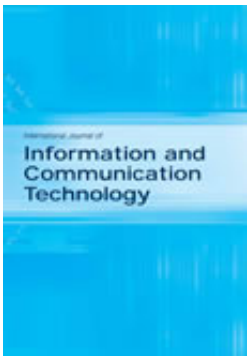


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Design of AI-enhanced hybrid storage engine for multimodal data management

Pinjie Liu, Jing Li

DOI: 10.1504/IJICT.2025.10072361

Article History:

Received:	23 May 2025
Last revised:	07 June 2025
Accepted:	08 June 2025
Published online:	25 July 2025

Design of AI-enhanced hybrid storage engine for multimodal data management

Pinjie Liu* and Jing Li

School of Computer Engineering,
Guangzhou Huali College,
Guangzhou 511325, China
Email: liupinjie0922@163.com
Email: wwlijing@163.com
*Corresponding author

Abstract: To enhance the management effect of multimodal data and increase the data access speed, this paper first uses dynamic random access memory (DRAM) to complete the caching of non-volatile memory (NVM) in the hybrid storage module. When there is a cache missing in DRAM, a high-speed acquisition card is used to collect historical access records of NVM. After encoding historical access records into access vectors, they are used as the input of the deep learning model. The spatio-temporal attention mechanism is introduced to enhance access coding features and improve the prediction accuracy of the access frequency. The multimodal data with prediction results higher than the set threshold are read into the hybrid storage module for storage. Experimental outcome implies that the average performance of the offered approach in sequential reading is at least 1.1 times that of the benchmark approach, significantly improving the access speed.

Keywords: multimodal data management; hybrid storage engine; non-volatile memory; NVM; deep learning; attention mechanism.

Reference to this paper should be made as follows: Liu, P. and Li, J. (2025) 'Design of AI-enhanced hybrid storage engine for multimodal data management', *Int. J. Information and Communication Technology*, Vol. 26, No. 28, pp.67–83.

Biographical notes: Pinjie Liu received her Master's degree from the Lanzhou University of Technology in 2023. She is currently an Associate Professor in the School of Computer Engineering at Guangzhou Huali College. Her research interests include database technology and deep learning.

Jing Li received her Master's degree from the Harbin Institute of Technology in 1997. She is currently a Professor in the School of Computer Engineering at Guangzhou Huali College. Her research interests include data mining and analysis, machine learning, and artificial intelligence.

1 Introduction

In the current era when the digital wave is sweeping the world, multimodal data is growing at an unprecedented rate. These data are widely applied in key fields such as intelligent security, medical diagnosis, autonomous driving, and intelligent education,

becoming the core driving force for technological innovation in artificial intelligence (AI) and the digital transformation of industries (He et al., 2024). However, multimodal data has characteristics such as structural heterogeneity, semantic diversity, large data volume, and high real-time requirements (Schweinar et al., 2024). When dealing with these data, traditional storage systems expose problems such as low data processing efficiency, insufficient utilisation rate of storage resources, and weak semantic understanding ability, making it difficult to meet the growing demands for multimodal data management (Siddiqa et al., 2017). In recent years, the quick growth of AI technique has brought new opportunities to the field of data management. Integrating AI technique into the storage system can achieve intelligent analysis of multimodal data, optimise storage strategies and efficient retrieval, significantly improving the efficiency of data management (Molas and Nowak, 2021). Meanwhile, the hybrid storage architecture combines the advantages of different storage media and can flexibly meet the diverse requirements of multimodal data, providing more efficient and reliable solutions for data storage and management (Aman et al., 2024).

Chen et al. (2022) used a hybrid architecture of Redis and HDFS to store file data. By merging files of the same type through Redis, when the large files obtained from the merging reach the pre-set threshold, the files will be written to HDFS, but the reliability needs to be improved. Lai et al. (2014) proposed a big data storage scheme based on Hadoop, which effectively processed file-based data by utilising Hadoop's distributed computing and storage capabilities. However, the method is mainly designed for batch processing and is not suitable for real-time data with low access scalability. Wang et al. (2019) proposed a unified storage model based on MongoDB. This model utilises the GridFS mechanism of MongoDB to store file data and uniformly stores various types of patient data in the set, but the optimisation difficulty is relatively large. Hennecke (2020) designed a storage scheme for semi-structured data, but this scheme could only meet the storage requirements of a single data type and could not be extended to other data types.

Due to the diverse types of stored objects in the database, the single-modal data storage method can no longer meet the current demands. Hybrid databases for multimodal data storage have emerged as the times require. They address the limitations of traditional databases in managing heterogeneous data by integrating multiple storage models. Lu et al. (2017) studied a hybrid storage system for multimodal data supporting massive small files, storing the location index of multimodal data through the hash function. However, the storage effect of this system for massive data is not very ideal. Zuo et al. (2024) designed a multimodal data edge hybrid storage engine based on consortium chains, combined with the greedy algorithm to determine the placement location of the cached content, and realised multimodal data storage. Jia et al. (2018) proposed a multimodal hybrid storage method based on edge computing. The data caching problem was decomposed into the caching placement problem based on the heuristic adaptive bit rate awareness algorithm and the access request scheduling problem based on low complexity, but the content access delay was relatively high. Bahn and Cho (2020) proposed a hybrid storage system based on non-volatile memory (NVM), designed hierarchical cache pages through the dynamic random access memory (DRAM) cache mechanism, and used RCache to migrate the data of NVM to DRAM for caching in the background thread, but reduced memory utilisation and access efficiency.

