



# A heterogeneous 3-D stacked PIM accelerator for GCN-based recommender systems

Xinyang Shen<sup>1</sup> · Yu Huang<sup>1,2</sup> · Long Zheng<sup>1,2</sup> · Xiaofei Liao<sup>1,2</sup> · Hai Jin<sup>1</sup>

Received: 9 June 2023 / Accepted: 2 January 2024  
© The Author(s) 2024

## Abstract

Modern recommendation systems integrate graph convolution neural networks (GCN) for enhancing embedding representation. Compared with widely deployed neural network-based models, the extra message propagation layer of GCN-based recommendation is featured with extensive computations and irregular memory access. However, architecture designs for prevailing deep neural network recommendation models assume simple pooling in the embedding layer. ReRAM-based GCN accelerators are specialized for graph-related operations. However, they are designed for general graphs, while GCN-based recommendation models mainly operate on the user-item graph. In this paper, we proposed a resistive random accessed memory (ReRAM) based processing-in-memory (PIM) accelerator, ReGCNR, for GCN-based recommendation. ReGCNR is featured with three key innovations. First, we exploit the 3-dimensional (3-D) stacked heterogeneous ReRAM to fit with the large-size embedding table and user-item graph. Then, we propose a joint degree mapping schema that maximizes the efficiency of the execution pipeline. After that, ReGCNR assembles a well-coordinated pipeline and hardware scheduling design to boost overall system performance. Results show that ReGCNR outperforms GPU by 69.83× and 56.67× in terms of average speedup and energy saving, respectively. In addition, ReGCNR outperforms state-of-the-art ReRAM-based solutions by 11.13× speedups and 7.22× energy savings on average.

**Keywords** GCN-based recommendation · ReRAM · Accelerator · Processing-in-memory · 3-D stacked PIM · Heterogeneous

## 1 Introduction

GCN-based recommender systems (Huang et al. 2021; He et al. 2020; Song et al. 2017; Berg et al. 2017; Wu et al. 2022) have emerged as a popular way for performing personalized recommendations. Those systems have been applied for lots of real-world services such as electric commerce (He et al. 2020; Wang et al. 2019; Feng et al. 2020; Li et al. 2019), content recommendation (Yang and Dong 2020), and advertising (Huang et al. 2021). The uniqueness of GCN-based recommendation models is that they integrate GCN for constructing user and item embeddings to capture the interaction information in embeddings for more accurate representations. It is already challenging to process recommender systems on general-purpose platforms due to large embedding tables and low operational tensity (Ke et al. 2020). The increasing complexity of embedding function (Huang et al. 2021; Song et al. 2018; Huang et al. 2022) for combining GCN on recommendation makes it more difficult.

---

✉ Long Zheng  
longzh@hust.edu.cn

Xinyang Shen  
xyhsen@hust.edu.cn

Yu Huang  
yuh@hust.edu.cn

Xiaofei Liao  
xfliao@hust.edu.cn

Hai Jin  
hjin@hust.edu.cn

<sup>1</sup> National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Clusters and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Luoyu Road, Wuhan 430074, Hubei, China

<sup>2</sup> Zhejiang Lab, Zhongtai Road, Hangzhou 311121, Zhejiang, China

Generally speaking, prevailing recommender systems, such as deep neural network (DNN) based recommendation models (Naumov et al. 2019), have a hybrid structure mainly composed of embedding lookup and neural network operations. The embedding lookup operation has been proved to be the system bottleneck for the DNN-based recommendation model (Gupta et al. 2020; Kal et al. 2021; Hwang et al. 2020; Naumov et al. 2019). Those systems have low operational tensity (Ke et al. 2020), which indicates that the embedding lookup layer induces extensive memory accesses in a very short time. Therefore, the overall DNN-based recommendation system performance is limited by the memory bandwidth on the general computing platform (Ke et al. 2020). On the contrary, GCN-based recommendations models have three layers including the embedding lookup, message propagation, and prediction layer (Huang et al. 2021; Wu et al. 2022). Hence, the situation becomes even more complex for GCN-based recommendations because the extra message propagation layer induces many more operations.

Specifically, the embedding lookup layer on GCN-based recommendation generates initial embeddings for users and items. Those initial embeddings are processed by the message propagation layer with two main kernels, aggregation and combination, to obtain propagated results. The aggregation kernel gathers messages to refine the user and item embeddings. The combination kernel transforms the aggregated results with weight matrixes. Both tow kernel requires extensive computations. Especially, the aggregation kernel induces random memory access while gathering messages along edges on the user-items bipartite graph. Those two kernels may be executed with multiple successive layers. Therefore, instead of embedding lookup operations, the unique message propagation layer becomes the GCN-based recommendation model bottleneck.

Several architecture works, such as RecNMP (Ke et al. 2020), Centaur (Hwang et al. 2020), and Space (Kal et al. 2021), address the system bottleneck caused by the embedding lookup layer for the recommendation system. For example, RecNMP (Ke et al. 2020) leverages the near-memory processing to execute pooling operation on computation logic near memory to mitigate the effect of limitation of memory bandwidth. However, those designs are specialized for embedding the lookup layer and may not be adaptable to GCN-based recommendation models. In GCN-based models, the message propagations dominate the system performance. In this work, we propose a ReRAM-based PIM architecture design for end-to-end GCN-based recommendation and focus on the propagation layer.

Meanwhile, there are GCN accelerators, such as HyGNC (Yan et al. 2020), PIMGCN (Yang et al. 2021), and Hetraph (Huang et al. 2022). Those accelerators make

special efforts to accelerate the combination and aggregation kernels (Yan et al. 2020; Yang et al. 2021; Huang et al. 2022) by exploiting multi-level parallelism (Yang et al. 2021) and reducing sparsity in aggregations kernels (Huang et al. 2022). However, those designs are designed for general GCN applications without considering the full procedure of GCN-based recommendation. GCN-based recommendation models, especially, generally operate on the user-item graph, which distinguishes them from general graphs. Compared with GCN accelerators, our work boosts the full procedure of the GCN-based recommender system on ReRAM heterogeneous PIM, focusing on aggregation and combination kernels operated for user-item graphs.

The most relevant work to our design is REREC (Wang et al. 2021), which also exploits ReRAM for accelerating recommendation. There are mainly two differences between our design and REREC. First, REREC focuses on interaction while our design focuses on the propagation layer. Second, REREC adopts a mapping schema that maps item embeddings on the crossbar while we map the user and item on the ReRAM crossbar, owing to the fact that both the user and item can be the aggregated destination. Furthermore, we exploit the heterogeneous 3D-stacked design and hardware scheduling to coordinate the mapping schema.

In summary, we make the following contributions:

- We identify the performance bottleneck of the GCN-based recommendation and characterize operation and access patterns.
- We present a ReRAM-based heterogeneous 3-D stacked PIM architecture design that integrates analog and memory ReRAM to satisfy the computation and on-chip memory access requirements for GCN-based recommendation.
- We propose a specific mapping schema and cooperative hardware scheduling to maximize the hardware efficiency.
- We evaluate ReGCNR on a range of models and datasets. Results show that ReGCNR outperforms state-of-art GPU solutions and ReRAM-based solutions by 69.83× and 11.13× in terms of average speedups, and by 56.67× and 7.22× in terms of energy savings.

## 2 Background and motivation

In this section, we introduce the background of GCN-based recommendation and ReRAM-based processing. After that, we make observations of gaps between the GCN-based recommendations properties and ReRAM-based PIM. Finally, we briefly introduce how we bridge those gaps.

### 2.1 GCN-based recommendation

Figure 1a shows the system overview of a GCN-based recommender system. A typical GCN recommender system model generally consists of an embedding layer, an embedding propagation layer, and a prediction layer. The embedding layer is composed of embedding lookup operations, which serves as an initial stage for building user embedding vector  $e_u$  and item embedding vector  $e_i$ , and can be defined as follows:

$$e_u = \text{Lookup}(\text{Tables}_u, fea_u) \tag{1}$$

where  $\text{Tables}_u$  denotes the user embedding tables and  $fea_u$  denotes the user features.

