in computer science. The garbage collector tries to reclaim the memory that the program has allocated but is no longer referenced. GC frees the programmer from having to deallocate and return objects to the memory system manually. GC can account for a considerable amount of a program's total processing time, resulting in a significant impact on performance. AI is introduced to make decisions on the objects and processes of collection to improve storage performance in response to limited storage resources and high real-time requirements at the time of collection. We divide the GC studies into device-level and system-level.

Device-level GC The demand for dense and dependable memory has become an urgent necessity as the amount of information created, accessed, and duplicated each year increases dramatically. SSDs require Garbage Collection operations due to the "erase and rewrite" nature of NAND, i.e., because NAND cannot be rewritten, only the old data can be erased first and then new data may be written to the same area. GC operations, on the other hand, can degrade storage performance and potentially cause long-tail latency, blocking subsequent I/O commands, due to restricted storage resources (e.g., storage internal I/O bandwidth) and write amplification. There are three kinds of approaches for the GC problem on SSDs using intelligent algorithms.

The first type of approach uses a greedy strategy to select the block with the fewest valid pages for GC. For small users with evenly distributed random writes, Bux and Iliadis (2010) proposed two distinct theoretical models of SSD operations to overcome the write amplification problem in garbage collection. The greedy model recovers the maximum free space by selecting the block with the smallest effective page count and is used to handle GC in large systems under no memory workload, and the Markov model helps to explore the performance characteristics of small and medium-sized systems. The simulation results support the analytical results and further reveal the behavior

and effectiveness of the greedy garbage collection strategy. Yang et al. (2015) demonstrated through theoretical analysis that the greedy GC algorithm is optimal for workloads with exponentially distributed page lifetimes.

The second type of method supports partial garbage collection in between running I/O read and write commands on the SSDs. Park et al. (2014) proposed an adaptive garbage collection technique that employs a Hidden Markov Model to predict the subsequent long idle time and triggers garbage collection in the background during the expected long idle time. This technique can improve performance by adding only a small number of erasures. In order to solve the possible bandwidth reduction problem caused by GC in SSDs, Park and Kim (2017) attempted to manage GC overhead at the operating system level. They used an ML model to devise a GC detecting mechanism at the operating system level and request TRIM operation from SSDs when GC is detected to reduce performance variance normally. Kim (2021) proposed a FTL-aware GCD host system. The system introduces a multi-layer perceptron with inputs to extract FTL metadata from storage to predict the expectation of garbage collection occurring after a host executes a write command. With the model prediction, the CD host system can allocate a higher storage cost for GC in advance. Kang et al. (2017) worked on the application of RL algorithms to GC. They first proposed RLGC Kang et al. (2017) that uses the Q-learning model to assist GC. RLGC learns the storage access behavior online and determines the number of GC operations in idle time. The limitation of RLGC is the cost, i.e., the more states it has, the more storage space is occupied by the Q-table. To solve this problem, they proposed a key state dynamic management technique Kang and Yoo (2018) to improve RLGC. The new method uses more fine-grained state information than RLGC, which makes GC prediction more accurate. Moreover, the method replaces the traditional Q-table with a small costly Q-table cache to store the Q-values of key states, and the cache is updated by LRU. Based on this work, they introduced the long-term history of the system and low-cost QTC to continue improving RLGC. The new method Kang and Yoo (2020) is equipped with a QP mini-network to take advantage of the long-term history and to provide good initialization of q values for QTC. The method is able to alleviate the long-tail delay problem of GC.

The last method reduces the GC time by predicting the characteristics of the data such as data access frequency, and then placing the data with minimum GC cost. Yang et al. (2019) proposed a scheme to place data according to its hot extent. The scheme utilizes an LSTM model to improve the prediction accuracy and also uses the K-means algorithm to assign blocks to different clusters based on the predicted hotness. The reduction in WAF in different types of workload scenarios illustrates the improvement in GC, while I/O performance is also improved. Luo et al. (2020) proposed the use of hybrid optimized echo state network (HOESN), a self-learning model, for hot data prediction to assist SSDs in garbage collection and wear balancing of NAND Flash Memories. The method utilizes hybrid optimized echo state network and quantum particle swarm algorithm in reservoir computing for hot data prediction. The results show that HOESN-based hot data prediction can reliably improve the access performance and persistence performance of NAND flash.

System-level GC GC affects the life of the system as it does for disks. The file system, as the user of GC, is most prominently affected in the cleaning of log data. Wu et al. (2019) proposed a background segment clean strategy (RLBC) based on the RL model, which adaptively decides when to trigger background segment clean by modeling the behavior of I/O workloads and the state of the logical address space. Yang et al. (2020) observed that flash friendly file system (F2FS), a file system designed for flash, suffers from severe fragmentation due to the highly synchronized, multi-threaded write behavior of its applications, resulting in a poor user experience. They proposed an Adaptive Reserved Space (ARS) scheme to select some specific files to update in the reserved space. ARS collects file characteristics associated with fragments to build the dataset and uses DTs to accurately select reserved files. Additionally, the dynamic reservation strategy and adjustable reserved space are employed. They proposed multi-log delayed writing and modied segment cleaning based on dynamically identified hotness (M2H Yang et al. (2021)) to optimize severe cleaning overhead due to its logging scheme writes. To determine the hotness, M2H uses file block update distance (FUD), which is based on the concept of IRR (Inter-Reference Recency). Due to too many file blocks and fluctuating workloads, the hotness is determined using sampling and K-means clustering.

3 AI for data organization

Data-specific research focuses on the items (data & metadata) provided by storage and demonstrates how AI technologies perceive and manage these items, as well as how to ensure effective retrieval via intelligent indexing. These jobs explore general transactions and add new components or methods to the original architecture to implement new functionality. AI technologies are introduced in the form of components (models) or approaches that supplement and improve traditional systems' functionality and efficiency.

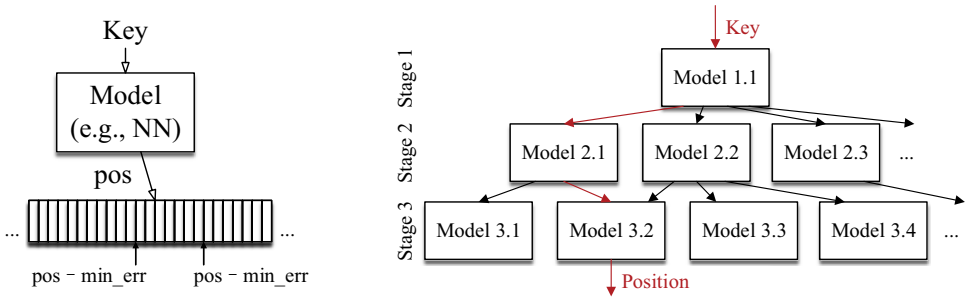

Fig. 3 Learned index and staged models

3.1 Index

More recently, there is an increasing requirement to read data "immediately" for an ever-increasing range of analytic queries. However, simply plugging in a fast parser into existing solutions does not help with read because of large parsed fields or heavy range indices, resulting in heavy write amplification, random I/Os, and CPU overheads Xie et al. (2019). Index-based optimization is the fundamental solution to this problem Chandramouli et al. (2018). AI algorithms have mostly focused on index construction and search methods based on index structures. The goal of these studies is to improve search speed in general or specific businesses. When the research employs AI algorithms, we classify the research that involves the generation of metadata and construction of index structures as *learned index structure*, and the work that investigates searching on index structures as *index search*.

3.1.1 Learned index

In this section, we collect research in 3 areas including learning-based consistent hashing, clustering-based index structures and ML-based range index (RMI) structures.

Learning-based Consistent Hashing Although learning-based hashing is widely used in multimedia, we focus on the learning-based consistent hashing for storage. DeepHash Gao et al. (2019) leverages the deep NN to learn a locality preserving hashing (LPH) Indyk et al. (1997) mapping which makes a ring-projection and angle-assignment on the namespace tree Liu et al. (2017) and can preserve the relative location of nodes from the metadata namespace tree to a linear hash space. To overcome the problem of the absence of training labels, DeepHash trains model by designed pair loss and triplet loss with the relative position relationships of metadata nodes. The hash codes generated by DeepHash can help recover the data when the master MDS crushes.

Clustering-based Index Structures The Clustering method is a common unsupervised intelligent algorithm, which has been used for indexing in specific scenarios for a long time due to their label-free characteristics. Intelligent Cluster Index (ICIx) is a multidimensional indexing and storage method. Its core idea is to combine methods from the field of AI (in particular nonparametric classification) with those from the field of database accessing Leuoth and Benn (2009). The clustering procedure uses representative parts or the complete database as a training set. The resulting hierarchical clustering will be exploited and particular cluster-ids with their associated tuple are transferred into a persistent management tree that is called V-Tree. It is used for all supported retrieval operations like point and range queries as well as nearest neighbor queries on the basis of a semantic similarity between the requested data objects Neubert et al. (2001). Based on V-Tree, C-Tree Leuoth and Benn (2009) was proposed to store the knowledge of the hierarchical clustering component, i.e., hierarchical Growing Neural Gas (GNG), for unsupervised content based classification. Similarly, NSSSD Gheisari et al. (2016) combines semantic web concepts and clusters sensor data to construct index structure, resulting in data aggregation along with aligning sensors in hierarchical form and reduction of consumption of energy (Table 3).

RMI RMI is a famous issue of learned index that replaces core components of a data management system through learned models Beutel et al. (2017). The premise of RMI is that a B-Tree-Index can be seen as a model to map a key to the position of a record within a sorted array, a Hash-Index as a model to map a key to a position of a record within an unsorted array, and a BitMap-Index as a model to indicate if a data record exists or not Kraska et al. (2018). In fact, it is not difficult to find the key by ML models. For monotonic models, the only thing we need to do is to execute the model for every key and remember the worst over- and under-prediction of a position to calculate the min- and max-error. In addition, the strong error bounds are not even needed. The data has to be sorted anyway to support range requests, so any error is easily corrected by a local search around the prediction (e.g., using exponential search) and thus, even allows for non-monotonic models. As shown in the left part of Fig. 3, the model-based pattern for B-Trees can be any other type of regression model, including linear regression or neural nets. The right part of Fig. 3 shows the real implementation of learned index structures that is the recursive regression model. This hierarchical model consists

Table 3 Learned index

Author	Algorithm	Model	Type	Improvement
Kraska et al. (2019)	SageDB	RMI	B-Tree	Scalability
Kipf et al. (2020)	RadixSpline	Linear spline	Radix tree	Build Speed
Stoian et al. (2021)	PLEX	Linear spline	Radix tree	Scalability
Spector et al. (2021)	RSS	Linear spline	string	Scalability
Ding et al. (2020)	ALEX	RMI	B+-Tree	Scalability
Li et al. (2019)	AIDEL	RMI	B+-Tree	Scalability
Wang et al. (2020)	SIndex	Linear regression	string	Scalability
Tang et al. (2020)	XIndex	RMI	B+-Tree	Scalability
Wei et al. (2020, 2021)	XSTORE	Linear regression	RDMA	Scalablity
Hadian and Heinis (2019)	–	–	Drift estimation	Update for dynamic workloads
Tang et al. (2019)	Doraemon	Linear regression	Model Training and data distribution	Update for dynamic workloads
Mishra and Singhal (2021)	RUSLI	Linear spline	B-Tree	Update for dynamic workloads
Wu et al. (2021)	LIPP	FMCD	B-Tree	Update for dynamic workloads
Ferragina and Vinciguerra (2020)	PGM-INDEX	Linear spline	B-Tree	Update for dynamic workloads
Hadian and Heinis (2021)	Shift-Table	Algorithm Layer	Model Correction	Update for dynamic workloads
Nathan et al. (2020)	Flood	RMI	Spatial data	Other
Li et al. (2020)	LISA	RMI	Spatial data	Other
Davitkova et al. (2020)	ML-Index	Linear regression	Spatial data	Other
Dai et al. (2020)	Bourbon	Linear regression	LSM-Tree	Other
Abu-Libdeh et al. (2020)	Big Table	Linear regression	BigTable	Other
Higuchi et al. (2021)	FIB	RMI	IP packet	Other

of multiple stages, where at each stage the model takes the key as an input and based on it picks another model, until the final stage predicts the position. SageDB Kraska et al. (2019) is the first system to implement RMI. By modeling the data distribution, workload, and hardware, SageDB learns the structure of the data and optimal access methods and query plans, resulting in high specialization for an application. Tim Kraska, who is a SageDB's author, proposed the benchmark of RMI Marcus et al. (2020) and demonstrated that RMI can indeed outperform non-learned indexes in read-only in-memory workloads over a dense array. Based on the benchmark, RadixSpline Kipf et al. (2020) is proposed, which is built in a single pass over the data to improve cumbersome index build efficiency. His team also proposed ALEX Ding et al. (2020) to deal with space-time trade-off. This index dynamically adapts RMI structure based on the workload and offers a model-based insertion by Gapped Array. ALEX uses exponential search to find the key at the leaf node to correct mispredictions of the RMI instead of using a binary search within the error bounds provided by the models. PLEX Stoian et al. (2021) is a combination of CHT and RadixSpline. PLEX builds a spline on the underlying data and indexes the spline points in a CHT. PLEX only has a single hyper-parameter δ (maximum prediction error) and offers a better trade-off between build and lookup time than state-of-the-art approaches. RSS Spector et al. (2021) is a learned index consisting of a tree of the learned index structure RadixSpline. RSS is a tree of RadixSpline with each indexing a fixed number of bytes. RSS uses the minimal string prefix to sufficiently distinguish the data, different from most learning approaches that index the entire string. Additionally, the bounded-error nature of RSS accelerates the last mile search and also enables a memory-efficient hash-table lookup accelerator. Note that the team of Tim Ding et al. (2020) proposed two key issues in RMI, including the scalability and update for dynamic workloads. We will describe other studies from these two aspects.

- *Scalability* The poor scalability comes from the heavy inter-model dependency and expensive retraining. Specifically, to keep all ordered data for range requests, inserting new data into learned indexes will change the positions of some data, thus leading to many data movements or even more probabilities of data lose. AIDEL Li et al. (2019) solves the challenges of significant inter-model dependency and costly retraining by building different models based on the data distribution, eliminating invalid and redundant models. AIDEL uses sorted lists which are appended behind existing data to improve insertion performance and achieve scalability. SIndex Wang et al. (2020) is proposed to learn indexes for variable-length string keys. SIndex leverages a greedy grouping strategy

to adaptively group keys with shared prefixes and uses. XIndex Tang et al. (2020) is a concurrent ordered index that can handle concurrent update operations, which uses a two-phase compaction scheme to merge the current data array and delta index into a new data array. Furthermore, XIndex adapts its structure according to runtime workload characteristics to support dynamic workload. XSTORE Wei et al. (2020, 2021) is an RDMA-based ordered key-value store with a new hybrid architecture that retains a tree-based index at the server to perform dynamic workloads and leverages a learned cache at the client to perform static workloads. It decouples ML model retraining from index updating using a layer of indirection from logical to actual positions of key-value pairs, resulting in the reduction of the end-to-end latency. XSTORE ensures correctness using a validation mechanism with a fallback path and further uses speculative execution to minimize the cost of cache misses.