To enhance the memory usage rate of the hybrid storage system, researchers have designed a detection method for the hybrid storage system based on deep learning algorithms, thereby enhancing the overall usage efficiency of the system. Deep learning

models can analyse historical data access patterns and predict future load trends. Based on the prediction results, the system can dynamically adjust the allocation of storage resources, such as migrating predicted hot data to high-speed storage media in advance, or optimising the load balancing of storage nodes. Through dynamic resource scheduling, the system can utilise storage resources more efficiently and avoid idle or overloaded resources, thus improving overall utilisation efficiency. Akgun et al. (2023) proposed a model based on autoencoders, which can learn the normal patterns of time series data, regard outliers as reconstruction errors different from the normal patterns, and has good detection performance. Shi et al. (2020) combined the generative adversarial network (GAN) and the transformer model to specifically predict the performance of the hybrid storage system, demonstrating excellent performance. Ebrahimi et al. (2021) predicted the CPU utilisation rate of virtual machines with the help of this network. The results fully demonstrated that recurrent neural network (RNN) showed relatively high accuracy. Zhou et al. (2022) proposed a storage anomaly detection method based on dual-attention contrastive representation learning, but abnormal events of different scales have not been considered yet. Ruan et al. (2023) used long short-term memory (LSTM) network to conduct multi-step prediction of the average CPU utilisation rate of servers and achieved accurate prediction on the historical dataset of real load of data centre servers provided by Google.

In light of the analysis of existing studies, it can be known that the read and write performance of traditional database storage methods often restricts the performance of the system. The main reason is that the performance between memory and external storage usually differs by several orders of magnitude, and the cost of reading and writing data from external storage is relatively high. Thus, for the purpose of enhancing the management effect of multimodal data and reduce access latency, this paper designs an AI-enhanced hybrid storage engine for multimodal data management. Firstly, in the hybrid storage module, DRAM is utilised to complete the caching of NVM. When there is cache missing in DRAM, the built-in high-speed acquisition card of the access monitoring module is used to collect the historical access records of frequently accessed data blocks on NVM. Then, the historical access records are encoded into access vectors to construct a training set, which is used as the input of the deep learning model to achieve the prediction of the access frequency. The feature vectors of different scales output by the temporal convolutional network (TCN) are taken as the input of LSTM. By taking advantage of the nonlinear fitting ability of LSTM, the interrelationships of multi-dimensional data are mined and the nonlinear features of the data are extracted. The spatio-temporal attention mechanism is integrated to enhance the access coding features and improve the prediction accuracy of the access frequency. The experimental outcome demonstrates that the offered approach has great advantages in hybrid storage and processing, and can meet the performance requirements of massive storage and multiple reads.

2 Relevant technologies

2.1 NVM technology

In the past, data management systems used volatile storage devices (DRAM) and persistent storage devices to build storage systems. All the data that needed to be

processed had to be scheduled from persistent devices to DRAM several orders of magnitude higher, which affected the performance of the entire system (Qian et al., 2021). However, the new storage technology, NVM, has broken this traditional design. NVM has the byte addressing and low latency characteristics of DRAM, as well as the persistence and large capacity characteristics of traditional persistent devices. NVM combines the byte-by-byte addressing and low latency characteristics of DRAM while overcoming the volatility limitations of DRAM. NVM does not lose data after a power outage, and does not need to rely on batteries or capacitors to maintain data as DRAM does. DRAM needs to be refreshed periodically to maintain data, which is power hungry and prone to data loss, while NVM does not need to be refreshed, which is low-power and long-lasting data. It narrates the performance gap between volatile devices and persistent devices, and can persist data at the same time. These characteristics bring new opportunities and challenges to traditional databases.

NVM, also known as storage-type memory and NVRAM, is a brand-new and highly promising storage device (Chen, 2016). Currently, NVM encompasses a variety of technologies, each with slightly different characteristics. However, they typically combine the low latency and word addressable nature of DRAM with the persistence of storage technologies such as solid state drive (SSD). In addition, the lifespan of NVM is limited. Repeatedly writing to the same location storage unit can lead to memory failure, as wear balancing needs to be considered in the same way as SSD.

2.2 Convolutional neural network

Convolutional neural network (CNN) is a commonly used deep learning model, which is particularly suitable for processing and classification tasks of high-dimensional data such as images, videos and voices (Kuo, 2016). Different from traditional fully connected neural networks, CNN utilises the structure of convolutional layers and pooling layers, enabling the network to automatically extract important features from the input raw data, thereby achieving efficient feature learning and classification. In addition, CNN significantly reduces the number of parameters, improves computational efficiency, and is able to automatically learn the spatial hierarchy of data through mechanisms such as local connectivity, parameter sharing, and hierarchical feature learning. These advantages make CNN more efficient and accurate than traditional fully connected neural networks in processing structured data such as images and speech. These advantages make CNN more efficient and accurate than traditional fully connected networks in processing structured data such as images and speech. The structure of CNN can consist of several convolutional levels, pooling levels and fully linked levels, among which the convolutional level is the core part of CNN (Liu et al., 2016). The convolutional level adopts convolution operation. Through a set of learnable convolution kernels, convolution operations are performed on the input data to extract local characteristics. Convolution kernels perform convolution at different positions of the input data to obtain the characteristic pictures of the corresponding positions. The output of the convolutional level passes through an activation function and is then input to the next level for processing.

The role of the pooling level is to downsample the output of the convolutional layer, reduce the number of parameters of the network, avoid overfitting, and retain key feature information. Common pooling operations include maximum pooling and average pooling. Fixed pooling sizes and steps can be selected respectively, or pooling sizes and

steps can be learned (Cong and Zhou, 2023). The output of the pooling level can be used as the input of the next convolutional level or directly as the input of the fully linked level. The fully linked level is usually the last level of the network. It flattens the outputs of the convolutional level and the pooling level into a one-dimensional vector, performs linear transformation and activation function transformation through a set of weight matrices, and finally outputs the forecasting level.