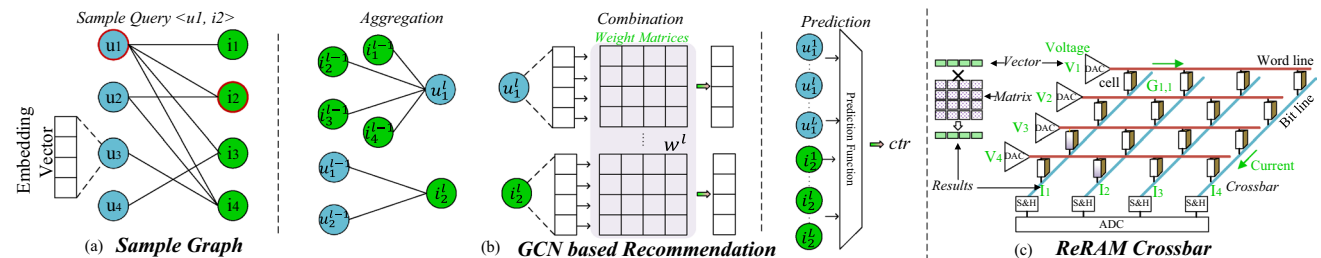
The embedding propagation layers play the role of refining the embedding vector by capturing information along the user-item graph. Each propagation layer is composed of an aggregation stage and a transformation, a.k.a. combination. Messages  $m_u^l$  for user  $u$  aggregated from it's neighbours  $i$  can be defined as follows:

$$m_u^l = \text{Agg}(e_i^{l-1} | i \in \text{Neighbour}(u)) \tag{2}$$

where  $l$  denotes the  $l$ -th propagation layer. The transformation can be defined as follows:

$$e_u^l = \sigma(\text{Combine}(m_u^l, W^l)) \tag{3}$$

where  $\sigma(\cdot)$  is an activation function such as  $\text{ReLU}(\cdot) = \text{Max}(0, \cdot)$ ,  $W^l$  is trainable weight matrix,  $e_u^l$  is user embedding representation in  $l$ -th propagation layer. We can obtain the item representation  $e_i^l$  with a similar process.



**Fig. 1** Overview of **a** A user-item graph example  $G$  and a sample query  $q < u_1, i_2 >$ . **b** GCN-based recommendation process of  $q$ . For layer  $l$ ,  $u_1$  and  $i_2$  aggregate from their neighbors to construct message  $m_{u_1}^l$  and  $i_2^l$ . The aggregated messages are then transformed in the com-

Based on the user and item representation from the propagation layer, the prediction layer makes predictions with a prediction function to make the final output as follows:

$$ctr = \text{Predict}(e_u^1, e_u^2, \dots, e_u^L, e_i^1, e_i^2, \dots, e_i^L) \tag{4}$$

where  $L$  is the number of propagation layers,  $ctr$  is the the click-through-rate.

The specific functions of different models of different layers are shown in Table 1. In the table,  $\odot$  denotes element-wise product,  $\oplus$  denotes concatenation operation,  $D_u$  denotes the degree of user  $u$ , and  $\alpha$  denotes layer-related model parameter. We can observe that there are mainly three types of operations in those functions: sum, matrix-vector multiplication, and vector multiplication. Those operations can be transformed into multiply-and-accumulation (MAC) operations. Although there are other operations, such as ReLU, MAC operations in those three layers dominate the overall performance of GCN-based recommender systems.

Besides, the user and the item can be destinations in the aggregation stage. This is because GCN-based recommender systems generate the final  $ctr$  by exploring the user-item interaction and dense information. Therefore, both the item and the user must be mapped on the crossbar to accumulate the incoming edge.

ReGCNR focuses on requirements for general GCN-based recommendation models (Huang et al. 2021; He et al. 2020; Wang et al. 2019; Berg et al. 2017). Those GCN-based recommender systems generate the final click-through rate by exploring the user-item interaction and dense information. The user-item interaction information directly

bination layer. Finally, the results of  $u_1, i_2$  for all  $L$  propagation layers are processed by the prediction function to generate the  $ctr$  for query  $q$ . (c) The computation principles of vector-matrix multiplication on ReRAM crossbar

**Table 1** Specific operations of key functions for different GCN-based recommender system models

Model	Aggregation	Combination	Prediction
GCMC	$m_u^l = \sum e_i^{l-1}$	$e_u^l = \frac{1}{ D_u } W^l m_i^{l-1}$	$e_u^1 e_i^1$
NGCF	$m_u^l = \sum e_i^{l-1} \odot e_u^{l-1}$	$\frac{1}{\sqrt{ D_u  D_i }} (W_1^l e_u^l + W_2^l m_u^l)$	$(e_u^1 \oplus e_u^2 \oplus \dots, \oplus e_u^L)^T (e_i^1 \oplus e_i^2 \oplus \dots, \oplus e_i^L)$
LightGCN	$m_u^l = \sum e_i^{l-1}$	$e_u^l = \frac{1}{\sqrt{ D_u  D_i }} W^l m_i^{l-1}$	$\sum \alpha_l e_u^l \sum \alpha_l e_i^l$

represents the user's preference for items. The user-user or item-item interaction is generally utilized as side information in emerging models (Wu et al. 2022). Thereby, the heterogeneous graph which has different user or item types, and the social recommendation, which considers user-user or item-item interactions can be considered as future work.

In conclusion, the GCN-based recommender system performance is dominated by the message propagation layer rather than the embedding lookup layer (Gupta et al. 2020; Ke et al. 2020) for DNN recommendation. This is caused by intensive computation in combination operation and sparse aggregation operations. Inside the message propagation layer, aggregation dominated overall system performance. Therefore, the message propagation layer needs to be considered as a preliminary layer in a GCN-based recommender system.

## 2.2 ReRAM-based PIM accelerators

ReRAM-based PIM (Wong et al. 2012; Niu et al. 2013; Hudec et al. 2016; Cheng et al. 2021) is widely deployed for graph (Song et al. 2018; Zheng et al. 2020; Challapalle et al. 2020; Lv et al. 2019), neural network (Chi et al. 2016; Song et al. 2017; Shafiee et al. 2016), and GCN (Huang et al. 2022; Arka et al. 2021; Zeng et al. 2023). ReRAM crossbar is well-suited for performing MAC operations and finishes a  $O(N^2)$  complexity operation at a time. Meanwhile, the computation is completed where the operator  $M$  is stored so that the data transferring cost between the computation and the memory device is reduced.

Figure 1b demonstrates the processing principle of analog ReRAM crossbar. For a given vector  $v$  by applying voltage on the word line, the result of  $VM$  can be obtained by sensing the current  $I_i = \sum I_j \cdot G_{i,j}$  on the bit line.  $G_{i,j}$  is the conductance of memristor cell on the  $(i, j)$  location of the crossbar with a shape of  $N$ .

## 2.3 Gaps between GCN-based recommendation and existing ReRAM-based PIM

*ReRAM-based graph processing accelerators:* In general, ReRAM-based PIM for graph processing (Song et al. 2018; Zheng et al. 2020; Challapalle et al. 2020) processes on data composed of edge ( $E$ ) and vertex ( $V$ ).  $V$  is composed of vertex attributes value and  $E$  is a sparse matrix.  $E$  is generally stored in compressed format, e.g., coordinate list (coo), for saving space. Due to the device size limitation, graph-based PIM accelerators generally split a graph into several partitions in a column or row manner and process one partition each time. For each partition, the vertex gathers data along incoming edges from its adjacent vertex to update its own vertex value. Such a gathering pattern provides opportunities for accelerating graph processing on ReRAM.

GraphR (Song et al. 2018) maps edge data on the crossbar, source vertex data are fed as the voltage on the word line, and the gathered information can be obtained on the bit line. However, those accelerators can not efficiently accelerate GCN-based recommender systems due to two main reasons. First, the recommender system processes multi-dimension vertex data while traditional graph processing focuses more on edge data (Huang et al. 2022; Naumov et al. 2019). For example, vertex data account for 59.51% of the model size for NGCF processed on the Gowalla dataset (Wang et al. 2019). Second, the GCN-based recommender system has much longer processing stages, as shown in Fig. 1. Those stages need to be well coordinated.