- *Update for dynamic workloads* Dynamic updates are a fundamental problem in adapting learning models to storage environments. A method for eliminating the drift Hadian and Heinis (2019), i.e., the error of learned index models caused by the updates to the index, is proposed to maintain performance of the learned model under higher update rates. They assume that the drift for a few points in the key set called the "reference points" is known. Thus, the drift can be estimated by its closest "reference points". Tang et al. (2019) is proposed to learn indexes for dynamic workloads. To improve the latency caused by skew queries, Doraemon augments the training data with access frequencies. To address the slow model re-training when data distribution shifts, Doraemon caches the previously-trained models and incrementally fine-tunes them for similar access patterns and data distribution. RUSLI Mishra and Singhal (2021) extends RadixSpline to build the updatable learned index while supporting real-time inserts in a data set without affecting the lookup time on the updated data set. RUSLI can update the index in constant time with an additional temporary memory of size proportional to the number of splines. Following, LIPP Wu et al. (2021) extends the tree structure when dealing with update operations to eliminate the deviation of location predicted by the models in the leaf nodes, and designs a dynamic adjustment strategy to ensure that the height of the tree index is tightly bounded. PGM-INDEX Ferragina and Vinciguerra (2020) is a fully-dynamic learned index, which orchestrates an optimal number of linear models based on a fixed maximum error tolerance in a novel recursive structure. PGM-INDEX is the first learned index job that supports predecessor, range queries, and updates within provably efficient time and space bounds in the worst case. It extends the RMI design by adapting to key access frequencies and distribution, resulting in improvement of space efficiency. Shift-Table Hadian and Heinis (2021) is proposed to make learned model fit for keys in out-of-distribution. It is an algorithmic layer that captures the micro-level data distribution and resolves the local biases of a learned model at the cost of at most one memory lookup. Shift-Table is combined with learned indexes to support low-latency processing of range queries. Shift-Table is optional and applied after the prediction. Hence, it can switch on or off without re-training the model.
- *Others* Other research focuses on spatial indexes or different basic index structures. Flood Nathan et al. (2020) is a RMI facing spatial indexes. Flood can automatically adapt to a specific dataset and workload by adjusting the index structure and data storage arrangement in tandem. It employs ML approaches to create an ideal layout that divides multidimensional data space into a grid of contiguous cells such that data in each cell is sorted together, and it uses linear models to speed up queries. Flood estimates query time using a cost function with the goal of query time reduction. Similarly, LISA Li et al. (2020) is a learned index structure designed for disk-resident spatial data. It leverages learned models to generate data layouts in the disk. LISA consists of a mapping function that maps spatial keys into 1-dimensional mapped values, a learned shard prediction function that partitions the mapped space into shards, and a series of local models that organize shards into pages. It is easy to select the page containing the lookup key by the corresponding local models. ML-Index Davitkova et al. (2020) utilizes clustering method to accelerate processing point, k-Nearest Neighbors (k-NN) and range queries. It partitions the data and transforms it into one-dimensional values relative to the distance to their closest reference point. Once scaled, the ML-Index utilizes a learned model to efficiently approximate the order of the scaled values. ML-Index uses a novel offset scaling method, which is more easily learnable compared to the existing scaling method of the iDistance approach. BOURBON Dai et al. (2020) is a log-structured merge (LSM) tree that utilizes ML to provide fast search. It employs greedy piece-wise linear regression to learn key distributions and applies a cost-benefit strategy to decide when learning will be worthwhile. A new approach to train a learned index for BigTable has been proposed Abu-Libdeh et al. (2020). They designate block boundaries during SSTable creation such that each record $<k, v>$ is placed in predicted block $f(k)$. Meanwhile, they ensure that the learned index always anticipates the correct block for a given key by writing the data to match the index. In addition, RMI also is used for the longest prefix matching in IP packet with key-value store Higuchi et al. (2021). The fundamental idea to apply a learned index to a forwarding informa-

tion base (FIB) is to simplify the complex longest prefix matching operation to a nearest address search operation.

3.1.2 Index search

In fact, there were studies applying intelligent algorithms to improve the search performance on traditional indexes before RMI. We skip the vast of representation-based retrieval and focus only on the methods of search themselves. EFANNA Fu and Cai (2016) combines the advantages of hierarchical structure and nearest-neighbor-graph. It is an extremely fast approximate nearest neighbor search algorithm based on k-NN Graph. Hybrid Indexing Jin et al. (2016) proposes an alternative approach that uses cluster-based retrieval to quickly narrow the search scope guided by version representatives and develops a hybrid index structure with adaptive runtime data traversal to speed up search. LazyLSH Zheng et al. (2016) uses a single base index to support the computations in multiple spaces, resulting in more accurate results for approximate k-NN search under fractional distance metrics with low maintenance overhead. PIT-based Index Hu et al. (2017), improving LSH function by transforming the original data to a new space, uses a preserving-ignoring transformation (PIT) function satisfying some rigorous conditions to convert original points to an interim space with strict distance preserving-ignoring capacity. It can optimize the query quality and time of k-NN search. Space-filling curve and Pivot-based B^+-tree (SPB-tree) Chen et al. (2017) employs a small set of so-called pivots to reduce significantly the number of distance computations, and uses a space-filling curve to cluster the data into compact regions, thus improving storage efficiency on k-NN search. Zoom Zhang and He (2018) is powered by a cluster routing layer (CRL) together with a product quantization layer (PQL) to address the latency challenge of quantization and reduces the cluster selection cost. The PQL compresses vectors through product quantization with optimized distance computation that improves overall system efficiency. GENIE Zhou et al. (2018) proposes a Count Priority Queue (c-PQ) structure on the GPU and a Tolerance-Approximate Nearest Neighbour (τ-ANN) search. It essentially is an inverted index on the GPU to efficiently support the match-count model between objects and queries and can reduce the time cost for similarity search. K-Means CT-Index (KMCTI) Luaces et al. (2019) constructs a binary tree of bitmaps from the CT-Index Klein et al. (2011) fingerprints, which has good performance for large queries, with the cost of increasing the index size and building time.

3.2 Cognitive storage

Storage system efficiency can be significantly improved by determining the value of data. A key concept is cognitive storage, or optimizing storage systems by better comprehending the relevance of data to user needs and preferences Cherubini et al. (2016). Since the user's requirements must be translated into semantic information, the process of acquiring relevance necessitates the usage of AI technologies. A popular storage management technique is to arrange semantically identical data in logically or physically similar locations to improve the efficiency of user reads for the same transaction once the storage system is aware of and has access to semantic information. This near-data processing places the processing power near the data, which contributes to breaking through the bottleneck of I/O bandwidth Hua et al. (2014). The near-data processing can be implemented at three levels: system-level, device-level, and component-level. Note that this research generally falls in the topic of "storage for AI" because it intends to speed up AI computing by designing structures for retrieval and recommendation systems Liang et al. (2019). Therefore, we mainly describe two representative papers that integrally offers semantic information abstraction, relevance acquisition, and storage architecture improvement at the system and device levels.

System-level Content Sifting Storage (CSS) Liu et al. (2020) is designed for analyzing large-scale image dataset, where the read overhead is reduced by sifting relevant data in advance. CSS transforms data content to similarity hash codes as metadata via deep learning and manages the metadata via Semantic Hamming Graph. The deep hashing method used in CSS is the deep self-taught hashing (DSTH) algorithm Wang et al. (2021); Liu et al. (2020, 2019); Zhou et al. (2015) that can learn a model to capture semantic information without hand-crafted labels. These hash codes can measure the content similarity by Hamming distance and provide the sifting metric. When an analysis requirement comes, DSTH generates the similarity hash codes for the requirement (delegating by images), and sends it along with corresponding radius to SHG. SHG retrieves a set of metadata according to the hash code and its radius, organizes the file names from the set of metadata into a subset, and then sends this subset to file storage layer. According to these file names, file storage system reads and returns the corresponding files to the memory for data analysis. Note that although previous research jobs, such as FAST Hua et al. (2014) and smartStore Hua et al. (2009), also employ the image features and the hash codes with Latent Semantic Index (LSI) to design the system, CSS is the first system learning with local data for metadata generation.

Device-level Cognitive SSD Liang et al. (2019), an energy-efficient engine for deep learning based unstructured data retrieval, aims to address high response latency and rising energy consumption. It utilizes a deep hashing method based on AlexNet and ResNet to generate hash code for each metadata. Simultaneously, the graph search method originates from Navigating Spreading-out Graph (NSG), which

well fits the limited memory space of the Cognitive SSD for large-scale multimedia data retrieval. Meanwhile, a flash-accessing accelerator named DLG-x is placed by the side of flash memory to achieve near-data deep learning and graph search. With the DLG-x compiler, Cognitive SSD allows the administrator to train the new deep learning model and generate corresponding DLG-x instructions offline. Then, the administrator can update the learning model running on the Cognitive SSD by updating the DLG-x instructions. The Cognitive SSD consists of a Cognitive SSD runtime embedded CPU, a DLG-x accelerator, and NAND flash controllers coupled to flash chips. Each NAND flash controller links one NAND flash module channel and corrects errors with an error correction code (ECC) engine. The internal bandwidth exceeds the I/O interface when the devices in each channel run in lockstep and are accessed in parallel. The DLG-x design sacrifices the flexibility of deep model updates but avoids reading data into the system cache during metadata generation, resulting in smaller I/O bandwidth. Different from CSS, Cognitive SSD is the in-storage acceleration device for data retrieval. With a similar idea, DeepStore Mailthody et al. (2019) is the first in-storage acceleration device for intelligent queries. In addition, EMB-SSD Kim and Lee (2020), RecSSD Wilkening et al. (2021), and GLIST Li et al. (2021) are in-storage acceleration devices for intelligent recommender systems, where the architectures can accelerate the running of graph-based intelligent algorithms.

4 AI for storage operation and maintenance

Improving the quality of storage service is critical to improving the user experience. The service of operation & maintenance ensures the normal operation of storage systems from a macro level depending on systems and data. Gartner proposes to select storage solutions with advanced AIOps[3] to automate IT functions. We will discuss how to utilize AI technologies for storage resource scheduling, configuration tuning for efficient system operation, and how to use AI technologies for anomaly detection, failure prediction, and root cause analysis under abnormal or sub-healthy system settings.

4.1 Resource scheduling

A large amount of data is stored in the storage system, and the method and location in which this data is stored has a direct impact on read efficiency and storage resource utilization. In order to improve the effectiveness of resource scheduling, some researches use artificial intelligence algorithms to achieve reasonable resource scheduling. The styles of studies consist of prediction for upper-level workloads and building performance models for underlying storage systems. Unlike the focus on managing data resources in Sect. 2.2, this section is concerned with the performance impact of scheduling of resources (e.g., device and virtual resources).

Cloud disk provided to users by cloud providers has been a prevalent form of cloud storage. Because the load characteristics of cloud disks are unknown at the time of disk creation, assigning disk storage space for cloud disks among available data warehouses remains a difficult task. Methods that simply evaluate capacity may result in resource under-utilization and load imbalance between warehouses. Wang et al. (2020) proposed a Smart Cloud Disk Allocation (S-CDA) approach, in which clustering and classification are used to anticipate load information for new cloud disks, and then multi-dimensional allocation based on Manhattan distance is realized. Experiments with real-world cloud workloads reveal that S-CDA improves overall storage/IOPS/disk bandwidth usage while reducing load imbalance when compared to the conventional one-dimensional allocation scheme.

In practice, the virtual machine (VM) virtualizes all physical disks to provide cloud storage capacity. The cloud disk is assigned to the specified repository according to some allocation policy, and the load of this cloud disk will directly affect the resource utilization and load balance of the whole repository, which will also affect the service quality of the cloud disk. Existing VM-based research focuses on resource management by predicting workloads. Cortez et al. (2017) analyzed VM workload characteristics for Microsoft Azure, demonstrated that VM behavior remains fairly consistent over multiple life-cycles, designed the Resource Central (RC) system to provide offline training with online workload prediction. The RC system was also applied to the VM scheduler and improved the VM over-subscription rate. They concluded that providers can exploit these workloads' characteristics and ML to improve resource management substantially. Poppe et al. (2020) proposed the Seagull system that predicts the time of lower workload on the server clients and the exact load during that time. This time period is then used as the backup working time for the database to ensure that the quality of service is not compromised during high loads. The Seagull infrastructure processes telemetry from individual servers, validates the data, then trains and deploys ML models. The models are used to forecast customer load per server for the next 24 hours and to improve service operations. Seagull constantly re-evaluates forecast accuracy, falls back on previously known good models, and sends out notifications as needed.

[3] https://www.gartner.com/en/information-technology/glossary/aiops-artificial-intelligence-operations.

Some studies use AI technologies to predict the performance of the underlying storage system under different workloads. Park et al. (2012) proposed Romano, which is used to optimize heterogeneous virtual data centers. Romano uses a probabilistic approach of linear regression to automatically build and tune performance models for approximate workloads of storage devices, and prediction intervals, Romano then uses a simulated annealing algorithm to explore the most efficient set of migration actions resulting in efficient IO load balancing. Ravandi and Papapanagiotou (2018) proposed a self-organizing resource allocation strategy in cloud storage scenarios, BSDS, which uses DTs and Bayesian networks to predict the performance of storage systems under different workloads, followed by a greedy policy to allocate virtual volumes on different data nodes. BSDS effectively guarantees QoS under both sequential read and random read/write loads.

4.2 Configuration tuning

The system's reliable operation is contingent on the system's compatibility with the hardware and business conditions. In the database industry, for example, database administrators (DBA) alter knobs (parameters) of configuration like cache pool size or concurrent logs to achieve efficient and stable operation in response to users' business needs Zhu et al. (2017); Aken et al. (2017); Zhang et al. (2019); Li et al. (2019); Zhang et al. (2021). The file system also contains a huge number of parameters that can be tuned. A fair tuning for these settings can improve system performance. As the complexity of the file system increases, the number of adjustable parameters grows exponentially, and the traditional manual tuning of parameters no longer achieves good results. The use of intelligent algorithms to automate the tuning process is becoming a popular research topic.

Behzad et al. (2013) proposed to construct an automatic tuning system for hierarchical data format version 5 (HDF5) applications using genetic algorithm (GA). The system finds the space of parameters in each layer of the I/O stack and determines a more efficient parameter setting. The parameter settings are then automatically applied to the HDF5 application. The experimental results show that the automatic tuning system achieves speedups ranging from 2 to 100 times for I/O write operations on different platforms, applications and concurrency. However, the problem of long running time of GA needs to be improved. To further solve this problem, Behzad et al. (2015) proposed to generate and train an empirically based prediction model using statistical methods. The experiments were conducted by running I/O kernels to obtain real measurement data and generate a nonlinear regression model based on the data to select the best parameters. Herodotou and Babu (2011) proposed fine-grained analytical models at the level of MapReduce phases, and then put the predictions of each phase together to predict the total execution time. However, this analytical model assumes that the execution time of each operation remains constant across different Hadoop configurations, which simplifies the complex relationship between configurations. Therefore, it is difficult to accurately predict the performance of Hadoop applications using simple analytical models. To solve this problem, Bei et al. (2016) proposed a tuning scheme RFHOC based on RF and GA for the optimization of Hadoop systems. The scheme first uses RF to learn the input of real Hadoop running data and builds a MapReduce/Hadoop programs performance prediction model. The model and GA are then used to find the optimal Hadoop configuration. Li et al. (2017) proposed an unsupervised parameter tuning system using deep RL. The prototype system was tested on Lustre Braam (2019) that is a large parallel distributed file system. The results show that the system training process can be automated online and the tuning work can be performed while the file system is running. Bagbaba (2020) developed an RF regression performance model for parameter tuning of each layer of the I/O stack for high performance computing applications to be used as performance prediction. The model analyzes the application and file system characteristics by capturing the behavior of parallel I/O. Prats et al. (2020) proposed a strategy for auto-tuning Spark systems that improved prediction by adding Spark workload runtime information. By examining the SparkEventlog, the technique generates feature vectors representing the relevant information about the Spark application without any manual extraction. Then, it is used to train and predict Bayesian Optimization (BO) based ML models. This method enhances prediction accuracy for unseen workloads. In addition, the BO algorithm has the problem of over-sampling the space with heavy overhead in resource-intensive scenarios. Li et al. (2018) proposed Metis to alleviate this problem. It is a robust tuning system using a Gaussian process model and BO-based method to reduce the tail latencies in cloud systems. The BO's strategy for picking the next configuration balances exploitation and exploration at each iteration, which selects regions with high probability and high uncertainty of containing optimum. Zhu and Liu (2019) proposed ClassyTune to auto-tune system by the classification model, which can introduce more training samples without increasing the input dimension. The core principle is to change configuration parameters that have a major impact on the system within short periods of testing. Cao et al. (2018) conducted a comparative study of various black-box auto-tuning techniques in file system tuning. The experiments are conducted in a parameter space consisting of nearly 25,000 different configurations and 450,000 data points, and five different tuning methods are compared in terms of finding near-optimal solutions, convergence time, and instantaneous system throughput. The experimental

results show that the optimal system configuration varies with hardware, software, and load, and that no one tuning method is necessarily better than the other. They also discovered that tuning can take a long time to converge when the parameter space is excessively vast. Thus, they proposed the Carver scheme Zhang et al. (2020) for selecting some of the parameters from the parameter set that have a greater impact on the system performance. The scheme quantifies the importance of parameters by a variance-based objective function, and uses Latin Hyper-cube Sampling to sample from a huge parameter space, and finally identifies the important parameters using greedy methods. The tests were repeated 500,000 times in seven different file systems and revealed that a small fraction of parameters has a bigger impact on system performance than others, that this subset varies with workload. In addition, there are interactions between parameters.