2.3 LSTM network

When traditional RNN processes long sequence data, due to the problems of vanishing or exploding gradients, it is difficult to learn the long-distance dependencies in the sequence. Through the design of the gating mechanism and cell state, LSTM can effectively retain and transmit long-term information. LSTM is very suitable for processing sequential data, such as time series data, text data and speech data (Chen, 2022). LSTM enables accurate modelling and prediction of sequence data through the synergy of cellular states and gating mechanisms. At its core, it dynamically balances information retention and forgetting to excel in long sequence tasks. Despite the problem of high computational overhead, LSTM is still widely used in natural language processing, time series analysis, speech recognition, etc. by combining the attention mechanism or using more efficient variants. In time series prediction, LSTM can predict future values based on the changing trend of historical data (Yu et al., 2019).

LSTM effectively solves the problem of gradient explosion by introducing memory units and gating mechanisms. Memory units control the inflow and outflow of information through input gates, forget gates and output gates. The network structure of LSTM is represented by the following formula:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \tag{1}$$

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \tag{2}$$

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right) \tag{3}$$

$$C_t = f_t \cdot C_{t-} + i_t \cdot \tilde{C}_t \tag{4}$$

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{5}$$

$$h_t = o_t \cdot \tanh\left(C_t\right) \tag{6}$$

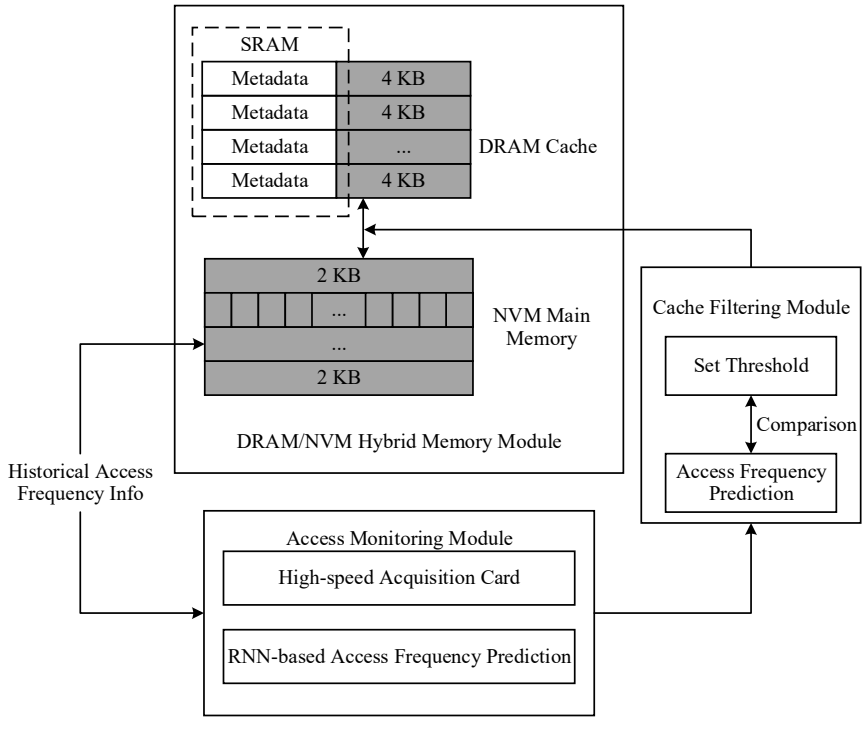
where σ is the sgmoid function; W and b are the weight matrix and the bias vector respectively, and the subscripts f, i and o represent the forget gate, input gate and output gate, respectively; C_t is the updated state of the memory unit, W_c and b_c are the weights and biases used to generate C_t.

3 Design of multi-level hybrid storage engine for multimodal data management

3.1 The overall structure of the hierarchical hybrid storage engine for multimodal data

To improve the management effect of multimodal data, increase the data access speed and reduce the database load, a multi-level hybrid storage engine for multimodal data management is designed. In the DRAM/NVM hybrid storage module, DRAM is utilised to complete the caching of the main memory NVM. When there is cache missing in DRAM, the built-in high-speed acquisition card of the access monitoring module is used to collect the historical access records of frequently accessing 4 KB data blocks on NVM. Then, the historical access records are encoded into access vectors to construct the training set, which is used as the input of the subsequent deep learning model for predicting the access frequency. In the cache filtering module, multimodal data with access frequency prediction results higher than the set threshold are read into DRAM for storage.

Figure 1 The overall structure of the hierarchical hybrid storage engine for multimodal data



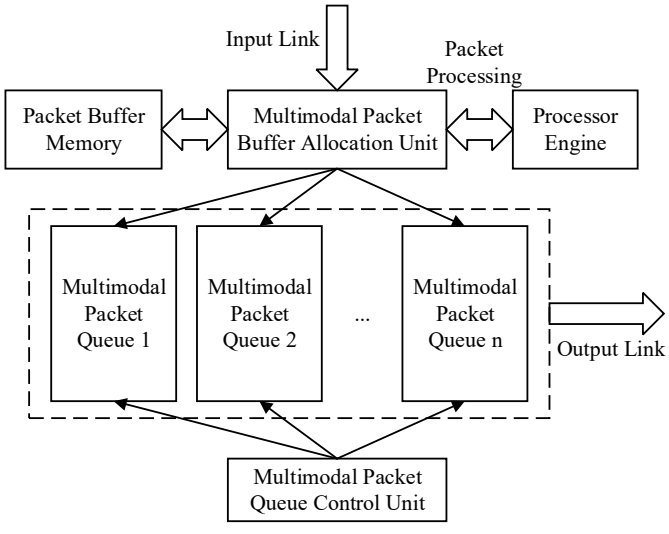
The multimodal data hierarchical hybrid storage engine includes a DRAM/NVM hybrid memory module, an access monitoring module, and a cache filtering module. The entire system framework is indicated in Figure 1. The DRAM/NVM hybrid storage module is composed of DRAM and NVM, and the cache part of the multimodal data main memory NVM is DRAM. Among them, the page size of NVM is 2 MB. DRAM directly saves the

metadata of multimodal data on the cache chip through a large granularity management method of 4 KB, which can effectively reduce the storage time of multimodal data. Intending to the issue that frequent data reading work occupies a large amount of memory, a storage filtering module is designed in the system. Only the NVM data with a higher access frequency is cached in DRAM to reduce the data exchange between DRAM and NVM. When there is a cache deficiency in DRAM, the cache filtering module determines the access frequency based on the access frequency prediction result obtained by the access monitoring module and combined with the set threshold, and reads the 4 KB multimodal data of the access frequency prediction result higher than the set threshold into DRAM for caching.