*ReRAM-based neural network accelerators:* Generally, the neural network is composed of multiple layers. Existing neural network accelerators (Chi et al. 2016; Song et al. 2017; Shafiee et al. 2016; Cheng et al. 2017; Qiao et al. 2018; Yang et al. 2019) exploit a spatial design that maps layers of the neural network on the crossbar to be processed in a pipelined way. Unlike graph processing, the neural network can be processed entirely on the accelerator or be duplicated for higher throughput because the parameter size is much smaller than graph data. Those accelerators are unsuitable for GCN-based recommender systems because the sparse matrix multiplication is hardly considered. Besides, the large embedding table size leads to larger model parameter discrepancy with traditional neural networks.

*ReRAM-based GCN accelerators:* Considering graph and neural network characteristics, existing ReRAM-based GCN accelerator (Huang et al. 2022; Yang et al. 2021) process graph data. In particular, GCN accelerators focus on accelerating the aggregation and combination kernel that dominate the overall system performance. Besides, specialized mapping designs for the vertex are considered in that accelerators (Huang et al. 2022). However, GCN-based accelerators are unsuitable for GCN-based recommender systems because the latter has a specialized computation phase. Meanwhile, GCN-based recommender systems mainly process user-item graphs.

*ReRAM-based recommendation accelerator:* REREC (Wang et al. 2021) focuses on accelerating the inner-product for feature interaction. Besides, MLP operations in the prediction layer are also considered in REREC (Wang et al. 2021) design. However, instead of interaction and prediction operations, the aggregation and combination operations in the message propagation layer dominate the system performance. For instance, the propagation layer accounts for 92.76% ~ 94.79% execution time for Neural Graph Collaborative Filtering (NGCF) (Wang et al. 2019) on the Gowalla dataset in our evaluation. Besides, considering the user and item vector access discrepancy, REREC chooses to map the item vectors on the ReRAM and apply user embedding as voltage. Differently, ReGCNR proposes

a joint degree mapping design that assigns the ReRAM resource by distinguishing user and item to coordinate the processing of each propagation layer.

### 2.4 Combining ReRAM-based PIM and GCN-based recommendation

To bridge those gaps, we propose ReGCNR with the following aspects.

First, ReGCNR adopts a unified MAC computation scheme for all the key components in GCN-based recommendation. For the combination kernel, the MAC is the multiplication of embedding  $e$  and each row vector  $M_i$  in matrix  $M$ . For the aggregation kernel, the MAC is the  $VM$ , where  $V$  is the vertex,  $M_i$  is the  $i$ -th column of the sparse adjacent matrix. For the prediction kernel, the MAC is the multiplication of embedding  $e$  and each row vector  $M_i$  in matrix  $M$ . For the inner-product kernel, the MAC is the  $e_u e_i$ , where  $e_u$  and  $e_i$  are user and item embeddings. While being unified into MACs, those kernels can be processed efficiently in the same computation manner. This lays a basis for integrating a GCN-based recommender system on ReRAM.

Second, we propose a specialized mapping schema for the entire model mapping (Sect. 4). Especially, a joint degree mapping for aggregation is proposed for assigning resources for both user and item aggregation among different propagation layers. For example, the inner production-based prediction maps the embedding vector on the crossbar, and neural network-based prediction maps the weight matrix on the crossbar.

Third, we design the query-driven pipeline and leverage the 3-D heterogeneous architecture for higher performance. The aggregation is executed in a way that separates user aggregation and item aggregation. Especially, this helps to balance latency for the aggregation kernel because we can assign different hardware for user and item aggregation, respectively. Therefore, the pipeline for processing one query can be more predictable. With the heterogeneous hardware design, the logic ReRAM layer is used to process

the MAC operations. The memory ReRAM layer is used to store the intermediate user and item embeddings. Therefore, both the large on-chip data storage requirements for embedding tables and the large computing requirements for the message propagation layer can be satisfied. What's more, the inter-layer connection of 3-D stacked architecture can reduce the embedding access latency.

## 3 Overall architecture

In this section, we first describe the architecture overview. Then, we give a brief introduction to the workflow of ReGCNR phase by phase.

### 3.1 Architecture overview

*ReRAM-based heterogeneous 3-D staked organization:* ReGCNR follows the design prototype of heterogeneous 3-D stacked architecture as shown in Fig. 2a. There are various heterogeneous 3-D stacked prototypes. We exploit a 3-tier logic-memory-logic (LML) heterogeneous 3-D stacked architecture (Kaul et al. 2022) based on the following considerations. Compared with one logic tier, LML can provide more logic ReRAM resources, which is helpful for computing kernels in GCN-based recommendation. Compared with LLL adopted in ReGraphX (Arka et al. 2021), LML had an essential memory layer for the large-scale embedding table. The logic layer consists of multiple PEs. The memory layer consists of multiple memory ReRAM units (MRU). Each memory unit provides 2 MBs of memory. There are 64 units on each layer in total. As shown in Table 2, GCNR provides 128 MBs on-chip memory and 16 MBs analog crossbar in total. Both those two kinds of resources decide the batch size that GCNR can process each time. Besides, the graph data is stored in a compressed graph data format to save space.

*Inter-layer connection:* Layers are connected by through-silicon vias (TSVs) (Kaul et al. 2022), which are dense and short interconnects offering high bandwidth for inter-layer

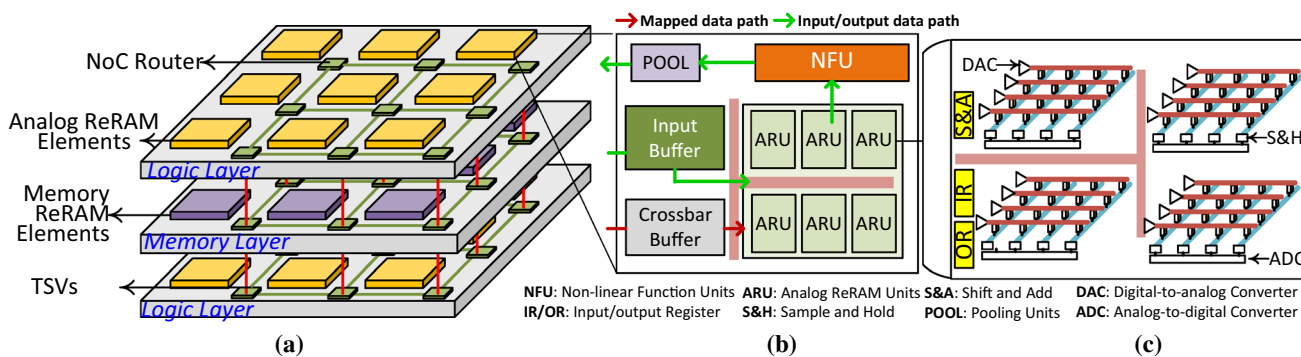


Fig. 2 Overview of a 3-tier heterogeneous 3-D stacked architecture, b processing elements, c analog ReRAM unit



communications. Prior works, such as Hetrapp (Huang et al. 2020) and ReGraphX (Arka et al. 2021), have utilized 3-D heterogeneous ReRAM to accelerate graph and GCN, respectively. The high bandwidth enables efficient data exchanges in embedding access across analog ReRAM and memory ReRAM. Due to the tensive access in embedding lookup operations, the shortened compute-memory distance further reduces the latency and the power consumption.

*Processing element:* The processing element consists of several analog ReRAM-based units (ARU), crossbar buffers, nonlinear function units, and pooling units. The ARU executes the main MAC operations of the entire system. The data buffer is used for buffering input data. The crossbar buffer is used for buffering the data to be mapped on the crossbar. The nonlinear and pooling function units are deployed for generality.

In ReGCNR, each ARU executes operations of one layer in a spatial way. Only the vertex to be aggregated will be mapped, and the combination and prediction kernel are generally assumed suitable for on-chip ARU resource (Huang et al. 2022; Shafiee et al. 2016). It's worth noticing that we do not need to process the entire graph in a partitioned way. This is because the recommender system processes one query each time, and the queries in a batch are random.