4.3 Anomaly detection

Anomaly detection (also known as outlier detection) is a concept derived from data analysis, which is the identification of rare items, events, or observations. In storage services, the observation of these rare items, events helps determine the status of the devices or systems, make immediate adjustments when the observation raises suspicion that it is significantly different from most data, and ensure running stability. Existing research focuses on hard disks (device-level) and systems (system-level) using anomaly detection algorithms to maintain storage stability. We will detail the anomaly detection technologies from these two aspects. Note that some studies of anomaly detection are aimed at fault prediction, and we have grouped these studies within the section of fault prediction.

Device-level HDDs are the primary data storage devices. The reliability of HDDs is critical for data reliability and security. Queiroz et al. (2018) proposed a semi-parametric statistical model, i.e., Fault Detection of HDDs based on Gaussian Mixture Model (FDGM), for disk anomaly detection. The method uses the Gaussian Mixture Model (GMM) to model the behavior of each healthy disk to obtain its similarity with the statistical model, and the disks with similarity exceeding a preset threshold are detected as anomalous objects. Pereira et al. (2019) evaluated the performance differences of several One-Class Classifiers (OCC) on disk anomaly detection tasks. Specifically, this work evaluated single GMMs and variants of GMMs in a single classification approach based on density techniques; *k*-NN and support vector data description (SVDD) in a single classification approach based on boundary evaluation; K-means, principal components analysis (PCA), and bottleneck neural networks (BNN) were evaluated in the single classification method based on the construction technique. The results of the comparison demonstrate that the single classification method based on reconstruction technique performs the best in disk anomaly detection. Although the single classification method based on the density technique and the reconstruction technique method have close area under curve (AUC) values, the reconstruction technique method has a higher true positive rate and a lower false alarm rate. An incremental learning model based on edge density was proposed by Gao et al. (2019). The method establishes the isolation region by K-nearest neighbors and uses Euclidean Distance (ED) to measure the isolation of the measured samples from the nearest training samples rather than the global anomaly samples. The experimental results show that the proposed scheme improves the recall rate compared with several unsupervised learning methods including the isolation forest algorithm (iForest).

System-level Anomaly detection at the system level is usually system-specific, especially in sensor-based systems. For example, IT system monitoring is a process of observing a system's quantifiable events and outputs and using them as a benchmark for determining the system's correct operation. Efficiency of detection is the most interesting issue in system-level studies. For quick initialization, Ma et al. (2021) introduced the Compressed Sensing technique to multivariate time series anomaly detection. They proposed JumpStarter as a method for creating a fast-starting anomaly detector. They created a shape-based clustering algorithm and an outlier-resistant sampling technique for JumpStarter. JumpStarter outperforms the state-of-the-art anomaly detection algorithms, with a much shorter initialization time of twenty minutes.

4.4 Failure prediction

Failure prediction is essential for the maintenance of storage systems due to its ability to prevent sudden system crashes and unexpected failure. Compare to anomaly detection, it relies on sharper pattern discovery to find out the latent failure of the future in advance. Research on this topic began early but still has been limited due to the number of failure samples and the unique characteristics of system devices. Remarkably, the employment of intelligent algorithms as a prediction broke such limits, not only providing a feasible and efficient way to figure out the failure pattern but also achieving a higher failure detection rate (FDR) and lower failure alarm rate (FAR). Table 4 list these works in terms of storage devices.

HDD Research on HDD failure prediction based on intelligent algorithms was first conducted in 2001. After that, this research grows rapidly, whose topics cover a wide task area, such as remaining useful life (RUL), health degree assessment (HDA) and latent sector error (LSE), whose standpoint focus on different problem scenarios likes the minority disk,

Table 4 Disk failure prediction algorithms using intelligent algorithms

Author	Prediction Algorithm	Storage media	Task	Problem scenarios	Used Intelligent Algorithm
Hamerly and Elkan (2001)	–	HDD	FP	–	BN
Murray et al. (2003)	–	HDD	FP	–	SVM; Clustering
Murray et al. (2005)	mi-NB	HDD	FP	–	NB; Multiple-Instance Learning
Zhao et al. (2010)	–	HDD	FP	–	HNM; HSMM
Pitakrat et al. (2013)	–	HDD	FP	–	ML
Agarwal et al. (2009)	MLRules	HDD	FP	Data insufficient, disk events	ML
Featherstun and Fulp (2010)	–	HDD	FP	Data insufficient, syslog	ML
Ganguly et al. (2016)	–	HDD	FP	Data insufficient, performance counters	DT; LR
Zhu et al. (2013)	–	HDD	FP	–	BPNN; AdaBoost; SVM
Yang et al. (2015)	HDoctor	HDD	FP	–	LR
Li et al. (2017)	–	HDD	FP	–	DT; GBRT
Pereira and dos, (2017)	–	HDD	FP	Minority disk	TL
Zhang et al. (2020)	TLDFP	HDD	FP	Minority disk	TL
Aussel et al. (2017)	–	HDD	FP	Unbalance dataset	SVM; GBRT; RF
Shen et al. (2018)	–	HDD	FP	Unbalance dataset	RF
Sun et al. (2019)	–	HDD	FP	Unbalance dataset	TCNN
Xu et al. (2018)	CDEF	HDD	FP	Unbalance dataset	MART
Botezatu et al. (2016)	–	HDD	FP	Heterogeneous disk	RGF; TL
Pâris et al. (2017)	–	HDD	FP	Heterogeneous disk	DT, NN; LR
Zhang et al. (2020)	HDDse	HDD	FP	Heterogeneous disk,minority disk, unbalance dataset	LSTM, SN
Xiao et al. (2018)	ORF	HDD	FP	Online learning, model aging	RF
Jiang et al. (2019)	SPA	HDD	FP	Cold start, model aging	GAN; CNN
Lu et al. (2020)	CNN-LSTM	HDD	FP	Performance metrics, location attributes	CNN; LSTM
Luo et al. (2021)	NTAM	HDD	FP	Neighbors' status data	Transformer
Li et al. (2016)	–	HDD	RUL	–	RNN; GBRT; DT
Chaves et al. (2016)	BaNHFaP	HDD	RUL	–	BN
Chaves et al. (2018)	–	HDD	RUL	–	BN
dos Santos Lima (2018)	–	HDD	RUL	–	LSTM; RNN; Elman NN
Anantharaman et al. (2018)	–	HDD	RUL	–	RF; RNN
Basak et al. (2019)	–	HDD	RUL	–	LSTM
Li et al. (2014)	–	HDD	HDA	–	CART; CT
Huang et al. (2015)	–	HDD	HDA	–	RT
Ji et al. (2015)	–	HDD	HDA	–	RT
Pang et al. (2016)	–	HDD	HDA	–	BPANN; ENN; CBN
Xu et al. (2016)	–	HDD	HDA	–	RNN
Züfle et al. (2020)	–	HDD	HDA	–	RF
Mahdisoltani et al. (2017)	–	HDD	LES	–	CART; RF; SVM; BP; LR
Jiang et al. (2019)	–	HDD	LES	–	LR; RF; SVM
Zhang et al. (2020)	TS	HDD	LES	–	LSTM
Narayanan et al. (2016)	–	SSD	FP	Datacenter	ML
Alter et al. (2019)	–	SSD	RUL	Manual repaired disk	ML
Chakrabortti and Litz (2020)	–	SSD	FP	Unbalance dataset	iForest;AE
Xu et al. (2021)	WEFR	SSD	RUL	Heterogeneous disk	RF
Giurgiu et al. (2017)	–	DRAM	FP	Uncorrected errors	RF
Boixaderas et al. (2020)	–	DRAM	FP	Uncorrected errors	RF
Baseman et al. (2016)	–	DRAM	FP	Corrected errors	BN; LR; RF; GBRF
Wang et al. (2021)	HiDEC	DRAM	FP	Corrected errors	DT
Mukhanov et al. (2019)	–	DRAM	FP	Corrected errors	SVM; *k*-NN; RDF
Wu et al. (2021)	–	DRAM	FP	Corrected errors	SVM; LR; iForest

BN represents the bayesian network. NB denotes naive bayes. DT is the decision tree model. TL represents transfer learning model. AE denotes the AutoEncoder model. iForest is the isolation forest model. RF represents the random forest model. SN represents the siamese network. RNN

Table 4 (continued)

denotes the recurrent neural network. CART represents the classication and regression trees model. CT denotes the classication tree model. MART represents multiple additive regression trees. GBRT denotes gradient boosted regression trees model

the unbalance dataset, the heterogeneous environment, the online learning, and the model aging, whose methodology shifted from the shallow easy-to-implement models to deep learning models and also to time-series models. According to the above problems, we summarize them in Table 4 and detail them in the following.

From the topic view of disk failure prediction (FP), Hamerly and Elkan (2001) firstly applied two Bayesian methods to predict HDD failures. They discovered that intelligent approaches outperform threshold-based methods in terms of prediction accuracy. Clearly, their results show limited benefit here but have greatly inspired the follow-up work. Recent studies provided insight about their limited improvement that possible mainly due to the limited sample setthat contain samples come form 1927 healthy and 9 failure disk data, or the part of the reason for their unsupervised methods. Murray et al. (2003) performed SVM, unsupervised clustering, rank sum test (RST), and reverse arrangement test (RAT) to predict failure . Their experimental results showed that the RST methods achieve the best performance. Based on this work, they further combined the Multiple-Instance Learning and Naive Bayes to propose a new method named Multiple-Instance Learning Naive Bayes (mi-NB) Murray et al. (2005), which performed considerably better than the unsupervised clustering algorithm.

Using the same dataset with Hamerly and Elkan (2001), Zhao et al. (2010) proposed the two time-series Markov models to predict HDD failure, and Pitakrat et al. (2013) compared 21 ML methods commonly used in disk failure prediction tasks in the WEKA. The former made optimization by considering observed data as time series, while the latter's experimental results showed such algorithms like the k-NN classifier, RF, c4.5, REPTree, RIPPER, PART, and K-Star are suitable for application scenarios that require high accuracy but are not limited by training time, such algorithms like BN and OneR are suitable for online learning because they require shorter training and prediction time, and the sequential minimal optimization (SMO) and SVM are the best choices for achieving a low or near-zero FAR.

Notably, the methods above predict failure with disk's SMART data which is recorded and provided by an embedded program named Self Monitor and Report Technology. Insight by that disk events also can be utilized for disk failure prediction with better performance, Agarwal et al. (2009) proposed the Maximum Likelihood Rules (MLRules) model trained with disk events from each layer of storage system stack and achieve a better performance. Similarly, with the system log syslog, Featherstun and Fulp (2010) performed the SVM with a spectrum kernel and proposed a sliding window algorithm to optimize the prediction process. SMART data by itself is not sufficient, Ganguly et al. (2016) combined SMART data with performance counters and provided a two-stage model for predicting disk failures. The first stage of the model uses a parameter-free DT model to reduce the prediction overhead, and the second stage uses logistic regression to improve the accuracy and recall of the whole model.

Fortunately, the problem of less accuracy caused by data insufficient gradually eased as various disk datasets are released by serval companies. Zhu et al. (2013) predicted HDD failures with the largest dataset during that time from the Baidu, which include 22,962 healthy drives and 433 failure drives. Their experimental results reveal that both ML methods obtain good prediction outcomes. Yang et al. (2015) designed the disk prediction system HDoctor for Google. This system is based on linear logistic regression. Li et al. (2017) proposed two failure prediction models based on DT and gradient boosted regression tree (GBRT). Their experiments use 121022 healthy disks and 679 failure disks, collected from two real-world data centers with smartmontools. The results show that the DT model get higher FDR score while the GBRT get near zero FAR score.

However, research on the well-acknowledged dataset BackBlaze suffers from various problem scenarios and these challenges drive much investigation. The minority disk is one of the problem scenarios where new disks gradually replace failed disks and their models suffer from data insufficiently. Pereira and dos, (2017) found that transfer learning (TL) is suitable for resolving the minority disk problem and suggested that clustering information of a similar disk model can be developed as a data source for the model training of minority disks. This work performed three different TL methods and revealed that TL can be utilized to improve disk failure prediction performance, especially for the minority disk. Zhang et al. (2020) also develop a TL model called TLDFP to predict minority disk failure. In their work, the definition of minority disk datasets was evaluated and the kullback leibler divergence (KLD) values were used to select proper source models.

The unbalance datasets are another problems scenarios in the Backblaze dataset. Aussel et al. (2017) found that the Backblaze contains 47,000 disk samples and 81 disk models from 5 vendors, but with an extremely unbalanced ratio of 5000:1 between healthy and failures samples. Their experimental results show that the highly imbalanced samples seriously affect the disk failure prediction

performance of the model, which cannot be improved even if trying to generate other training samples of a few classes by interpolating using the synthetic minority over-sampling (SMOTE) technique. To further solve the imbalance dataset problem, Shen et al. (2018) proposed a clustering-based sub-sampling method and used an RF method based on partial prediction for disk failure prediction. The experimental results show that the prediction algorithm used can respectively achieve acceptable performance on the three sub-datasets divided from the dataset of Backblaze and BaiDu. Considering that disk failure is often a time-dependent and asymptotic process, Sun et al. (2019) used a temporal convolution neural network (TCNN) with temporal data processing capability to predict system-level disk failures for the interference of failure latency with temporal learning models such as recurrent neural network (RNN) and LSTM. In addition, this study also tries to optimize the loss function to avoid gradient disappearance to solve the sample imbalance problem. Xu et al. (2018) pointed out that the ranking learning model is not sensitive to samples with unbalanced categories, so it can be used to solve the unbalanced samples problem. They proposed an online prediction algorithm CDEF (Cloud Disk Error Forecasting) based on multiple additive regression trees (MART) to predict disk errors (including sector failures) under cloud systems. Experimental results show that CDEF is more effective than baseline methods such as RF and SVM in predicting disk errors, and the proposed cost-sensitive ranking model achieves the lowest cost among all comparison methods of all data sets.

The heterogeneous scenarios problem causes limited applicability and lack of adaptability and is common in larger-scale storage. Different disk models have different SMART value distributions whether the disk comes from the same manufacturers or not. New models replace the old models also gradually change the sample distribution. Botezatu et al. (2016) use regularized greedy forests (RGFs) for disk failure prediction, and employ TL to solve the sample migration problem between different models of disk datasets. The results show that regularized greedy forests can predict disk failure up to 15 days ahead of time, and TL can reuse labeled data information from the manufacturer-specific disk model and then build a high-quality prediction model for a new disk model from the same manufacturer without labeled data. Pâris et al. (2017) compared the failure prediction performance of three ML methods including DTs, NNs, and logistic regression. But their experimental result is not ideal. Zhang et al. (2020) proposed a novel generic disk failure detection approach named HDDse for heterogeneous disks. They employ an LSTM based siamese network (SN) to learn the long-term behavior of disk healthy statues and to generate high dimensional state embeddings for failure detection. Because their motivations are to strengthen the general ability of the prediction model, the imbalanced dataset and the minority also take into consideration.