3.2 The hybrid storage structure of multimodal data

The multimodal data hybrid storage structure of DRAM/NVM is shown in Figure 2. Among them, the storage task scheduling work of multimodal data packets and the access control work of data packet buffer memory. All of these are accomplished through the multimodal data packet storage and distribution unit. The managed multimodal data packets are transmitted to the corresponding network interface for data packet storage queuing, and through the scheduling of the multimodal data packet queue management and control unit, the multimodal data packets are transmitted to the output link.

Figure 2 The multimodal data hybrid storage structure of DRAM/NVM



The multimodal data packet storage allocation unit adopts a queue caching mechanism based on shared storage (Suh et al., 2004), allocating buffer storage resources to the data packet queue. This mechanism shares the same cache space with all data transmission ports. After caching, the multimodal data packets only need to sort the storage addresses of the multimodal data packets transmitted to the same interface when sending. Form a sending queue and send in sequence. It has the advantages of simple operation and high resource utilisation rate.

The multimodal data packet queue control unit is responsible for the cache queue control of multimodal data packets transmitted to the same network interface, and can control the storage location of multimodal data packets in the hierarchical cache system. The queuing mode adopted in this unit is the output queuing mode. The multimodal data packet sending queue management is implemented in the form of a linked list, and the control is carried out during queuing and dequeuing through the multimodal data packet queue handle. There are two pointers in the handle for the head and tail of the sending queue. When a new queued data packet or a dequeued data packet is added to the multimodal data packet sending queue, the tail pointer and the head pointer need to be modified respectively to ensure that the data is sent first after queuing.

The storage location of the queue handle is determined based on the network interface encoding for sending multimodal data packets. It begins to write the queue handle when receiving the multimodal data packet. Each time a multimodal data packet in the queue is added, the first letter of the handle long word of the previous multimodal data packet is rewritten first. Then, the handle tail pointer is rewritten to the handle address of the new multimodal data packet, and at the same time, the handle long word 2 is increased by 1. In this way, a queue handle is formed to complete queue control, and then the multimodal data packets are stored in the order of the queue.

4 Access frequency prediction of the hybrid storage engine based on the improved time convolutional network and LSTM

4.1 The encoding of historical access records of the hybrid storage engine

For the goal of enhancing the storage utilisation rate of the designed hybrid storage engine system, this paper uses the high-speed acquisition card built into the access monitoring module to collect the historical access records of frequently accessing 4 KB data blocks on the hybrid storage engine. After encoding the historical access records into access vectors, a training set is constructed and used as the input for improving the hybrid deep learning model of TCN and LSTM to predict the access frequency. In the cache filtering module, the 4 KB multimodal data with access frequency prediction results higher than the set threshold is read into DRAM/NVM for caching.

The access monitoring module uses a high-speed acquisition card to collect the historical access records of frequently visiting 4 KB data blocks in the hybrid storage engine. After encoding the historical access records into access vectors to construct a training set, it serves as the input for improving TCN-LSTM to implement the access frequency prediction of the hybrid storage engine system. Encode the historical access records of the hybrid storage engine into format $S_{op} = \{op_1, op_2, \ldots, op_{m-1}\}$ that is convenient to be used as the input data of the LSTM neural network, where op_i represents the i^{th} access code in the access sequence and m represents the number of accesses in the historical access records of the hybrid storage engine. For this purpose, the multi-fork tree traversal of the historical access record of the hybrid storage engine can be described as follows:

$$S_{op} = \left\{T_2, \lambda T_1, k < 100, Hash, \overline{T} = T_1 \leftrightarrow T_2, Materialise, \overline{T} \leftarrow T_3, Sort, Group\right\} \quad (7)$$

where T_1, T_2 and T_3 respectively represent the source tables of the hybrid storage engine used in the 1th, 2th, and 3th visits in the historical access record; λ and $\leftrightarrow$ represent selection and connection operations, respectively; $\overline{T}$ and k respectively represent the result of connecting T_1 with T_2. and the primary key of the source table. The historical access record multitree of each hybrid storage engine can be converted into a unique access sequence S_{op}; conversely, S_{op} can also be transformed into a unique multi-way tree of access records. Each leaf node of the tree contains the source table information of the hybrid storage engine.

After traversal, an access sequence is generated for each access in the historical access record of the hybrid storage engine. During this process, the key features in the historical access record of the hybrid storage engine need to be extracted to complete the v encoding of the access vector. v includes the following contents:

1 Type of access n_0, access the corresponding source table n_1 in the hybrid storage engine.

2 Access the column n_2 designed for the corresponding source table in the hybrid storage engine.

3 Access the mean width n_3 of the row where the corresponding result is located.

4 Access the selection rate n_4 for multimodal data.

In the above content, n_0, n_1 and n_2 respectively represent the structures for accessing multimodal data; n_3 and n_4 respectively represent the scale of accessing multimodal data. The historical access records are represented by vector $S_v = (v_0, v_1, v_2, \ldots, v_{m-1})$, and v_i is the i^{th} access in the encoding. Vector S_v is taken as the encoding result of the access vector of the historical access records of the hybrid storage engine and input into the improved TCN-LSTM model for access frequency prediction.

4.2 Improvement of the TCN-LSTM model Based on the spatio-temporal attention mechanism

For the goal of enhance the access vector coding results of the hybrid storage engine, this paper introduces the temporal attention mechanism (SE) and the spatial attention mechanism (TPA) (Lin et al., 2020) to improve the TCN-LSTM model (ETCN-LSTM). Compared with the traditional TCN-LSTM model, the ETCN-LSTM model can adaptively learn the temporal and spatial information in historical access records.