*Memory ReRAM layer:* The memory ReRAM layer (MRL) is composed of several memory ReRAM elements (MRE) and 3-D on-chip NoC. The MRE is designed to store intermediate embedding vector and on-chip edge data for both the user and the item. The MRL is integrated in the middle of two LRL layers so that the data transfer cost can be reduced. This is because aggregation for the user (item) takes  $e_i^l$  ( $e_u^l$ ) as  $l$ -th propagation layer output and generates  $e_u^{l+1}$  ( $e_i^{l+1}$ ). The MRE size and the number of propagation layers of the model decide the system throughput.

*Discussion on off-chip memory access:* The LML 3-D stacked architecture can provide a large amount of on-chip memory. However, it still requires access to off-chip memory that stores the original embedding vector and graph edge data. Specifically, the edge data is stored in two kinds of compressed formats for user-based aggregation and item-based aggregation. ReGCNR only processes queried user item embeddings. The size of batches is decided by the hardware resource and model size. The advantage of such a pattern is the multiple intermediate results for the propagation layer do not require to be written back to off-chip memory until the final result is obtained. The disadvantage is that the intermediate data will limit the number of queries that can be processed on-chip due to the multiple-layer structure of propagation.

What's more, a natural question is raised: will the off-chip memory access eliminate the in-memory processing benefits of ReGCNR? Such consideration is reasonable for

a DNN-based recommender system as embedding table lookup operation dominates the overall system performance. Since the performance bottleneck of the GCN-based recommender system lies in combination and aggregation layers, ReGCNR can promote the system performance by processing those layers in parallel.

### 3.2 Workflow

ReGCNR processes one batch of queries each time. Therefore, ReGCNR is essentially a data parallel accelerator. Each query is composed of user and item embeddings and is used to generate a *ctr*.

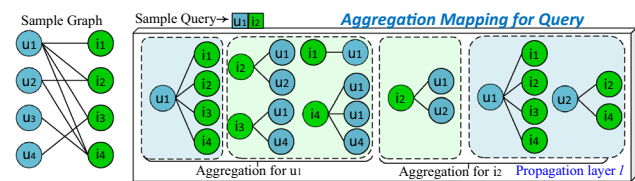
For each query, there are mainly three stages. The first is the aggregation stage. Only the embedding queried will be loaded from off-chip memory. The user aggregation and item aggregation are processed by different aggregation engines on LRE in ReGCNR design because those two kernels operate on different compressed formats, and the aggregation destinations are user and item, respectively. The aggregated embeddings are then written into MRE for combination. The second stage is the combination stage, and the combination engine fetches the aggregated result from MRE and executes the combination operation. The combination engine maps combination weight on the crossbar and fed embedding as inputs. The third is the prediction stage. After each layer of combination has been finished, the result will be written to MRL for subsequent prediction.

## 4 Model mapping

In this section, we discuss the algorithm mapping of ReGCNR. It is worth noticing that aggregation mapping is dominant in each query procedure. This is because the aggregation resource for each query is not fixed, while the resource of combination and prediction is based on the model parameter. After the mapping of aggregation is determined, the mapping of the other two kernels can be decided.

### 4.1 Aggregation mapping

We propose a joint degree mapping schema. The key idea of the schema is that the aggregation resource for each query is assigned based on the destination degree and the degree of its source. As demonstrated in Fig. 3, for propagation layer  $l$ , user  $u$  are mapped for aggregation to get  $e_u^{l+1}$  on the user aggregation kernel. User's connected items  $i$  are mapped on the item aggregation kernel at the same time to get  $e_i^{l+1}$ . With such a mapping schema, the user  $u$  aggregation for  $e_u^{l+2}$  in layer  $l + 1$  can be finished in one time of mapping because the all connected  $e_i^{l+1}$  have been calculated. This helps to



**Fig. 3** Overview of joint degree mapping designed for batch-processing in ReGCNR. For one query for that aggregate  $u_1$  and  $i_2$  on the sample graph, aggregation for  $u_1$  and  $i_2$  are calculated, respectively, together with its adjacent vertex

manage resource assignments for each query at the very first propagation layer.

Meanwhile, there is a well-known sparse problem that induces processing on invalid edges for graph-related applications. Existed work has proposed mapping the vertex on the crossbar rather than the edge to mitigate overhead caused by the sparse problem for GCN on ReRAM (Huang et al. 2022). We adopt the same vertex-based mapping schema because it is more suitable for models with multi-dimensional vertex vectors. However, the sparsity problem still causes unused rows on the crossbar. This problem is severe when the vertex degree is low.

Fortunately, low-degree vertices are further less accessed than high-degree vertex in the recommender system. This fact helps to reduce the overheads of the sparsity problem of the low-degree vertex. Besides, the sparsity overheads can be mitigated by reducing the crossbar size. However, this may generate extra overhead for aggregating high-degree vertex. Consequently, there is an interesting trade-off between high-degree and low-degree by varying the crossbar size  $\epsilon$ . We further explore the relationship between  $\epsilon$  sensitivity study in Sect. 6.

Finally, the mapping strategy adopted by ReGCNR only incurs one time of ReRAM crossbar write for aggregation in each propagation layer for each query. This is because only the vertex to be aggregated will be loaded in our query-driven design (Sect. 5.1). ReGCNR can support hundreds of billions of total inferences on average by conservatively assuming the endurance of the memristor cells (Niu et al. 2013; Qiao et al. 2018) is  $10^{12}$ .

### 4.2 Combination mapping

In the combination stage, our aim is to provide enough resources for every combination matrix of every propagation layer for each query. This is because REGCNR processes a batch of queries where each query can be in a random partition. This inherent property means ReGCNR can not be processed in a partitioned way. Consequently, it is more efficient for ReGCNR to work in an end-to-end way that provides enough resources for the entire procedure. Therefore,

it is necessary to map all weights of each propagation layer on the crossbar.

### 4.3 Prediction mapping

There are generally two kinds of prediction kernels: MLP-based kernel and inner-product-based interaction kernel for GCN-based recommender systems. The mapping of MLP-based prediction can be done as traditional ReRAM-based neural network accelerators, and we mainly discuss the inner-product-based prediction mapping below. As demonstrated in Table 1, the inner product multiplication is composed of vector multiplication between user and item embeddings of every propagation layer. Based on the computation pattern, we map the item embedding on the crossbar bar once the combination engine generates item embedding. To leverage the MAC property of the crossbar, the item is mapped on the same column of the crossbar. The user embeddings are fed into PEs mapped with item embeddings as input in a way that each embedding dimension corresponds to one row of the crossbar once the one layer of combination finishes.

It's worth noticing that despite the fact that each column of the crossbar is mapped with item embedding, only the column corresponding to the input data is activated for multiplication. This is because the interaction executes only between user embedding and item embedding of the same propagation layer  $l$ . Such a prediction mapping can only generate one result of one column each time. This can be tolerated because the prediction result does not take part in the subsequent layer of propagation. Therefore, the same PE for the prediction can process propagated results from a different query.

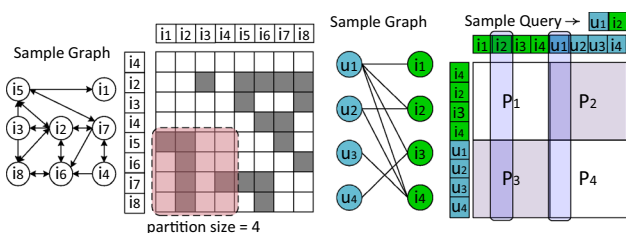
## 5 Pipeline and hardware scheduling

In this section, we describe the pipeline execution and hardware scheduling of the proposed architecture.

### 5.1 Query-driven pipeline

As demonstrated in Fig. 4, general GCN processes one partition each time. However, the GCN-based recommender system processes one query each time, and partition-based processes will generate ineffective results on columns. For processing the mapped PEs, we propose the query-driven pipeline execution as the basic processing model, which matches the nature of query processing in recommender systems.