The traditional offline method no longer updates the model after the training model is obtained, and there will be a model aging problem. Using online learning methods can effectively solve the model aging problem. Xiao et al. (2018) introduced online random forest (ORF) method in disk failure prediction and proposed an automatic online labeling algorithm for collecting and labeling samples of the last few days. At the same time, they proposed to use two Poisson distributions to model the sequence arrival process of positive and negative samples respectively, so that negative samples have less chance to be selected and updated by the model. The experimental results showunder the continuous test for tens of months, the online RF model can significantly improve the FDR compared with the offline model . Moreover, Zhang et al. (2020) proposed a Semi-supervised disk failure Prediction via Adversarial training (SPA) based on generative adversarial network (GAN) and CNN. This model avoids the cold start problem by using only healthy disk samples for training.

Intuitively, the addition of metrics such as location attributes can increase the prediction power. Recently, Lu et al. (2020) conducted disk prediction using CNN with LSTM (CNN-LSTM). The model training not only uses SMART data but also uses disk performance metrics and location attributes. The experimental results show that the performance metric and location attributes of the disk can effectively improve the quality of disk failure prediction, and the CNN-LSTM method can predict the failure disk within ten days in advance on average, and the F-Measure and MCC metric values are both higher than 0.90. Luo et al. (2021) propose a Transformer based approach named Neighborhood-Temporal Attention Model (NTAM), which not only uses disk own status data but also captures the temporal nature from neighbors' status data based on attention.

Different from failure prediction, research on RUL refers to estimating HDD life from the current state to the failure. With the RUL as an indicator, the imminent failure disks can migrate or back up their data orderly, and there is plenty of time for the data center or system to conduct disk replacement more economically. Li et al. (2016) used three ML models to predict disk RUL and proposed migration rate (MR) and miss migration rate (MMR) to compare three models' performance. Their experimental results show that compared with the state-of-the-art RNN model, the GBRT model has a poor predictive performance, it performs better on MR and MMR while the RNN model has migrated more data. Chaves et al. (2016) proposed a BN based HDD Failure Prediction (BaNHFaP) based on a BN to predict RUL of the disk. In their method, disk RUL is evaluated using its working time and SMART attribute. The experimental results show that the RUL predicted by the BaNHFaP is close to

the truth. To predict the RUL more accurately, they further proposed an RUL prediction model based on Chaves et al. (2018). This work differs from the previous because it not only inputs SMART attributes during model training but also inputs the changing trends of some SMART attributes as features. Therefore, the solution model can also extract the timing relationship of the SMART attributes, and thus can more accurately predict disk RUL. Similarly, dos Santos Lima (2018) proposed LSTM to extract the long-term dependency of SMART data and used it to predict the RUL of disks. The research first divides RUL of the disk into buckets, and unequally classifies each bucket. The closer the disk is to the point of failure, the smaller the life bucket is set. Then use LSTM, RNN, and Elman NN models to experiment. The experimental results showed that compared with Elman NN and RF method, LSTM is better at dealing with long-term dependency prediction tasks. Anantharaman et al. (2018) used RF and LSTM to predict RUL. Basak et al. (2019) proposed a data-driven architecture based on the deep LSTM architecture for the online evaluation of RUL. The experimental results also show that their model has a certain degree of versatility and portability, and can be used to predict the RUL of another model of disks from the same manufacturer.

Compared to a simple failure prediction method, the health degree assessment is viewed as multi-class classification problem, which is more valuable in practice because it enables system managements to schedule the recovery of different hard drives according to the level of urgency. Li et al. (2014) proposed to use CART to classify disk health status. It predicts disk failures by using a classication tree (CT) model, and uses regression trees (RT) to score the current health status of the disk. Experimental results show that in this scenario the FDR and FAR are excellent. However, the prediction performance of the health model using regression trees cannot completely outperform the two-classifier model, and other methods need to be further tried. Huang et al. (2015) found that disk failures can be classified according to the unique manifestations and attributes of disk failures, and then deduced the degradation characteristics of each failure category to describe the degradation process from health to failure. In their work, regression tree model can not only predict each type of disk failure, but also predict performance degradation in different stages. The research ranks the health of disks and shows how to find the types of disk failures, and successfully characterizes the failure degradation process in the data center. Similarly, Ji et al. (2015) also built a multi-classification model based on regression trees to assess the risk of disk failures, and proposed a self-scheduling migration mechanism (Self-Scheduling Migration, SSM) based on the different risks of disk failures. Considering that the tasking the probability of disk failure as the criterion of ealth degree is not rigorous enough, Züfle et al. (2020) proposed to use the RUL of the disk to define the health degree. They proposed to train four separate classifiers using backpropagation artificial neural network (BPANN), evolutionary neural network (ENN), SVM, and CT distribution, and then use the Combined BN (CBN) implements a combined classifier to predict disk failure and its health. Xu et al. (2016) on the same team as Li et al. (2014) continued the research work of the above-mentioned disk gradual change process. Based on RUL of the disk, a discrete classification method was used to divide the disk health status into 6 grades. Subsequently, the work proposed to use RNNs to predict disk failures, and use the gradually changing sequence of SMART attributes to evaluate the operating status of the disk. Experimental results show that compared with other sequence independent models and short-term sequence correlation models, the RNN-based method can achieve higher prediction accuracy. Moreover, Züfle et al. (2020) proposed to use RF to predict the time window of disk failure. Specifically, they classified the close disk failure time into one category to form multiple time window classes, and predicted the time window for the impending disk failure.

Different from the whole disk failure, sector error is a kind of potential disk error that is difficult to be directly detected. It is often necessary to scan each sector with the help of disk scrubbing technology to find the failed sector. Mahdisoltani et al. (2017) used classification regression trees, RF, SVM, NNs, and logistic regression to predict disk sector error, and proposed that the disk scrubbing rate can be accelerated or slowed based on the prediction results. Jiang et al. (2019) proposed an unleveling disk scrubbing scheme, which uses logistic regression, RF, and SVM to predict disk sector errors, and then dynamically adjust the disk scrubbing rate according to disk status. Zhang et al. (2020) designed a Tier-Scrubbing (TS) based on the long short-term memory and piggyback scrubbing strategy, achieving higher reliability and lower scrubbing cost.

SSD SSDs employ integrated circuit assemblies to store data in a durable manner, generally using flash memory, and serve as secondary storage in a computer's storage hierarchy. Since the physical addressing method is different from HDD, the SMART attribute becomes less reliable for SSD state description, so it is unreliable to predict failed SSD only with SMART attribute. The problem of failure prediction for SSDs becomes a new focus. Note that because the SSD research boom coincided with the rapid development of ML, the research in SSDs has involved ML from the beginning. Narayanan et al. (2018) present an extensive SSD failure characterization by examining data from over half a million SSDs spanning different generations and distributed across several datacenters hosting a variety of workloads for nearly three years. It's the first in-depth examination of the what, when, and why of SSD failures in real datacenters. In the impact evaluation of all relevant factors on failures, they

employed ML models and graphical causal models. With failure signatures from tens of important factors and their threshold values, failed devices can be differentiated from healthy ones with high precision and recall score. Alter et al. (2019) investigated the SSD retirement process by studying drive failures that require manual intervention and repairs. They investigated this by analyzing daily performance logs for three multi-level cell (MLC) models collected over the course of six years in a Google production data center. They used ML predictors to figure out which quantitative measurements provide good indicators of impending failures. According to experimental results, they believed that the drive age plays a key role in model learning. To deal with the extremely low percentage of failure samples in SSD failure prediction, Chakrabortti and Litz (2020) advised to turn the prediction problem into an anomaly detection problem. They proposed the use of 1-class iForest and AutoEncoder (AE) -based anomaly detection techniques for predicting previously unseen SSD failure types. It is found that the anomaly detection method based on AE can effectively improve the failure prediction accuracy of a system in a dynamic environment. In addition, correctable_errors are most associated with SSD failures, which may be caused by the frequent triggering of ECC mechanisms by correctable_errors. Xu et al. (2021) focus on the problems of disk heterogeneity and manual feature selection in SSD failure prediction. WEFR, a system for automatic feature selection based on SSD traits, was proposed. Not only does WEFR pick different features for different types of SSD, but it also updates the features that are best for model learning over time. The remarkable contribution is that they study the relationship between the wear-out degree and SSD failures. MWI_N, the indication of the percentage of the remaining erase cycles for SSDs, is introduced as a learning feature. With the RF model and MWI_N, they achieved high precision and F-0.5 score.

DRAM DRAM errors are one of the most common hardware failures in the modern large-scale data center. Once DRAM errors exceed the correction ability of the ECC, it will cause the uncorrectable error and further break down the server. Recently much works based on ML for DRAM failure prediction has been introduced. According to predictive scenarios, DRAM failure can be classified into uncorrected DRAM errors (UEs) and corrected DRAM error (CEs). Table 4 lists them and we detail them in the following.

The first research on DRAM uncorrectable errors prediction is provided by Giurgiu et al. (2017). They Giurgiu et al. (2017) proposed an RF predictive model to predict uncorrectable errors based on daily event logs and sensor measurements. Their experimental results showed that the RF model achieved high precision. Boixaderas et al. (2020) are the first work to show that only a small fraction of uncorrected error also can lead to node failure. They compared six ML models and choose RF as a classifier to predict DRAM error. Their results show that the performance of DRAM error prediction is highly dependent on the system and its workload characteristics.

With the goal to categorize DRAM corrected errors such as Single bit, Single word, Single row, Single column, Single bank, Multi bank or Multi rank. Baseman et al. (2016) explored the predictive performance of models based on Naive Bayes, logistic regression, RF, and gradient boosted RF. Their result showed that the predictive performance has been improved with the AI approaches compared with the rule-based. Workloads running on top of the system have an impact on DRAM failure patterns and have been viewed as important features for DRAM failure prediction. Wang et al. (2021) proposed Hierarchical DRAM Error Code (HiDEC) to represent the DRAM access pattern and utilized the DT model to predict DRAM failure. The results show that their features used to train models from macroscopical and microscopical aspects can bring significant improvements to the prediction performance. Mukhanov et al. (2019) extract 249 features from memory operating parameters or programs and then apply SVM, KNN, and RDF to construct the DRAM error prediction model. Their results show with the correct choice of program features and an ML model can predict the single-bit failures and find the system crash triggered by uncorrectable errors. On the one hand, Wu et al. (2021) viewed DRAM error prediction as a multi-class classification problem and then evaluated the performance of the state-of-the-art classifiers including SVM, LR, RF, DT, GBDT, XGB, LGBM. On the other hand, they formulate the same problems as the anomaly detection task and compared three anomaly detection methods, such as iForest, HBOS, and COPOD.

4.5 Root cause analysis

The process of uncovering the root causes of problems in order to develop appropriate solutions is known as root cause analysis (RCA). RCA believes that consistently preventing and resolving core issues is far more successful than treating ad hoc symptoms and putting out fires. A collection of ideas, approaches, and methodologies can be used to locate the root causes of an event or trend. RCA can reveal where processes or systems failed or produced an issue in the first place, going beyond simple cause and effect. Existing research on RCA using AI technologies mainly faces services and systems. We list these studies in Table 5 and describe them in detail in the following paragraphs.

Service-Oriented Architecture (SOA) The first RCA research using AI technologies was applied to website architecture. Since websites are predominantly built as an SOA, Kim et al. (2013) proposed the MonitorRank algorithm for root cause detection of SOA. The MonitorRank algorithm uses two types of data: metric data for each sensor (latency,

Table 5 Root cause analysis research using intelligent algorithms

Author	Framework/Algorithm	Environment	Used Intelligent Algorithms	Contribution
Kim et al. (2013)	MonitorRank	SOA	Clustering; RW	Deployment in a runtime environment
Luo et al. (2014)	–	SOA	*k*-NN	Online analysis
Zhang et al. (2019)	MSCRED	SOA	AE; ConvLSTM	Correlation between multivariate time series data
Weng et al. (2018)	–	SOA	RW	Non-intrusive way to capture dependencies of multitier services
Liu et al. (2019)	FluxRank	SOA	Clustering; Logistical Regression	Generalization of framework
Lin et al. (2018)	FacGraph	Microservice architecture	PC	Low cost and parallel framework
Zhou et al. (2019)	MEPFL	Microservice architecture	RF; *k*-NN; MLP	Using trace log
Ma et al. (2019)	MS-Rank	Microservice architecture	PC; RW	Update metric weights considering multiple metrics
Ma et al. (2020)	AutoMAP	Microservice architecture	PC; RW	Automatic weight assignment of metrics using history data
Gan et al. (2019)	Seer	Microservice architecture	CNN; LSTM	Solution of microservices in cloud services
Qiu et al. (2020)	–	Microservice architecture	PC	Optimization of PC algorithm
Wu et al. (2020)	MicroRCA	Microservice architecture	Birch; Personalized PageRank	Fusion of service performance and resource utilization rate
Meng et al. (2020)	MicroCause	Microservice architecture	PCTS; TCORW	Improve PC and RW algorithms by event serials
Liu et al. (2020)	TraceAnomaly	Microservice architecture	Deep BN; VAE	interpretability
Cheng et al. (2016)	–	Distributed system	ARX	Joint dynamically changing network information
Pham et al. (2017)	–	Distributed system	*k*-NN	Targeted fault injection
Ni et al. (2017)	CRD	Network system	Clustering; ARX	localization of multiple causal anomalies
Sun et al. (2018)	HotSpot	Network system	MCTS	Pruning
Wang et al. (2018)	CouldRanger	Cloud system	PC; RW	Optimization of PC and RW
Wang et al. (2019)	GRANO	Cloud system	Clustering;	RCA based on ML
Arzani et al. (2016)	NetPoirot	Data center	DT	Lightweight monitoring solution with low cost
Cai et al. (2019)	–	Data center	Clustering	Off-line modeling and on-line analysis

SOA is the service-oriented architecture. RW represents the random walking algorithm. ARX denotes AutoRegressive eXogenous model. DT represents the decision tree algorithm. ML is the machine learning model

error count, throughput) and a graph of calls between sensors. The MonitorRank algorithm periodically generates call graphs and extracts external factors based on a pseudo-anomaly clustering algorithm for historical metric relevance. Using the similarity of metric data between anomaly sensors and their corresponding downstream sensors as a key feature, a random walking (RW) is implemented on the call graph based on the similarity score. At last, the frequency of accessing each sensor is used as the score of the root cause. This approach represents the dominant framework for RCA studies, i.e., graph construction and visit-frequency collection. Luo et al. (2014) investigated the correlation between time series data and event series data in order to resolve faults in time and to maximize loss reduction. This correlation is analyzed to find out the possible KPIs at the time of the failure. The algorithm introduces a nearest neighbor algorithm to determine the two sample distinctions, which in turn is used to determine the correlation between the event, the time series data for a period of time before an event or after an event. Zhang et al. (2019) paid attention to the scalable of RCA for multivariate time series data and proposed a Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED). MSCRED introduces AE and ConvLSTM with temporal patterns and attention into RCA, simultaneously addressing the three issues of anomaly detection, root cause identification, and anomaly severity (duration). Weng et al. (2018) argued that the performance degradation of multilevel services in cloud platforms could be caused by the resource competition. They argued that cloud providers could be in a better position to address this issue since

providers have access to information on resource competition among different tenants. The algorithm introduces similarity scoring and RW, which is able to find two factors that cause public cloud anomalies: component bugs in anomalous services and performance interference from other tenants. Liu et al. (2019) observed that mainstream methods of root cause detection (building dependency graphs, using ML methods, etc.) have problems such as difficulty in collecting data and poor interpretation. Therefore, they proposed FluxRank, a system that collects data on KPIs in software services for automatic root cause localization. There are three primary steps in the FluxRank system. Firstly, determine the time of anomaly and the degree of anomaly; Secondly. build feature vectors for several machines and cluster them; At last, rank the clustering results so that the ranking results reflect the root cause.