To cope with the issue of weight distribution between historical access records and the input at the current moment and enhance the adaptability of the time channel of the model, the SE module is introduced in this paper. The SE module can effectively assist the model in allocating weights between historical traffic data and the traffic data input at the current moment, enabling the model to have better learning efficiency and robustness in extracting temporal features. Moreover, due to the relatively small computational load of the SE module, only one global pooling and two fully connected operations are required. Compared with other time attention mechanism models, it has less resource consumption, and the SE module can also better meet the real-time requirements.

Due to the significant differences among the storages of different modalities, when the model learns spatial features, it will extract different features simultaneously, resulting in the model learning excessive noise information and thus leading to a decrease

in model accuracy. To cope with the above issues, this paper suggests a TPA module for the scenario of predicting the access frequency of the hybrid storage engine. The TPA module divides the access encoding results into individual matrix data of different types, uses convolution to extract spatial information from different access matrices respectively, and then concatenates different access matrices to restore the original access matrix data structure. Finally, the Hadama product is performed between the extracted spatial information weights and the matrix data on the backbone road for weight allocation.

4.3 Access frequency prediction based on the improved TCN-LSTM model

The TCN-LSTM model improved based on the above-mentioned spatio-temporal attention mechanism takes the historical access record encoding vector of the hybrid storage engine as the input data of the ETCN-LSTM model, effectively utilises the spatio-temporal attention mechanism to enhance the characteristics of TCN and LSTM to capture the time series dependence of data access and achieve access frequency prediction. Combined with the set threshold to determine the multimodal data access frequency of the hybrid storage engine, the 4 KB multimodal data in the part where the predicted access frequency result is higher than the set threshold is read into DRAM for caching, reducing unnecessary cache replacement operations, thereby improving the performance and efficiency of the system.

Each residual module in TCN is composed of three one-dimensional convolutional levels, Conv0, Conv1 and Conv2. The first convolutional level, Conv0, performs preliminary processing on the input and selects to activate it using the ReLU function, with the output being C_0. The input of the second convolutional level Conv1 is C_0, and it is called C_1 after convolutional feature extraction. The inputs of C_0 and C_1 need to take into account the expanded convolution parameter d of TCN, and then input Conv2 after stepwise element multiplication. The output of Conv2 can be added to the module input p_t to obtain another output R_t. Therefore, R_t is represented as follows:

$$C_{0,t} = \mathrm{ReLU}\left(p_t \otimes W_0 + b_0\right) \tag{8}$$

$$C_{1,t} = C_{0,t} \otimes W_1 + b_1 \tag{9}$$

$$C_{2,t} = \left(C_{0,t} \cdot C_{1,t}\right) \otimes W_2 + b_2 \tag{10}$$

$$R_t = \mathrm{ReLU}\left(p_t + C_{2,t}\right) \tag{11}$$

where W_0, W_1 and W_2 are convolution kernel matrices; b_0, b_1 and b_2 are biases; $\otimes$ is the convolution operation; $\cdot$ is the product of elements one by one; the number of elements in R_t is n, denoted as $R_t = [r_{t1}, r_{t2}, \ldots, r_{tn}]^T$.

Considering that a TCN level can be composed of multiple residual modules, when stacking multiple residual modules, it is necessary to find the output R_{k-1} of the $k-1^{\text{th}}$ residual module as the input of the k^{th} residual module. The output of the final module is H_n. Nonlinear calculations are carried out using the ReLU activation function as the output G_t of the TCN level. Therefore, G_t can be expressed as follows:

$$R_{t,k} = Res\left(R_{t,k-1}\right) \tag{12}$$

$$G = \mathrm{ReLU}\left(R_{t,k}\right) \tag{13}$$

where $R_{t,k}$ represents R_t output by the k^{th} residual module; *Res* represents the residual module. The number of elements in G_t is n, denoted as $G_t = [G_{t,1}, G_{t,2}, \ldots, G_{t,n}]^T$.

The output G_t of the TCN module is concatenated with the input x using the concatenate function as the input of LSTM, denoted as I_t, where *concatenate* is the concatenation function.

$$I_t = concatenate\left(x, G_t\right) \tag{14}$$

At time t, the LSTM module learns the power load characteristic vector I_t output by the concatenate module. Let the hidden layer state of the LSTM module at time t be $H_{t,i}$, and $H_{t,i}$ is represented as follows:

$$H_{t,i} = LSTM\left(H_{t,i-1}, I_t, C_{t,i-1}\right) \tag{15}$$

where $H_{t,i}$ represents the hidden state of the i^{th} input step at time t; $C_{t,i-1}$ represents the cell state of the $i-1^{\text{th}}$ input step at time t.

The input of the spatio-temporal attention mechanism is the obscured level state h_t calculated by the LSTM module. The calculation of the attention weight $A_{t,i}$ is shown in equation (16). Let the output of the spatio-temporal attention module at time t be S_t. The output S_t of the spatio-temporal attention module and the output H_t of the LSTM module are multiplied bit by bit through the *Multiply* function to achieve the dynamic weighting process of the obscured level units, which are denoted as L_t and S_t, as shown below:

$$A_{t,i} = \frac{\exp\left(H_{t,i}^T, q_t\right)}{\sum_{k=1}^{T} \exp\left(H_{t,i}^T, q_t\right)} \tag{16}$$

$$S_t = \sum_{i=1}^{T} A_{t,i} H_{t,i} \tag{17}$$

$$L_t = Multiply\left(S_t, H_t\right) \tag{18}$$

where $A_{t,i}$ represents the hidden state of the i^{th} input time step in the attention level at time t; q_t represents the randomly initialised attention weight matrix; *Multiply* means to multiply elements one by one.