*Pipeline on ReRAM:* Assuming each query is composed by  $e_u^l$  and  $e_i^l$ , which both has a length of  $d$ . For the user embedding  $e_u$ , the aggregation kernel for layer



**Fig. 4** Partition-based processing for general GCN processes one partition each time

1 is executed on all connected item embeddings, which includes one time of loading the source embedding and aggregation operation. Then, the aggregated result is written to the input buffer of PE mapped with the subsequent combination kernels. After that, the combination kernels for layer 1 are utilized for processing the aggregated results to get  $e_u^2$ . Then, the  $e_u^2$  is written to MRL for access to other processing elements.

After layer 1 is finished, the aggregation for layer 2 can start. In fact, the aggregation kernel for layer 2 can start as long as it has finished the aggregation task of layer 1 and the item embedding for layer 2 is ready. It's worth noticing that ReGCNR adopts the joint degree mapping schema that assigns enough resources for every propagation kernel at the very beginning of the propagation layer. Besides, this mapping schema ensures item embeddings are processed on the item aggregation kernel as the user embedding for layer 1 is processed. Therefore, the long tail problem on large-degree vertex aggregation can be mitigated, and the aggregation can start the next layer operation within one operation cycle. Similar to layer 1, the combination kernel of layer 2 can be done once the input data is ready. Such procedures are executed for the subsequent layers until all layers are processed.

Meanwhile, it's optional to process the prediction operation, which is decided by the model. The prediction is not in the pipeline as the result of the prediction does not take part in the subsequent layer of propagation. For example, for NGCF (Wang et al. 2019), the inner-product of  $e_u^l$ ,  $e_i^l$  can be executed to get the prediction result, a scalar  $s^l$  of layer  $l$ . Then  $e_p^l$  is written to MRE for the final ctr.

**Batch size:** The batch size is decided by the on-chip hardware resource and the specific model. Considering the hardware, the loaded graph has to ensure that the intermediate data of all queries in the loaded batch will not exceed the MRL capacity. Besides, the loaded queries should not exceed the ARU resource capacity. To cooperate with the querying-driven pipeline for a specific

model, we only load two kinds of embeddings each time: 1) the embeddings of the queried user and items, 2) the embeddings of the user (item) are connected to the queried item (users). Besides the embeddings, the edge data are stored on-chip in compressed format for the aggregation kernel.

## 5.2 Hardware scheduling

The query time for the individual user determines the quality of service for the recommender system in real-world applications. Therefore, we process the user and item on different hardware simultaneously so that the end-to-end query time can be reduced to nearly half. To achieve this, we propose the differentiated-user-item (DUI) execution for user and item processing. This is feasible because the processing pipelines of  $e_u$  and  $e_i$  have a unified processing pipeline. This time reduction can not improve the system throughput because two times of hardware is occupied for processing user and item embedding simultaneously. Specifically, the DUI engine processes each query in the following way. For aggregation, user aggregation and item aggregation are executed on separate hardware simultaneously. For combination we assign hardware for the user and item separately to ensure there are no hardware conflicts for combination.

**Degree-based execution schema for DUI engine on 3-tier accelerator:** To maximize the system performance under a hybrid execution pattern, we exploit the 3-D interleaved architecture to adopt the execution. We choose the 3-tier logic-memory 3-D architecture (Sect. 3) to harvest the following benefits. First, compared with 2-D architecture, LML architecture provides higher computing density and larger on-chip memory, which is essential for the GCN-based recommendation. Second, accessing the embedding on MRU connected with the same TSV connection can reduce communication overhead. Third, compared to one logic layer 3-D stacked architecture, LML architecture can provide more ARU resources, which satisfies large computation requirements in aggregation, combination, and prediction kernels.

However, the aggregated message volumes for the user and item can be different. Therefore, intuitively executing the user and item engine on each logic tier can cause unbalanced resource assignment between two tiers. Therefore, we propose the degree-based execution of DUI Engine on a 3-tier accelerator. Specifically, assuming the degree of the destination of one execution is  $d_{engine}$ , we can calculate the total degree of all engines for one logic tier  $d_{tier}$ . The schema ensures the  $d_{tier}$  of both logic tiers are close to each other. This can be done by choosing which tier to load the aggregation and combination of the query. With such a degree-based design, the hardware on both tiers can be fully exploited. Meanwhile,



the vertical transferring ability of both sides of MRL can be fully exploited because there is a large amount of data transferring for intermediate embeddings.

## 6 Evaluation

In this section, a comprehensive evaluation is conducted to assess the performance and energy efficiency of ReGCNR, in comparison with state-of-art solutions. Furthermore, the resource usage and sensitivity of ReGCNR are examined.

### 6.1 Setup

*ReGCNR setting:* Table 2 shows the hardware configuration of ReGCNR. There are three layers of ReRAM in ReGCNR architecture. Both two logic layers have the same hardware setting. Each logic ReRAM layer has 64 processing elements. Each processing element is composed of 8 analog ReRAM units, which are set with 32 crossbars. The crossbar array has a size of  $64 \times 64$ , where each cross point is a TaOx ReRAM cell that has a 29.31 ns read latency and a 50.88 ns write latency. (Zheng et al. 2020; Shafiee et al. 2016; Song et al. 2018). The cell precision is set conservatively with 2 bits for operation reliability. (Song et al. 2018; Xu et al. 2015). Table 2 assumes ReGCNR configurations.

Following the specification for LML heterogeneous stacked architecture (Kaul et al. 2022), we built a cycle-accurate simulator to model the on-chip behavior of components on ReGCNR. The on-chip network is modeled using Booksim 2.0 (Jiang et al. 2010). We model the ReRAM crossbar arrays energy via nvsim (Dong et al. 2012). We set a 32 nm process for a buffer, which is the same as REREC (Song et al. 2018), and estimate its latency, area, and power via CACTI (Thoziyoor et al. 2008). We set ADC and DACs with 8-bit and 2-bit precision, respectively, and their energy and area overheads are taken from ISAAC (Shafiee et al. 2016). The pooling and non-linear function parameters are adopted from (Shafiee et al. 2016). The time of model execution on the GPU platform is estimated based on the open-sourced framework (Berg et al. 2017; Wang et al. 2019; He et al. 2020). The CPU energy consumption is estimated based on Intel Product Specifications. The energy consumption of GPU is estimated with NVIDIA System Management Interface.

*Datasets and benchmarks:* Table 3 demonstrates 4 widely-used real-world datasets: MovieLens-100K (ML1) (Harper and Konstan 2015), Gowalla (GOW) (He et al. 2020), Yelp (YEL) (Huang et al. 2021), MovieLens-10 M (ML2) (Harper and Konstan 2015) for GCN-based

**Table 2** Hardware configurations of ReGCNR

Component	Param.	Spec.	Pow. (mW)	Area ( $mm^2$ )
ARU properties (8 ARUs per Tile)				
ADC	Number	32	64	0.00384
	Resolution	8 b		
DAC	Number	$32 \times 64$	8	0.00034
	Resolution	2 b		
S & H	Number	$32 \times 64$	0.020	0.000080
Crossbar	Number	32	6.2	0.0005
	Size	$64 \times 64$		
	bits/cell	2		
S & A	Number	16	0.80	0.00096
IR	size	4 KB	2.32	0.0038
OR	size	512 B	0.42	0.0014
PE properties (64 PEs per chip)				
ARU Total	Number	8	653.92	0.36384
I-Buffer	size	16 KB	5.17	0.021
C-Buffer	size	128 KB	39.6	0.135
NFU	Number	8	2.08	0.0024
POOL	Number	8	3.20	0.0019
Chip properties				
LRE total	number	64	90.11K	33.544
MRE total	size	128 MB	2.245K	22.016
NoC	Flit_size	128 b	75	0.58
Chip total	-	-	92.430K	34.124

**Table 3** Dataset parameters of different datasets

Dataset	User	Item	Interaction	Density
ML-100K	943	1682	100,000	0.0630
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp	31,668	38,048	1,561,406	0.00130
ML-10M	69,878	10,677	10,000,054	0.0134

recommendation model. All four datasets are widely used and cover a range of data by varying in terms of data size and density. We conduct experiments with several representative GCN-based recommendation models: GCMC (Berg et al. 2017), NGCF (Wang et al. 2019), and LightGCN (He et al. 2020) for real-world workloads. Considering the marginal improvements and even the overfitting problem, we set each model with a 3-layer GCN as default. Besides, the hidden features are set with 64 dimensions (Wang et al. 2019).