A microservice architecture, which is a variation of the SOA structural style, divides an application into loosely connected services. The services in a microservices architecture are fine-grained, and the protocols are light. The idea is for teams to be able to bring their services to life without relying on others. Lin et al. (2018) proposed a frequent pattern mining algorithm on anomaly correlation graph, named FacGraph. FacGraph designs an anomaly graph-based frequent pattern mining algorithm (FSM) for root cause detection, while employing breadth-first ordered string (BFOS) to reduce the time-consumption of FSM. FacGraph uses the PC algorithm to build up the correlation graph, and then extracts the frequent system correlation graph (SCG) from the correlation graph using FSM. FSM takes the failed front-end service as the root node, and generates other SCGs by adding other nodes. Then, it determines whether the count of this SCG in all graphs is greater than the threshold value. If it is greater, the node connected to the root node in this SCG is identified as the root cause. Zhou et al. (2019) proposed the MEPFL algorithm to discover multi-instance faults, configuration faults, and asynchronous interaction faults by analyzing and learning trace logs. MEPFL is also divided into two parts: offline training, where a series of traces are obtained by fault injection and debugging, features are extracted, and a prediction model is trained using ML methods. Online prediction, on the other hand, implements monitoring trace logs, collecting data, extracting features, and then using the prediction model for prediction and diagnosis. In order to solve the inaccuracy of root cause detection results caused by a single metric in the microservice architecture, Ma et al. (2019) proposed the MS-Rank algorithm, which enables the system to complete impact graph construction, RW for root cause detection, resulting in precision calculation and updating metric weights considering multiple metrics. They also designed the AutoMAP Ma et al. (2020) system to select metrics of microservices using historical data. With the use of historical data analysis, this system can complete the automatic weight assignment of metrics. The AutoMAP system generates the behavior graph using a PC method and completes the RW on this graph. The system defines the graph's addition and subtraction, then extracts the anomalous profiles and service profiles as historical analysis data from the graph. The most similar anomalous service profiles can be found by analyzing similarity and utilized to determine the index weights for automatic index assignment in the new round of RCA. Gan et al. (2019) proposed the Seer debugging system to diagnose impending QoS violations in an online scalable manner, which copes with the increasing complexity of cloud computing. Seer employs a deep learning model that contains a set of convolutional layers and LSTM layers. Qiu et al. (2020) proposed to use the knowledge graph to help construct relationship graphs. RCA is implemented on the relational graph with knowledge graph, PC algorithm, and breadth-first algorithm. Wu et al. (2020) proposed the MicroRCA system, an application-independent system designed for container-based microservice environments. The system continuously collects application- and system-level metrics and detects service-level metrics anomalies. Once anomalies are detected by an online clustering algorithm (Birch), the anomaly propagation is modeled by constructing an attribute graph. After that, communication service anomaly symptoms are correlated with the associated resource utilization and potential anomalous services are inferred using the graph centrality algorithm Personalized PageRank. The system is effective in capturing anomaly propagation. Meng et al. (2020) found that the traditional PC algorithm regards two data intercepted in two time periods as i.i.d, resulting in ignorance of the causality with anomalous propagation delays. They proposed the MicroCause framework, which proposes an improvement based on temporal characteristics for both the PC algorithm and the RW algorithm, resulting in the accurate results of RCA. Liu et al. (2020) designed a model of deep BN with posterior flow to finish the anomaly detection and introduced a novel trace representation to intuitively locate root cause.

Systems and Data Centers For the distributed systems, Cheng et al. (2016) proposed an invariant network model-based framework for ranking causal anomalies. This label propagation theory-based method can effectively model fault propagation over the invariant network and derive anomaly scores by reconstructing the vanishing invariant correlations. Thus, this model is able to locate the high-confidence anomalies that actually cause the vanishing correlations and can compensate for the unstructured measurement noise in the system. Pham et al. (2017) proposed a fusion of Targeted Fault Injection (TFI) techniques and data analysis techniques for automatic fault diagnosis in distributed systems. It provides the location of faults close to the actual errors and can be easily used to find and fix the actual root cause of the problem.

For the network systems, Ni et al. (2017) proposed the CRD algorithm. Firstly, probabilistic clustering uses to identify impaired node groups in the invariant network. Then, a network diffusion model is constructed to retrace the root cause in each degraded cluster. The previous invariant model usually assumes only a global single fault propagation in the whole network, while this model can capture different local parallel fault propagation and is suitable for locating multiple causal anomalies. Sun et al. (2018) proposed the Hotspot method to solve the root cause search when anomalies occur in Additive KPIs with multidimensional attributes. Hotspot introduces the MCTS algorithm that uses the potential root cause score based on ripple effect, achieving root cause estimation and prediction while narrowing the search space by hierarchical pruning.

For the cloud systems, Wang et al. (2018) proposed the CouldRanger framework including anomaly detection, graph construction, correlation calculation, and root cause detection. CloudRanger automatically constructs the relationship graph with PC algorithm and uses second-order RW for root cause detection. Wang et al. (2019) proposed an end-to-end anomaly detection and RCA system, GRANO, including detection layer, anomaly graph layer and Grano application layer. The anomaly layer constructs graph and uses Score Propagation to identify component-level root causes. The result will be presented at the application layer.

For the data centers, Arzani et al. (2016) proposed the NetPoirot system to locate faults and performance bottlenecks in complex services. NetPoirot uses lightweight non-intrusive continuous monitoring of TCP metrics only on client machines, the key to which is that different types of faults will cause TCP to react differently. NetPoirot uses DTs to classify TCP-based metrics. However, it can only identify the entity (network, client, remote service) that caused the failure, and cannot pinpoint the specific physical device. Cai et al. (2019) customized the RCA system for Alibaba data centers into two parts: offline modeling and online analysis. The offline modeling part finds the canonical patterns by data preprocessing and process discovery. The online analysis part finds the root cause by comparison based on the normative process found by offline models and the graph matching. Finally, root cause is confirmed by linking other information. To solve the incident routing problems in the data center, Gao et al. (2020) proposed an approach using per-team Scouts. The overall idea of RCA is to decompose event routing into events for each team. The team can build a scout for team event monitoring, feature extraction and prediction. Scout can reject events that team is not responsible for and get running events. For most transactions, Scout uses supervised learning like RF. For new or rare transactions, Scout uses unsupervised learning.

5 Discussion and future work

The research of storage using AI algorithms is not a new issue, as seen by the above survey, but the employment of learning models has emerged as the popular style. Nonetheless, we find a common phenomenon that the use of learning models in storage is cautious and demanding. Next, we describe the relationships between storage requirements and learning models to inspire future studies.

Deep learning models are leading the development in current AI technologies, and their most significant advantage is their capacity to extract the most relevant features based on the task at hand. For example, CNN is capable of retrieving locally sensitive information from images, which is why it can dramatically improve the performance of image processing. Nevertheless, CNN does not exactly match the data characteristics in storage service. Since there is no dedicated feature extraction model for storage service data, the storage industry has only benefited from the advancement of learning styles, such as unsupervised learning, self-learning, few-shot learning, zero-shot learning, etc., for small samples, in addition to conventional learning models (e.g., LSTM, RNN, and GRU models for temporal tasks). It is not sufficient for most storage services, albeit in the breakthrough of anomaly detection and failure prediction tasks with few training samples. Deep understanding of the data characteristics of storage services and designing deep learning models based on these characteristics is the key to making the storage industry truly enjoy AI technologies.

In fact, the state-of-the-art intelligent algorithm Vaswani et al. (2017) focuses on the accuracy of feature extraction and ignores efficiency and cost, resulting in a mismatch between the development of storage applications and intelligent algorithms. We believe that storage requirements for AI technologies mainly include efficiency, overhead, scalability and reliability.

- *efficiency* The learning model combines the advantages of offline learning and the efficiency of online running. The learning model can improve the quality of service through empirical data in storage circumstances where the empirical data does not change substantially or the change pattern is steady. In this way, the caching, cloud disk, and failure prediction tasks are all benefited.
- *overhead* Lightweight models such as Bayesian models, DTs, and lightBGM models are popular because of the restricted resources in the storage system. Deep learning models that require the use of GPUs are strictly limited to the only research on Near Data Processing. More Ridiculously, deep RL models that need to explore large search spaces seem to be useless. How-

ever, to the best of our knowledge, the advantage of the model relies on structure rather than arithmetic power Yu et al. (2021), as illustrated by the use of AE in data compression, i.e., inputting rational data and training data in the right ways can take advantage of models while reducing overhead. Deep RL models naturally match most decision-oriented tasks in storage scenarios, especially those with time-series information (e.g., cache, failure prediction, GC, etc.). The model should have an advantage in these studies due to its ability to directly pursue performance goals rather than relevant features must be found.

- *scalability* Because storage services are geared toward a wide range of upper-layer processes, the data distribution of stored data is frequently in flux, posing two major issues. First, the learning model must be updated on a frequent basis because only the latest empirical data is reliable. For some online storage services, such as cache, it is not available due to the generally long training time of the learning model. Luckily, incremental and fine-tune learning styles can be used as a compromise to reduce the update time. The other situation is very tricky. The new data distribution alters the model's input dimension, rendering the original model useless. In conventional image processing, this can be solved by resizing the image to the same size. However, for storage service data, this approach inevitably results in a significant loss, as the new data is often not linearly related to the existing data. This remains a pressing issue and is important for studies like the learned index.
- *reliability* Because learning models are probabilistic predictions, the application scenario must be fault-tolerant, such as cache, failure prediction, GC and range search tasks. But for exact search, lossless compression, and deduplication tasks, failure decisions are fatal. Further advancements in AI technologies are required to resolve this unsolvable conflict.

In brief, the application of AI technology aiming at solving storage problems is still in its infancy. The quantitative surge in research has not yet triggered a real breakthrough point, as AI technologies adapted to storage are still investigating. The most significant issue is understanding the properties of storage service data and breaking through storage resource limitations. We believe that more AI scholars will focus on the service data in storage to design targeted and intelligent algorithms. Meanwhile, we expect that the storage industry will continue to loosen resource constraints in the device update, embracing the AI algorithms.[4]

[4] http://idsm.wnlo.hust.edu.cn/index.htm.

Acknowledgements This work is supported by the National Natural Science Foundation of China No.61902135 and No.62172180, and the Joint Founds of ShanDong Natural Science Funds (Grant No.ZR2019LZH003). Thanks to Professor Hong Jiang for his advice on classification issues. Thanks to all students in *Intelligent Data Storage and Management Laboratory*[4].

References

C., C.A.R., Pâris, J., Vilalta, R., Cheng, A.M.K., Long, D.D.E.: Disk failure prediction in heterogeneous environments. In: International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS, pp. 1–7. IEEE, Seattle, WA, USA (2017)

Abu-Libdeh, H., Altinbüken, D., Beutel, A., Chi, E.H., Doshi, L., Kraska, T., Li, X., Ly, A., Olston, C.: Learned indexes for a google-scale disk-based database. CoRR **abs/2012.12501** (2020)

Agarwal, V., Bhattacharyya, C., Niranjan, T., Susarla, S.: Discovering rules from disk events for predicting hard drive failures. In: International Conference on Machine Learning and Applications, ICMLA, pp. 782–786. IEEE Computer Society, Miami Beach (2009)

Aken, D.V., Pavlo, A., Gordon, G.J., Zhang, B.: Automatic database management system tuning through large-scale machine learning. In: International Conference on Management of Data, SIGMOD, pp. 1009–1024. ACM, Chicago (2017)

Alter, J., Xue, J., Dimnaku, A., Smirni, E.: SSD failures in the field: symptoms, causes, and prediction models. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, pp. 75–17514. ACM, Denver (2019)

Anantharaman, P., Qiao, M., Jadav, D.: Large scale predictive analytics for hard disk remaining useful life estimation. In: International Congress on Big Data, BigData Congress, pp. 251–254. IEEE Computer Society, San Francisco (2018)

Arzani, B., Ciraci, S., Loo, B.T., Schuster, A., Outhred, G.: Taking the blame game out of data centers operations with netpoirot. In: SIGCOMM, pp. 440–453. ACM, Florianopolis, Brazil (2016)

Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E., Chabridon, S.: Predictive models of hard drive failures based on operational data. In: International Conference on Machine Learning and Applications, pp. 619–625. IEEE, Cancun, Mexico (2017)

Bagbaba, A.: Improving collective I/O performance with machine learning supported auto-tuning. In: International Parallel and Distributed Processing Symposium Workshops, IPDPSW, pp. 814–821. IEEE, New Orleans (2020)

Basak, S., Sengupta, S., Dubey, A.: Mechanisms for integrated feature normalization and remaining useful life estimation using lstms applied to hard-disks. In: International Conference on Smart Computing, SMARTCOMP, pp. 208–216. IEEE, Washington (2019)

Baseman, E., DeBardeleben, N., Ferreira, K.B., Levy, S., Raasch, S., Sridharan, V., Siddiqua, T., Guan, Q.: Improving DRAM fault characterization through machine learning. In: 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN Workshops, pp. 250–253. IEEE Computer Society, Toulouse, France (2016)

Behzad, B., Luu, H.V.T., Huchette, J., Byna, S.: Prabhat, Aydt, R.A., Koziol, Q., Snir, M.: Taming parallel I/O complexity with auto-tuning. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, pp. 68–16812. ACM, Denver (2013)

Behzad, B., Byna, S., Wild, S.M.: Prabhat, Snir, M.: Dynamic model-driven parallel I/O performance tuning. In: International