The output module is a fully linked level, with the linear function as the activation function. After processing, the predicted access frequency value of the mixed storage engine system at the $t+1$ moment is obtained, denoted as y. The calculation equation of the output level can be expressed as follows:

$$y = linear\left(wL_t + b\right) \tag{19}$$

where w is the convolution kernel matrix, and b is bias.

5 Experimental results and analyses

To verify the storage performance of the AHSE method proposed in this paper, the NVMain local memory simulator is used to simulate the storage performance of the system. It mainly simulates the access behaviour of the mixed storage of DRAM and NVM in the system proposed in this paper, and combines the global simulator to simulate and generate multimodal data with different distribution situations and access patterns for the system cache experiment. Meanwhile, simulate the application situations of each functional module of the system, such as queue management, etc. The hardware and software configuration of this system includes Hadoop version 3.1.1, the operating system version Ubuntu 16.04, and the CPU version Core i7-10700. Set the learning efficiency of the deep learning model for learning different data to 0.01, the feature matching coefficient to 0.1, and the sampling interval of the deep learning training time to 1 second.

This paper records the time delays of AHSE, MCVS (Jia et al., 2018) and GAN-TRANS (Shi et al., 2020) in uploading and downloading multimodal data. The size of the text is 4.5 M, the size of the image is 25.9 M, and the size of the video is 325.9 M. The experimental results of uploading and downloading data in different storage systems are shown in Table 1. The experiment found that when uploading data, the larger the file, the faster the upload speed. Similarly, the larger the file is, the faster its download speed will be. This is because AHSE adopts a distributed storage method, dividing multimodal documents into multiple data blocks and storing them on different nodes, thereby achieving parallel data transmission and improving the efficiency of upload and download. Furthermore, this storage method can also ensure the reliability of data. Even if a certain node fails, it will not affect the integrity and availability of the entire file. Therefore, AHSE has a significant advantage in hybrid storage and processing, and can meet the performance requirements of massive storage and multiple reads.

Table 1 Comparison of data download speeds

Method	*Text*		*Picture*		*Video*	
	Upload	*Download*	*Upload*	*Download*	*Upload*	*Download*
MCVS	97 ms	62 ms	183 ms	105 ms	9,180 ms	7,051 ms
GAN-TRANS	52 ms	39 ms	106 ms	67 ms	7,200 ms	5,216 ms
AHSE	35 ms	17 ms	59 ms	31 ms	4,240 ms	2,340 ms

This experiment tested the performance of AHSE, MCVS and GAN-TRANS in sequential writing and random writing. The test uses one million pieces of data, with values ranging in size from 64 B to 32 Kb, as shown in Figure 3. Figure 3(a) implies the comparison results in terms of sequential write performance. Regardless of the size of the values, the sequential write throughput of AHSE is significantly higher than that of MCVS and GAN-TRANS. When the value size reaches 4 kB, the sequential write throughput of AHSE is almost stable at 200 MB/s, which is much lower than the sequential write speed of the hard disk. However, for MCVS and GAN-TRANS, as the value size increases, the sequential write throughput reaches 800–900 MB/s. The average performance of AHSE in sequential writing is 3.1 times that of MCVS and 1.4 times that of GAN-TRANS, respectively.

Figure 3 The (a) sequential write and (b) random write results of different methods (see online version for colours)

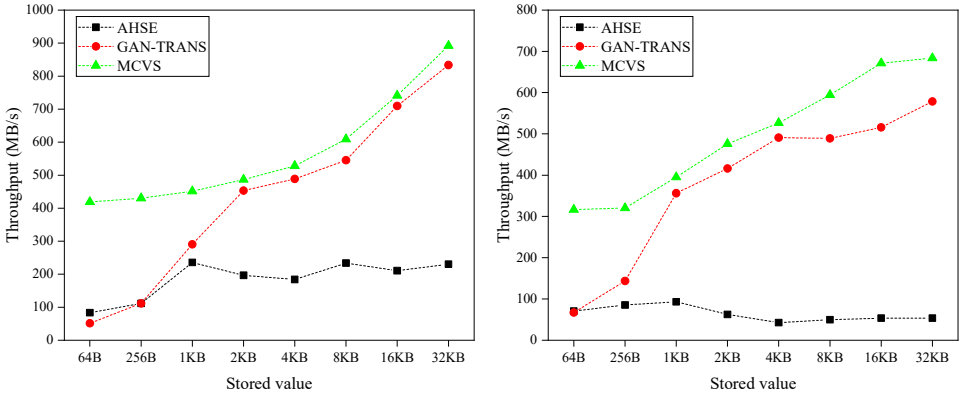


Figure 3(b) shows the comparison in terms of random write performance. The overall random write throughput of MCVS remains at 60–90 MB/s and decreases with the increase of the value size. The reason for this phenomenon is that as the value size increases, the storage scale also increases, thereby leading to an increase in the merging operation time of MCVS and affecting the write performance. On the contrary, since AHSE and GAN-TRANS adopt hierarchical storage technology, the access scale remains at a relatively small level. Therefore, in terms of on-machine write, the write throughput of AHSE and GAN-TRANS is much higher than that of MCVS. The average performance of AHSE in random writing is 7.6 times that of MCVS and 1.3 times that of GAN-TRANS, respectively.