*Baseline:* We compare ReGCNR with a state-of-the-art ReRAM-based accelerator for recommender system, REREC (Wang et al. 2021). To make an apple-to-apple comparison with REREC, we set ReGCNR with approximately the same ReRAM resource as REREC. Besides, we deploy the benchmarks on the GPU platform, implemented

**Table 4** Hardware configurations of CPU, GPU, and REREC

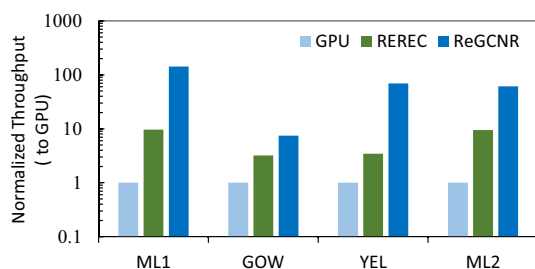
CPU	Intel Xeon CPU E5-2680 v4, 28 cores, 2.4 GHz
Cache	L1 64KB, L2 256KB, L3 35MB
Main memory	256GB DDR4
GPU	Tesla P100, 56 SMs $\times$ 64 cores, 1.33 GHz
Cache	L1 64 KB per SM, L2 4MB
GPU memory	16GB HBM2
REREC	419 MB MLP/memory arrays, 1048 KB inner-product arrays

by the constantly-updated tensorflow framework as a baseline on the general-purpose computing platform. The hardware specifications of GPU and REREC are listed in Table 4.

## 6.2 Performance evaluation

*ReGCNR vs GPU:* Figure 5 demonstrates the performance comparisons among GPU, REREC, and ReGCNR. ReGCNR focuses on throughput as the main performance improvement target because the real-word recommender systems are required to process numerous queries in a large batch simultaneously. The throughput result is equal to the execution time divided by the number of total queries in each test dataset. As the number of queries of the dataset remains the same for ReGCNR and GPU-based experiments, the throughput is proportional to execution time. Therefore, we can regard throughput as time performance speedup.

ReGCNR achieves  $7.44\times \sim 141.79\times$  ( $69.83\times$  on average) speedup over GPU. The main benefit comes from the crossbar structure, in-memory processing, and the architecture. In particular, ReGCNR achieves the highest speedup on ML1, which is induced by two reasons. First, ML1 is the smallest dataset among all four datasets, which means the least off-chip memory access for ReGCNR. Consequently, ReGCNR can harvest relatively high in-memory processing benefits. Such trends can also be seen on relatively small-size datasets YEL and ML2, where the speedup is  $68.93\times$  and  $61.20\times$ , respectively. The YEL is an exception because YEL has

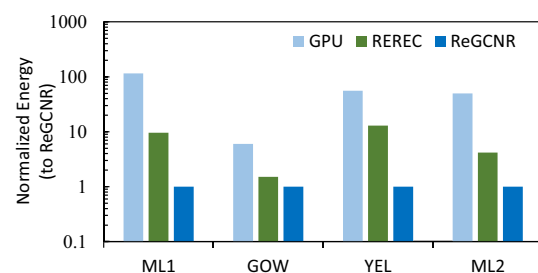
**Fig. 5** Performance comparisons among GPU, REREC, and ReGCNR

the lowest density, which induces a severe sparsity problem for ReRAM. Second, ML1 has the highest average degree, which induces a large amount of irregular memory access in the aggregation phase. The mapping schema adopted in ReGCNR can efficiently reduce data movements to mitigate the effects of irregular access.

*ReGCNR vs REREC:* As shown in Fig. 5, ReGCNR achieves  $2.32\times \sim 19.95\times$  ( $11.13\times$  on average) speedup over REREC. The speedup comes from the aggregation phase optimizations, including the mapping schema and the hybrid execution design. Besides, the high-speed layer-wise vertical communication of LML architecture also contributes to the speedup. It's worth noticing that we assume equally analog ReRAM resources for REREC for a fair comparison. Specifically, ReGCNR achieves a relatively high speedup on YEL and ML1, which is  $19.94\times$  and  $14.76\times$ , respectively. This is caused by two sides of reasons. The first side of the reason is that datasets with a larger degree, such as ML1, can harvest more aggregation phase benefits on ReGCNR. Second, the REREC mapping schema is more suitable for datasets where items are more accessed. Specifically, ReGCNR adopts a mapping schema that considers user and item, while REREC mainly maps items on the crossbar. Therefore, ReGCNR achieves a relatively low speedup on ML2 ( $6.44\times$ ), where the average item degree is much larger than the user degree.

## 6.3 Energy savings

Figure 6 depicts the energy savings of ReGCNR over GPU and REREC. Compared with GPU, ReGCNR achieves  $6.03\times \sim 115.05\times$  ( $56.67\times$  on average) energy savings. The energy savings mainly come from two sides. First, the data movement reduction of in-situ processing on the crossbar costs less energy. Therefore, GPU consumes  $1.66\times$  power over ReGCNR, despite the fact that ReGCNR is equipped with three layers of ReRAM. Meanwhile, the throughput advantage of ReGCNR brings an overall processing time reduction. Both two reasons contribute to the overall energy saving of ReGCNR. The highest energy saving ( $115.05\times$ ) is achieved on ML1, which has a relatively large size of irregular memory access in the aggregation phase.

**Fig. 6** Energy comparisons among GPU, REREC, and ReGCNR

Compared with REREC, ReGCNR achieves  $1.51\times \sim 12.95\times$  ( $7.22\times$  on average) energy savings. The main energy saving comes from the throughput improvement and the intermediate embedding accessing on the memory layer. The former reduces the overall execution time. The latter reduces the energy cost for aggregating embedding via planer NoC. Thus YEL and ML1 achieve relatively high energy savings, which are  $12.95\times$  and  $9.59\times$  respectively.

### 6.4 Resource breakdowns

*Resource breakdowns of kernels:* To demonstrate the adaptability of the mapping schema, Fig. 7 compares the analog ReRAM resource occupation breakdowns for different kernels. Each bar is composed of the blue part (aggregation), brown part (combination), and green part (prediction). In most of the dataset, aggregation occupies the major part of the chip resource, which is 95.92% (ML2), 74.02% (ML1), 57.76% (YEL), and 45.50% (GOW), respectively. Such an assignment coincides with the GCN-based recommendation property.

The GOW allocates the least resource for aggregation because the GOW has the lowest average degree. Therefore, a relatively smaller size of ARU is enough for the aggregation of one destination user or item. On the contrary, the combination and prediction kernel are not affected by the average degree. This indicates that the effort to reduce sparsity encounters a marginal effect on system performance because the combination shall dominate the overall system performance.

*Resource breakdowns in terms of users and items:* We also explore mapping and execution efficiency by exploring the hardware assignment pattern in terms of user and items. As depicted in Fig. 8, REGCNR manages to assign resources for different datasets. The resource difference between the user and item of ML2 is that there is a large difference between the user and item degree. The proposed mapping scheme is designed to address such situations

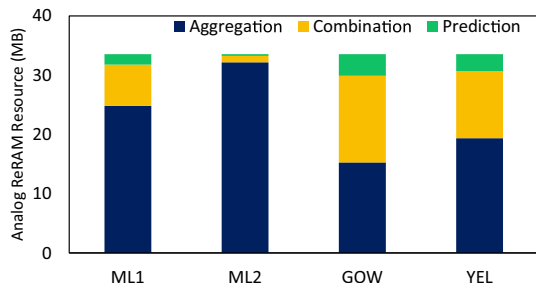


Fig. 7 Analog ReRAM resource occupation of aggregation, combination, and prediction for different datasets

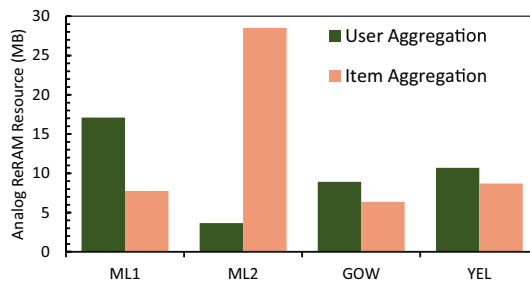


Fig. 8 Resource assignment difference of aggregation kernels on user and item

to balance the processing time cost of the user and item sides. By combining the mapping and hardware scheduling, ReGCNR achieves throughput improvements and energy savings for different datasets.