Conference on Cluster Computing, CLUSTER, pp. 184–193. IEEE Computer Society, Chicago (2015)
Bei, Z., Yu, Z., Zhang, H., Xiong, W., Xu, C., Eeckhout, L., Feng, S.: RFHOC: A random-forest approach to auto-tuning hadoop's configuration. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1470–1483 (2016)
Berger, D.S.: Towards lightweight and robust machine learning for CDN caching. In: Workshop on Hot Topics in Networks, HotNets, pp. 134–140. ACM, Redmond (2018)
Berger, D.S., Sitaraman, R.K., Harchol-Balter, M.: Adaptsize: Orchestrating the hot object memory cache in a content delivery network. In: Symposium on Networked Systems Design and Implementation, NSDI, pp. 483–498. USENIX Association, Boston (2017)
Beutel, A., Kraska, T., Chi, E., Dean, J., Polyzotis, N.: A machine learning approach to databases indexes. In: ML Systems Workshop, Annual Conference on Neural Information Processing Systems, NIPS, Long Beach, CA, USA (2017)
Bhatia, E., Chacon, G., Pugsley, S.H., Teran, E., Gratz, P.V., Jiménez, D.A.: Perceptron-based prefetch filtering. In: International Symposium on Computer Architecture, ISCA, pp. 1–13. ACM, Phoenix (2019)
Boixaderas, I., Zivanovic, D., Moré, S., Bartolome, J., Vicente, D., Casas, M., Carpenter, P.M., Radojkovic, P., Ayguadé, E.: Cost-aware prediction of uncorrected DRAM errors in the field. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, p. 61. IEEE/ACM, Virtual Event / Atlanta, Georgia, USA (2020)
Botezatu, M.M., Giurgiu, I., Bogojeska, J., Wiesmann, D.: Predicting disk replacement towards reliable data centers. In: International Conference on Knowledge Discovery and Data Mining, SIGKDD, pp. 39–48. ACM, San Francisco (2016)
Braam, P.: The lustre storage architecture. CoRR **abs/1903.01955** (2019)
Braun, P., Litz, H.: Understanding memory access patterns for prefetching. In: International Workshop on AI-assisted Design for Architecture (AIDArc), Held in Conjunction with ISCA, Phoenix, AZ, USA (2019)
Bux, W., Iliadis, I.: Performance of greedy garbage collection in flash-based solid-state drives. Perform. Eval. **67**(11), 1172–1186 (2010)
Cai, Z., Li, W., Zhu, W., Liu, L., Yang, B.: A real-time trace-level root-cause diagnosis system in alibaba datacenters. IEEE Access **7**, 142692–142702 (2019)
Cao, Z., Tarasov, V., Tiwari, S., Zadok, E.: Towards better understanding of black-box auto-tuning: A comparative analysis for storage systems. In: 2018 USENIX Annual Technical Conference, USENIX ATC 2018, , July 11-13, 2018, pp. 893–907. USENIX Association, Boston, MA, USA (2018)
Cao, S., Gao, Y., Gao, X., Chen, G.: Adam: An adaptive fine-grained scheme for distributed metadata management. In: International Conference on Parallel Processing, ICPP, pp. 37–13710. ACM, Kyoto (2019)
Chakrabortti, C., Litz, H.: Learning I/O access patterns to improve prefetching in ssds. In: Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track - European Conference, ECML PKDD, Lecture Notes in Computer Science, vol. 12460, pp. 427–443. Springer, Ghent (2020)
Chandramouli, B., Prasaad, G., Kossmann, D., Levandoski, J.J., Hunter, J., Barnett, M.: FASTER: A concurrent key-value store with in-place updates. In: International Conference on Management of Data, SIGMOD, pp. 275–290. ACM, Houston (2018)
Chaves, I.C., de Paula, M.R.P., de,: Moura Leite, L.G., Gomes, J.P.P., Machado, J.C.: Hard disk drive failure prediction method based on A bayesian network. In: International Joint Conference on Neural Networks, IJCNN, pp. 1–7. IEEE, Rio de Janeiro, Brazil (2018)
Chaves, I.C., de Paula, M.R.P., de,: Moura Leite, L.G., Queiroz, L.P., Gomes, J.P.P., Machado, J.C.: Banhfap: A bayesian network based failure prediction approach for hard disk drives. In: Brazilian Conference on Intelligent Systems, BRACIS, pp. 427–432. IEEE Computer Society, Recife, Brazil (2016)
Chen, L., Gao, Y., Li, X., Jensen, C.S., Chen, G.: Efficient metric indexing for similarity search and similarity joins. IEEE Trans. Knowl. Data Eng. **29**(3), 556–571 (2017)
Cheng, P., Lu, Y., Du,: Y., Chen, Z., Liu, Y.: Optimizing data placement on hierarchical storage architecture via machine learning. In: Network and Parallel Computing, NPC, Lecture Notes in Computer Science, vol. 11783, pp. 289–302. Springer, Hohhot, China (2019)
Cheng, W., Zhang, K., Chen, H., Jiang, G., Chen, Z., Wang, W.: Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations. In: International Conference on Knowledge Discovery and Data Mining, KDD, pp. 805–814. ACM, San Francisco (2016)
Cherubini, G., Jelitto, J., Venkatesan, V.: Cognitive storage for big data. Computer **49**(4), 43–51 (2016)
Chledowski, J., Polak, A., Szabucki, B., Zolna, K.T.: Robust learning-augmented caching: An experimental study. In: International Conference on Machine Learning, ICML, Proceedings of Machine Learning Research, vol. 139, pp. 1920–1930. PMLR, Virtual Event (2021)
Cohn, D.A., Singh, S.P.: Predicting lifetimes in dynamically allocated memory. In: Advances in Neural Information Processing Systems, NIPS, pp. 939–945. MIT Press, Denver (1996)
Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., Bianchini, R.: Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Symposium on Operating Systems Principles, SOSP, pp. 153–167. ACM, Shanghai (2017)
Dai, Y., Xu, Y., Ganesan, A., Alagappan, R., Kroth, B., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: From wisckey to bourbon: A learned index for log-structured merge trees. In: Symposium on Operating Systems Design and Implementation, OSDI, pp. 155–171. USENIX Association, Virtual Event (2020)
Davitkova, A., Milchevski, E., Michel, S.: The ml-index: A multidimensional, learned index for point, range, and nearest-neighbor queries. In: International Conference on Extending Database Technology, EDBT, pp. 407–410. OpenProceedings.org, Copenhagen, Denmark (2020)
Ding, J., Minhas, U.F., Yu, J., Wang, C., Do, J., Li, Y., Zhang, H., Chandramouli, B., Gehrke, J., Kossmann, D., Lomet, D.B., Kraska, T.: ALEX: an updatable adaptive learned index. In: International Conference on Management of Data, SIGMOD, pp. 969–984. ACM, Portland (2020)
dos Santos Lima, F.D.: Pereira, F.L.F., Chaves, I.C., Gomes, J.P.P., de Castro Machado, J.: Evaluation of recurrent neural networks for hard disk drives failure prediction. In: Brazilian Conference on Intelligent Systems, BRACIS, pp. 85–90. IEEE Computer Society, São Paulo (2018)
Featherstun, R.W., Fulp, E.W.: Using syslog message sequences for predicting disk failures. In: Large Installation System Administration Conference, LISA. USENIX Association, San Jose, CA, USA (2010)
Ferragina, P., Vinciguerra, G.: The pgm-index: a fully-dynamic compressed learned index with provable worst-case bounds. Proc. VLDB Endow. **13**(8), 1162–1175 (2020)
Fu, C., Cai, D.: EFANNA : An extremely fast approximate nearest neighbor search algorithm based on knn graph. CoRR **abs/1609.07228** (2016) 1609.07228

Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., Delimitrou, C.: Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In: International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS, pp. 19–33. ACM, Providence (2019)

Ganguly, S., Consul, A., Khan, A., Bussone, B., Richards, J., Miguel, A.: A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters. In: International Conference on Big Data Computing Service and Applications, pp. 105–116. IEEE Computer Society, Oxford, United Kingdom (2016)

Gao, J., Yaseen, N., MacDavid, R., Frujeri, F.V., Liu, V., Bianchini, R., Aditya, R., Wang, X., Lee, H., Maltz, D.A., Yu, M., Arzani, B.: Scouts: Improving the diagnosis process through domain-customized incident routing. In: Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM, pp. 253–269. ACM, Virtual Event, USA (2020)

Gao, X., Zha, S., Li, X., Yan, B., Jing, X., Li, J., Xu, J.: Incremental prediction model of disk failures based on the density metric of edge samples. IEEE Access **7**, 114285–114296 (2019)

Gao, Y., Gao, X., Chen, G.: Deephash: An end-to-end learning approach for metadata management in distributed file systems. In: International Conference on Parallel Processing, ICPP, pp. 36–13610. ACM, Kyoto (2019)

Gheisari, M., Movassagh, A.A., Qin, Y., Yong, J., Tao, X., Zhang, J., Shen, H.: NSSSD: A new semantic hierarchical storage for sensor data. In: International Conference on Computer Supported Cooperative Work in Design, CSCWD, pp. 174–179. IEEE, Nanchang (2016)

Giurgiu, I., Szabó, J., Wiesmann, D., Bird, J.: Predicting DRAM reliability in the field with machine learning. In: Zhu, X., Roy, I. (eds.) Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track, pp. 15–21. ACM, Las Vegas, NV, USA (2017)

Guan, Y., Zhang, X., Guo, Z.: CACA: learning-based content-aware cache admission for video content in edge caching. In: International Conference on Multimedia, MM, pp. 456–464. ACM, Nice (2019)

Hadian, A., Heinis, T.: Shift-table: A low-latency learned index for range queries using model correction. In: International Conference on Extending Database Technology, EDBT, pp. 253–264. OpenProceedings.org, Nicosia, Cyprus (2021)

Hadian, A., Heinis, T.: Considerations for handling updates in learned index structures. In: International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD, pp. 3–134. ACM, Amsterdam (2019)

Hamerly, G., Elkan, C.: Bayesian approaches to failure prediction for disk drives. In: International Conference on Machine Learning (ICML, pp. 202–209. Morgan Kaufmann, Williams College, Williamstown, MA, USA (2001)

Hashemi, M., Swersky, K., Smith, J.A., Ayers, G., Litz, H., Chang, J., Kozyrakis, C., Ranganathan, P.: Learning memory access patterns. In: International Conference on Machine Learning, ICML, Proceedings of Machine Learning Research, vol. 80, pp. 1924–1933. PMLR, Stockholmsmässan (2018)

Herodotou, H., Babu, S.: Profiling, what-if analysis, and cost-based optimization of mapreduce programs. Proc. VLDB Endow. **4**(11), 1111–1122 (2011)

Higuchi, S., Takemasa, J., Koizumi, Y., Tagami, A., Hasegawa, T.: Feasibility of longest prefix matching using learned index structures. SIGMETRICS Perform. Eval. Rev. **48**(4), 45–48 (2021)

Hu, G., Shao, J., Zhang, D., Yang, Y., Shen, H.T.: Preserving-ignoring transformation based index for approximate k nearest neighbor search. In: International Conference on Data Engineering, ICDE, pp. 91–94. IEEE Computer Society, San Diego (2017)

Hua, Y., Jiang, H., Zhu, Y., Feng, D., Tian, L.: Smartstore: a new metadata organization paradigm with semantic-awareness for next-generation file systems. In: Conference on High Performance Computing, SC. ACM, Portland, Oregon, USA (2009)

Hua, Y., Jiang, H., Feng, D.: FAST: near real-time searchable data analytics for the cloud. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, pp. 754–765. IEEE Computer Society, New Orleans (2014)

Huang, S., Fu, S., Zhang, Q., Shi, W.: Characterizing disk failures with quantified disk degradation signatures: An early experience. In: International Symposium on Workload Characterization, IISWC, pp. 150–159. IEEE Computer Society, Atlanta (2015)

Indyk, P., Motwani, R., Raghavan, P., Vempala, S.S.: Locality-preserving hashing in multidimensional spaces. In: Symposium on the Theory of Computing, STOC, pp. 618–625. ACM, El Paso (1997)

Jain, R., Panda, P.R., Subramoney, S.: A coordinated multi-agent reinforcement learning approach to multi-level cache co-partitioning. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 800–805. IEEE, Lausanne (2017)

Ji, X., Ma, Y., Ma, R., Li, P., Ma, J., Wang, G., Liu, X., Li, Z.: A proactive fault tolerance scheme for large scale storage systems. In: International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP, Lecture Notes in Computer Science, vol. 9530, pp. 337–350. Springer, Zhangjiajie (2015)

Jiang, T., Zeng, J., Zhou, K., Huang, P., Yang, T.: Lifelong disk failure prediction via gan-based anomaly detection. In: International Conference on Computer Design, ICCD, pp. 199–207. IEEE, Abu Dhabi (2019)

Jiang, T., Huang, P., Zhou, K.: Scrub unleveling: Achieving high data reliability at low scrubbing cost. In: Teich, J., Fummi, F. (eds.) Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 1403–1408. IEEE, Florence (2019)

Jiménez, D.A., Teran, E.: Multiperspective reuse prediction. In: International Symposium on Microarchitecture, MICRO, pp. 436–448. ACM, Cambridge (2017)

Jin, X., Agun, D., Yang, T., Wu, Q., Shen, Y., Zhao, S.: Hybrid indexing for versioned document search with cluster-based retrieval. In: International Conference on Information and Knowledge Management, CIKM, pp. 377–386. ACM, Indianapolis (2016)

Kang, W., Yoo, S.: Dynamic management of key states for reinforcement learning-assisted garbage collection to reduce long tail latency in SSD. In: Design Automation Conference, DAC, pp. 8–186. ACM, San Francisco (2018)

Kang, W., Yoo, S.: q -value prediction for reinforcement learning assisted garbage collection to reduce long tail latency in SSD. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst **39**(10), 2240–2253 (2020)

Kang, W., Shin, D., Yoo, S.: Reinforcement learning-assisted garbage collection to mitigate long-tail latency in SSD. ACM Trans. Embed. Comput. Syst. **16**(5s), 134–113420 (2017)

Kim, M., Lee, S.: Reducing tail latency of dnn-based recommender systems using in-storage processing. In: SIGOPS Asia-Pacific Workshop on Systems, pp. 90–97. ACM, Tsukuba, Japan (2020)

Kim, J.: An ftl-aware host system alleviating severe long latency of NAND flash-based storage. In: International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, pp. 189–194. IEEE, Houston (2021)

Kim, M., Sumbaly, R., Shah, S.: Root cause detection in a service-oriented architecture. In: International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS, pp. 93–104. ACM, Pittsburgh (2013)

Kim, Y., More, A., Shriver, E., Rosing, T.: Application performance prediction and optimization under cache allocation technology.

In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 1285–1288. IEEE, Florence (2019)
Kipf, A., Marcus, R., van Renen, A., Stoian, M., Kemper, A., Kraska, T., Neumann, T.: Radixspline: a single-pass learned index. In: Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD, pp. 5–155. ACM, Portland (2020)
Kirilin, V., Sundarrajan, A., Gorinsky, S., Sitaraman, R.K.: Rl-cache: Learning-based cache admission for content delivery. In: Proceedings of the 2019 Workshop on Network Meets AI & ML, NetAI@SIGCOMM 2019, pp. 57–63. ACM, Beijing, China (2019)
Kirilin, V., Sundarrajan, A., Gorinsky, S., Sitaraman, R.K.: Rl-cache: Learning-based cache admission for content delivery. IEEE J. Sel. Areas Commun. **38**(10), 2372–2385 (2020)
Klein, K., Kriege, N.M., Mutzel, P.: Ct-index: Fingerprint-based graph indexing combining cycles and trees. In: International Conference on Data Engineering, ICDE, pp. 1115–1126. IEEE Computer Society, Hannover (2011)
Kraska, T., Alizadeh, M., Beutel, A., Chi, E.H., Kristo, A., Leclerc, G., Madden, S., Mao, H., Nathan, V.: Sagedb: A learned database system. In: Biennial Conference on Innovative Data Systems Research, CIDR. www.cidrdb.org, Asilomar, CA, USA (2019)
Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The case for learned index structures. In: International Conference on Management of Data, SIGMOD, pp. 489–504. ACM, Houston (2018)
Leuoth, S., Benn, W.: A self-adaptive insert strategy for content-based multidimensional database storage. In: GI-Workshop on Foundations of Databases (Grundlagen Von Datenbanken). Preprints aus dem Institut für Informatik, vol. CS-02-09, pp. 75–79. Universität Rostock, Mecklenburg-Vorpommern, Germany (2009)
Leuoth, S., Benn, W.: Towards SISI - a self adaptive insert strategy for the intelligent cluster index (icix). In: Machine Learning and Data Mining in Pattern Recognition, MLDM, pp. 141–155. ibai Publishing, Leipzig, Germany (2009)
Li, P., Hua, Y., Zuo, P., Jia, J.: A scalable learned index scheme in storage systems. CoRR **abs/1905.06256** (2019) 1905.06256
Li, J., Ji, X., Jia, Y., Zhu, B., Wang, G., Li, Z., Liu, X.: Hard drive failure prediction using classification and regression trees. In: International Conference on Dependable Systems and Networks, DSN, pp. 383–394. IEEE Computer Society, Atlanta (2014)
Li, J., Stones, R.J., Wang, G., Li, Z., Liu, X., Xiao, K.: Being accurate is not enough: New metrics for disk failure prediction. In: Symposium on Reliable Distributed Systems, SRDS, pp. 71–80. IEEE Computer Society, Budapest (2016)
Li, Y., Chang, K., Bel, O., Miller, E.L., Long, D.D.E.: CAPES: unsupervised storage performance tuning using neural network-based deep reinforcement learning. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, pp. 42–14214. ACM, Denver (2017)
Li, J., Stones, R.J., Wang, G., Liu, X., Li, Z., Xu, M.: Hard drive failure prediction using decision trees. Reliab. Eng. Syst. Saf. **164**, 55–65 (2017)
Li, Z.L., Liang, C.M., He, W., Zhu, L., Dai, W., Jiang, J., Sun, G.: Metis: Robustly tuning tail latencies of cloud systems. In: Annual Technical Conference, USENIX ATC, pp. 981–992. USENIX Association, Boston (2018)
Li, G., Zhou, X., Li, S., Gao, B.: Qtune: A query-aware database tuning system with deep reinforcement learning. Proc. VLDB Endow. **12**(12), 2118–2130 (2019)
Li, P., Lu, H., Zheng, Q., Yang, L., Pan, G.: LISA: A learned index structure for spatial data. In: Maier, D., Pottinger, R., Doan, A., Tan, W., Alawini, A., Ngo, H.Q. (eds.) International Conference on Management of Data, SIGMOD, pp. 2119–2133. ACM, Portland (2020)
Li, C., Wang, Y., Liu, C., Liang, S., Li, H., Li, X.: GLIST: towards in-storage graph learning. In: Annual Technical Conference, ATC, pp. 225–238. USENIX Association, Ho Chi Minh City (2021)
Liang, S., Wang, Y., Lu, Y., Yang, Z., Li, H., Li, X.: Cognitive SSD: A deep learning engine for in-storage data retrieval. In: Annual Technical Conference, ATC, pp. 395–410. USENIX Association, Renton (2019)
Lin, W., Ma, M., Pan, D., Wang, P.: Facgraph: Frequent anomaly correlation graph mining for root cause diagnose in micro-service architecture. In: International Performance Computing and Communications Conference, IPCCC, pp. 1–8. IEEE, Orlando (2018)
Liu, J., Wang, R., Gao, X., Yang, X., Chen, G.: Anglecut: A ring-based hashing scheme for distributed metadata management. In: Database Systems for Advanced Applications, DASFAA, Lecture Notes in Computer Science, vol. 10177, pp. 71–86. Springer, Suzhou (2017)
Liu, Y., Song, J., Zhou, K., Yan, L., Liu, L., Zou, F., Shao, L.: Deep self-taught hashing for image retrieval. IEEE Trans. Cybern. **49**(6), 2229–2241 (2019)
Liu, P., Chen, Y., Nie, X., Zhu, J., Zhang, S., Sui, K., Zhang, M., Pei, D.: Fluxrank: A widely-deployable framework to automatically localizing root cause machines for software service failure mitigation. In: International Symposium on Software Reliability Engineering, ISSRE, pp. 35–46. IEEE, Berlin (2019)
Liu, Y., Jiang, H., Wang, Y., Zhou, K., Liu, Y., Liu, L.: Content sifting storage: Achieving fast read for large-scale image dataset analysis. In: Design Automation Conference, DAC, pp. 1–6. IEEE, San Francisco (2020)
Liu, Y., Wang, Y., Song, J., Guo, C., Zhou, K., Xiao, Z.: Deep self-taught graph embedding hashing with pseudo labels for image retrieval. In: International Conference on Multimedia and Expo, ICME, pp. 1–6. IEEE, London (2020)
Liu, W., Cui, J., Liu, J., Yang, L.T.: Mlcache: A space-efficient cache scheme based on reuse distance and machine learning for nvme ssds. In: International Conference On Computer Aided Design, ICCAD, pp. 58–1589. IEEE, San Diego (2020)
Liu, P., Xu, H., Ouyang, Q., Jiao, R., Chen, Z., Zhang, S., Yang, J., Mo, L., Zeng, J., Xue, W., Pei, D.: Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks. In: International Symposium on Software Reliability Engineering, ISSRE, pp. 48–58. IEEE, Coimbra (2020)
Lu, S., Luo, B., Patel, T., Yao, Y., Tiwari, D., Shi, W.: Making disk failure predictions smarter! In: Conference on File and Storage Technologies, FAST, pp. 151–167. USENIX Association, Santa Clara, CA, USA (2020)
Luaces, D., Viqueira, J.R.R., Pena, T.F., Cotos, J.M.: Leveraging bitmap indexing for subgraph searching. In: International Conference on Extending Database Technology, EDBT, pp. 49–60. OpenProceedings.org, Lisbon, Portugal (2019)
Luo, C., Zhao, P., Qiao, B., Wu, Y., Zhang, H., Wu, W., Lu, W., Dang, Y., Rajmohan, S., Lin, Q., Zhang, D.: NTAM: neighborhood-temporal attention model for disk failure prediction in cloud platforms. In: The Web Conference, WWW, pp. 1181–1191. ACM / IW3C2, Virtual Event / Ljubljana, Slovenia (2021)
Luo, C., Lou, J., Lin, Q., Fu, Q., Ding, R., Zhang, D., Wang, Z.: Correlating events with time series for incident diagnosis. In: International Conference on Knowledge Discovery and Data Mining, KDD, pp. 1583–1592. ACM, New York (2014)
Luo, Q., Fang, X., Sun, Y., Ai, J., Yang, C.: Self-learning hot data prediction: Where echo state network meets NAND flash memories. IEEE Trans. Circuits Syst. I Regul. Pap. **67**(I(3)), 939–950 (2020)
Lykouris, T., Vassilvitskii, S.: Competitive caching with machine learned advice. In: International Conference on Machine Learning, ICML, Proceedings of Machine Learning Research, vol. 80, pp. 3302–3311. PMLR, Stockholmsmässan (2018)