Figure 4 Sequential read the comparison results in terms of performance (see online version for colours)

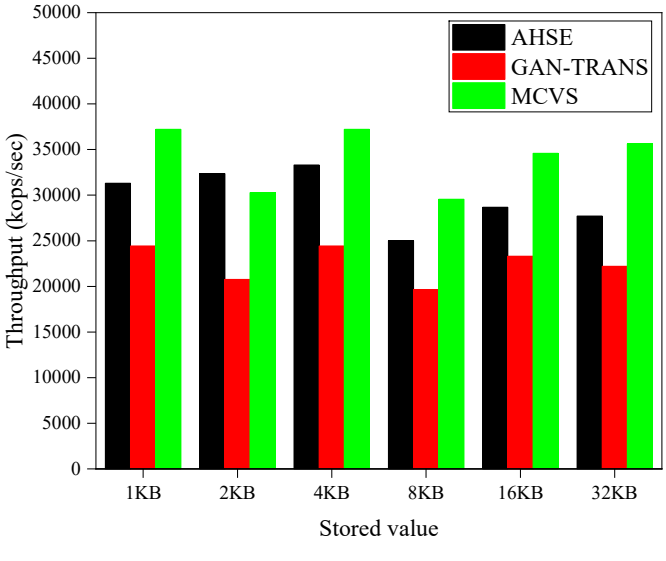
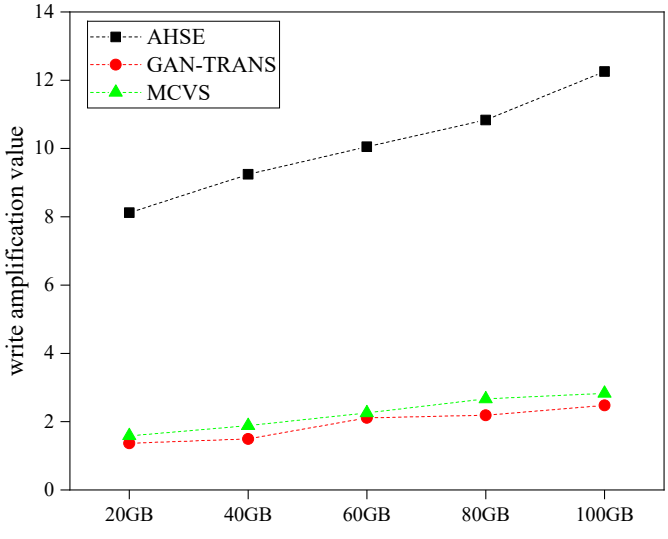


Figure 4 shows the comparison results in terms of sequential reading performance, among which the throughput of AHSE is significantly higher than that of MCVS and GAN-TRANS. GAN-TRANS improves the reading performance by sorting the data at each level. However, MCVS separates the values into the value log file, causing their values to be stored in an unordered manner, thereby affecting the reading performance. To cope with this issue, AHSE adopted the parallel reading technology of hybrid storage to improve the reading performance. The average performance of AHSE in sequential reading is 1.1 times that of MCVS and 1.5 times that of GAN-TRANS, respectively.

This experiment tested the write amplification of MCVS, GAN-TRANS, and AHSE under a mixed workload, as shown in Figure 5. As the total data volume increases from 20 GB to 100 GB, the write amplification of MCVS increases from 8 to 11. In contrast, GAN-TRANS and AHSE have successfully reduced write amplification through hierarchical storage technology, and the write amplification values of both remain between 1 and 3. However, since AHSE adopts access monitoring technology and detects the storage of abnormal access, the storage efficiency has been greatly improved.

Figure 5 The enlarged result graphs of writing by different methods (see online version for colours)



6 Conclusions

As the big data quickly growing, the storage and management of multimodal data are facing huge challenges. The traditional database storage methods have a high cost of reading and writing from external memory and a slow access speed. To solve the above problems, this paper designs an AI-enhanced hybrid storage engine for multimodal data management. Firstly, a multimodal data hierarchical hybrid storage engine was designed. The DRAM/NVM hybrid storage module is composed of DRAM and NVM, and the cache part of the multimodal data main memory NVM is DRAM. DRAM/NVM directly saves the metadata of multimodal data on the cache chip through the large granularity

management method, which can effectively reduce the storage time of multimodal data. When there is cache missing in DRAM, the built-in high-speed acquisition card of the access monitoring module is used to collect the historical access records of frequently accessed data blocks on NVM. Then, the historical access records are encoded into access vectors to construct a training set, which is used as the input of the deep learning model to achieve the prediction of the access frequency. The feature vectors of different scales output by TCN are taken as the input of LSTM. By taking advantage of the nonlinear fitting ability of LSTM, the interrelationships of multi-dimensional data are mined and the nonlinear features of the data are extracted. The spatio-temporal attention mechanism is integrated to enhance the access coding features and improve the prediction accuracy of the access frequency. The experimental outcome implies that the suggested approach has significant advantages in multimodal data storage and management. It can effectively improve data access efficiency and reduce query response time, providing strong support for the large-scale storage and application of multimodal data.

The hybrid storage engine designed in this paper has achieved significant performance advantages. However, the layout and size of other data were not redesigned. Subsequent work can consider how to design the layout and size of other data, as well as further fine-grained management of data blocks in DRAM/NVM, thereby improving the space utilisation rate.

Acknowledgements

This work is supported by the Guangdong Higher Education Society '14th Five-Year Plan' 2024 Annual Higher Education Research Project (No. 24GYB119), 2024 Teaching Quality Engineering Construction Project of Guangzhou Huali College named: Database Principles Course Teaching and Research Office, Outstanding Teaching Team of Guangzhou Huali College named: Database Teaching Innovation Team, and 2025 Teaching Quality and Teaching Reform Project of Guangzhou Huali College named: Curriculum Reform Practice of Database Principles Based on Trinity Integration: Ideological and Political Education Guidance, Technology Empowerment, and Industry-Academia Collaboration Driven.

Declarations

All authors declare that they have no conflicts of interest.

References

Akgun, I.U., Aydin, A.S., Burford, A., McNeill, M., Arkhangelskiy, M. and Zadok, E. (2023) 'Improving storage systems using machine learning', *ACM Transactions on Storage*, Vol. 19, No. 1, pp.1–30.

Aman, S.S., Agbo, D.D.A., N'guessan, B.G. and Kone, T. (2024) 'Design of a data storage and retrieval ontology for the efficient integration of information in artificial intelligence systems', *International Journal of Information Technology*, Vol. 16, No. 3, pp.1743–1761.