### 6.5 Sensitivity study

Figure 9 demonstrates the system performance under difference crossbar size  $\epsilon$ . For small  $\epsilon$ , the peripheral circuit, such as ADC and DAC, occupies most of the on-chip area. Thereby, only a small amount of ReRAM can be integrated into the chip. Therefore, such a small  $\epsilon$  throughput is the smallest for all datasets even though the sparsity problem can be mitigated with a smaller crossbar size.

A larger crossbar indicates that more ReRAM accounts for the overall chip area. Besides, for relatively larger  $\epsilon$ , our aggregation mapping schema helps to mitigate the sparsity problem. However, there is a marginal effect of crossbar size on the system performance as  $\epsilon$  increases larger than 512. This is because the embedding aggregation of a low degree causes a more severe effect on the overall system throughput. Meanwhile, the variation and current leakage problem in the ReRAM device will cause severe effects on device reliability and endurance when the  $\epsilon$  is too large (Wong et al. 2012; Niu et al. 2013).

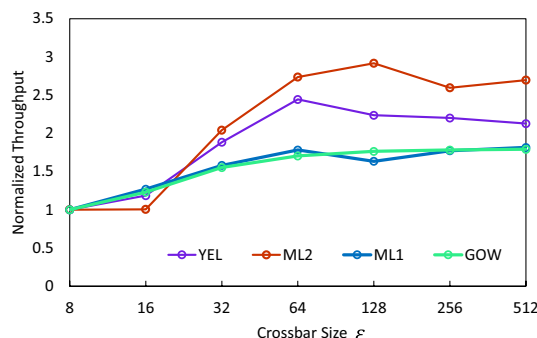
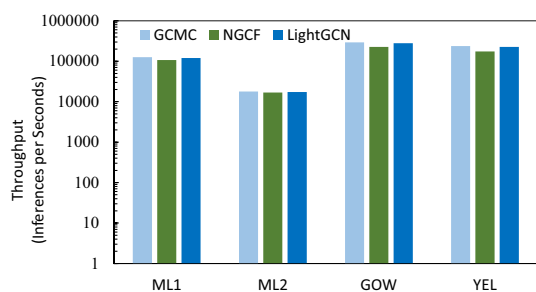


Fig. 9 Performance of ReGCNR on NGCF with varying crossbar size  $\epsilon$



**Fig. 10** System performance of ReGCNR on popular GCN-based recommendation models

Therefore,  $\epsilon = 64$  strikes a sweet spot for performance and reliability.

## 6.6 Generality

To show the ReGCNR generality for different models, we investigate the generality of the design by comparing the overall system performance for GCMC, NGCF, and LightGCN, as depicted in Fig. 10. ReGCNR achieves the highest throughput on GCMC for all other three models, which is  $1.26 \times$  over NGCF. The main difference lies in the operations of message construction in the propagation layer and the final embedding representation, as shown in Table 1. Unlike GCMC, NGCF adopts an element-wise multiplication that can exploit the MAC operation pattern on the crossbar. As for LightGCN, it achieves better performance than NGCF due to simpler combination operation.

## 7 Conclusion

In this paper, we propose a ReRAM-based accelerator, ReGCNR, for GCN-based recommendations. ReGCNR is featured with the following designs. First, ReGCNR exploits a 3-D heterogeneous architecture to fit the large-size intermediate data and extensive embedding access on GCN-based recommendation models. Second, ReGCNR proposes a mapping schema for the full execution stages of GCN-based recommendation with a specialized design for user-item graphs. Third, the hardware scheduling is leveraged to coordinate the execution of GCN recommender systems on the proposed heterogeneous 3-D ReRAM accelerator. Results show that ReGCNR outperforms state-of-the-art ReRAM-based solutions in terms of performance and energy.

**Acknowledgements** This work is supported by the National Key Research and Development Program of China under Grant No.

2023YFB4503400 and the National Natural Science Foundation of China under Grant No. 62322205, 62072195, and 61825202. This work is also supported by Zhejiang Lab (Grant No. 2022P10AC02).

**Data availability** The data that support the findings of this study are available upon request from the authors.

## Declarations

**Conflict of interest** The authors declare that there is no conflict of interest relevant to the content of this article.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Arka, A.I., Doppa, J.R., Pande, P.P., Joardar, B.K., Chakrabarty, K.: ReGraphX: NoC-enabled 3D heterogeneous ReRAM architecture for training graph neural networks. In: Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp. 1667–1672 (2021)
- Berg, R.V.d., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv preprint [arXiv:1706.02263](https://arxiv.org/abs/1706.02263) (2017). Accessed 10 June 2023
- Challapalle, N., Rampalli, S., Song, L., Chandramoorthy, N., Swaminathan, K., Sampson, J., Chen, Y., Narayanan, V.: GaaS-X: Graph analytics accelerator supporting sparse data representation using crossbar architectures. In: Proceedings of the International Symposium on Computer Architecture (ISCA). IEEE, pp. 433–445 (2020)
- Cheng, M., Xia, L., Zhu, Z., Cai, Y., Xie, Y., Wang, Y., Yang, H.: Time: A training-in-memory architecture for memristor-based deep neural networks. In: Proceedings of the Design Automation Conference (DAC). ACM, pp. 1–6 (2017)
- Cheng, C., Tiw, P.J., Cai, Y., Yan, X., Yang, Y., Huang, R.: In-memory computing with emerging nonvolatile memory devices. *Sci. China Inf. Sci.* **64**, 1–46 (2021)
- Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y., Xie, Y.: Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory. *SIGARCH Comput. Archit. News* **44**(3), 27–39 (2016)
- Dong, X., Xu, C., Xie, Y., Jouppi, N.P.: Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **31**(7), 994–1007 (2012)
- Feng, Y., Hu, B., Lv, F., Liu, Q., Zhang, Z., Ou, W.: Atbrg: adaptive target-behavior relational graph network for effective recommendation. In: Proceedings of the International SIGIR Conference on Research and Development in Information Retrieval (SIGIR). ACM, pp. 2231–2240 (2020)