Ma, M., Xu, J., Wang, Y., Chen, P., Zhang, Z., Wang, P.: Automap: Diagnose your microservice-based web applications automatically. In: The Web Conference, WWW, pp. 246–258. ACM / IW3C2, Taipei, Taiwan (2020)

Ma, M., Zhang, S., Chen, J., Xu, J., Li, H., Lin, Y., Nie, X., Zhou, B., Wang, Y., Pei, D.: Jump-starting multivariate time series anomaly detection for online service systems. In: Annual Technical Conference, ATC, pp. 413–426. USENIX Association, Virtual Event (2021)

Ma, M., Lin, W., Pan, D., Wang, P.: Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications. In: International Conference on Web Services, ICWS, pp. 60–67. IEEE, Milan (2019)

Maas, M., Andersen, D.G., Isard, M., Javanmard, M.M., McKinley, K.S., Raffel, C.: Learning-based memory allocation for C++ server workloads. In: Architectural Support for Programming Languages and Operating Systems, ASPLOS, pp. 541–556. ACM, Lausanne (2020)

Mahdisoltani, F., Stefanovici, I.A., Schroeder, B.: Proactive error prediction to improve storage system reliability. In: Silva, D.D., Ford, B. (eds.) Annual Technical Conference, ATC, pp. 391–402. USENIX Association, Santa Clara (2017)

Mailthody, V.S., Qureshi, Z., Liang, W., Feng, Z., Gonzalo, S.G.D., Li, Y., Franke, H., Xiong, J., Huang, J., Hwu, W.: Deepstore: In-storage acceleration for intelligent queries. In: International Symposium on Microarchitecture, MICRO, pp. 224–238. ACM, Columbus (2019)

Marcus, R., Kipf, A., van Renen, A., Stoian, M., Misra, S., Kemper, A., Neumann, T., Kraska, T.: Benchmarking learned indexes. Proc. VLDB Endow. **14**(1), 1–13 (2020)

Meng, Y., Zhang, S., Sun, Y., Zhang, R., Hu, Z., Zhang, Y., Jia, C., Wang, Z., Pei, D.: Localizing failure root causes in a microservice through causality inference. In: International Symposium on Quality of Service, IWQoS, pp. 1–10. IEEE, Hangzhou (2020)

Mishra, M., Singhal, R.: RUSLI: real-time updatable spline learned index. In: Bordawekar, R., Amsterdamer, Y., Shmueli, O., Tatbul, N. (eds.) Workshop in Exploiting AI Techniques for Data Management, aiDM, pp. 1–8. ACM, Virtual Event, China (2021)

Monjalet, F., Leibovici, T.: Predicting file lifetimes with machine learning. In: High Performance Computing - ISC High Performance 2019 International Workshops, Lecture Notes in Computer Science, vol. 11887, pp. 288–299. Springer, Frankfurt (2019)

Mukhanov, L., Tovletoglou, K., Vandierendonck, H., Nikolopoulos, D.S., Karakonstantis, G.: Workload-aware DRAM error prediction using machine learning. In: International Symposium on Workload Characterization, IISWC, pp. 106–118. IEEE, Orlando (2019)

Murray, J.F., Hughes, G.F., Kreutz-Delgado, K.: Hard drive failure prediction using non-parametric statistical methods. In: ICANN/ICONIP (2003)

Murray, J.F., Hughes, G.F., Kreutz-Delgado, K.: Machine learning methods for predicting failures in hard drives: a multiple-instance application. J. Mach. Learn. Res. **6**, 783–816 (2005)

Narayanan, I., Wang, D., Jeon, M., Sharma, B., Caulfield, L., Sivasubramaniam, A., Cutler, B., Liu, J., Khessib, B.M., Vaid, K.: SSD failures in datacenters: What, when and why? In: SIGMETRICS, pp. 407–408. ACM, Antibes Juan-Les-Pins, France (2016)

Narayanan, A., Verma, S., Ramadan, E., Babaie, P., Zhang, Z.: Deepcache: A deep learning based framework for content caching. In: Workshop on Network Meets AI & ML, NetAI@SIGCOMM, pp. 48–53. ACM, Budapest (2018)

Nathan, V., Ding, J., Alizadeh, M., Kraska, T.: Learning multi-dimensional indexes. In: International Conference on Management of Data, SIGMOD, pp. 985–1000. ACM, Portland (2020)

Neubert, R., Görlitz, O., Benn, W.: Towards content-related indexing in databases. In: Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Informatik Aktuell, pp. 305–321. Springer, GI-Fachtagung (2001)

Ni, J., Cheng, W., Zhang, K., Song, D., Yan, T., Chen, H., Zhang, X.: Ranking causal anomalies by modeling local propagations on networked systems. In: International Conference on Data Mining, ICDM, pp. 1003–1008. IEEE Computer Society, New Orleans (2017)

Pang, S., Jia, Y., Stones, R.J., Wang, G., Liu, X.: A combined bayesian network method for predicting drive failure times from SMART attributes. In: International Joint Conference on Neural Networks, IJCNN, pp. 4850–4856. IEEE, Vancouver (2016)

Park, N., Ahmad, I., Lilja, D.J.: Romano: autonomous storage management using performance prediction in multi-tenant datacenters. In: Symposium on Cloud Computing, SOCC, p. 21. ACM, San Jose, CA, USA (2012)

Park, J.K., Kim, J.: A method for reducing garbage collection overhead of SSD using machine learning algorithms. In: International Conference on Information and Communication Technology Convergence, ICTC, pp. 775–777. IEEE, Jeju Island (2017)

Park, S., Kim, D., Bang, K., Lee, H., Yoo, S., Chung, E.: An adaptive idle-time exploiting method for low latency NAND flash-based storage devices. IEEE Trans. Comput. **63**(5), 1085–1096 (2014)

Paschos, G.S., Destounis, A., Vigneri, L., Iosifidis, G.: Learning to cache with no regrets. In: Conference on Computer Communications, INFOCOM, pp. 235–243. IEEE, Paris (2019)

Peled, L., Mannor, S., Weiser, U.C., Etsion, Y.: Semantic locality and context-based prefetching using reinforcement learning. In: International Symposium on Computer Architecture, ISCA, pp. 285–297. ACM, Portland (2015)

Peled, L., Weiser, U.C., Etsion, Y.: A neural network prefetcher for arbitrary memory access patterns. ACM Trans. Archit. Code Optim. **16**(4), 37–13727 (2020)

Pereira, F.L.F., dos,: Santos Lima, F.D., de Moura Leite, L.G., Gomes, J.P.P., de Castro Machado, J.: Transfer learning for bayesian networks with application on hard disk drives failure prediction. In: Brazilian Conference on Intelligent Systems, BRACIS, pp. 228–233. IEEE Computer Society, Uberlândia, Brazil (2017)

Pereira, F., Teixeira, D., Gomes, J.P., Machado, J.C.: Evaluating one-class classifiers for fault detection in hard disk drives. In: Brazilian Conference on Intelligent Systems, BRACIS, pp. 586–591. IEEE, Salvador (2019)

Pham, C., Wang, L., Tak, B., Baset, S., Tang, C., Kalbarczyk, Z.T., Iyer, R.K.: Failure diagnosis for distributed systems using targeted fault injection. IEEE Trans. Parallel Distrib. Syst. **28**(2), 503–516 (2017)

Pitakrat, T., van Hoorn, A., Grunske, L.: A comparison of machine learning algorithms for proactive hard disk drive failure detection. In: International ACM Sigsoft Symposium on Architecting Critical Systems, ISARCS, pp. 1–10. ACM, Vancouver (2013)

Poppe, O., Amuneke, T., Banda, D., De, A., Green, A., Knoertzer, M., Nosakhare, E., Rajendran, K., Shankargouda, D., Wang, M., Au, A., Curino, C., Guo, Q., Jindal, A., Kalhan, A., Oslake, M., Parchani, S., Ramani, V., Sellappan, R., Sen, S., Shrotri, S., Srinivasan, S., Xia, P., Xu, S., Yang, A., Zhu, Y.: Seagull: An infrastructure for load prediction and optimized resource allocation. Proc. VLDB Endow. **14**(2), 154–162 (2020)

Prats, D.B., Portella, F.A., Costa, C.H.A., Berral, J.L.: You only run once: Spark auto-tuning from a single run. IEEE Trans. Netw. Serv. Manag. **17**(4), 2039–2051 (2020)

Qiu, J., Du, Q., Yin, K., Zhang, S.-L., Qian, C.: A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications. Appl. Sci. **10**(6), 2166 (2020)

Queiroz, L.P., Gomes, J.P.P., Rodrigues, F.C.M., Brito, F.T., Chaves, I.C., de Moura Leite, L.G., Machado, J.C.: Fault detection in