Bahn, H. and Cho, K. (2020) 'Implications of NVM based storage on memory subsystem management', *Applied Sciences*, Vol. 10, No. 3, pp.62–75.

Chen, A. (2016) 'A review of emerging non-volatile memory (NVM) technologies and applications', *Solid-State Electronics*, Vol. 125, pp.25–38.

Chen, D. (2022) 'Cloud computing database and travel smart platform design based on LSTM algorithm', *Mobile Information Systems*, Vol. 20, No. 3, pp.51–69.

Chen, L., Zhao, J., Wang, C., Cao, T., Zigman, J., Volos, H., Mutlu, O., Lv, F., Feng, X. and Xu, G.H. (2022) 'Unified holistic memory management supporting multiple big data processing frameworks over hybrid memories', *ACM Transactions on Computer Systems (TOCS)*, Vol. 39, No. 4, pp.1–38.

Cong, S. and Zhou, Y. (2023) 'A review of convolutional neural network architectures and their optimizations', *Artificial Intelligence Review*, Vol. 56, No. 3, pp.1905–1969.

Ebrahimi, S., Salkhordeh, R., Osia, S.A., Taheri, A., Rabiee, H.R. and Asadi, H. (2021) 'RC-RNN: reconfigurable cache architecture for storage systems using recurrent neural networks', *IEEE Transactions on Emerging Topics in Computing*, Vol. 10, No. 3, pp.1492–1506.

He, W., Xia, T., Song, S., Huang, X. and Wang, J. (2024) 'Multimodal data encoding and compression in Apache IoTDB', *International Journal of Software & Informatics*, Vol. 14, No. 1, pp.1–16.

Hennecke, M. (2020) 'Daos: a scale-out high performance storage stack for storage class memory', *Supercomputing Frontiers*, Vol. 40, pp.1–12.

Jia, G., Han, G., Du, J. and Chan, S. (2018) 'A maximum cache value policy in hybrid memory-based edge computing for mobile devices', *IEEE Internet of Things Journal*, Vol. 6, No. 3, pp.4401–4410.

Kuo, C-C.J. (2016) 'Understanding convolutional neural networks with a mathematical model', *Journal of Visual Communication and Image Representation*, Vol. 41, pp.406–413.

Lai, W.K., Chen, Y-U., Wu, T-Y. and Obaidat, M.S. (2014) 'Towards a framework for large-scale multimedia data storage and processing on Hadoop platform', *The Journal of Supercomputing*, Vol. 68, pp.488–507.

Lin, T., Pan, Y., Xue, G., Song, J. and Qi, C. (2020) 'A novel hybrid spatial-temporal attention-LSTM model for heat load prediction', *IEEE Access*, Vol. 8, pp.159182–159195.

Liu, M., Shi, J., Li, Z., Li, C., Zhu, J. and Liu, S. (2016) 'Towards better analysis of deep convolutional neural networks', *IEEE Transactions on Visualization and Computer Graphics*, Vol. 23, No. 1, pp.91–100.

Lu, W., Wang, Y., Jiang, J., Liu, J., Shen, Y. and Wei, B. (2017) 'Hybrid storage architecture and efficient MapReduce processing for unstructured data', *Parallel Computing*, Vol. 69, pp.63–77.

Molas, G. and Nowak, E. (2021) 'Advances in emerging memory technologies: from data storage to artificial intelligence', *Applied Sciences*, Vol. 11, No. 23, pp.12–27.

Qian, Z., Wei, J., Xiang, Y. and Xiao, C. (2021) 'A performance evaluation of DRAM access for in-memory databases', *IEEE Access*, Vol. 9, pp.146454–146470.

Ruan, L., Bai, Y., Li, S., He, S. and Xiao, L. (2023) 'Workload time series prediction in storage systems: a deep learning based approach', *Cluster Computing*, Vol. 6, pp.1–11.

Schweinar, A., Wagner, F., Klingner, C., Festag, S., Spreckelsen, C. and Brodoehl, S. (2024) 'Simplifying multimodal clinical research data management: introducing an integrated and user-friendly database concept', *Applied Clinical Informatics*, Vol. 15, No. 2, pp.234–249.

Shi, Z., Dang, H., Liu, Z. and Zhou, X. (2020) 'Detection and identification of stored-grain insects using deep learning: a more effective neural network', *IEEE Access*, Vol. 8, pp.163703–163714.

Siddiqa, A., Karim, A. and Gani, A. (2017) 'Big data storage technologies: a survey', *Frontiers of Information Technology & Electronic Engineering*, Vol. 18, pp.1040–1070.

Suh, G.E., Rudolph, L. and Devadas, S. (2004) 'Dynamic partitioning of shared cache memory', *The Journal of Supercomputing*, Vol. 28, No. 1, pp.7–26.

Wang, S., Li, G., Yao, X., Zeng, Y., Pang, L. and Zhang, L. (2019) 'A distributed storage and access approach for massive remote sensing data in MongoDB', *ISPRS International Journal of Geo-Information*, Vol. 8, No. 12, p.533.

Yu, Y., Si, X., Hu, C. and Zhang, J. (2019) 'A review of recurrent neural networks: LSTM cells and network architectures', *Neural Computation*, Vol. 31, No. 7, pp.1235–1270.

Zhou, Q., Yang, G., Song, H., Guo, J., Zhang, Y., Wei, S., Qu, L., Gutierrez, L.A. and Qiao, S. (2022) 'A BiLSTM cardinality estimator in complex database systems based on attention mechanism', *CAAI Transactions on Intelligence Technology*, Vol. 7, No. 3, pp.537–546.

Zuo, R., Zheng, C., Li, F., Zhu, L. and Zhang, Z. (2024) 'Privacy-enhanced prototype-based federated cross-modal hashing for cross-modal retrieval', *ACM Transactions on Multimedia Computing, Communications and Applications*, Vol. 20, No. 9, pp.1–19.