- Gupta, U., Wu, C.-J., Wang, X., Naumov, M., Reagen, B., Brooks, D., Cottel, B., Hazelwood, K., Hempstead, M., Jia, B., Lee, H.-H. S., Malevich, A., Mudigere, D., Smelyanskiy, M., Xiong, L., Zhang, X.: The architectural implications of Facebook's DNN-based personalized recommendation. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 488–501 (2020)
- Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. *ACM Trans. Interact. Intell. Syst.* **5**(4), 1–19 (2015)
- He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: simplifying and powering graph convolution network for recommendation. In: Proceedings of the International SIGIR Conference on Research and Development in Information Retrieval (SIGIR). ACM, pp. 639–648 (2020)
- Huang, Y., Zheng, L., Yao, P., Zhao, J., Liao, X., Jin, H., Xue, J.: A heterogeneous PIM hardware-software co-design for energy-efficient graph processing. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS). IEEE, pp. 684–695 (2020)
- Huang, T., Dong, Y., Ding, M., Yang, Z., Feng, W., Wang, X., Tang, J.: Mixgcf: an improved training method for graph neural network-based recommender systems. In: Proceedings of the SIGKDD Conference on Knowledge Discovery & Data Mining (KDD). ACM, pp. 665–674 (2021)
- Huang, Y., Zheng, L., Yao, P., Wang, Q., Liao, X., Jin, H., Xue, J.: Accelerating graph convolutional networks using crossbar-based processing-in-memory architectures. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 1029–1042 (2022)
- Hudec, B., Hsu, C.-W., Wang, I.-T., Lai, W.-L., Chang, C.-C., Wang, T., Fröhlich, K., Ho, C.-H., Lin, C.-H., Hou, T.-H.: 3D resistive RAM cell design for high-density storage class memory—a review. *Sci. China Inf. Sci.* **59**, 1–21 (2016)
- Hwang, R., Kim, T., Kwon, Y., Rhu, M.: Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations. In: Proceedings of the International Symposium on Computer Architecture (ISCA). IEEE, pp. 968–981 (2020)
- Jiang, N., Michelogiannakis, G., Becker, D., Towles, B., Dally, W.J.: *Booksim 2.0 user's guide*. Stanford University, p. q1 (2010)
- Kal, H., Lee, S., Ko, G., Ro, W.W.: Space: Locality-aware processing in heterogeneous memory for personalized recommendations. In: Proceedings of the International Symposium on Computer Architecture (ISCA). IEEE, pp. 679–691 (2021)
- Kaul, A., Luo, Y., Peng, X., Manley, M., Luo, Y.-C., Yu, S., Bakir, M.S.: 3-D heterogeneous integration of RRAM-based compute-in-memory: impact of integration parameters on inference accuracy. *IEEE Trans. Electron Devices* **70**(2), 485–492 (2022)
- Ke, L., Gupta, U., Cho, B. Y., Brooks, D., Chandra, V., Diril, U., Firoozshahian, A., Hazelwood, K., Jia, B., Lee, H.-H. S., Li, M., Maher, B., Mudigere, D., Naumov, M., Schatz, M., Smelyanskiy, M., Wang, X., Reagen, B., Wu, C.-J., Hempstead, M., Zhang, X.: Recnmp: Accelerating personalized recommendation with near-memory processing. In: Proceedings of the International Symposium on Computer Architecture (ISCA). IEEE, pp. 790–803 (2020)
- Li, C., Jia, K., Shen, D., Shi, C.-J. R., Yang, H.: Hierarchical representation learning for bipartite graphs. In: Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), vol. 19. AAAI Press, pp. 2873–2879 (2019)
- Lv, X., Xiao, W., Zhang, Y., Liao, X., Jin, H., Hua, Q.: An effective framework for asynchronous incremental graph processing. *Front. Comput. Sci.* **13**, 539–551 (2019)
- Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., Dzhulgakov, D., Malleevich, A., Cherniavskii, I., Lu, Y., Krishnamoorthi, R., Yu, A., Kondratenko, V., Pereira, S., Pereira, X., Chen, W., Rao, V., Jia, B., Xiong, L., Smelyanskiy, M.: Deep learning recommendation model for personalization and recommendation systems. arXiv preprint [arXiv:1906.00091](https://arxiv.org/abs/1906.00091) (2019). Accessed 24 May 2023
- Niu, D., Xu, C., Muralimanohar, N., Jouppi, N. P., Xie, Y.: Design of cross-point metal-oxide ReRAM emphasizing reliability and cost. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD). IEEE, pp. 17–23 (2013)
- Qiao, X., Cao, X., Yang, H., Song, L., Li, H.: AtomLayer: a universal ReRAM-based CNN accelerator with atomic layer computation. In: Proceedings of the Design Automation Conference (DAC). ACM, pp. 1–6 (2018)
- Shafiee, A., Nag, A., Muralimanohar, N., Balasubramonian, R., Strachan, J.P., Hu, M., Williams, R.S., Srikumar, V.: ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *SIGARCH Comput. Archit. News* **44**(3), 14–26 (2016)
- Song, L., Qian, X., Li, H., Chen, Y.: Pipelayer: a pipelined ReRAM-based accelerator for deep learning. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 541–552 (2017)
- Song, L., Zhuo, Y., Qian, X., Li, H., Chen, Y.: GraphR: accelerating graph processing using ReRAM. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 531–543 (2018)
- Thoziyoor, S., Muralimanohar, N., Ahn, J. H., Jouppi, N. P.: CACTI 5.1. Tech. rep., Technical Report HPL-2008-20, HP Labs (2008)
- Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.: Neural graph collaborative filtering. In: Proceedings of the International SIGIR conference on Research and development in Information Retrieval (SIGIR). ACM, pp. 165–174 (2019)
- Wang, Y., Zhu, Z., Chen, F., Ma, M., Dai, G., Wang, Y., Li, H., Chen, Y.: REREC: In-ReRAM acceleration with access-aware mapping for personalized recommendation. In: Proceedings of the International Conference On Computer Aided Design (ICCAD). IEEE, pp. 1–9 (2021)
- Wong, H.-S.P., Lee, H.-Y., Yu, S., Chen, Y.-S., Wu, Y., Chen, P.-S., Lee, B., Chen, F.T., Tsai, M.-J.: Metal-oxide RRAM. *Proc. IEEE* **100**(6), 1951–1970 (2012)
- Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: a survey. *ACM Comput. Surv.* **55**(5), 1–37 (2022)
- Xu, C., Niu, D., Muralimanohar, N., Balasubramonian, R., Zhang, T., Yu, S., Xie, Y.: Overcoming the challenges of crossbar resistive memory architectures. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 476–488 (2015)
- Yang, Z., Dong, S.: HAGERec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *Knowl. Based Syst.* **204**, 106194 (2020)
- Yang, T.-H., Cheng, H.-Y., Yang, C.-L., Tseng, I.-C., Hu, H.-W., Chang, H.-S., Li, H.-P.: Sparse ReRAM engine: joint exploration of activation and weight sparsity in compressed neural networks. In: Proceedings of the International Symposium on Computer Architecture (ISCA). ACM, pp. 236–249 (2019)
- Yan, M., Deng, L., Hu, X., Liang, L., Feng, Y., Ye, X., Zhang, Z., Fan, D., Xie, Y.: HyGCN: A GCN accelerator with hybrid architecture. In: Proceedings of the International Symposium on High Performance Computer Architecture (HPCA). IEEE, pp. 15–29 (2020)
- Yang, T., Li, D., Han, Y., Zhao, Y., Liu, F., Liang, X., He, Z., Jiang, L.: PIMGCN: A ReRAM-based PIM design for graph

convolutional network acceleration. In: Proceedings of the Design Automation Conference (DAC). ACM, pp. 583–588 (2021)

Zeng, Y., Li, Z., Chen, Z., Ma, H.: Aspect-level sentiment analysis based on semantic heterogeneous graph convolutional network. *Front. Comput. Sci.* **17**(6), 176340 (2023)

Zheng, L., Zhao, J., Huang, Y., Wang, Q., Zeng, Z., Xue, J., Liao, X., Jin, H.: Spara: An energy-efficient ReRAM-based accelerator for sparse graph analytics applications. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS). IEEE, pp. 696–707 (2020)



**Xinyang Shen** is currently a Ph.D. student in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST) in China. His research interests include ReRAM-based processing in memory, graph processing and recommendation systems.



**Yu Huang** received a BS degree from the Huazhong University of Science and Technology (HUST), in 2016. He is now working toward a PhD degree in the School of Computer Science and Technology, HUST, in China. His research interests focus on distributed stream processing and graph processing.



**Long Zheng** is now an Associate Professor in the School of Computer Science and Technology at Huazhong University of Science and Technology (HUST) in China. He received his Ph.D. degree at HUST in 2016. His current research interests include runtime systems, and configurable computer architecture.



**Xiaofei Liao** received his Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2005. He is currently a Professor in the School of Computer Science and Technology at HUST. He has served as a reviewer for many conferences and journal papers. He was the recipient of the Excellent Youth Award from the National Science Foundation of China in 2018 and the CCF-IEEE CS Young Computer Scientist Award in 2017. He is a

member of the IEEE. His research interests are in the areas of system software, P2P system, cluster computing, graph processing, and streaming services.



**Hai Jin** is a Cheung Kung Scholars Chair Professor of computer science and engineering at Huazhong University of Science and Technology (HUST) in China. Jin received his Ph.D. in computer engineering from HUST in 1994. Jin worked at The University of Hong Kong between 1998 and 2000 and as a visiting scholar at the University of Southern California between 1999 and 2000. Jin is the chief scientist of ChinaGrid, the largest grid computing project in China. Jin is an IEEE Fellow,

CCF Fellow, and a member of the ACM. He has co-authored 22 books and published over 800 research papers. His research interests include computer architecture, virtualization technology, big data processing, data storage, and system security.