hard disk drives based on a semi parametric model and statistical estimators. New Gen. Comput **36**(1), 5–19 (2018)
Rahman, S., Burtscher, M., Zong, Z., Qasem, A.: Maximizing hardware prefetch effectiveness with machine learning. In: International Conference on High Performance Computing and Communications, HPCC, International Symposium on Cyberspace Safety and Security, CSS, International Conference on Embedded Software and Systems, ICESS, pp. 383–389. IEEE, New York (2015)
Ravandi, B., Papapanagiotou, I.: A self-organized resource provisioning for cloud block storage. Future Gen. Comput. Syst. **89**, 765–776 (2018)
Ren, J., Chen, X., Liu, D., Tan, Y., Duan, M., Li, R., Liang, L.: A machine learning assisted data placement mechanism for hybrid storage systems. J. Syst. Archit. **120**, 102295 (2021)
Rodriguez, L.V., Yusuf, F.B., Lyons, S., Paz, E., Rangaswami, R., Liu, J., Zhao, M., Narasimhan, G.: Learning cache replacement with CACHEUS. In: Conference on File and Storage Technologies, FAST, pp. 341–354. USENIX Association, Virtual Event (2021)
Sethumurugan, S., Yin, J., Sartori, J.: Designing a cost-effective cache replacement policy using machine learning. In: International Symposium on High-Performance Computer Architecture, HPCA, pp. 291–303. IEEE, Seoul (2021)
Shen, J., Wan, J., Lim, S., Yu, L.: Random-forest-based failure prediction for hard disk drives. Int. J. Distrib. Sens. Netw. **14**(11) (2018)
Shi, H., Arumugam, R.V., Foh, C.H., Khaing, K.K.: Optimal disk storage allocation for multi-tier storage system. In: 2012 Digest APMRC, pp. 1–7 (2012)
Shi, W., Cheng, P., Zhu, C., Chen, Z.: An intelligent data placement strategy for hierarchical storage systems. In: International Conference on Computer and Communications (ICCC), pp. 2023–2027 (2020). IEEE
Shi, Z., Jain, A., Swersky, K., Hashemi, M., Ranganathan, P., Lin, C.: A hierarchical neural model of data prefetching. In: International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS, pp. 861–873. ACM, Virtual Event, USA (2021)
Shi, Z., Huang, X., Jain, A., Lin, C.: Applying deep learning to the cache replacement problem. In: International Symposium on Microarchitecture, MICRO, pp. 413–425. ACM, Columbus (2019)
Song, Z., Berger, D.S., Li, K., Lloyd, W.: Learning relaxed belady for content distribution network caching. In: Symposium on Networked Systems Design and Implementation, NSDI, pp. 529–544. USENIX Association, Santa Clara (2020)
Spector, B., Kipf, A., Vaidya, K., Wang, C., Minhas, U.F., Kraska, T.: Bounding the last mile: Efficient learned string indexing. CoRR **abs/2111.14905** (2021)
Srivastava, A., Lazaris, A., Brooks, B., Kannan, R., Prasanna, V.K.: Predicting memory accesses: the road to compact ml-driven prefetcher. In: International Symposium on Memory Systems, MEMSYS, pp. 461–470. ACM, Washington (2019)
Stoian, M., Kipf, A., Marcus, R., Kraska, T.: Plex: Towards practical learned indexing. (2021) arXiv preprint arXiv:2108.05117
Subedi, P., Davis, P.E., Duan, S., Klasky, S., Kolla, H., Parashar, M.: Stacker: an autonomic data movement engine for extreme-scale data staging-based in-situ workflows. In: International Conference for High Performance Computing, Networking, Storage, and Analysis, SC, pp. 73–17311. IEEE / ACM, Dallas (2018)
Sun, X., Chakrabarty, K., Huang, R., Chen, Y., Zhao, B., Cao, H., Han, Y., Liang, X., Jiang, L.: System-level hardware failure prediction using deep learning. In: Design Automation Conference, DAC, p. 20. ACM, Las Vegas, NV, USA (2019)
Sun, Q., Jin, T., Romanus, M., Bui, H., Zhang, F., Yu, H., Kolla, H., Klasky, S., Chen, J., Parashar, M.: Adaptive data placement for staging-based coupled scientific workflows. In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC, pp. 65–16512. ACM, Austin (2015)
Sun, Y., Zhao, Y., Su, Y., Liu, D., Nie, X., Meng, Y., Cheng, S., Pei, D., Zhang, S., Qu, X., Guo, X.: Hotspot: Anomaly localization for additive kpis with multi-dimensional attributes. IEEE Access **6**, 10909–10923 (2018)
Tang, C., Dong, Z., Wang, M., Wang, Z., Chen, H.: Learned indexes for dynamic workloads. CoRR **abs/1902.00655** (2019) 1902.00655
Tang, C., Wang, Y., Dong, Z., Hu, G., Wang, Z., Wang, M., Chen, H.: Xindex: a scalable learned index for multicore data storage. In: Gupta, R., Shen, X. (eds.) SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP, pp. 308–320. ACM, San Diego (2020)
Teran, E., Wang, Z., Jiménez, D.A.: Perceptron learning for reuse prediction. In: International Symposium on Microarchitecture, MICRO, pp. 2–1212. IEEE Computer Society, Taipei (2016)
Thomas, L., Gougeaud, S., Rubini, S., Deniel, P., Boukhobza, J.: Predicting file lifetimes for data placement in multi-tiered storage systems for HPC. ACM SIGOPS Oper. Syst. Rev. **55**(1), 99–107 (2021)
Tomes, E., Rush, E.N., Altiparmak, N.: Towards adaptive parallel storage systems. IEEE Trans. Comput. **67**(12), 1840–1848 (2018)
Tsai, L., Franke, H., Li, C., Liao, W.: Learning-based memory allocation optimization for delay-sensitive big data processing. IEEE Trans. Parallel Distrib.Syst. **29**(6), 1332–1341 (2018)
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Annual Conference on Neural Information Processing Systems, NIPS, Long Beach, CA, USA, pp. 5998–6008 (2017)
Vietri, G., Rodriguez, L.V., Martinez, W.A., Lyons, S., Liu, J., Rangaswami, R., Zhao, M., Narasimhan, G.: Driving cache replacement with ml-based lecar. In: Workshop on Hot Topics in Storage and File Systems, HotStorage. USENIX Association, Boston, MA, USA (2018)
Wang, H., He, H., Alizadeh, M., Mao, H.: Learning caching policies with subsampling. In: NeurIPS Machine Learning for Systems Workshop (2019)
Wang, X., Li, Y., Chen, Y., Wang, S., Du,: Y., He, C., Zhang, Y., Chen, P., Li, X., Song, W., Xu, Q., Jiang, L.: On workload-aware DRAM failure prediction in large-scale data centers. In: VLSI Test Symposium, VTS, pp. 1–6. IEEE, San Diego, CA, USA (2021)
Wang, P., Xu, J., Ma, M., Lin, W., Pan, D., Wang, Y., Chen, P.: Cloudranger: Root cause identification for cloud native systems. In: International Symposium on Cluster, Cloud and Grid Computing, CCGRID, pp. 492–502. IEEE Computer Society, Washington (2018)
Wang, H., Yi, X., Huang, P., Cheng, B., Zhou, K.: Efficient SSD caching by avoiding unnecessary writes using machine learning. In: International Conference on Parallel Processing, ICPP, pp. 82–18210. ACM, Eugene (2018)
Wang, H., Nguyen, P., Li, J., Köprü, S., Zhang, G., Katariya, S., Ben-Romdhane, S.: GRANO: interactive graph-based root cause analysis for cloud-native distributed data platform. Proc. VLDB Endow. **12**(12), 1942–1945 (2019)
Wang, H., Yang, Y., Huang, P., Zhang, Y., Zhou, K., Tao, M., Cheng, B.: S-CDA: A smart cloud disk allocation approach in cloud block storage system. In: Design Automation Conference, DAC, pp. 1–6. IEEE, San Francisco (2020)
Wang, H., Zhang, J., Huang, P., Yi, X., Cheng, B., Zhou, K.: Cache what you need to cache: Reducing write traffic in cloud cache via "one-time-access-exclusion'' policy. ACM Trans. Storage **16**(3), 18–11824 (2020)
Wang, Y., Tang, C., Wang, Z., Chen, H.: Sindex: a scalable learned index for string keys. In: SIGOPS Asia-Pacific Workshop on Systems, APSys, pp. 17–24. ACM, Tsukuba (2020)

Wang, Y., Song, J., Zhou, K., Liu, Y.: Unsupervised deep hashing with node representation for image retrieval. Pattern Recognit. **112**, 107785 (2021)
Wei, X., Chen, R., Chen, H.: Fast rdma-based ordered key-value store using remote learned cache. In: Symposium on Operating Systems Design and Implementation, OSDI, pp. 117–135. USENIX Association, Virtual Event (2020)
Wei, X., Chen, R., Chen, H., Zang, B.: Xstore: Fast rdma-based ordered key-value store using remote learned cache. ACM Trans. Storage **17**(3), 18–11832 (2021)
Weng, J., Wang, J.H., Yang, J., Yang, Y.: Root cause analysis of anomalies of multitier services in public clouds. IEEE/ACM Trans. Netw. **26**(4), 1646–1659 (2018)
Wilkening, M., Gupta, U., Hsia, S., Trippel, C., Wu, C., Brooks, D., Wei, G.: Recssd: near data processing for solid state drive based recommendation inference. In: International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS, pp. 717–729. ACM, Virtual Event, USA (2021)
Wu, Z., Xu, H., Pang, G., Yu, F., Wang, Y., Jian, S., Wang, Y.: DRAM failure prediction in aiops: Empiricalevaluation, challenges and opportunities. CoRR **abs/2104.15052** (2021)
Wu, C., Ji, C., Xue, C.J.: Reinforcement learning based background segment cleaning for log-structured file system on mobile devices. In: International Conference on Embedded Software and Systems, ICESS, pp. 1–8. IEEE, Las Vegas (2019)
Wu, L., Tordsson, J., Elmroth, E., Kao, O.: Microrca: Root cause localization of performance issues in microservices. In: Network Operations and Management Symposium, NOMS, pp. 1–9. IEEE, Budapest (2020)
Wu, J., Zhang, Y., Chen, S., Chen, Y., Wang, J., Xing, C.: Updatable learned index with precise positions. Proc. VLDB Endow. **14**(8), 1276–1288 (2021)
Xiao, J., Xiong, Z., Wu, S., Yi, Y., Jin, H., Hu, K.: Disk failure prediction in data centers via online learning. In: International Conference on Parallel Processing, ICPP, pp. 35–13510. ACM, Eugene (2018)
Xie, D., Chandramouli, B., Li, Y., Kossmann, D.: Fishstore: Faster ingestion with subset hashing. In: International Conference on Management of Data, SIGMOD, pp. 1711–1728. ACM, Amsterdam (2019)
Xu, C., Wang, G., Liu, X., Guo, D., Liu, T.: Health status assessment and failure prediction for hard drives with recurrent neural networks. IEEE Trans. Comput. **65**(11), 3502–3508 (2016)
Xu, Y., Sui, K., Yao, R., Zhang, H., Lin, Q., Dang, Y., Li, P., Jiang, K., Zhang, W., Lou, J., Chintalapati, M., Zhang, D.: Improving service availability of cloud systems by predicting disk error. In: Annual Technical Conference, ATC, pp. 481–494. USENIX Association, Boston (2018)
Xu, R., Jin, X., Tao, L., Guo, S., Xiang, Z., Tian, T.: An efficient resource-optimized learning prefetcher for solid state drives. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 273–276. IEEE, Dresden (2018)
Xu, F., Han, S., Lee, P.P.C., Liu, Y., He, C., Liu, J.: General feature selection for failure prediction in large-scale SSD deployment. In: International Conference on Dependable Systems and Networks, DSN, pp. 263–270. IEEE, Taipei (2021)
Yan, G., Li, J.: Rl-bélády: A unified learning framework for content caching. In: Chen, C.W., Cucchiara, R., Hua, X., Qi, G., Ricci, E., Zhang, Z., Zimmermann, R. (eds.) International Conference on Multimedia, MM, pp. 1009–1017. ACM, Virtual Event/Seattle (2020)
Yang, P., Xue, N., Zhang, Y., Zhou, Y., Sun, L., Chen, W., Chen, Z., Xia, W., Li, J., Kwon, K.: Reducing garbage collection overhead in SSD based on workload prediction. In: Workshop on Hot Topics in Storage and File Systems, HotStorage. USENIX Association, Renton, WA, USA (2019)
Yang, W., Hu, D., Liu, Y., Wang, S., Jiang, T.: Hard drive failure prediction using big data. In: Symposium on Reliable Distributed Systems Workshop, SRDS, pp. 13–18. IEEE Computer Society, Montreal (2015)
Yang, Y., Misra, V., Rubenstein, D.: On the optimality of greedy garbage collection for ssds. SIGMETRICS Perform. Eval. Rev. **43**(2), 63–65 (2015)
Yang, L., Wang, F., Tan, Z., Feng, D., Qian, J., Tu, S.: ARS: reducing F2FS fragmentation for smartphones using decision trees. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 1061–1066. IEEE, Grenoble (2020)
Yang, L., Tan, Z., Wang, F., Tu, S., Shao, J.: M2H: optimizing F2FS via multi-log delayed writing and modified segment cleaning based on dynamically identified hotness. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 808–811. IEEE, Grenoble (2021)
Ye, J., Li, Z., Wang, Z., Zheng, Z., Hu, H., Zhu, W.: Joint cache size scaling and replacement adaptation for small content providers. In: Conference on Computer Communications, INFOCOM, pp. 1–10. IEEE, Vancouver (2021)
Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. CoRR **abs/2111.11418** (2021)
Yuan, D., Yang, Y., Liu, X., Chen, J.: A data placement strategy in scientific cloud workflows. Fut. Gen. Comput. Syst. **26**(8), 1200–1214 (2010)
Zeng, Y., Guo, X.: Long short term memory based hardware prefetcher: a case study. In: International Symposium on Memory Systems, MEMSYS, pp. 305–311. ACM, Alexandria (2017)
Zhang, M., He, Y.: Zoom: Multi-view vector search for optimizing accuracy, latency and memory. Technical Report MSR-TR-2018-25 (August 2018). https://www.microsoft.com/en-us/research/publication/zoom-multi-view-vector-search-for-optimizing-accuracy-latency-and-memory/
Zhang, J., Huang, P., Zhou, K., Xie, M., Schelter, S.: Hddse: Enabling high-dimensional disk state embedding for generic failure detection system of heterogeneous disks in large data centers. In: Annual Technical Conference, ATC, pp. 111–126. USENIX Association, Virtual Event (2020)
Zhang, X., Wu, H., Chang, Z., Jin, S., Tan, J., Li, F., Zhang, T., Cui, B.: Restune: Resource oriented tuning boosted by meta-learning for cloud databases. In: International Conference on Management of Data, SIGMOD, pp. 2102–2114. ACM, Virtual Event, China (2021)
Zhang, J., Liu, Y., Zhou, K., Li, G., Xiao, Z., Cheng, B., Xing, J., Wang, Y., Cheng, T., Liu, L., Ran, M., Li, Z.: An end-to-end automatic cloud database tuning system using deep reinforcement learning. In: International Conference on Management of Data, SIGMOD, pp. 415–432. ACM, Amsterdam (2019)
Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., Chawla, N.V.: A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: Conference on Artificial Intelligence, AAAI, pp. 1409–1416. AAAI Press, Honolulu (2019)
Zhang, S., Roy, R., Rumancik, L., Wang, A.A.: The composite-file file system: decoupling one-to-one mapping of files and metadata for better performance. ACM Trans. Storage **16**(1), 5–1518 (2020)
Zhang, J., Zhou, K., Huang, P., He, X., Xie, M., Cheng, B., Ji, Y., Wang, Y.: Minority disk failure prediction based on transfer learning in large data centers of heterogeneous disk systems. IEEE Trans. Parallel Distrib. Syst. **31**(9), 2155–2169 (2020)
Zhang, Y., Huang, P., Zhou, K., Wang, H., Hu, J., Ji, Y., Cheng, B.: OSCA: an online-model based cache allocation scheme in cloud block storage systems. In: Gavrilovska, A., Zadok, E. (eds.)

Annual Technical Conference, ATC, pp. 785–798. USENIX Association, Virtual Event (2020)
Zhang, J., Wang, Y., Wang, Y., Zhou, K., Schelter, S., Huang, P., Cheng, B., Ji, Y.: Tier-scrubbing: An adaptive and tiered disk scrubbing scheme with improved MTTD and reduced cost. In: Design Automation Conference, DAC, pp. 1–6. IEEE, San Francisco (2020)
Zhang, Y., Zhou, K., Huang, P., Wang, H., Hu, J., Wang, Y., Ji, Y., Cheng, B.: A machine learning based write policy for SSD cache in cloud block storage. In: Design, Automation & Test in Europe Conference & Exhibition, DATE, pp. 1279–1282. IEEE, Grenoble (2020)
Zhao, Y., Liu, X., Gan, S., Zheng, W.: Predicting disk failures with HMM- and hsmm-based approaches. In: International Conference on Data Mining, ICDM, Lecture Notes in Computer Science, vol. 6171, pp. 390–404. Springer, Berlin (2010)
Zheng, Y., Guo, Q., Tung, A.K.H., Wu, S.: Lazylsh: Approximate nearest neighbor search for multiple distance functions with a single index. In: International Conference on Management of Data, SIGMOD, pp. 2023–2037. ACM, San Francisco (2016)
Zhou, K., Liu, Y., Song, J., Yan, L., Zou, F., Shen, F.: Deep self-taught hashing for image retrieval. In: Conference on Multimedia Conference, MM, pp. 1215–1218. ACM, Brisbane (2015)
Zhou, J., Guo, Q., Jagadish, H.V., Krcál, L., Liu, S., Luan, W., Tung, A.K.H., Yang, Y., Zheng, Y.: A generic inverted index framework for similarity search on the GPU. In: International Conference on Data Engineering, ICDE, pp. 893–904. IEEE Computer Society, Paris (2018)
Zhou, X., Peng, X., Xie, T., Sun, J., Ji, C., Liu, D., Xiang, Q., He, C.: Latent error prediction and fault localization for microservice applications by learning from system trace logs. In: Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/SIGSOFT FSE, pp. 683–694. ACM, Tallinna (2019)
Zhu, Y., Liu, J.: Classytune: A performance auto-tuner for systems in the cloud. CoRR **abs/1910.05482** (2019)
Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., Ma, J.: Proactive drive failure prediction for large scale storage systems. In: Symposium on Mass Storage Systems and Technologies, MSST, pp. 1–5. IEEE Computer Society, Long Beach (2013)
Zhu, Y., Liu, J., Guo, M., Bao, Y., Ma, W., Liu, Z., Song, K., Yang, Y.: Bestconfig: tapping the performance potential of systems via automatic configuration tuning. In: Symposium on Cloud Computing, SoCC, pp. 338–350. ACM, Santa Clara (2017)
Züfle, M., Krupitzer, C., Erhard, F., Grohmann, J., Kounev, S.: To fail or not to fail: Predicting hard disk drive failure time windows. In: Measurement, Modelling and Evaluation of Computing Systems MMB, Lecture Notes in Computer Science, vol. 12040, pp. 19–36. Springer, Saarbrücken (2020)